IBM

Enterprise Multiplatform Auditing

- Auditing tools available for the enterprise
- Auditing in a WebSphere Application Server environment
- How to audit the privilege users

Paola Bari
David Edwards
Thomas Hackett
Jack Jones
Klaus Heinrich Kiwi
Zhu Min
Hanne Wesselsen
Eran Yona

Redbooks

ibm.com/redbooks

International Technical Support Organization

Enterprise Multiplatform Auditing

June 2008

SG24-7472-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (June 2008)

This book is based on z/OS 1.9. The following products were also used:
- Red Hat Enterprise Linux Release 5
- SuSE Linux Enterprise Server Release 9
- Tivoli Access Manager for Operating Systems
- DB2 for z/OS V8
- WebSphere Application Server V6.1
- Tivoli Access Manager for e-business

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the products and/or the programs described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ®
AIX 5L™
AIX®
CICS®
Cloudscape®
DB2 Universal Database™
DB2®
developerWorks®
DFSORT™
Domino®
IBM®
MVS™
OS/390®
Parallel Sysplex®
RACF®
Redbooks®
System z™
Tivoli Enterprise Console®
Tivoli®
WebSphere®
z/OS®
z/VM®
zSeries®
Linux®
Windows®
Microsoft®
Java™
UNIX®

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

EJB, J2EE, J2SE, Java, JavaScript, JDBC, JSP, Solaris, Sun, Trusted Solaris, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Access, Excel, Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

In today's flattened business world businesses need to demonstrate compliance to different laws like Sarbanes Oxley, HIPPA, EU s 8 directive, privacy laws, and so on, in an enterprise (horizontal) environment. The environments become more and more complex with the rapid growth of e-business, and they often span several geographies. Most IT organizations are still very much vertical, often with different organizations within each country, with little cooperation between them. Lack of cross-platform skills are also a major issue.

Integration and automation at the infrastructure layer is key to enable e-business. With the growing number of security databases, it is very complex to prove compliance across the enterprise. This is also true for auditors, as they need very deep knowledge of IT and a variety of solutions and IT Infrastructures to be able to do a reliable audit.

This IBM® Redbooks® publication is the result of our efforts to effectively audit user activity in an enterprise. It includes the generation of auditable activity, the capture and consolidation of the audit records, and performing reporting, such as compliance reporting, against this data.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Paola Bari is an Advisory Programmer at the International Technical Support Organization, Poughkeepsie Center. She has 26 years of experience as a Systems Programmer in OS/390®, z/OS®, and Parallel Sysplex®, including several years of experience in WebSphere® MQ and WebSphere Application Server.

David Edwards is a Consulting IT Specialist with the Tivoli® Global Response team. He is based in Australia. He has 20 years of experience in IT covering areas as diverse as application development, CICS® systems programming, distributed systems management, and being a product specialist in the Tivoli Security products. He holds a Bachelor of Science (chemistry and applied mathematics) degree from Monash University and a graduate diploma in computer science from Swinburne University. He has written extensively on the Tivoli Systems Management and Security products, including co-authoring four IBM Redbooks, as well as authoring a number of IBM Redpapers and developerWorks® articles.

Thomas Hackett is a Senior Software Engineer at the IBM System z™ New Technology Center in Poughkeepsie. He has nine years of experience in technical sales support covering service-oriented architecture (SOA) and e-business products on the z/OS operating system. He is known worldwide for his expertise on WebSphere Application Server security, having co-authored two previous Redbooks in that area. Tom has worked at IBM for 27 years, including positions in development, test, and management, all focused on the mainframe. He is a Certified Information Systems Security Professional (CISSP) and holds a Bachelor of Arts degree in Philosophy from Drew University, a Master of Divinity from Wesley Theological Seminary, and a Master of Science degree in Computer and Information Science from Syracuse University.

Jack Jones is an IT Specialist who has worked for IBM for over 30 years. He has held several technical jobs within IBM including Programmer, System Programmer, Technical Field Support Representative, Instructor, working IBM services, and now is a Solution

Architect. He currently works for the Advanced Technical Support Organization in System z security. He has written several security Redbooks and regularly presents at worldwide conferences and user groups.

Klaus Heinrich Kiwi is a Software Engineer working for IBM Linux® Technology Center in security development and is based in Brazil. His experience includes working as an IT consultant in Network and Security Management deployments, contributions in the Red Hat Enterprise Linux 5 and Novell SuSE Enterprise Linux 10.1 Common Criteria certification efforts, and more recently with software development for the Linux Audit subsystem. Klaus holds a computer engineering degree from Campinas State University (2004) and has articles published in the IBM developerWorks Web site.

Zhu Min (朱 闽) is a Software Specialist for IBM System z in the IBM China software group, where she performs technical support for major banking customers. She holds a bachelor's degree in international economics and finance, as well as a Master of Science degree in Computer Science from Beijing University. Her areas of expertise include System z operation system base, DB2® UDB, DB2, and Information M.

Hanne Wesselsen is an IBM Certified Consulting IT Specialist, Software. She started working in IBM Norway in January 1998 as SYSTEMS PROGRAMmer working on OS/390 and SO/Service Delivery. Through the years, she has held positions like Nordic Lead Domain Architect Tivoli Identity and Access™ Management, Lead Architect Identity and Access Management within Security, Asset and Risk Management in IT Delivery Global Technology Integration and Management and IT Delivery Early Skills Enablement Programme lead Tivoli Security Pan IOT (SWG programme to enhance skills on IBM SW including education, mentoring, and lab training). Hanne is a member of the WW IBM Global Services/Tivoli Governance Board/Security for Pan IOT.

Eran Yona is an IT Infrastructure Architect from the Israeli Ministry of Defense. He has 16 years of experience in IT. He has a Bachelor of Arts degree in business from the Collage of Management in Israel. His areas of expertise include Datacenter management, Datacenter Infrastructure, Disaster Recovery solutions, and virtualization.

Thanks to the following people for their contributions to this project:

Mike Buzzetti
IBM Poughkeepsie

Axel Buecker
International Technical Support Organization, Austin

Peter Birk
IBM Austin US

Monique Conway
IBM Software Group, Tivoli

Rich Conway
International Technical Support Organization, Poughkeepsie

Marcelo Eliseu
IBM Brazil

Elisa Ferracane
IBM Austin US

Jan-Anders Thorsveen
IBM Norway

Michael Tebolt
IBM Poughkepsie

Alyson Comer
IBM Endicott US

Michael Gray
IBM Tivoli Development, Australia

Kerry Gunn
IBM Tivoli Development, Australia

Reese Meadows
IBM Software Group, Tivoli

Guillermo Mejía
IBM Software Group, Tivoli

Mark Nelson
IBM Poughkeepsie

John Dayka
IBM Poughkeepsie

Roger St Denis
IBM Silicon Valley

Special thanks to Mark Hahn and Max Rodrigez for the zSecure and TCIM training.

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Strategy overview and security roles

This chapter provides an overview of the System z security strategy, outlining the different security phases and putting the correct emphasis on the auditing and compliance tasks that are the objectives of this chapter. We explore many of the System z products and their relevance to enterprise-wide auditing.

1.1 IBM System z security strategy

The System z platform has offered many services that have provided enhanced computing services required by production business applications since its creation in 1964. One of these services that the System z platform has always been associated with is an extremely high level of system security. With the introduction of RACF® and the system integrity statement in 1973, the System z platform and its predecessors have always been perceived as the most secure platform within the computing enterprise.

The distributed nature of today's computing environment and introduction of many new standards and legal requirements has made IT security implementation much more complex and difficult. Providing a secure enterprise that can be monitored, measured, and audited is becoming an almost impossible task for today's CSO, CISO, security administrators, and corporate auditors. How can the System z and its security features be utilized to assist with these new requirements on the enterprise?

For many years, the System z strategy has been a strategy of centralization. Historically, most of today's production commercial business data is stored within the bounds of the System z security controls such as RACF. A centralized view of data, the most valuable of an enterprise's resources, provides a much easier way to control the administration of the security rules to access the data and monitor and audit who is using the resources. The advent of tools such as IBM Tivoli Identity Manager also assisted with the management of identities across the enterprise and to some extent with the auditing of a user's authority. But as applications became more and more distributed across the enterprise, and as it became common for transactions to cross between the System z and other platforms, the centralized strategy began to have major difficulties in the areas of auditing and identity propagation. It became common for users to authenticate themselves on a distributed platform and then to engage a J2EE™ application running within the security confines of WebSphere Application Server and then request data within DB2 on a System z platform. More often than not, access to the DB2 data on the System z platform would be done with a server or application RACF user ID and the audit trail would not be able to capture the authenticated user who initiated the business request from the distributed application. This is not acceptable in today's IT environment and cannot be completely and practically solved with a System z security strategy that only uses centralization. Nor can System z accept that it is the most secure platform and pretend that all these difficulties are somebody else's to solve without the assistance of System z.

So how is System z planning to handle this new complex IT environment and to address these ever-changing problems? In a simple word, it is simplification. But this is a long-term statement of the goals and an understatement of the problem. Also, things may look far from easy for a some time before the dawn of simplification begins to show itself to the security administrators.

The System z strategy focuses on three major points:

- Strengthening the core

  As stated, System z has always been viewed and continues to be viewed as the most secure platform within the enterprise. As the foundation for the System z security strategy, the core quality of service (QoS) of System z security and integrity must be maintained, enhanced, and strengthened. Without this basic core, the rest of the security strategy falters. The System z platform has been built with an internal coordination between the System z hardware and the z/OS (or z/VM®, or both) operating system. This has allowed highly integrated cooperation where the System z hardware features can be utilized to protect and isolate the z/OS operating system. As long as the z/OS operating system is isolated from the applications and business processes by the System z hardware, then its

integrity can be insured. This is the basis of the IBM statement of integrity for the z/OS operating system. To paraphrase the z/OS statement of integrity: As long as nothing is done to inhibit this coordination between the System z hardware and z/OS operating system, IBM can assert that all the resource managers within the z/OS operating system will ask the appropriate security questions before allowing a requestor to access a resource.

RACF, working with the z/OS operating system, provides the system security. But the system security is only as good as the underlaying system integrity. With its history in system integrity and system security, z/OS has to insure this core foundation as it extends itself into the realm of today's new technologies, standards, and legal and business requirements. As these new features and functions are added into the evolving z/OS and System z environments, every precaution is taken to be sure that the core basic security beliefs are designed, developed, and tested into all the new advancements. When UNIX® System Services and TCP/IP with all its networking applications were added onto the z/OS system, all the basic system integrity and security requirements were fulfilled. Even some enhancements such as Application Transparency-Transport Level Security (AT-TLS) have been introduced, which with the introduction of the policy agent will lead to a simplification of the z/OS network security policy management. When Public Key Infrastructure (PKI) was announced as the choice for a secure authentication method, a complete PKI solution was developed as a component of the z/OS operating system. But it was done within the infrastructure of the z/OS system integrity and security confines, utilizing the hardware and crypto coprocessors, as well as the RACF environment, to their fullest potential for performance and security capabilities. The introduction of IBM Tivoli zSecure products are also an excellent example of how z/OS is moving to simplify the management and administration of RACF and z/OS security and integrity. These are just a few of the examples that demonstrate how z/OS evolution has and will continue to add new features and capabilities to its business solution and, not but that the evolution will be done it such a way as to exploit and enhance the core System z integrity and security. It is the base of System z security strategy and a fundamental feature of System z.

- Extending the Security QoS into the enterprise

  With the core of the z/OS and System z environment secure, the next phase of the System z security strategy is to insure that security for the complete IT enterprise is sound. It is a well-known fact that a computing environment is only as secure as its weakest server and poorest developed application. The System z hardware and operating systems cannot provide the security environment for every application and server within the installation's environment, but it can help protect the installations' most valuable resources—its data. As long as the data is on the System z platform and is protected with RACF and the z/OS operating system, it can be protected by the most capable environment as described in the first step of this strategy. But what about when the data has to be transported off the platform to the user or application? In previous strategies and arguments, it was a common theme to indicate that the application should be on z/OS so that it could protect the data as it should be protected. Copying data only leads to multiple security rules for the same data, leading to security management, auditing, and access problems. Centralization is a valid argument and has many advantages. But it is sometimes impractical and, even if you could move all your applications to the z/OS platform, there are still many unanswered questions such as identity propagation of that authentication distributed user requesting to run the desired z/OS application.

  IBM has developed and continues to develop many features in this effort to extend its security domain out into the distributed enterprise. One excellent example of this is the IBM Encryption Facility. This facility encrypts data that is going to be transported off the z/OS system whether the reason for the transport is for backups or use by a distributed application. The key that is used to protect the data before it goes on the transport media is stored and protected within the RACF and z/OS environment. If the data is requested back on a z/OS platform then the key can be requested from z/OS. If the data is requested

from a distributed server, there is a Java™ client application that can be used to authenticate to the z/OS system and request the appropriate key if the user is authorized.

Identity authentication, propagation, and management is another area where z/OS has been working to ease the complexity of today's security requirements. There are examples of this that have existed for several years, such as the LDAP server, which has extended the capabilities of distributed applications and servers to use RACF user and group information for authentication or potentially for some authorization. The PKI solution has also been mentioned above and is another good example of z/OS being able to handle many types of identity management solutions. IBM Tivoli components that work with identity management and propagation have also moved onto the z/OS system with IBM Tivoli Identity Manager and IBM Tivoli Federated Identity Manager being able to run on z/OS. More recently, applications such as DB2 v9 and WebSphere Application Server have used the z/OS capabilities to map distributed identities to local RACF identities so that z/OS can give a complete audit record with the distributed authenticated user's identity in the audit record.

z/OS continues to work on these security advancements to ease the labyrinthine that is enterprise security management. System z will move forward to allow z/OS resources to be protected as well as they can when they leave the security domain of z/OS, and to completely validate all users who enter the z/OS system no matter where they are authenticated.

- Ensuring compliance and enterprise auditing

  A requirement that gained a lot of attention recently is auditing of the enterprise security environment. Payment Card Industry Data Security Standards (PCI-DSS) and the large disclosure of our private and personal data has been a major reason for this new emphasis on compliance and auditing. z/OS and RACF have always used a centralized protected security audit log called System Management Facility (SMF) that has provided an accurate audit trail of users and resources of the z/OS system. The security records that are recorded in the SMF security log have been enhanced and updated as new z/OS features and characteristics have evolved. With the recent introduction of the IBM Tivoli's zSecure Audit and Alert, reporting and monitoring of the z/OS systems has become easier and more manageable. zSecure Audit also does more than reporting on just the security of the z/OS system, as it will also report on the system integrity features and critical system configuration requirements. All of this goes hand-in-hand with a newer component of z/OS that has been evolving called health checker. This component takes periodic checks of the z/OS system and reports on how it currently is running and how its critical configuration files are being protected.

  New enhancements to the TCP/IP applications such as the LDAP server have also added the option of being able to extend the z/OS auditing and reporting capabilities to distributed environments. The newest version of the z/OS LDAP server, called Tivoli Directory Server for z/OS (TDS for z/OS), has introduced an audit enhancement so that security changes and authentication can be logged to the SMF security log. A version of this new LDAP server also runs on z/VM with RACF/VM so that SMF security records can be logged for the z/VM system and merged with the z/OS SMF records to provide a complete and consist report of all the operating systems that record to SMF.

  All of these advances, along with recent developments such as the Linux audit plug-in, which can format the Linux audit log into SMF records, and a JACC provider for WebSphere Application Server, which uses JAAS to build a complete subject for your J2EE application and will log the RACF usage in SMF, allow for a more complete and consistent audit of everything that enters and uses the z/OS system and its resources.

  It is this third phase of the System z security strategy that we demonstrate with a simple example of a typical e-business environment.

1.2 Security aspects

Information security is a very broad topic that involves protecting the confidentiality, integrity, and availability (CIA) of information. The focus of this book is on auditing system use and how this can assist compliance. Tied to auditing is knowledge of who is using the systems and what resources they are accessing (for example, who has been authenticated and what they have been authorized to access). The following sections look at the interrelated areas of authentication, authorization, and audit as they relate to Information Security.

1.2.1 Authentication

The act of authenticating something or somebody is to establish or confirm that the subject is authentic. In computer terms, this means that the person, server, or message that is being authenticated is who or what it claims to be. This implies a few different elements based upon the factors involved, but all of the factors are trying to establish that we trust the identity and validity of the object we authenticating. For example, if we are authenticating a message, then the authentication process might involve not only validating the message, but also its source and integrity. When we authenticate a person or server, we might be involved more with validating the claim that the person or server is who they say they are with various methods and authentication factors.

Authentication has been an issue for a long time. The origin of the word dates back the Greek word *authentikos*. Authenticating someone implies two items: First, the person or server must make a claim that they are a certain person or server. Second, we validate this claim by one of several possible authentication factors such as:

- Something the user knows

  This is currently the most commonly method used and includes a password or a personal identification number (PIN). This is also viewed as the weakest of authentication factors and is sometimes referred to as basic authentication.

- Something the user has

  This is a fast-growing method that uses items that the person has such as digital certificates or security tokens. This is viewed as much stronger than basic authentication and is sometimes referred to as strong authentication.

- Something the user is or does

  This requires the person to provide an action or a part of themselves for authentication such as a fingerprint or retinal scan, or a signature or voice recognitions. This is also viewed as much stronger than basic authentication but has some cost and management issues.

All of these authentication factors have good and bad points, and security specialists are always looking for ways to enhance the authentication methodologies. One of the interesting methods that is currently being investigated is *someone the user knows* or social networking for authenticating a person. But more commonly, if stronger authentication methods are needed, then the factors are combined into what is called two-factor authentication. This is where the user may be asked to provide a password (something the user knows) for authentication, and based upon this form of authentication they are allowed to perform certain actions within the computer system or application. But when the user tries to perform more sensitive operations or access critical data, then he is asked for a second item such as a digital certificate (something the user has) to provide a stronger form of authentication.

Since this book is not about security in general, the method and elements of authenticating the users and servers are not reviewed here. But the fact that the user has been

authenticated and the act of authentication is logged and can be audited is critical. That was our attempt to audit the level of trust between the users and the different servers or applications within this document.

1.2.2 Authorization

Authorization is the act of allowing a user or consumer to access a resource based upon a set of rules that have been established. Resources are applications, data, or anything on a computer system that is needed by that user to do their job. Resources are protected by rules or access control lists that identify the users or groups of users who have access to the resource and the level of access that the users have access to the resource. The management of the protection of the resources is the implementation of the enterprise's security policy. In this book we do not look into the management of these resources and their access control lists, except to the point out that we need to be able to log and audit the changes to the security policy. Examples of questions that need to be addressed by the auditors are:

- Who is changing the access control lists?
- Who is providing users with the authority to access the system?
- What resources can a user access and at what level?
- Is the enterprise security policy implemented appropriately?

More of this is discussed in the following section, but basically we need to be able to identity who can access which resources, when, and at what level of access. That is, did the user just read the data or did they change the data?

One of the factors behind authorizing users to resources is that the users have been authenticated and classified by the type of user they are. Users should only be allowed access to the resources that they need to do their jobs and they should only have access to the resource at the appropriate level. There are basically three types of users:

- Anonymous users, who should have little or no access to only public resources
- Privileged users, who have special privileges on the system and can change system configuration, enterprise policies, and application behavior
- Regular users, who are partially trusted users who need access to the appropriate resources at the correct level to do their jobs

All of these different types of users and their access to resources needs to be logged and audited at the appropriate level and time, based upon the enterprise's security policy and the current activities. The privileged users should always be logged for their privileged activities. This is one of the special cases that will be investigated in this document.

1.2.3 Auditing

Auditing of a computer system is the systematic technical assessment of the enterprise security policy's implementation within that computer environment. The audit can be done on many different levels from one user or application to an entire enterprise's computer resources. An audit is usually a manual, after-the-fact event instead of the monitoring activities that are usually more real time. An audit is based on the log records that are written during the security activities, such as authentication of a user and authorization to a resource. One of the issues that has to be addressed is the lack of standard logging and audit recording formats on the different servers, applications, and computers. Even in our simple environment, the free form of the logging records presented challenges that had to be addressed.

But the audit reports have recently become a more critical process due to the required legal demands for compliance and certification. Also, an audit view of an entire enterprise and not just a single server or computer has become an important requirement within these legal requirements. To meet these requirements, a tool was needed that would:

- Provide an enterprise view of the transaction and the resources it used.
- Be able to handle many different types of log records from many different systems.
- Be an off-the-shelf auditing tool that requires little or no coding to be able to handle our environment and the different log records.
- Be able to report an individual user, especially a privileged user, as well as application activities and resource usage.

For this book, the individual systems did some auditing on their privileged users using the individual system auditing tools. For example, to audit on the DB2 environment the DB2 Auditing tool was used to do some auditing on the DBAs. The same was done for some of the other security environments, such as IBM Tivoli Access Manager and the WebSphere Application Server's administrators. But for an enterprise view of the users, resources, and transactions, the Tivoli Compliance Insight Manager (TCIM) was used in an attempt to answer the typical auditing questions and to get details of all activities across the computing environment.

1.3 Policy and compliance

In today's global world, auditing is becoming a requirement as a first step to satisfy the need to be in compliance with, for example, laws like Sarbanes Oxley, Hippa, the European Union's 8t. directive, privacy laws, and so on. The IT environments become more and more complex with the rapid growth of e-business, and they often span several geographies.

Most IT organizations are still very vertical, often with different organizations within each country, with little cooperation between them. Lack of cross-platform and cross-enterprise skills are also a major issue.

So what is needed to get compliant? The IT Governance Institute defines IT Governance as "Achieving the organization's goal by adding the value while balancing risk versus return over IT and its processes." IT Governance measures are "Key goal indicators, Key performance Indicators, Critical Success Factors, and Maturity Models." Simply put, IT Governance and Compliance is good for businesses. Several reports and measurements says that businesses that have governed their IT show better results than those that do not.

So what is security and how can we define it? Security is a function of:

- People
- Processes
- Technology

Administration and control of identity, access, and compliance can be a complex undertaking that covers several platforms, technologies, systems, organization, and people.

Integration and automation of the infrastructure layer are key for enabling e-business. With the growing number of security databases, proving compliance across the enterprise is a complex task. This is also true for auditors, as they need a very deep knowledge of the business *and* the supporting IT systems, across a variety of solutions and IT Infrastructures, to be able to do a reliable audit.

Choosing the best tools for compliance is important, but even more important is that the tools can deliver integration across the enterprise and the different solutions.

But tools can only cover approximately 20% of the compliance tasks necessary for being compliant. Before even thinking of any additional tools or software, an enterprise needs to have processes and preferably business roles in place. Putting all processes and business roles in place for governing and administrating the IT systems is a time-consuming task.

Today's picture

Most of today's IT environments consist of several platforms, systems, and applications, usually dispersed across different countries and built on non-standardized applications and systems. The biggest challenges for these installations are:

- Dealing with extremely complex and varying laws. Different countries often have their own laws, such as the European Union and the United States.
- Dealing with the fact that each application and system has its own user registry and user interface.
- Dealing with different processes.

Every organizational unit and department administers access in its own way. Often, even a global company has several sister companies in different countries that:

- Have different human resources systems, in different formats, and with different person objects.
- Often do not a differentiate between consultants and employees (consultants are not registered in the HR system).
- Use different approval mechanisms or systems.
- Are paper based or stored on e-mail.
- Use distributed security and access control.
- Use different reporting routines.

What is needed on the organization side

Take central ownership and sponsorship for the business organization, institute the role of a compliance officer role administrator (organized under a compliance officer), and centralize identity and access administration and control. In such a vision, security should be a part of the central management team.

Audit and compliance for identity and access management can be divided in three areas:

- The W7 scenario:
  - Who
  - What
  - Where
  - When
  - On what
  - Where from
  - Where to

  For example, for the data, you need to control the following elements:

  - What are your most critical data?
  - Where are they?
  - Who owns them?
  - Who gives access to them?
  - Who has access to them?

- What kind of access do they have?
- How are the privileged users and their access controlled?

If you have reliable information, it is easy to categorize, automate, measure, and do incident reporting and monitoring.

In our environment we used Tivoli Compliance InSight Manager to measure on W7, which is discussed later.

- Life-cycle identity and access management

  The basis of this task is defining the business roles. If your installation is not operating this way, the suggested way to implement the change is to start with a small pilot environment using the following methodology:

  - Each department needs to be interviewed to find the business roles needed to be aligned into the different IT systems.
  - Each person in each department needs to be linked to one or more business roles.
  - The business roles then need to be aligned with the different data resources and groups at the IT application/system level.
  - The groups must be connected to the correct data resources.
  - The department persons are then connected to the groups.
  - The processes can then be updated with the new roles and eventual new approval structure, administered by the compliance officer/role administrator. All new roles or changes to existing ones should go through a formal change process to avoid loosing control.

  There are tools on the market that can help with defining the roles, but they are not discussed here. When your pilot for roles is done, automation with the help of a life-cycle identity management tool (for example, IBM Tivoli Identity Manager (TIM)) can start.

  You can expand your role-based architecture, business unit by business unit, implementing and automating the roles and business processes into the life-cycle identity management solution.

  Tip: For more information about Tivoli Identity Manager see:

  http://www.ibm.com/software/tivoli/products/identity-mgr

- System infrastructure audit and health checking

  Several tools exist to perform this task. These tools need to be used by persons with in-depth knowledge of the infrastructure and the solutions that are investigated.

  Tip: For more information about governance and compliance, see the following sources:

  - IT Governance Institute:

    http://www.itgi.org/
    http://www.isaca.org

  - IBM Data Governance Council:

    http://www.ibm.com/software/tivoli/governance/servicemanagement/data-governance.html

Part 1

Tools

In this part of the book we discuss the tools available for auditing in a mainframe-centric environment. While the focus of this book is on auditing, auditing consists of the processes and tools used to ensure that a company is compliant with policy. Thus, we have extended the discussion of the tools to include information about how they can define or implement policy and how they can do compliance-based reporting.

The information in this part of the book tie together the high-level view and the specific scenarios. This part provides a technical overview of each of the tools, or products, in a consistent format, so that tools may be compared. Where possible, we have addressed each of the following aspect of the tools:

- Overview
  - Introduction - introduction to the features and functions of the product
  - Architecture - the major components of the product and how do they work together
- Policy, auditing, and compliance aspects
  - Policy definition and enforcement - what the product provides for policy definition and enforcement
  - Audit capture - what the product provides for audit record capture and storage
  - Audit consolidation - what the product provides for audit record consolidation and standardization
  - Audit reporting - what the product provides for reporting, both in terms of pre-canned reports and ad hoc reporting, such as forensic investigation
  - Compliance reporting - what the product provides for compliance checking

This approach allows the tools to be compared to help you build a policy/auditing/compliance solution that may be similar to, but not exactly the same as, our scenarios in Part 2, “Scenarios” on page 145.

We have limited the scope of this part to IBM platform tools (such as RACF and the Linux plug-in) and IBM software products (such as the Tivoli tools). There are many more tools and products than this available on the market that may fill niche auditing requirements.

2

Enterprise-wide auditing tools

This chapter discusses the tools that are intended to be deployed across the entire enterprise. They may generate audit data on multiple platforms, consolidate data from many different types of systems, or provide a single reporting location for all of the enterprise audit data.

2.1 Tivoli Compliance Insight Manager (TCIM)

IBM Tivoli Compliance Insight Manager is a security application that fits in the most common information security models. A number of security standards can be managed and handled with this product, including ISO 17799, Basel II, HIPAA, and Sarbanes-Oxley. Tivoli Compliance Insight Manager helps organizations meet audit and logging requirements. It provides reliable, verifiable log data collection and centralizes security log data from heterogeneous sources.

Tivoli Compliance Insight Manager centralizes, consolidates, and normalizes security log data that originates throughout the enterprise. The Tivoli Compliance Insight Manager, by collecting information from many components, is capable of presenting an horizontal view of the security operations throughout the enterprise. The event log data that is collected and analyzed is compared with established security policies. Policy breaches or reasons for further action are brought to your attention through a variety of alerting and reporting mechanisms.

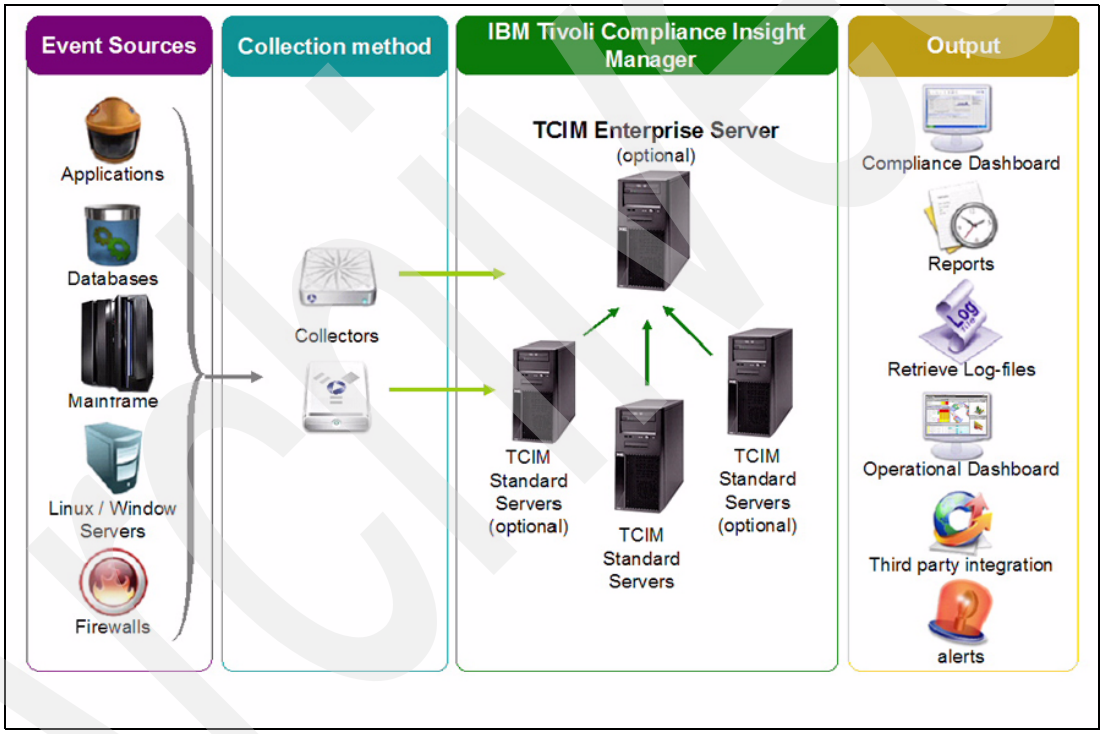


Figure 2-1 The TCIM product overview

Tivoli Compliance Insight Manager uses the Generic Event Model (GEM) and the W7 language to consolidate, normalize, and analyze vast amounts of user and system activity. Tivoli Compliance Insight Manager is able to deliver alerts and reports on who touched what information and how those actions may violate external regulations or internal security policies. By revealing who touched what within the organization and comparing that activity to an established internal policy or external regulation defining appropriate use, security specialists can successfully implement the first layer of defense for information protection, thereby accelerating compliance efforts.

2.1.1 Architecture

The Tivoli Compliance Insight Manager environment includes a number of key components:

- Enterprise server
- Standard server
- Actuators
- Management console
- Webportal (iView)

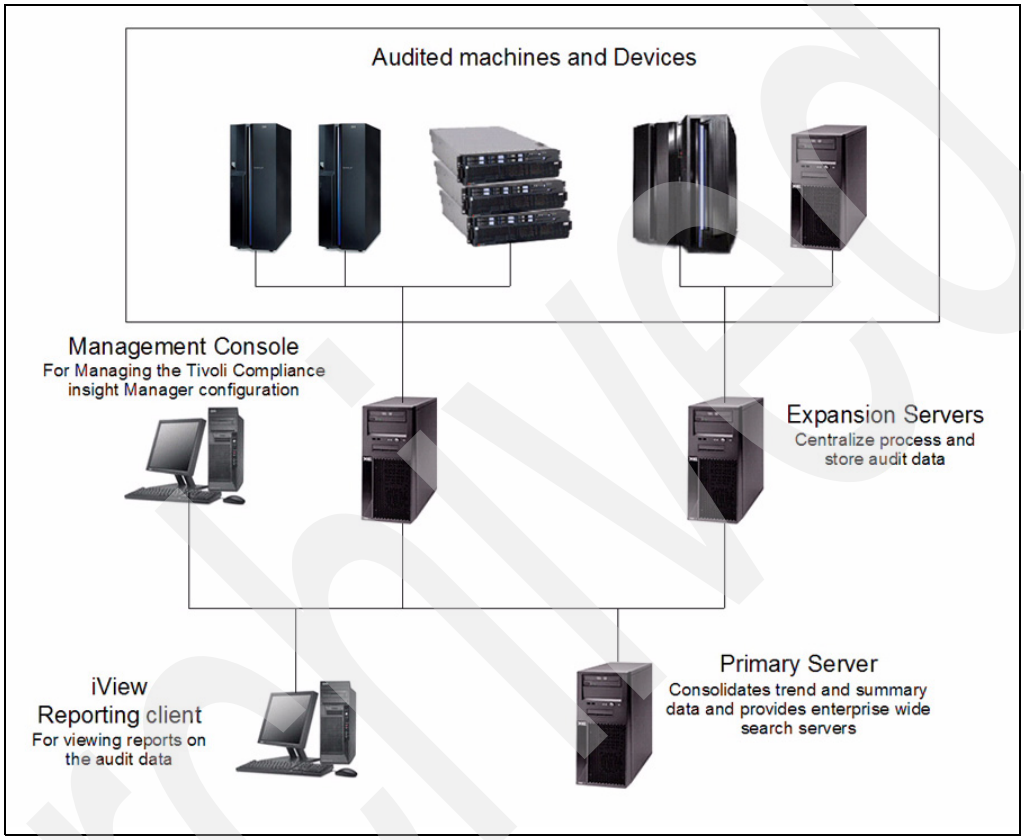


Figure 2-2 TCIM architecture overview

Enterprise server

The Tivoli Compliance Insight Manager Enterprise Server is a Windows®-based server that provides centralized log management and forensic functions, allowing these features to operate across multiple Tivoli Compliance Insight Manager Standard Servers.

Enterprise server offers consolidated log management facilities over all connected Tivoli Compliance Insight Manager Standard Servers. From one enterprise server you can get a consolidated view of log collections and log continuity. The centralized management feature provides a point of access to query and download the original log data collected by standard servers.

The enterprise server also provides the forensic search capabilities. The enterprise server allows you to search the archived logs for evidence without using the GEM and W7 tools. Sometimes you may want to look for the raw traces without going through the report preparation process.

Standard server

Tivoli Compliance Insight Manager uses a centralized Windows-based server, called the *standard server*, as the heart of its security audit and compliance system. The standard server performs the following main functions:

- Collects security logs from the audited event sources
- Archives the logs
- Normalizes the event data and loads it into the reporting databases
- Sends e-mail alerts when a high-severity event is detected
- Creates reports

Note: In our environment we used only a standard server that collected data from the z/OS environment as well as from TAMos and TAM for e-business.

Actuator

Depending on the platform, actuator software is installed on audited systems as a service or daemon. Each actuator consists of an agent and numerous actuator scripts. The agent is responsible for maintaining a secure link with the agents running on the Tivoli Compliance Insight Manager Server and other audited systems. The actuator scripts are invoked by the agent (at the request of the Tivoli Compliance Insight Manager Server) to collect the log for a particular event source. There is a different script for every supported event type.

Each system on which a Tivoli Compliance Insight Manager actuator is running is a *point of presence*. This designation includes the server systems and all systems that are running the actuator. The point of presence is capable of collecting data from a system and the systems that are connected to it.

The management console

The management console is responsible for configuration and management of the enterprise server and the standard servers. The management console can operate locally or in a distributed manner. All that is required for remote operation apart from the management console itself is a local point of presence to which it can communicate.

You can use the management console to perform numerous tasks related to the configuration and management of the Tivoli Compliance Insight Manager servers:

- Activate the agents and have them collect audit trails from different platforms.
- Define the security policy and attention rules.
- Define users and their access rights.
- Start the preparations of the reports.

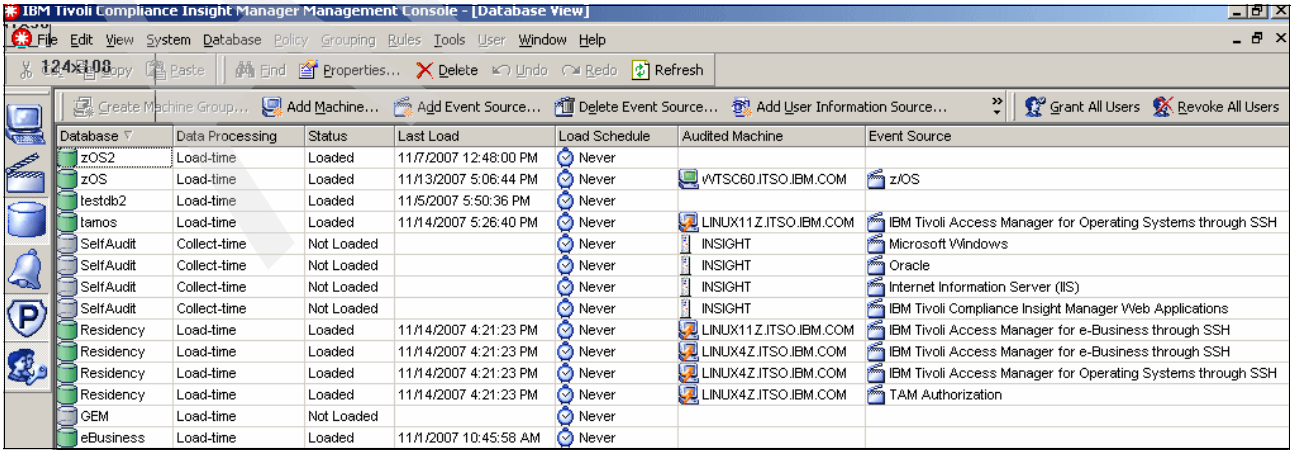


Figure 2-3 TCIM management console

The management console is the user interface for the Tivoli Compliance Insight Manager server. After the reports have been prepared by the server, a Tivoli Compliance Insight Manager user may generate the specific reports using the iView component.

iView

The events found in the logs are normalized and stored in databases. The data in the databases is available for further investigation through the Web-based tool called iView. iView is a reporting application that Tivoli Compliance Insight Manager administrators can use to generate specific reports on compliance level and policy violations. It uses an HTTP server, authorizing users to view reports through their Web browsers.

On the iView, you can use a dashboard to see a summary of your security events, and investigate events by using predefine reports or customized report. The iView is capable of showing you events base on the W7 fields (for example, you can check who was doing what on which resource and at what time).

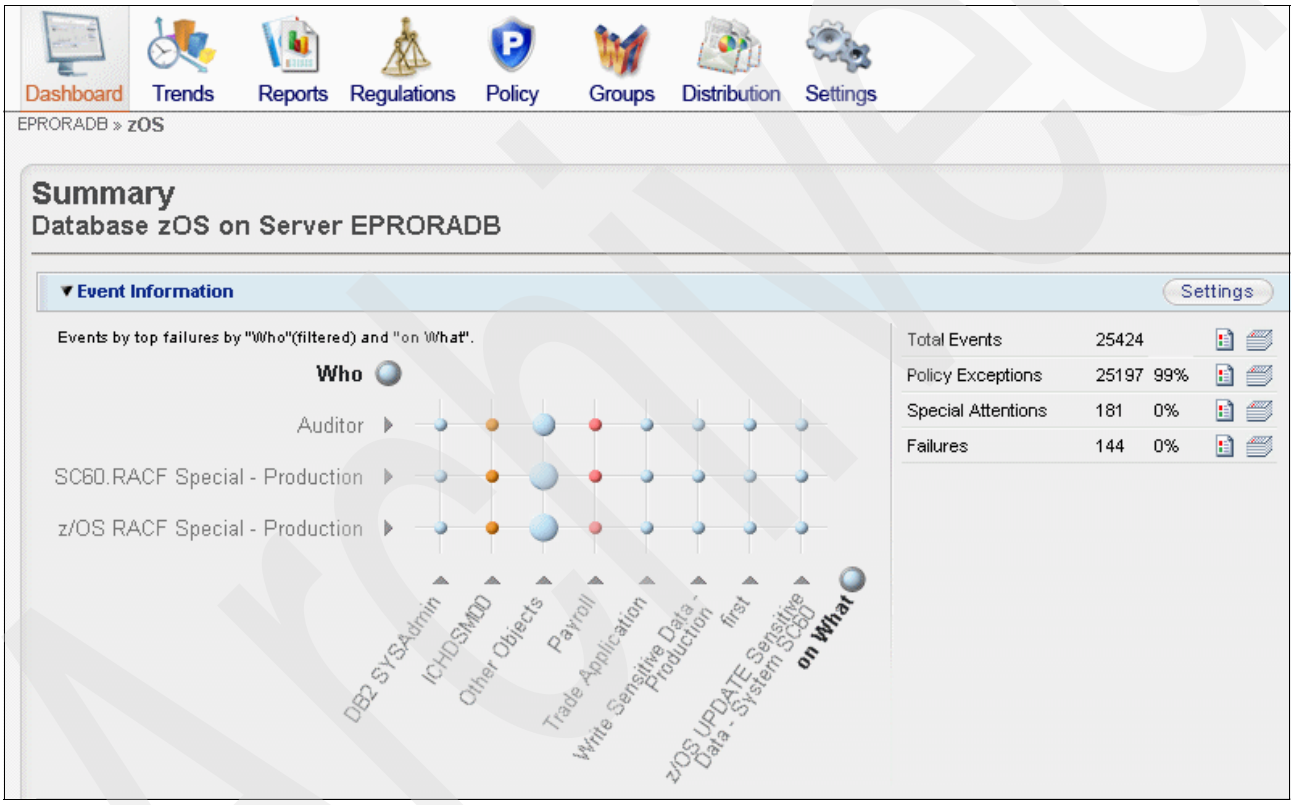


Figure 2-4 iView - part of the dashboard view

2.1.2 Defining and enforcing policy

You can determine whether an event deserves special treatment by comparing its classifications in the W7 groups and a set of user-defined rules. Policy rules and attention rules represent two different ways of evaluating events:

Policy rules Rules that describe allowed behavior. A well-defined policy includes policy rules that completely and exactly cover all allowed behavior.

Attention rules Rules that identify events that deserve special investigating attention.

By applying these two rule methods, the mapping process determines whether an event complies with the policy rules. If not, the event is marked as a policy exception. An event,

whether or not it is an exception, may also be a special attention event. Depending on grouping and policy evaluation, an event receives a significance number, indicating the amount of attention it deserves. Significances range from 1 to 99, with 1–9 for ordinary events and 10–99 for policy exceptions and special attentions.

A security policy consists of group definition sets, policy rules, and attention rules defined for one or more platforms. When systems whose activity is audited are registered, Tivoli Compliance Insight Manager applies the policy and attention rules in your security policy to load audit data from each system into a GEM database, organizing the data using the groups you defined, and displaying the results in iView.

You can define a policy, create a new empty policy, duplicate or edit a policy, or delete or rename a policy.

When creating a policy, you must specify the following information:

- Group definitions for each platform
- Attention rules
- Platforms to be audited
- Policy rules

Groups

To create an effective security policy, you must define groups. Group definition sets are created to organize audit data into standardized groups for efficient analysis. You can create a group definition set for an entire policy or create a set for each platform to be audited. You can define who, what, where, when, and onwhat groups. In each group you can assign resources from your environment. For example, all users who perform administration tasks should be assigned to the administrator's group. Under the who and OnWhat groups you can define customer data (data files), and on when groups you can define business hours and non-business hours.

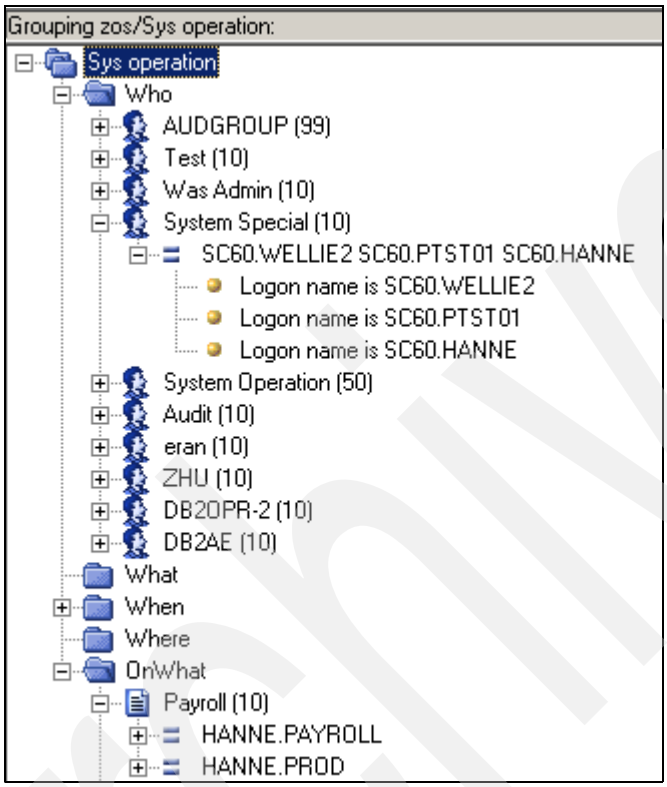


Figure 2-5 An example of the grouping that we used for our environment

When defining new groups you can use advanced definitions to define more complicated and more narrow groups. When defining a new group you have to add at least one condition. Inside the condition you need to define at least one requirement. If you add more than one requirement in the same condition, TCIM will read it as: `First requirement AND the Second requirement`. If you add more than one condition, TCIM will get it as `First condition OR the Second condition`. See Figure 2-6.

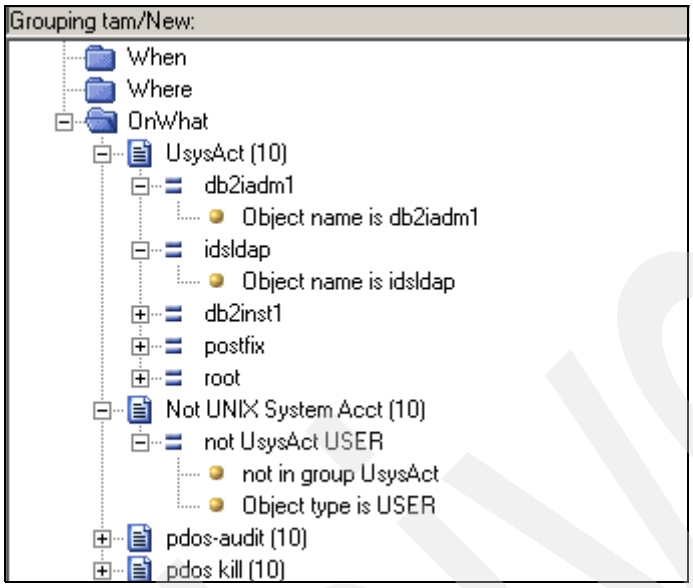


Figure 2-6 Using OR and AND to define complex groups

In the above example we had to define a group that will include all the users, except of a list of users, who an action was made on them. We did that in the following steps:

1. We define the groups in the onWhat section because we are looking for the users for which an action was made.
2. We defined the group called UsysAct. In this group we added five conditions, each of which reflects a different object and all of which are grouped in an OR statement. In this case we get information about the objects db2iadm1, idsldap, db2inst, and so on.
3. We defined a second group called Not UNIX System Acct. In this group we defined one condition called not UsysAct USER, which has two requirements:
   - The first object is not in group UsysAct (not the objects that are defined in group UsysAct).
   - The second requirement is that the object type is USER.

   TCIM will take the two requirements and join them together into one statement. All the users, except the one defined in the Usys Act group, will be contained in the statement.

Tivoli Compliance Insight Manager provides you with out-of-the box default grouping files that provide solid ideas for the what and when groups. You can also use the grouping wizard, which can help you define groups based on log information, which is especially useful to create who and OnWhat groups. You can start the wizard with a preloaded grouping file, or you can create a completely new grouping file with the wizard. The grouping wizard is also a very powerful tool to analyze your security information because it shows the who, OnWhat, when, and where information for each type of operation. You can use the wizard to determine which administrators added a user, or which systems had audit policy modifications.

Attention rules

Attention rules are used to highlight instances of events that are critical to the organization. One typical application for these rules is to monitor change management activities even if the events are allowed by your policy rules. Actions that match an attention rule generate actions. For example, by looking for a specific instance of a data attribute in any of the W7 dimensions for certain events, you can set an alert to notify someone of a change to a server's configuration.

In our case we used the attention rules to get information about actions that are made on the payroll file outside of business hours. The payroll file definition was made in the onWhat group and the non-business hours were defined in a when group. (The group was built by five conditions that were joined into one OR statement.)

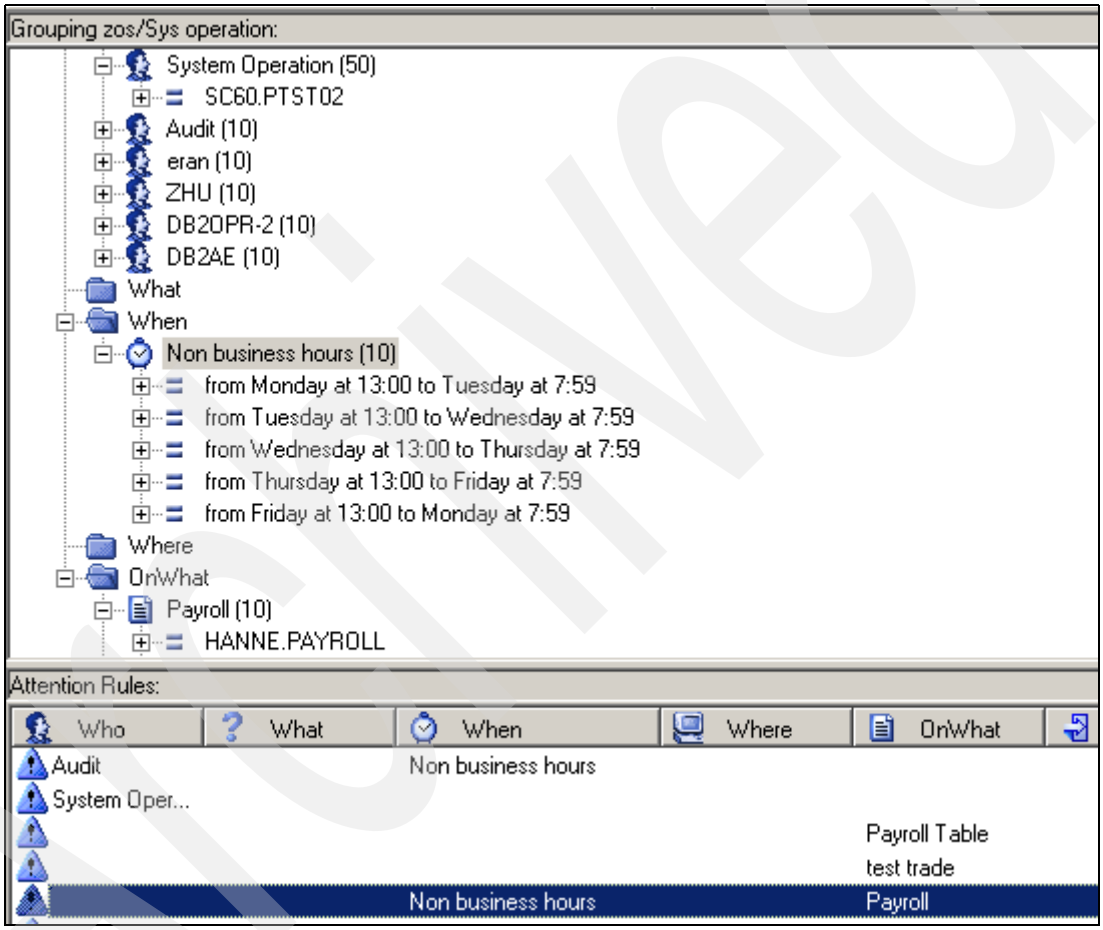


Figure 2-7 An attention rule that collect information about an action that was made during non-business hours and on the payroll file

Policy rules

Policy rules are used to monitor the way that information and processes are being used within an organization. That is, they specify which actions can be performed by which people on which systems at what times. Actions that do not match a policy rule generate policy exceptions. Policy rules have an associated priority that can be set to enable differentiation so that policy violations and other exceptions can be processed according to their severity or importance. This allows security administrators and auditors to focus on addressing those events that have the most significant impact on the business.

By refining policy rules, you can ensure that existing policies are effective and can even establish new policies that reflect the actual behavior of users, as opposed to theoretical activities contained in policy manuals and non-automated tracking systems.

Automatically applying the policy rules makes it easy to quickly determine whether each monitored action complies with policy.

2.1.3 Audit logs capture and storage

A *collect* is the process of centralizing event data by retrieving it from the audited systems and applications and archiving it in a central place on the server, called the depot. The most common mechanism for retrieving security log data is through a process called *batch collect*.

A collect is being preformed by the actuator and can be preformed on the same machine on which the actuator is installed or on a remote server that the actuator has permission to collect the data from. This is called a *remote collect*.

A security log is created on the audited system by the application that is being audited. Such logs contain records of many events, which are all processed as a batch. The Tivoli Compliance Insight Manager system retrieves and centralizes this batch as a whole.

After receiving the security logs, the server archives the security logs in the depot. Each operating system or application that is configured for monitoring is called an *event source*. In our case the mainframe logs, which are generated from the SMF data, are collected by the actuator for z/OS, which is installed on the z/OS system. This log is a z/OS event source. Each event source has an associated schedule that determines how often event data is collected. Typically, the collect schedule is set to occur once each hour.

When log data is centralized in the depot, it can be processed and analyzed. To make the data accessible, it is translated (or normalized) into an easy-to-understand data model called the Generic Event Model (GEM) and is stored in a relational GEM database. These two steps are called mapping and loading. Because mapping precedes and serves loading, the combination of the two is also called *load*.

Security log data consists of records. Each record usually describes one event that happened on the audited system. Before loading the event into the GEM database, you need to classify the events according to their W7 attributes. Because log formats are very different for different event sources, logic that is specific to the event source does this translation.

Data and statistics, spanning a longer period, are maintained by a process called aggregation. The aggregation process builds a special database called the aggregation database from which trends and summaries can be extracted.

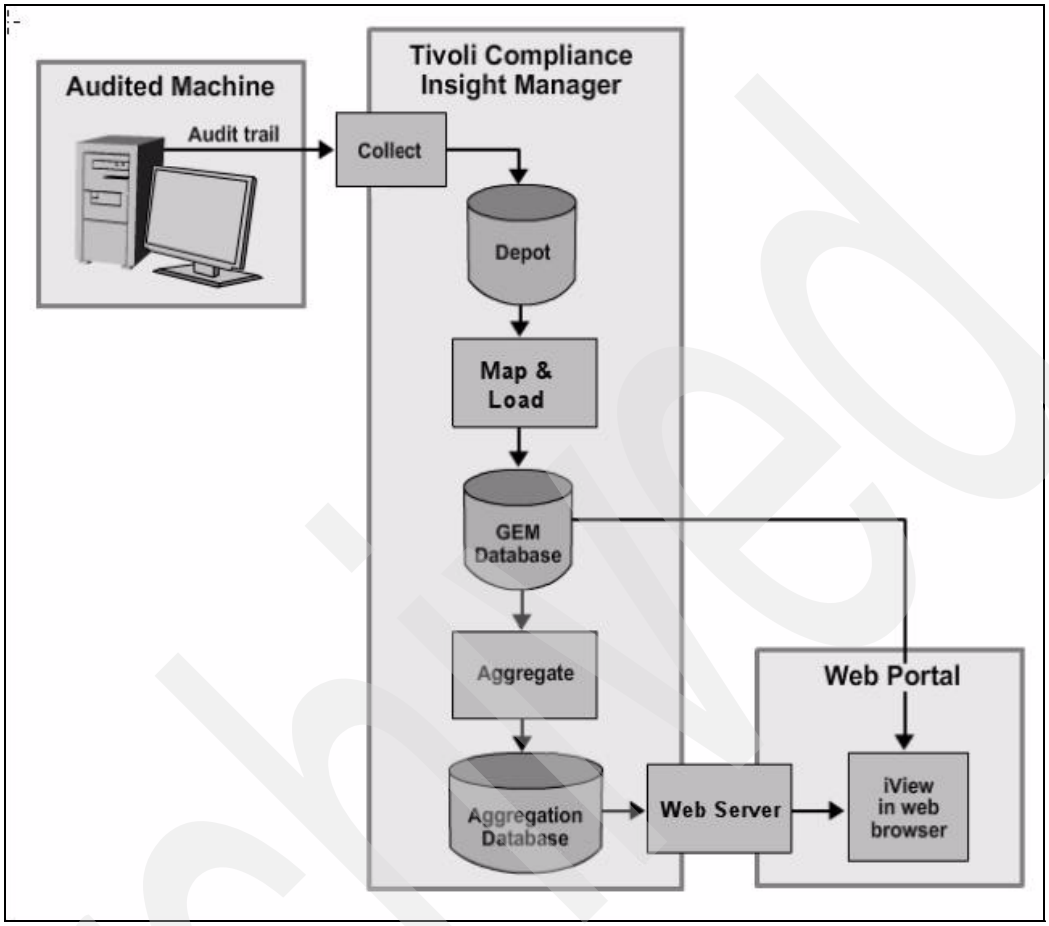


Figure 2-8 TCIM Logical workflow

2.1.4 Audit records consolidation and standardization

All the data that is being collected by the TCIM is stored in the server in databases. These databases store the audit data from security logs and other sources of event information. In the flow from collection to archive, audit data is indexed and normalized to facilitate analysis, forensics, information retrieval, reporting, and so on.

- Depot

  Logs that are being collected are stored in the depot directory. The data is compressed similar to a file system based log repository. You can download a log from the depot using the log manager on the TCIM portal and choosing the investigate option to look for the right log.

- Reporting database

  Data that has been mapped into the W7 format is stored in an instance of an embedded database. The database is periodically emptied and then filled with more recent data. Typically, this refresh cycle is done on a daily scheduled basis, meaning that data from the previous period is present and available for analysis and reporting. Data from a depot can be mapped and manually loaded into the reporting database for processing.

- Aggregation database

  The aggregation process takes a large number of individual events and duplicates them into a more manageable set of information. Aggregation is performed as part of the normal scheduled load processing. After a successful scheduled load, aggregation is performed for each reporting database. Aggregation vastly reduces the amount of event information that needs to be online, and allows users to have an organization view of security using iView on the TCIM portal. Additionally, these aggregated statistics are used for providing long-term trending information and are typically held for several years (dictated by local or statutory requirements). This is highly valuable data and provides a historical database of an organization performance against defined security policies and regulations.

- Consolidation database

  The consolidation database consolidates all the aggregation databases in a TCIM cluster. This provides an overall view of all servers in the cluster for trending and statistical purposes.

2.1.5 Reporting and compliance checks

TCIM has many ways to show and report data (such as by using TCIM portal options and by sending alerts when a high severity event is detected). The TCIM portal has the main panel shown in Figure 2-9.

Figure 2-9 IBM TCIM portal

This menu has two sections:

- The IBM Tivoli Compliance Insight Manager, which contains links to the principal components of the TCIM. These are:
  - iView - A reporting application that Tivoli Compliance Insight Manager administrators can use to generate specific reports on compliance level and policy violations. The iView has sub-options that help you see and analyze the data. iView is discussed in more detail in “iView” on page 25.
  - Log Manager - This option allows you to check the status of the load and collect procedures that were done by TCIM. You also have the ability to download the data that was collected by the TCIM actuators and investigate (the files are allocated in the depot directory under server).
  - Policy Generator - To define new policies based on the data that has been collected.
  - Scoping - This section allows you to scope the viewing rights of reports to protect user privacy.
- The section IBM Tivoli Compliance Insight Manager Management Modules - This contains links to the management modules, tailored to help you meet SOX, GLBA, HIPAA, and ISO 17799 requirements. These are not discussed in detail in this book.

2.1.6 iView

The iView allows you to investigate and drill down into events that have been collected by the TCIM. The data that is being collected by TCIM can be presented as a report or as filtered data on the panel. Both ways can be saved in one of the following options:

- PDF
- CSV
- XLS
- HTML

On most of these panels and the reports you can filter the information using W7 attributes. Figure 2-10 shows the filter setting panel. You can use a wild card (* or ?) when you do not know the exact phrase that you are looking for.

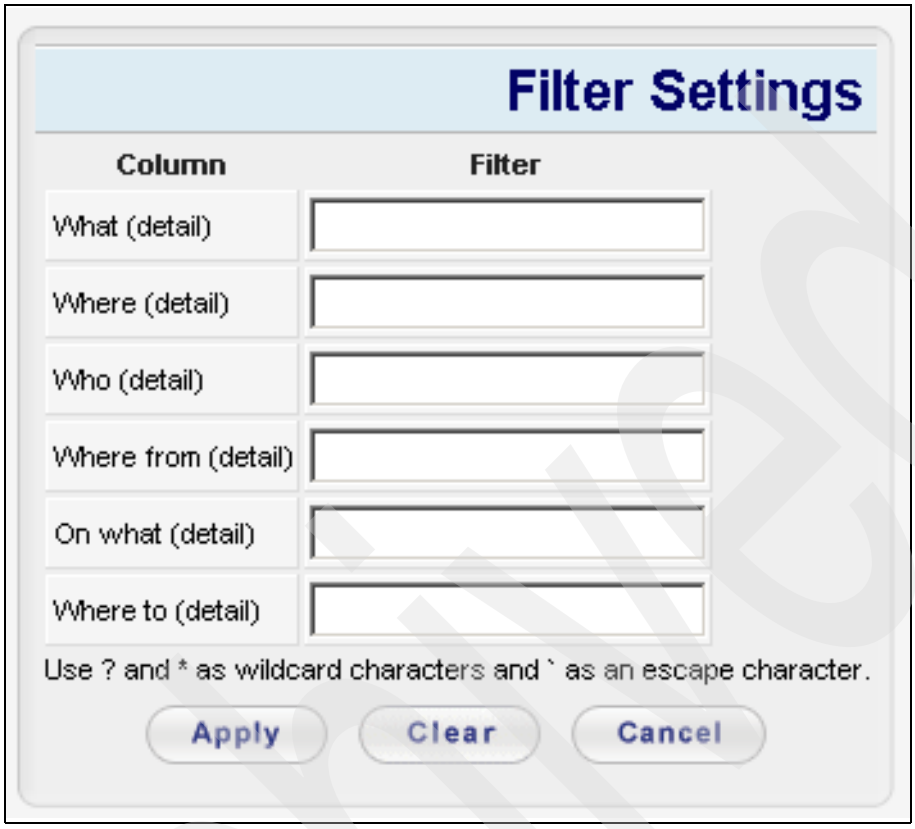


Figure 2-10 Filter setting

In order to initiate the filter setting, click the small cube (circled in Figure 2-11).

Figure 2-11 Filter setting

When using a filter view, the small cube is shown in red, as in Figure 2-12.

Figure 2-12 Using a filter symbol

Figure 2-13 shows the iView main options followed by a brief description of each of the options.

Figure 2-13 iView main options

The main menu of the iView is the compliance dashboard, which allows you to have an high-level view of the events that were reported to each database that you have created. If you drill down into one of the databases, you can retrieve the following information:

- Total events - the option to see all the events that were reported to the TCIM. You can see a list of all the reports or get a W7 summary.
- Policy Exception - This option lets you see a list of all the policy exception events that were reported to TCIM using the policy that you have configured, or a summary of the policy exception.
- Special Attention - A list of all the special attention events that were generated by the policy definition. As with the above two options, you can see a list of the events or a summary.
- Failure - A list of all the failure events that happened in the audited machine.

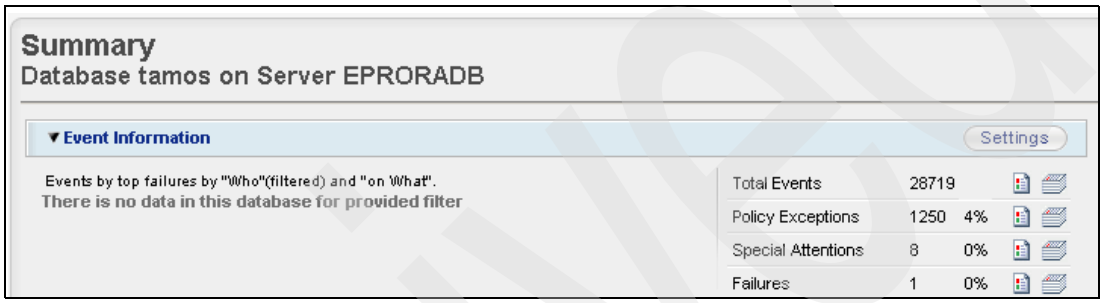


Figure 2-14 iView - database summary menu

- Trends - This option allows you to see a graphical view of the data grouped by the main options that were discussed in the dashboard section. The data can be viewed as yearly hourly.

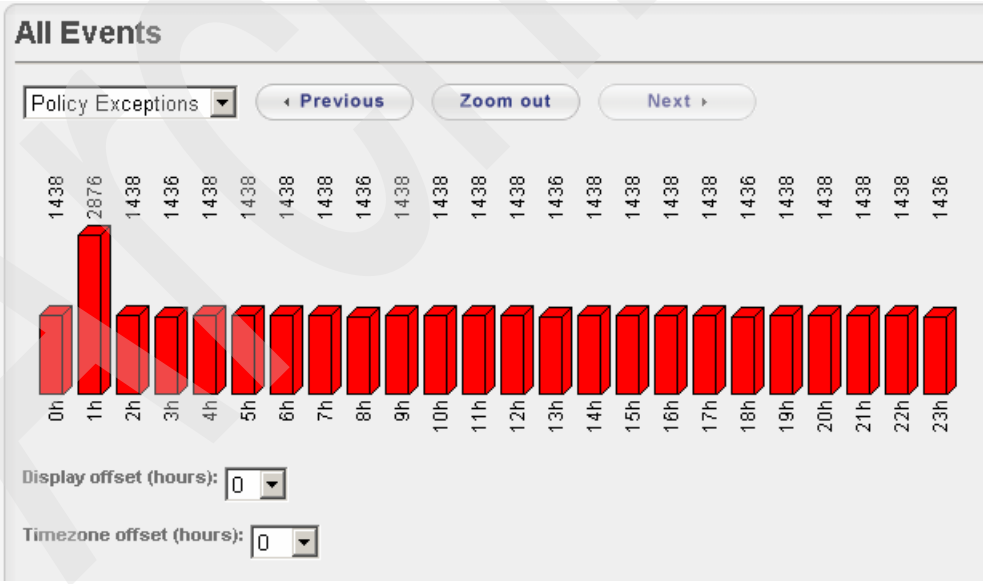


Figure 2-15 iView - trend menu - hourly view

- Reports - The report view allows you to view the data in the database by predefined reports or self-customized reports. You have the ability to define a report according to almost every field of data that the TCIM has.

  Figure 2-16 is an example of a customized report, enabling us to see the z/OS SMF. Figure 2-17 on page 29 shows the result. The symbol A in Figure 2-16 represents the field where you can add information, and the symbol B is the added value.

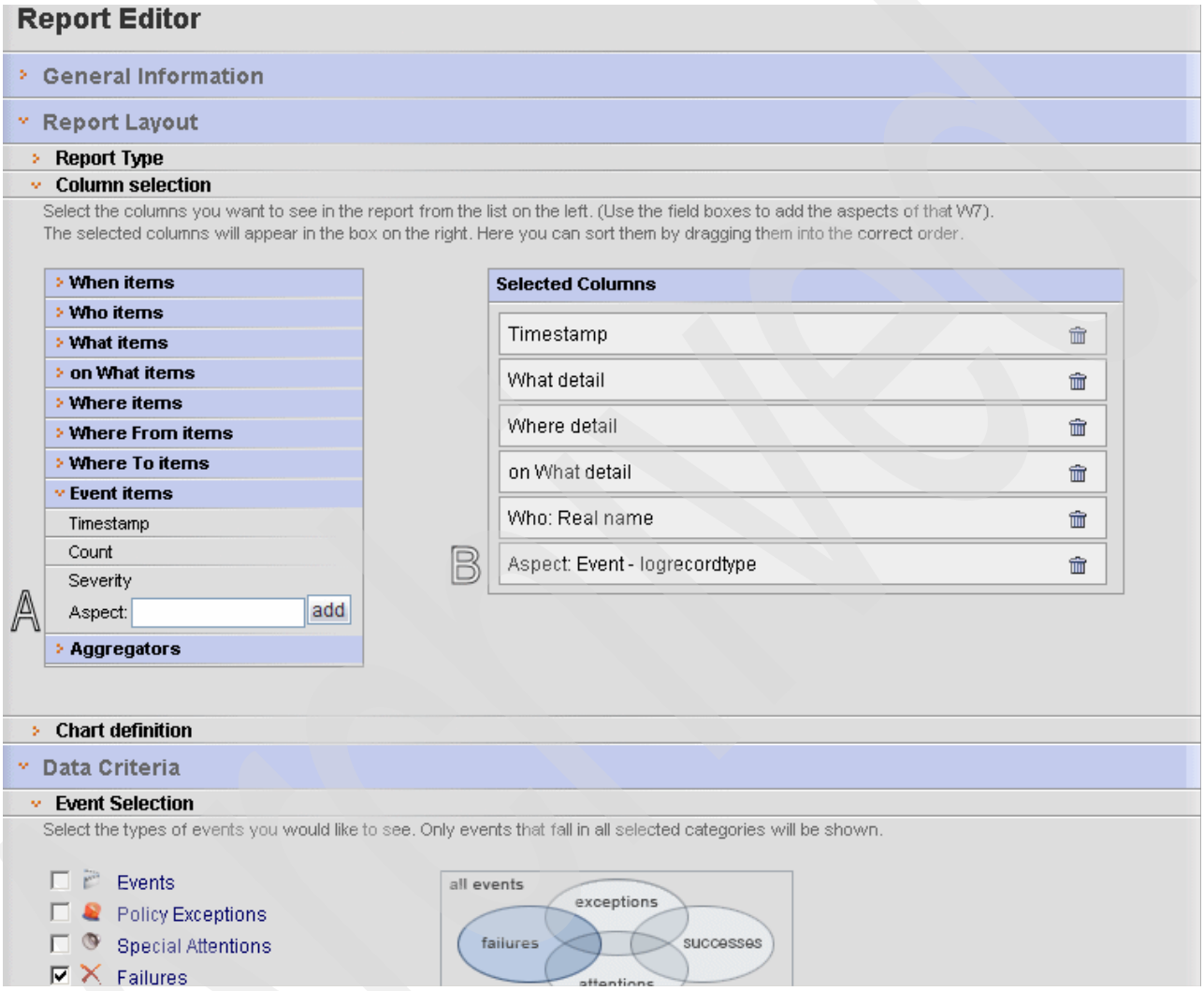


Figure 2-16 iView report editor

Figure 2-17 represents the report generated according to those input fields.

SMF type
Database zOS on Server EPRORADB

Setup:

	Month	Day	Year	Hour	Min.
Start time	November	2	2007	23	50
End time	November	17	2007	14	20

Execute Reset

Time zone: Event time zone

What (detail)	Where (detail)	On what (detail)	Who (real name)	logrecordty (event aspect)
Open : Directory / Failure	SC60 (z/OS)	DIRECTORY : /WebSpherePA1/V6R1/AppServer/properties/service/product/WebSphere/backup / tmp	DAVID EDWARDS/BARI	SMF,80-29
Open : Directory / Failure	SC60 (z/OS)	DIRECTORY : /WebSpherePA1/V6R1/AppServer/properties/service/product/WebSphere/backup / tmp	DAVID EDWARDS/BARI	
Open : Directory / Failure	SC60 (z/OS)	DIRECTORY : /WebSpherePA1/V6R1/AppServer/profiles/default/installedApps/cl6601 / AClockWebClientEAR.ear	DAVID EDWARDS/BARI	SMF,80-29
Open : Directory / Failure	SC60 (z/OS)	DIRECTORY : /WebSpherePA1/V6R1/AppServer/profiles/default/installedApps/cl6601 / AClockWebClientEAR.ear	DAVID EDWARDS/BARI	
Open : Directory / Failure	SC60 (z/OS)	DIRECTORY : /WebSpherePA1/V6R1/AppServer/profiles/default/installedApps/cl6601 / AClockWebEAR.ear	DAVID EDWARDS/BARI	SMF,80-29
Open : Directory / Failure	SC60 (z/OS)	DIRECTORY : /WebSpherePA1/V6R1/AppServer/profiles/default/installedApps/cl6601 / AClockWebEAR.ear	DAVID EDWARDS/BARI	

Figure 2-17 The SMF type report

2.2 Tivoli Directory Server (TDS)

This section introduces IBM Tivoli Directory Server and discusses how it fits into the audit architecture.

The Lightweight Directory Access Protocol (LDAP) is a directory access specification. It dictates message structures and interfaces to access directory services over TCP/IP. When we say that a directory is an LDAP directory, we generally mean that a directory is LDAP compliant. IBM has a number of LDAP-compliant directories, such as IBM Tivoli Directory Server and Domino®.

A directory is a data store that is generally read-optimized, meaning that it is much better at performing read operations than write operations. This is different from traditional data stores such as relational databases. This characteristic makes a directory ideally suited to a central store of data that many people will read, such as a phone directory.

A great introduction to directories can be found in Part 1, "Directories and LDAP," of *Understanding LDAP - Design and Implementation*, SG24-4986.

Many IBM software products can use LDAP-compliant directories as identity stores, such as IBM Tivoli Access Manager, IBM Tivoli Identity Manager, IBM WebSphere Application Server, and IBM WebSphere Portal Server.

As the focus of this book is on auditing in a mainframe-centric environment, we focus our discussion on the three IBM LDAP-compliant directories that run on z/OS and Linux on

System z, the IBM Tivoli Directory Server (ITDS), the z/OS Integrated Security Services (ISS) LDAP, and the IBM Tivoli Directory Server for z/OS.

The distributed ITDS runs on many platforms including Linux on System z, x86 Linux, and various flavors of UNIX and Windows. It provides standard directory features and is implemented using DB2 Universal Database™ (UDB) as the back-end data store.

On z/OS we are seeing a migration from the older z/OS ISS LDAP to the ITDS for z/OS. ISS LDAP is the heritage directory on z/OS and has been available since the later OS/390 releases. It has not changed since z/OS v1r6. ITDS for z/OS is the successor and is being functionally enhanced. Both ISS LDAP and ITDS for z/OS are shipped with the v1r9 release of z/OS.

Both z/OS products provide a variety of back-end data stores (see the following section). This includes RACF, which means distributed applications have access to RACF data via an LDAP interface. The directories also support native authentication, where a person object in the directory can be linked to a RACF identity and users can authenticate to LDAP with their RACF password. A good example of z/OS LDAP being used as a central authentication service with native authentication can be found in *Linux on IBM zSeries and S/390: Securing Linux for zSeries with a Central z/OS LDAP Server (RACF)*, REDP-0221.

The ITDS on z/OS introduced a number of new features for use by other products such as an Extended Operation (EXOP) interface for writing audit records into SMF using a LDAP client.

All three directories support high-availability features such as directory replication. They support a changelog mechanism where data changes are logged and can trigger other activity, such as password cascading to other repositories.

From the perspective of auditing, directories are often used as a central authentication service, so we should be able to audit authentication attempts.

2.2.1 Architecture of distributed TDS

Figure 2-18 shows the components of the distributed ITDS.

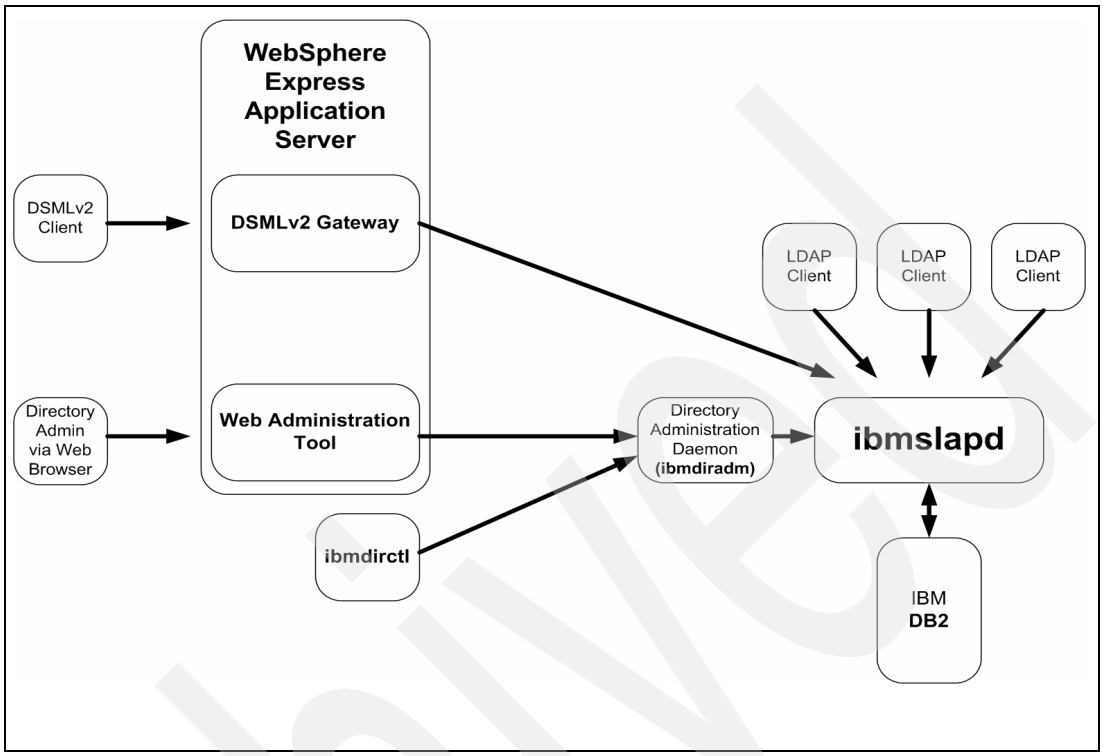


Figure 2-18 TDS distributed components

The central process of the directory server is the ibmslapd process. It communicates with the DB2 database holding the directory data. Any LDAP v3-compliant client can access the directory. This includes the clients shipped with the ITDS. There is also an administrative deamon, ibmdiradm, that processes directory administration calls from the command-line interface, ibmdirctl, and the Web Administration Tool (hosted on WebSphere Application Server). The directory also supports the Directory Services Markup Language (DSML) v2 protocol via a DSML v2 gateway.

2.2.2 Architecture of z/OS ISS LDAP

Figure 2-19 shows the components of the z/OS ISS LDAP.

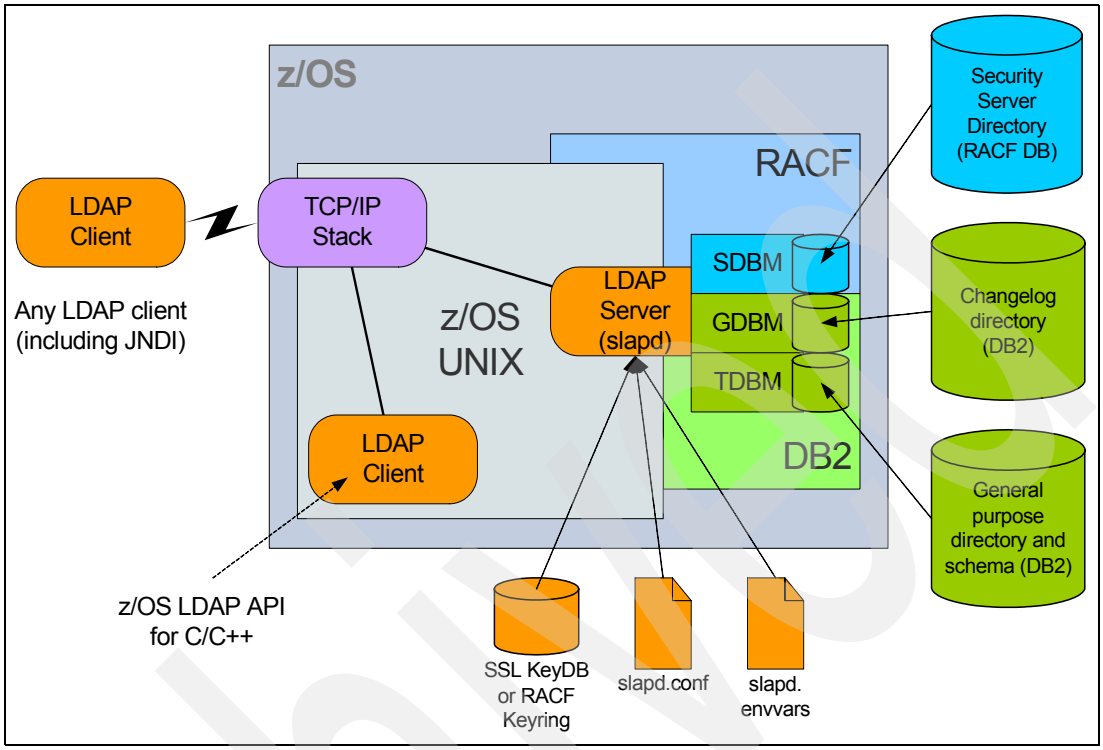


Figure 2-19 z/OS ISS LDAP components

Like its distributed counterpart, the z/OS ISS LDAP has a central server, called slapd, that clients connect to over TCP/IP. The server runs as a started task in z/OS but uses z/OS UNIX functions such as the TCP/IP stack. It ships a client that runs in z/OS UNIX, which supports C/C++ API calls.

Like the distributed ITDS, it can use DB2 as the back-end data store for its data and changelog. Unlike the distributed ITDS, it supports RACF as a back-end data store. This means that the directory data, such as user details, could be stored in DB2, RACF, or spread across both.

2.2.3 Architecture of ITDS for z/OS

Figure 2-20 shows the new ITDS for z/OS components.

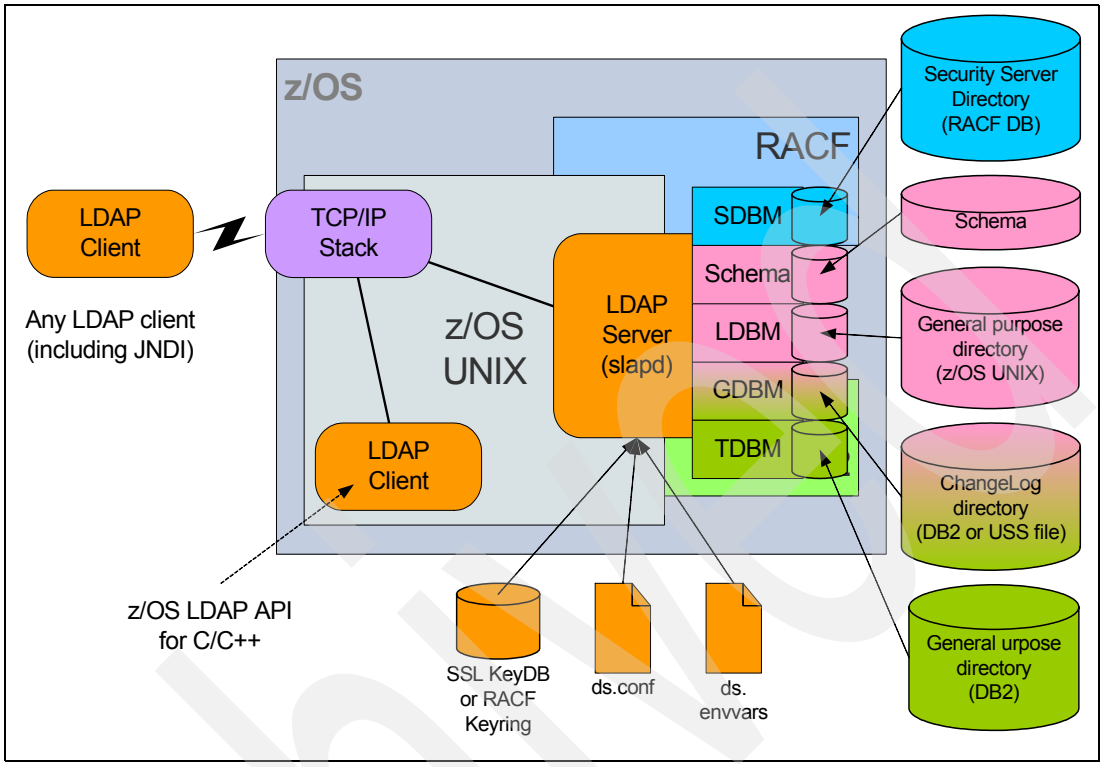


Figure 2-20 ITDS for z/OS components

The ITDS for z/OS is similar to the ISS LDAP, with some key changes. With ISS LDAP the directory schema was local to each of the data stores. In ITDS for z/OS a single directory schema is used and held in a z/OS UNIX file. This single schema concept brings the z/OS ITDS into line with the distributed one.

The other key change is the addition of the z/OS UNIX files as a backed data store. In ITDS for z/OS the directory data could be held in RACF, a DB2 database, a z/OS UNIX file, or a combination of these. The changelog for ITDS on z/OS can be held in DB2 or a z/OS UNIX file.

The next section looks at how the three directories support auditing.

2.2.4 Policy, auditing, and compliance aspects of TDS

This section discusses how TDS fits into a broad audit solution.

Policy definition and implementation

These directories may be used as a central authentication service. While they could be configured to support authorization data, none of the directories are configured to act as an authorization service. The only policy that can be implemented with these directories is password strength, which is normally set at a global level.

Audit capture

When talking about these three directories, there are two types of activities that would be of interest for auditing. First are the administrative changes made to the data, such as who has changed the data schema, and who has added, modified, and deleted entries. This is relevant irrespective of the user of the directory. The other type of activity is the authentication attempts when the directory is acting as an authentication service. These *binds*, connections to the LDAP server, and *unbinds* may be a critical link in the chain of activity for a user in a system.

The distributed ITDS provides audit logging as part of its logging utilities (see Chapter 15, "Logging Utilities," in the *IBM Tivoli Directory Server Administration Guide*, SC32-1674). Auditing is set or modified using either the Web Administration Tool or by directly modifying the LDAP entries using an **ldapmodify** command. There are two audit logs generated, if configured, by the product:

- The adminaudit.log contains audited activity from the ITDS administration deamon (Figure 2-18 on page 31).
- The audit.log contains audited activity from the ITDS server, such as the BIND and UNBIND operations.

Example 2-1 shows some records in the distributed ITDS audit.log file.

Example 2-1 Distributed ITDS audit.log entries

```
AuditV3--2006-05-11-11:59:50.763+10:00--V3 unauthenticated Bind--bindDN:
cn=root--client: 192.168.254.90:23424--connectionID: 2--received:
2006-05-11-11:59:50.762+10:00--Inappropriate authentication
name: cn=root
authenticationChoice: simple
AuditV3--2006-05-11-12:00:07.986+10:00--V3 anonymous Modify--bindDN:
<*CN=NULLDN*>--client: 192.168.254.90:23680--connectionID: 3--received:
2006-05-11-12:00:07.935+10:00--Insufficient access
controlType: 1.3.6.1.4.1.42.2.27.8.5.1
criticality: false
object: uid=3317816,cn=users,dc=racv
replace: uid
replace: userPassword
replace: objectclass
replace: sn
replace: cn
```

There are two activities shown here: a bind and a modify.

There is no auditing feature with the z/OS ISS LDAP directory.

The ITDS for z/OS has audit logging that can be enabled. When configured it will write SMF Type 83 Subtype 3 records. The level of auditing enabled is controlled by the parameter file settings, arguments passed at startup time, or through runtime changes. As with the distributed ITDS, auditing can be enabled for different operations, such as bind and unbind, and set for all, error, or none.

Audit consolidation

There is no mechanism to consolidate the distributed ITDS audit log records.

As the ITDS for z/OS audit records are written to SMF, they are considered already consolidated. In order to manage the SMF records with TCIM, you would need to create a

manual feeding mechanism since the current version of the TCIM z/OS Enablers does not manage SMF Type 83 records.

Audit reporting

There is no tool for reporting against the distributed ITDS audit log records.

For audit reporting against the ITDS for z/OS audit records in SMF you could use zSecure Audit, although you would need to develop custom CARLa scripts to format the data, as zSecure is not currently instrumented for the SMF Type 83 records.

Compliance reporting

There is currently no compliance reporting supplied that can be run against the distributed or z/OS ITDS audit records. You could build compliance reports for the z/OS ITDS data in SMF with the zSecure products. If you could consolidate the data to TCIM, you could build compliance reporting in TCIM.

2.3 Tivoli Directory Integrator (TDI)

This section provides an overview of TDI, covering its features and high-level architecture, and discusses how it fits into the audit picture.

TDI enables you to integrate data from different repositories in an easy and flexible way and can also be used to provide a *metadirectory* or *virtual directory* services. Do not be deceived by the word *directory* in its name. TDI enables integration of data from different formats and from different types of repositories, not only from directories.

TDI is a Java-based framework for the development of data flows to join data sources. It uses an assembly-line analogy, where a piece of data is picked from one source and goes through a set of predefined steps and is placed on another source, possibly in a modified form. Data may be pulled from multiple sources, it may be combined or enhanced from different sources, and it may be written to different data repositories.

The logical components of TDI include:

- A single data flow called an *AssemblyLine*. AssemblyLines may call other AssemblyLines to provide distributed functionality.
- AssemblyLines use *connectors* to connect to a data source and may have *parsers* to parse the data as it flows from or to a data source. For example, you may use a file system connector to write data to some text files, but one file must be in XML format, whereas another may need to write as comma-separated. Parsers are used to convert the canonical TDI data format into the XML and CSV format required.
- Business logic within an AssemblyLine may include:
  - Logic coded in JavaScript™ in *hooks*
  - Flow *components*, such as branch, loop, and case components
  - TDI *servers* or *EventHandlers* for specific event-driven functionality, such as HTTP servers
  - *Libraries* of commonly used components, such as script libraries and connectors
  - *Property stores*, such as external property files, to externalize parameters from the data flow
  - *SystemStores*, such as the Cloudscape® database, for holding and managing data

A logical grouping of these logical components is often referred to as a *TDI config*, and is held within a single (config) file.

TDI ships with almost 50 connectors, 12 EventHandlers, 12 parsers, and numerous other components, meaning that most data flow requirements can be met using the supplied components. Where there are requirements for additional functions, the product can be extended. There is an extensive user community that actively shares custom components and sample AssemblyLines.

Architecture of TDI

The physical implementation of TDI is very straightforward. The physical components are shown in Figure 2-21.

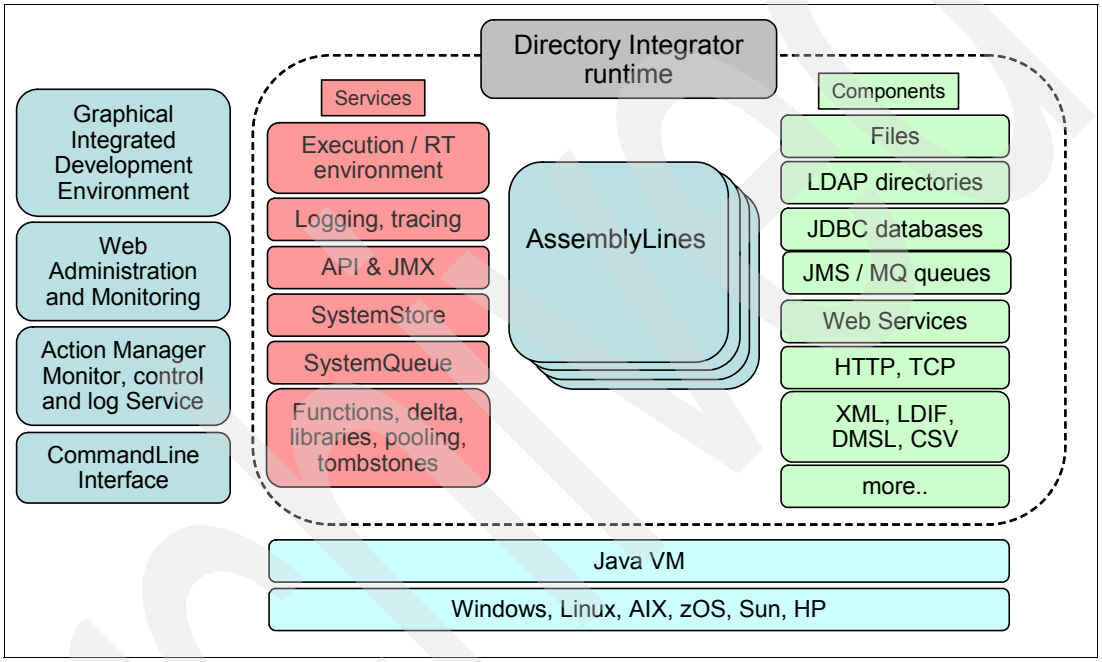


Figure 2-21 TDI components

Central to TDI is the runtime component, or server. This ibmdisrv process runs a single TDI configuration that consists of one or more of the logical components discussed in 2.2.3, “Architecture of ITDS for z/OS” on page 33, such as AssemblyLines and EventHandlers. These are shown within the dotted line in Figure 2-21. The TDI runtime is a Java process.

The other physical components that may be used in a TDI deployment are:

- The *Graphical Integrated Development Environment*, or Configuration Editor (CE), is started by the **ibmditk** command. This Java-based UI is used for developing and testing the AssemblyLines, EventHandlers, and other components of data flows.
- The *Web Administration and Monitoring* console (AMC) is an operations console to remotely start, stop, and manage TDI Configs and AssemblyLines.
- The *Action Manager* (AM) is a standalone Java application that allows you to monitor multiple TDI Configs and AssemblyLine execution.
- The *command-line interface* (CLI) to TDI, called the tdisrvctl utility, is designed for remotely managing Configs, AssemblyLines, and so on.

Depending on the data being accessed, the TDI components may be installed locally or remotely. For example, say that you needed to use TDI to centralize audit log files to a central

server. If the data sources support remote access (such as SSH to access an OS file or a database that supports remote access) you could deploy TDI to the central system and pull the data in. Otherwise, you may need to deploy TDI to the source machines and push the data.

> **Tip:** For more detailed technical information, refer to the product manuals, which are available at the following Web site:
>
> http://www.ibm.com/software/tivoli/products/directory-integrator/

2.3.1 Policy, auditing, and compliance aspects of TDI

This section discusses how TDI fits into a broad audit solution.

Policy definition and implementation

TDI does not do any policy definition or implementation.

Audit capture

As TDI is basically a framework on which you implement custom functionality, it does not produce audit records. If the function that you are implementing performs some activity that would require an audit trail, you could implement an auditing mechanism using the provided connectors (such as the file system or database connectors).

Audit consolidation

The strength of TDI is its ability to quickly connect multiple data sources and pass data between them. It is ideally suited to capturing and consolidating audit data from disparate sources and forwarding it to a central repository, such as a database or a log file depot.

There are a number of sample AssemblyLines available that take audit log data from a source (such as the Tivoli Common Auditing and Reporting Service, or CARS), map it to the TCIM W7 data format, and write out a comma-separated file (.csv). This file would be defined as a data source to be loaded into TCIM.

> **Tip:** The AssemblyLine is posted on an IBM internal Wiki at the following Web site and through OPAL for external users:
>
> https://w3.webahead.ibm.com/w3ki/display/TDI/Create+custom+adapter+for+TCIM+using+TDI)

Audit reporting

TDI does not do any audit reporting.

As it is a framework, functionality could be implemented onto it to gather log records and produce reports. Connecting to the data sources is straightforward given the variety of connectors available. TDI has its own data store, the Cloudscape DB, or could connect to another database for data manipulation or mining. Presentation TDI can act as a Web server for publishing Web-based reports. It also has connectors to write data in xml/xsl format.

Compliance reporting

TDI does not do any compliance reporting. As above, functionality could be implemented to do this.

3

e-business model auditing tools

This section discusses the tools that are intended to be deployed as part of a typical e-business model, with back-end or early business applications and data being made available via Web-based applications.

3.1 Overview of the e-business model

A simple representation of the e-business model is shown in Figure 3-1.

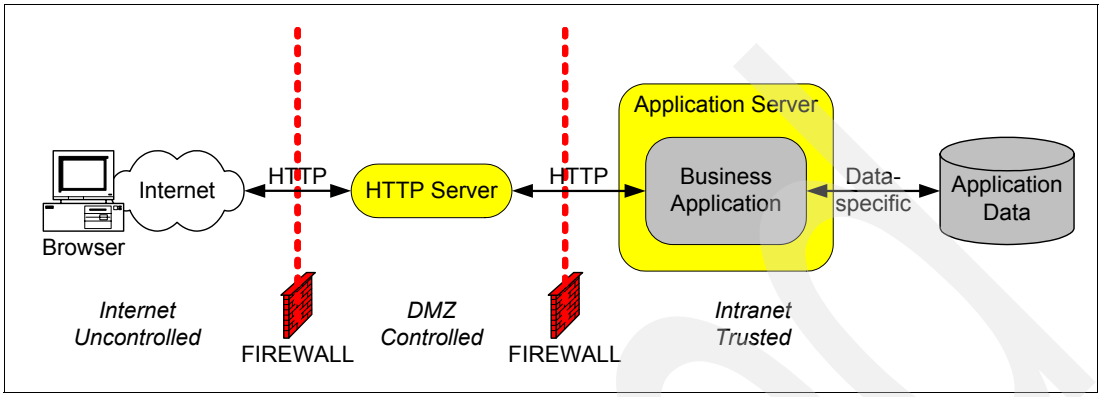


Figure 3-1 Simple e-business model

In this model a client, such as a customer, uses a Web browser to access the Web-enabled application over the Internet. The http traffic goes via the http server. This http server is normally deployed in a DeMilitarized Zone (DMZ) to reduce the exposure of the application and data to hackers on the Internet. The http server may serve static content (such as images and html pages) or route the request to back-end applications hosted on an application server. These applications may then access data directly or work with earlier applications to deliver content to the client.

Often the business application may implement its own security (authentication and authorization). There may also be authentication and authorization required to access the back-end data and earlier applications.

This basic module has some security exposures that may be addressed by additional products or extensions to existing products. Figure 3-2 shows an enhanced security e-business model.

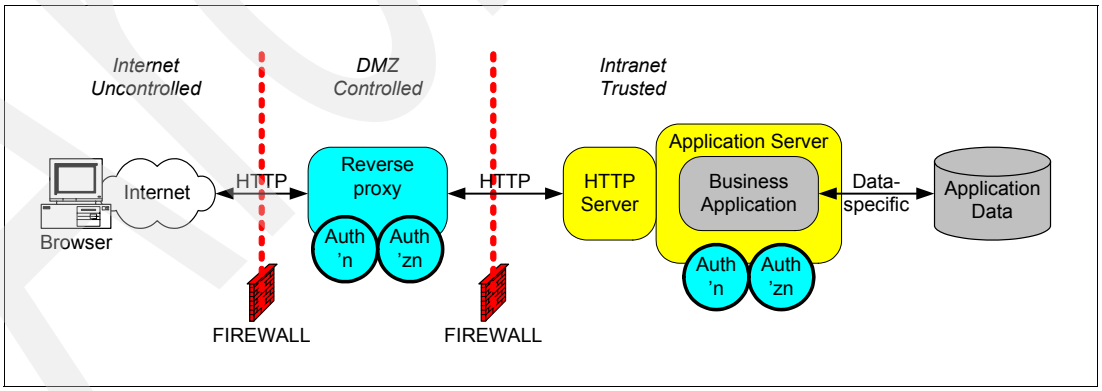


Figure 3-2 Enhanced security e-business model

In this model:

- The http server and its static content has been moved from the DMZ into the intranet (a trusted zone in the network) so that two firewalls must be traversed to access even the static content.

- A reverse proxy security server (RPSS) is deployed into the controlled DMZ. This server implements authentication of incoming http requests and authorization of http access to back-end resources.
- The application server implements the authentication and authorization on behalf of the applications. This enables application developers to get focus on business logic and not worry about implementing their own flavor of security, and to use a common security model across a range of business applications.

In many e-business deployments the reverse proxy security server function is provided by *IBM Tivoli Access Manager for e-business* (TAMeb) and the application server function is provided by *IBM WebSphere Application Server*. TAMeb uses its own authentication and authorization frameworks, and manages the generation of audit records associated with authentication and authorization. WebSphere Application Server implements J2EE mechanisms to perform authentication and authorization, and these may change depending on platform and external security frameworks available. The audit trail generated by the WebSphere Application Server authentication and authorization components depends on the components, not on WebSphere Application Server.

In this chapter we first look at how TAMeb implements authentication, authorization, and auditing. Then we look at two common implementations of WebSphere Application Server, with the TAM JACC Provider, and with SAF authentication and authorization.

3.2 Tivoli Access Manager for e-business

This section introduces IBM Tivoli Access Manager for e-business (TAMeb) and discusses how it fits into the audit picture.

TAMeb provides robust, policy-based security to a corporate Web environment. Its key functions are authentication and authorization of HTTP traffic. It provides authentication of users and authorization of access rights to Web resources. It uses the common Tivoli Access Manager (TAM) authentication and authorization frameworks, shared by other TAM products such as Tivoli Access Manager for Operating Systems (TAMOS). This means that the suite of TAM products can share common policy, user and group definitions and mechanisms to access this information.

Figure 3-3 shows the major components in the TAMeb authorization and authentication model. The figure shows *WebSEAL*, but the middle component could also be the *Plug-in for Web servers* module.

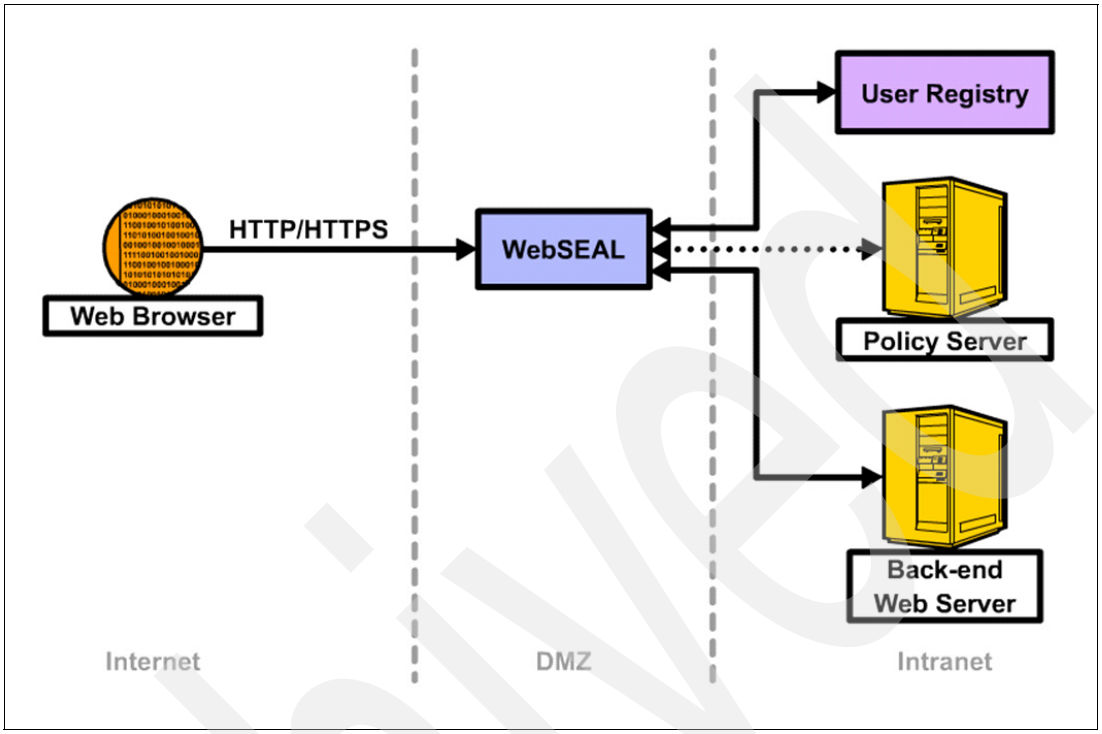


Figure 3-3 TAMeb WebSEAL for HTTP/HTTPS authentication and authorization

In Figure 3-3 a user is attempting to access a URL on a back-end Web server. WebSEAL knows the level of authentication required for this back-end *junction* and if required forces the user to authenticate against the *TAM user registry*. This authentication will result in a set of credentials that include the user and group identities for this user. The next phase is authorization, where these credentials and the requested access are checked against a local copy of the policy (replicated from the *policy server*). If the user is authorized to access this URL the request is passed to the back-end Web server for processing.

It is normal to deploy WebSEAL to a DMZ to isolate the untrusted Internet zone from the trusted internal network.

TAMeb supports *single sign-on* (SSO) for backed Web applications. This may take the form of passing user credentials to back-end applications or using generic accounts to establish trust. For applications running on WebSphere Application Server, there is a TAMeb-specific *Trust Association Interceptor* (TAI) module shipped with WebSphere Application Server that will take a user's TAM credentials and use them to establish the user context in WebSphere Application Server.

As the TAMeb reverse proxy components (standalone *WebSEAL* or the *Plug-in for Web servers*) are configured as the entry points to e-business applications, it can be an effective audit point and has extensive auditing capabilities.

For application integration TAMeb provides several plug-ins, such as those for Microsoft® .NET and BEA WebLogic. These provide advanced capabilities to manage access control at the application level.

The WebSphere Application Server 6.x versions ship with a Java Authorization Contract for Containers (JACC) module that uses the TAM authorization framework for application-level authorization. This is called the *Access Manager JACC Provider*, and is discussed in 3.3, “WebSphere Application Server and TAM JACC Provider” on page 46.

For a typical e-business application deployed on WebSphere Application Server with TAMeb performing the reverse-proxy function (WebSEAL or Web Plug-ins) you would normally find that:

- TAMeb (WebSEAL or a Web Plug-in module) would perform the user authentication,
- TAMeb (WebSEAL or a Web Plug-in module) would perform the coarse-grained authorization, determining which application or major application component could be accessed based on the URL.
- The WebSphere Application Server Access Manager JACC provider would perform the fine-grained authorization, determining which methods could be accessed based on user role, where identity relevant to the role has been passed via the junction from TAMeb.

The remainder of this section focuses on the functions provided by WebSEAL and the Web Plug-in modules.

Architecture of TAMeb

TAMeb comprises the *TAM base components* and the *Web security* components (normally WebSEAL or the Plug-in for Web servers). Figure 3-4 shows the components associated with a WebSEAL TAMeb deployment.

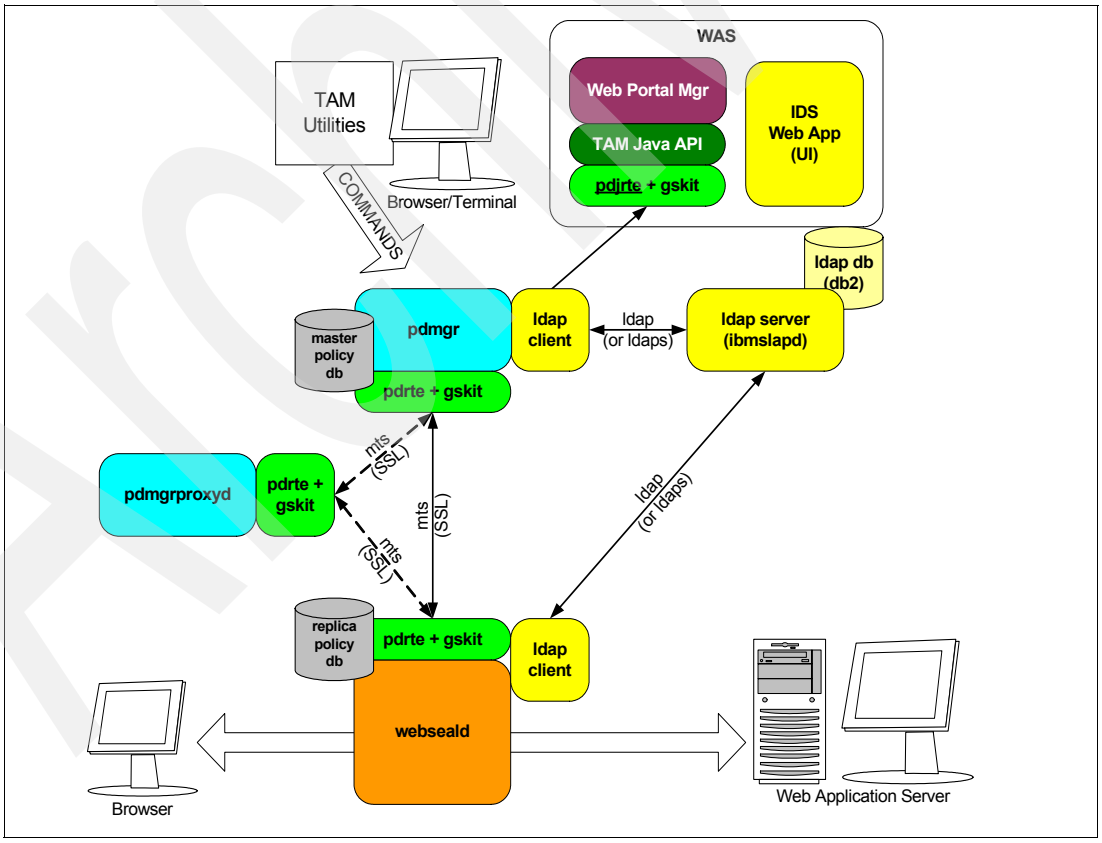


Figure 3-4 TAMeb components

The Web security components are shown at the bottom of Figure 3-4 on page 43. They include:

- WebSEAL implementing the HTTP authentication and authorization functions
- A local copy of the TAM policy database
- The user repository client, which is normally an LDAP V3 client
- The TAM base framework components, consisting of the TAM runtime and Global Security kit (gskit) libraries

Also shown is the TAM user repository. In most cases this is an LDAP server, such as IBM Tivoli Directory Server (ITDS). It contains the users and groups used by the TAM components.

The rest of Figure 3-4 on page 43 shows the TAM base components that are common to any TAMeb (and TAMOS deployment). The components are:

- The TAM Policy Server, or **pdmgrd**. This module manages the policy database and replication to the policy database replicas (such as the one used by WebSEAL).
- The TAM Policy Proxy Server, or **pdmgrproxyd**. This module proxies TAM internal SSL communications.
- The TAM runtime components. The runtime for the C-based TAM components is called **pdrte**. The runtime for the Java-based TAM components is called **pdjrte**. As the internal TAM communication is SSL, the Global Security kit (gskit) libraries are installed with the runtime components.
- The TAM utilities, such as the command-line administration command **pdadmin**.
- The TAM Web-based user interface called Web Portal Manager (WPM). This is a Java application deployed to WebSphere Application Server.

For further information about TAMeb, see the *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-60144.

3.2.1 Policy, auditing, and compliance aspects of TAMeb

This section discusses how TAMeb fits into a broad audit solution.

Policy definition and implementation

TAMeb implements policy through the centralized TAM policy framework. It can define user roles and map them to access rights that apply to Web resources. As the TAM policy database may be shared across various TAM-based products, such as TAMOS, TAM for Business Integration, and the WebSphere Application Server Access Manager JACC Provider, it may be possible to implement enterprise-wide roles that apply access rights across multiple systems to cross-system users.

Roles are implemented in the form of TAM groups. Users or groups are mapped to resources in the protected object space via *access control lists* (ACLs) and *protected object policies* (POPs). ACLs dictate what access a user or group has on a resource. POPs dictate a level of service on that access, such as time-of-day restrictions or auditing levels.

Policy can be implemented to control both the TAM base administrative functions and the Web security access. The TAM base administration policy is detailed in Chapter 4, “Default security policy,” of the *Tivoli Access Manager 6.0 Administration Guide,* SC32-1686.

The policy implemented through TAMeb focuses on access control for http traffic and includes:

Read Can the user view a Web object?

Execute Can the user run a cgi program?

Delete Can the user remove a Web object from the Web space?

Modify Can the user PUT an HTTP object (place, or publish, an HTTP object in the WebSEAL object space)?

List Can the user can see the directory contents when the default index.html page is not present?

Definition of a policy related to Web access would involve defining user roles and mapping these to the URL-accessible parts of a Web application. There is a good discussion of this in Chapter 1, “IBM Tivoli Access Manager WebSEAL overview,” of the *Tivoli Access Manager for e-business 6.0 WebSEAL Administration Guide*, SC32-1687.

In addition to the types of access listed above, additional object policy may be applied. This may control when a resource may be accessed, audit settings, and quality of protections settings. The latter controls the encryption of traffic to the back-end resource and may be set to enforce data integrity (signed but not encrypted) or privacy (encrypted data).

WebSEAL can also implement password policy for password activity it controls. This includes password strength checking when passwords are changed and login failure lockout. Details of this policy can be found in Chapter 10, “Password processing,” of the *Tivoli Access Manager for e-business 6.0 WebSEAL Administration Guide*, SC32-1687.

Audit capture

From an audit perspective both WebSEAL (and equally the Plug-in for Web servers) and the TAM Policy Server generate audit records if configured to do so. WebSEAL audits the http-related activity (authentication and authorization), while the TAM Policy Server generates audit records related to administrative activity. The TAM base and WebSEAL audit levels available include:

audit.authz Authorization events for WebSEAL servers. These are the authorization events for access to the http resources.

audit.azn Authorization events for base servers (such as the TAM Authorization Server, `pdacld`).

audit.authn Authentication, credential acquisition authentication, password change, and logout events. This level may also be split into successful and unsuccessful.

audit.http HTTP access events. This level may also be split into successful and unsuccessful.

audit.mgmt Management events. This generates audit events for administration of the TAM policy, users, and groups (such as activity performed through the `pdadmin` command).

http HTTP logging.

http.clf HTTP request information in common log format.

http.ref HTTP referer header information.

http.agent HTTP user agent head information.

http.cof HTTP information in NCSA combined output format (cof) with time stamp and appends the quoted referer and agent strings to the common log format.

You would normally choose a combination of audit.* audit levels and one of the http.* levels depending on the output format you want. The important audit types are the audit.* levels.

These audit records can be logged locally or routed to the Tivoli Common Auditing and Reporting Service (CARS). As the CARS functionality is being replaced by TCIM, we do not discuss CARS further. We focus on the audit event logging. For details on TAMeb auditing and CARS see the *Tivoli Access Manager for e-business 6.0 Auditing Guide*, SC32-2202.

The audit events that are generated are written into XML-formatted files, either on the local system or routed to another system.

Audit consolidation

As with TAMOS, audit logs can be forwarded to another server that is running the TAM authorization service. You could also use a tool such as TDI for log routing.

TCIM can gather the TAMeb logs and consolidate them in its log depot using either a TAMeb actuator running on the TAMeb machines (such as the policy server machine and the WebSEAL machine), or a TAMeb SSH actuator running on another machine and connecting via SSH to gather the logs. See 2.1, “Tivoli Compliance Insight Manager (TCIM)” on page 14 for more details of TCIM and its log-gathering mechanisms.

Audit reporting

There are no utilities provided to view or report on the XML-formatted log files. To report on the audit log files you need to collect them using TCIM.

Compliance reporting

There are no compliance reports specific to TAMeb.

3.3 WebSphere Application Server and TAM JACC Provider

This section introduces WebSphere Application Server security concepts relating to authentication and authorization and then looks at the first deployment model of WebSphere Application Server where the Access Manager JACC Provider is used for authorization.

3.3.1 Introduction to WebSphere Application Server security

When discussing WebSphere Application Server security we need to understand the security features implemented in WebSphere Application Server and when the external authentication and authorization providers plug in to the model. For someone new to Java and WebSphere Application Server, terms like Java2 security model, JAAS, and JACC are can be confusing. This section introduces the WebSphere Application Server security features and the J2EE security plug-in points. If you are familiar with WebSphere Application Server security concepts you may want to skip straight to 3.3.2, “Overview of WebSphere Application Server security and TAM JACC Provider” on page 51.

WebSphere Application Server authentication features

WebSphere Application Server uses a pluggable authentication module consisting of an *authentication mechanism* and an appropriate *user registry*. The authentication mechanism is

an implementation of *Java Authentication and Authorization Service (JAAS) login modules*. Only one authentication mechanism and user registry can be enabled at one time.

There are four types of JAAS login modules supported with WebSphere Application Server v6:

SWAM The Simple WebSphere Authentication Mechanism, for simple, non-distributed, single application server environments

LTPA The Lightweight Third Party Authentication mechanism, for distributed, multiple application server environments

Custom User-developed implementation

Each of these have mechanisms to accept a set of credentials from clients, such as a user ID/password pair, token, or certificate, and establish the WebSphere Application Server identify.

WebSphere Application Server supports three types of user registries under WebSphere Application Server Global Security:

LocalOS The WebSphere Application Server principals (users and groups) are held in the local OS repository (such as the UNIX /etc/passwd and /etc/group files or RACF on z/OS).

LDAP An LDAP-compliant directory is used to store the WebSphere Application Server principals (such as the IBM Tivoli Directory Server).

Custom A custom repository is deployed and integrated with WebSphere Application Server.

Beginning with WebSphere Application Server V6.1, you can federate multiple user repositories in a cell's security realm. The federated registries can be file-based, LDAP, or database-based.

WebSphere Application Server Trust Association Interceptor (TAI)

The Trust Association Interface implementation is a user-customizable interface to make use of external authentication. It relies on another party to perform the user authentication and it trusts this party. When it receives credentials from that party is validates the origin and performs any required mapping. This is shown in Figure 3-5.

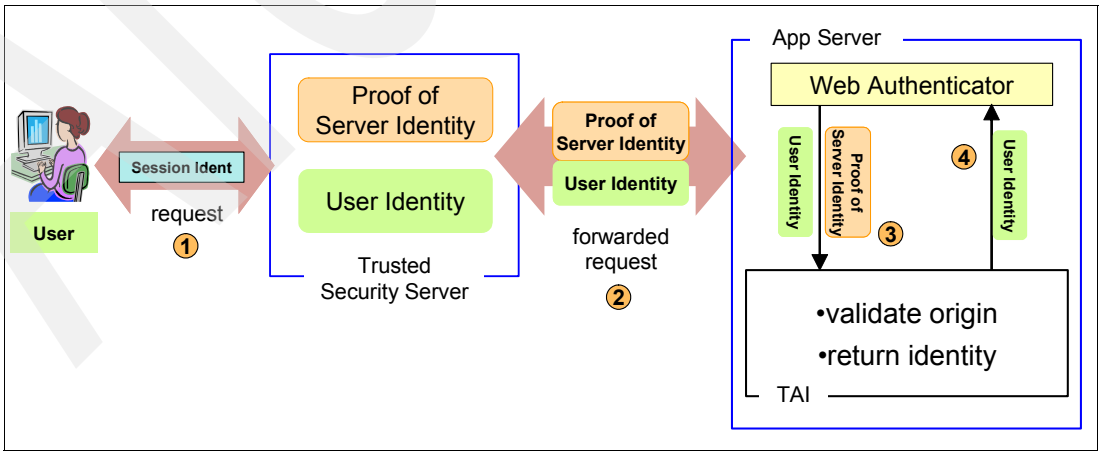


Figure 3-5 Generic Trust Association Interceptor model

WebSphere Application Server 6.x ships with a TAI module called the Trust Association Interceptor Plus (TAI++) that will accept credentials passed from TAMeb WebSEAL in the

http header, perform a validation against a TAM authorization server, and then create the WebSphere Application Server PDPrincipal based on the identity that authenticated in WebSEAL, as shown in Figure 3-6. This allows the same identity to be used in WebSEAL policy and WebSphere Application Server policy.

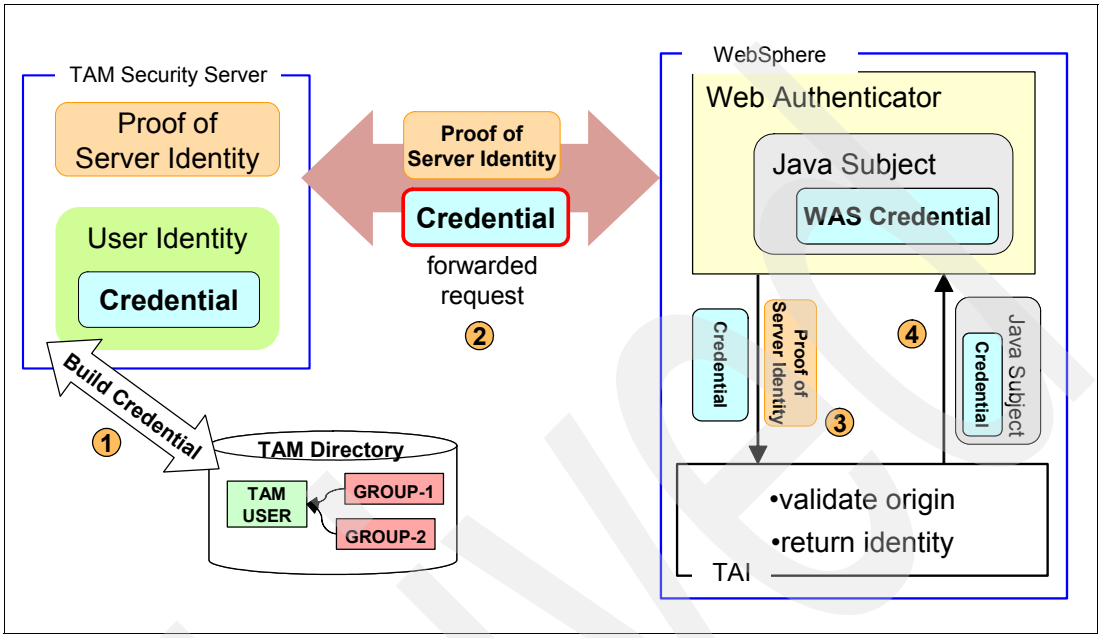


Figure 3-6 WebSphere Application Server TAI++ implementation

For a detailed explanation of the TAI++ module, see Chapter 11, "WebSphere Application Server security," in *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-6014.

It is conceivable that a WebSphere Application Server deployment could have a TAI and one or more JAAS login modules to establish different identities for the WebSphere Application Server principal.

WebSphere Application Server authorization features

WebSphere Application Server authorization generally consists of Java2 Security, J2EE security, and the Java Authorization Contract for Containers (JACC). One or more of these may be deployed.

Java2 Security provides a policy-based, fine-grained access control mechanism. It is part of the Java 2 Standard Edition (J2SE™) and is independent of J2EE role-based authorization. It enforces access control based on the location of the code and that signed it, not on the principal. It is defined in a set of policy files and enforced at run time.

The J2EE 1.4 security features implemented in WebSphere Application Server 6.0 include:

Role-based security J2EE implements role-based security and moves the burden of the implementation out of the application code and into the application *assemblers* and *deployers*. It enables them to define security roles, sets of permissions for access to Web resources, and specific EJB™ methods.

The J2EE specification defines a security role as a logical grouping of users that is defined by an application component provider or assembler. It is then mapped by a deployer to security identities (for

example, principals or groups) in the operational environment (such as WebSphere Application Server).

Declarative security The declarative security mechanisms, as part of J2EE, are stored in a document called the deployment descriptor using declarative syntax. Global security roles for a WebSphere Application Server application are stored in the XML deployment descriptor. Security roles for WebSphere Application Server components are stored in their corresponding deployment descriptors inside the EAR, Java archives (JARs), and Web archives (WARs).

WebSphere Application Server uses method permissions to describe security roles for EJBs. For a particular EJB resource, method permissions are the association of role names with the sets of methods, based on what types of permissions should be required to invoke the methods. For example, a deployment descriptor may define a role of Teller mapped to the method getBalance of the AccountBean EJB. To access the getBalance method of AccountBean, a user must be mapped to the Teller role.

For Web resources (servlet, JSP™ and URL), security constraints are the association of role names with the sets of HTTP methods based on the types of permissions that should be required to access the resource. These are defined in the WARs deployment descriptor. For example, you could map the role SalesPerson to the POST and GET http methods in the Sales URL.

Programmatic security

The use of role-based and declarative security does not require an application developer to implement security code. However, the business logic may dictate the need for different levels of security within a single EJB. For example, a method may implement a money transfer facility, but transfers over a certain amount may require additional security checks. This requires security to be implemented into the application.

The API for programmatic security in J2EE consists of four methods for performing security checks:

- isCallerInRole (EJBContext)
- getCallerPrincipal (EJBContext)
- isUserInRole (HttpServletRequest)
- getUserPrincipal (HttpServletRequest)

These methods enable components to make business logic decisions based on the security role of the caller or remote user.

In summary, WebSphere Application Server authorization uses a role-based model. WebSphere Application Server treats a role as a set of permissions to access a particular resource. Figure 3-7 shows the various mappings associated with WebSphere Application Server authorization.

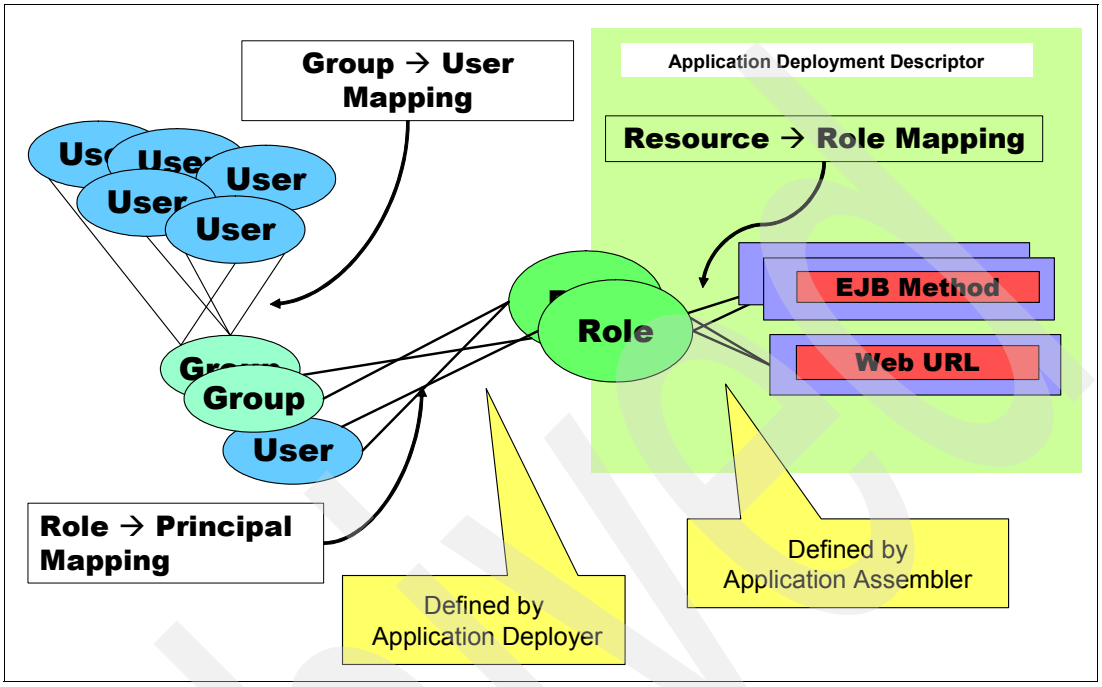


Figure 3-7 WebSphere Application Server role-based security

These are normally external to an application, based purely on the application structure. However, there are methods provided by the J2EE specification to allow programmatic access to role and principal information.

The z/OS WebSphere Application Server uses a custom authorization module called the SAF Authorization Provider. This is discussed in 3.4, “WebSphere Application Server using z/OS SAF” on page 58.

Java Authorization Contract for Containers (JACC)

The Java Authorization Contract for Containers was introduced in J2EE 1.4 to allow third-party authorization service providers to plug into application servers like WebSphere Application Server using standard interfaces for policy configuration and access decisions. It describes a standard *contract* (interfaces and rules) that authorization framework providers must meet. Figure 3-8 shows the relationships between the J2EE Container and the JACC Provider.

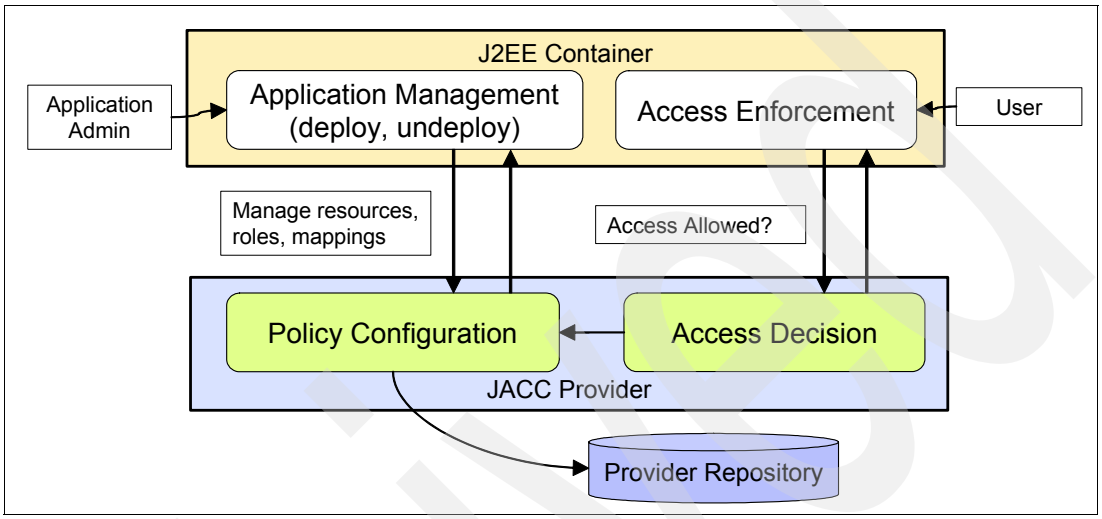


Figure 3-8 Java Authorization Contract for Containers

WebSphere Application Server ships with the Access Manager JACC Provider that utilizes the TAM authorization framework, also used by TAMeb and TAMOS, to implement the authorization mechanism. This is discussed in the remainder of this section.

3.3.2 Overview of WebSphere Application Server security and TAM JACC Provider

This section introduces some of the WebSphere Application Server security concepts and how the Tivoli Access Manager (TAM) JACC Provider and other components implement authentication and authorization.

WebSphere Application Server with the TAM JACC Provider introduction

WebSphere Application Server 6 ships with a JACC Provider that uses the Tivoli Access Manager authorization framework for the container-level authorization. The authorization policy is implemented in the same policy database as the other TAM products in the TAM domain. This means that the same users and groups can be mapped to WebSEAL policy, which may secure both WebSphere Application Server-based and other http resources, and WebSphere Application Server security policy, such as the J2EE roles and resources.

The J2EE roles, resources, and mappings are defined in the application descriptors and implemented in TAM at the time of installing the application (Figure 3-7 on page 50). The users and groups can be administered using TAM utilities or through the WebSphere Application Server administration console. The principal to role mapping is only performed through the WebSphere Application Server administrative console.

WebSphere Application Server with the TAM JACC Provider architecture

The components involved with the TAM JACC Provider in WebSphere Application Server are shown in Figure 3-9. It extends the generic JACC design as shown in Figure 3-8 on page 51.

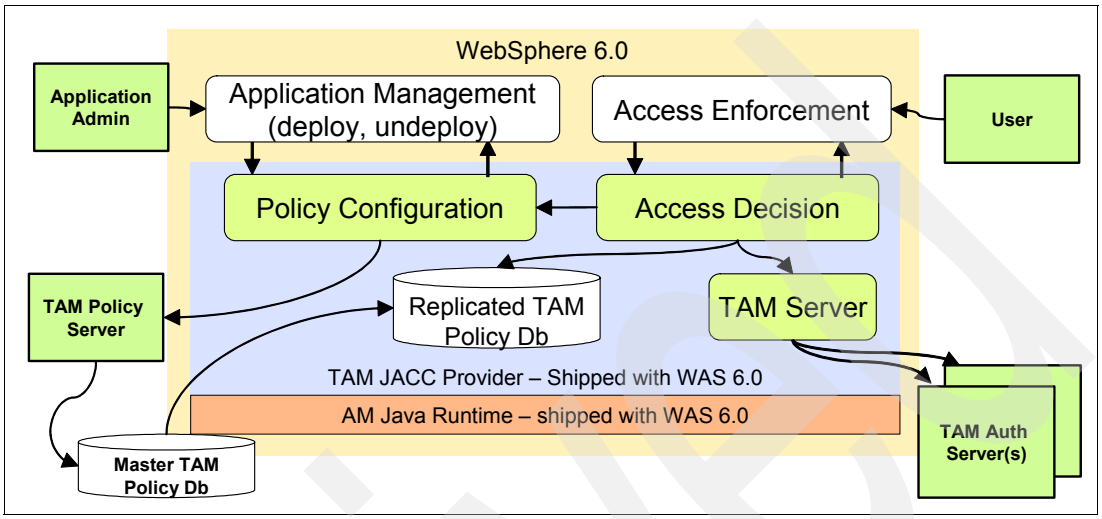


Figure 3-9 WebSphere Application Server and Tivoli Access Manager JACC Provider

This implementation includes:

- The TAM JACC Provider. This is shipped with WebSphere Application Server 6 and is implemented as a series of Java classes. It includes an embedded TAM client that will hold a local copy of the TAM policy database. The local copy of the TAM policy database will be replicated by the TAM policy server in the same way as for WebSEAL and other TAM clients.
- The TAM Java Runtime (TAMJrte) libraries. These enable communication between TAM components, such as the TAM policy server and TAM authorization servers.
- Remote TAM authorization servers (one or more).
- The TAM policy server and master policy database.

The TAM JACC Provider also adds a JAAS login module to verify the TAM credentials. This uses the remote authorization servers to build the PdPrincipal.

The TAM JACC Provider implementation requires that the user has authenticated to WebSphere Application Server with a TAM identity. This may be through user authentication directly against WebSphere Application Server, but is more likely to be via WebSEAL SSO where a user has authenticated to WebSEAL and the WebSEAL credentials have been passed to the TAI++ module to built the WebSphere Application Server credentials. This is shown in Figure 3-6 on page 48.

3.3.3 Policy, auditing, and compliance of WebSphere Application Server with the TAM JACC Provider

This section discusses how this implementation model fits into a broad audit solution.

Policy definition and implementation

Policy for WebSphere Application Server applications involves the application structure (resources), roles to access the resources, and principals (users and groups) mapped to the roles, as shown in Figure 3-7 on page 50.

The roles, resources, and mapping between them is defined in the Application Deployment Descriptor and loaded into TAM when the application is installed.

The users and groups are defined in TAM and are normally users with access to other resources in TAM, such as the WebSEAL URL resources. The definition of these is done through TAM mechanisms (such as the **pdadmin** command).

The principal (user or group) to role mapping is performed within the WebSphere Application Server Administration Console. It may be viewed in the TAM object space (through **pdadmin** or the Web Portal Manager UI) or through the WebSphere Application Server Administration Console.

Audit capture for distributed platforms

WebSphere Application Server provides a limited audit capability. You can use the *Diagnostic Trace Service* to set the level of tracing. The levels include:

Off	No logging.
Fatal	Only log fatal messages.
Severe	Log severe and fatal messages.
Warning	Log warning, severe, and fatal messages.
Audit	Log audit, warning, severe, and fatal messages.
Info	Log informational, audit, warning, severe, and fatal messages.
All	Maximum trace logging.

To enable any auditing, you need to enable at least the audit level (for example, *=audit). You can also specify the component to set the trace level against.

This level of trace is indicated by an A (for audit) in the log entries before the error code. Many of the components will issue audit messages, so you cannot identify audit-related messages by the error code, only the trace level flag. Example 3-1 shows some of the security-related audit events in SystemOut.log showing TAI++ and LDAP information.

Example 3-1 WebSphere Application Server security audit events in SystemOut.log

```
[11/9/07 11:48:52:605 EST] 0000000a TrustAssociat A   SECJ0122I: Trust Association
Init Interceptor signature: WebSeal Interceptor Version 1.1
[11/9/07 11:48:52:613 EST] 0000000a TrustAssociat A   SECJ0121I: Trust Association
Init class com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus loaded
successfully
[11/9/07 11:49:01:787 EST] 0000000a LdapRegistryI A   SECJ0419I: The user registry
is currently connected to the LDAP server ldap://9.12.4.18:3391.
[11/9/07 11:49:02:698 EST] 0000000a PluggableAuth A   SECJ0157I: Loaded Vendor
AuthorizationTable: com.tivoli.pdwas.websphere.PDWebSphere Application
ServerAuthzManager
```

Example 3-2 shows some administrative audit messages in SystemOut.log.

Example 3-2 WebSphere Application Server administration audit events in SystemOut.log

```
[11/9/07 11:16:33:792 EST] 0000009e AdminHelper   A   ADMN1021I: An attempt is
made to stop the server1 server.
[11/9/07 11:32:03:598 EST] 0000000a RMIConnectorC A   ADMC0026I: The RMI Connector
is available at port 2809
[11/9/07 11:46:41:440 EST] 00000036 FileRepositor A   ADMR0010I: Document
cells/linux4zNode01Cell/security.xml is modified.
```

While there are some messages that may be useful from an audit perspective, such as security.xml being modified, the messages do not identify who made the changes. This makes the native WebSphere Application Server auditing mechanism of limited value.

For more details on the WebSphere Application Server trace audit level, see Chapter 5, “Diagnosing problems (using diagnostic tools),” in the *IBM WebSphere Application Server Base V6 Troubleshooting and Support* manual. We do not discuss these audit records further.

From a user audit perspective we need to rely on the auditing generated by the WebSEAL integration (through TAI++) and the (TAM) JACC Provider. The TAI++ and TAM JACC Provider components do not cut their own audit records. The identity coming into TAI++ has been authenticated (and authorized) by WebSEAL or the Web Plug-in module, and that component will cut an audit record. The TAM JACC Provider is configured by default into local-mode, which means that it holds a local copy of the TAM policy database and performs its own authorization. In this mode there will be no authorization records written showing the Java container objects being accessed. It can be configured to operate in remote mode, where all authorization decisions are sent off to a TAM authorization service (the pdacld deamon), and this can generate audit records. However, this is undocumented and might have performance implications.

Example 3-3 shows an authentication audit entry for user dedwards in the WebSEAL audit log.

Example 3-3 WebSEAL audit entry for authenticating dedwards

```
<event rev="1.2">
<date>2007-11-09-19:03:21.059-05:00I-----</date>
<outcome status="0">0</outcome>
<originator blade="webseald" instance="default"><component
rev="1.4">authn</component>
<event_id>101</event_id>
<action>0</action>
<location>linux11z.itso.ibm.com</location>
</originator>
<accessor name="">
<principal auth="IV_LDAP_V3.0" domain="Default">dedwards</principal>
<name_in_rgy>cn=David Edwards, ou=Employees,
o=ITSO</name_in_rgy><session_id>5915ae14-8f20-11dc-a687-0200
00000004</session_id><user_location>9.12.5.148</user_location><user_location_type>
IPV4</user_location_type></accessor><target
resource="7"><object></object></target>
<authntype>basicAuthRFC2617</authntype><data>
</data>
</event>
```

When the transaction arrives in WebSphere Application Server, the TAI++ module verifies the identity of the SSO user associated with the junction and then builds the credentials of the login user. For normal user access through the TAI++ and JACC modules you will see three audit records:

- The junction sso user being authenticated by TAI++
- The junction sso user being authorized by TAI++
- The login (TAM) user being authorized by TAI++

Example 3-4 show examples of the audit records from the pdacld audit file for user dedwards accessing a resource through TAI++ and JACC. Example 3-4 shows the junction sso user being authenticated.

Example 3-4 TAI++ WebSEAL sso user authentication audit record

```
<event rev="1.2">
<date>2007-11-12-12:02:19.423-05:00I-----</date>
<outcome status="0">0</outcome>
<originator blade="pdacld"><component rev="1.2">authn</component>
<event_id>101</event_id>
<action>0</action>
<location>linux4z.itso.ibm.com</location>
</originator>
<accessor name="">
<principal auth="IV_LDAP_V3.0" domain="Default">webseal_sso</principal>
</accessor><target resource="7"><object></object></target>
<data>
</data>
</event>
```

Example 3-5 shows the junction sso user being authorized.

Example 3-5 TAI++ WebSEAL sso user authorization audit record

```
<event rev="1.2">
<date>2007-11-12-12:02:19.472-05:00I-----</date>
<outcome status="0">0</outcome>
<originator blade="pdacld"><component rev="1.2">azn</component>
<event_id>110</event_id>
<action>0</action>
<location>linux4z.itso.ibm.com</location>
</originator>
<accessor name="">
<principal auth="IV_LDAP_V3.0" domain="Default">webseal_sso</principal>
<name_in_rgy>cn=webseal_sso,o=itso</name_in_rgy></accessor><target
resource="3"><object>IV_LDAP_V3.0:webseal_sso</object></target>
<data>
azn_id_get_creds</data>
</event>
```

Example 3-6 shows the TAM user (the one that logged into WebSEAL) being authorized. This is mused to build the WebSphere PDPrincipal object.

Example 3-6 TAI++ WebSEAL login user authentication audit record

```
<event rev="1.2">
<date>2007-11-12-12:02:19.825-05:00I-----</date>
<outcome status="0">0</outcome>
<originator blade="pdacld"><component rev="1.2">azn</component>
<event_id>110</event_id>
<action>0</action>
<location>linux4z.itso.ibm.com</location>
</originator>
<accessor name="">
<principal auth="IV_LDAP_V3.0" domain="Default">dedwards</principal>
<name_in_rgy>cn=David Edwards, ou=Employees,
o=ITSO</name_in_rgy></accessor><target
resource="3"><object>IV_LDAP_V3.0:dedwards</object></target>
<data>
azn_id_get_creds</data>
</event>
```

As these are all the records written for the WebSphere Application Server part of the transaction, you will notice that there is no mention of the resource to which JACC is authorizing access.

Note: With the default configuration of the TAI++ and JACC TAM components, you do not see what resources the user is being authorized to access, just which user is performing the access request. You could configure the TAM JACC Provider to operate in remote-mode, which will forward all authorization requests to a TAM authorization deamon (pdacld) to generate records to show what Java container resource was accessed, but the steps are not documented, may be unsupported, and could have performance implications.

The final component that may be involved in the WebSphere Application Server auditing is LDAP. If LDAP is the user repository for TAM, there will be a record of a bind operation when WebSEAL performs the authentication (which should duplicate the record cut by WebSEAL to its audit log). Also, where LDAP is the authentication source for WebSphere Application Server Global Security and users access an application without going through WebSEAL, an audit record of the LDAP bind may be the only record of authentication.

The level of auditability depends on the LDAP being used:

- For the z/OS ISS LDAP, there is a debug level of CONNs that will log connection activity to the SYSPRINT DD. See Chapter 8, “Customizing the LDAP server configuration,” in *z/OS Integrated Security Services LDAP Server Administration and Use*, SC24-5923.
- For the z/OS ITDS LDAP, there is SMF auditing available for the LDAP operations, including bind. See Chapter 9, “Running the LDAP server,” in *z/OS IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191.
- For the distributed ITDS LDAP, audit records can be written to the LDAP audit log (slapdlog) for the LDAP operations, including bind. See Chapter 15, “Logging Utilities,” in the *IBM Tivoli Directory Server Administration Guide*, SC32-1674.

The components involved in the audit trail are shown in Figure 3-10.

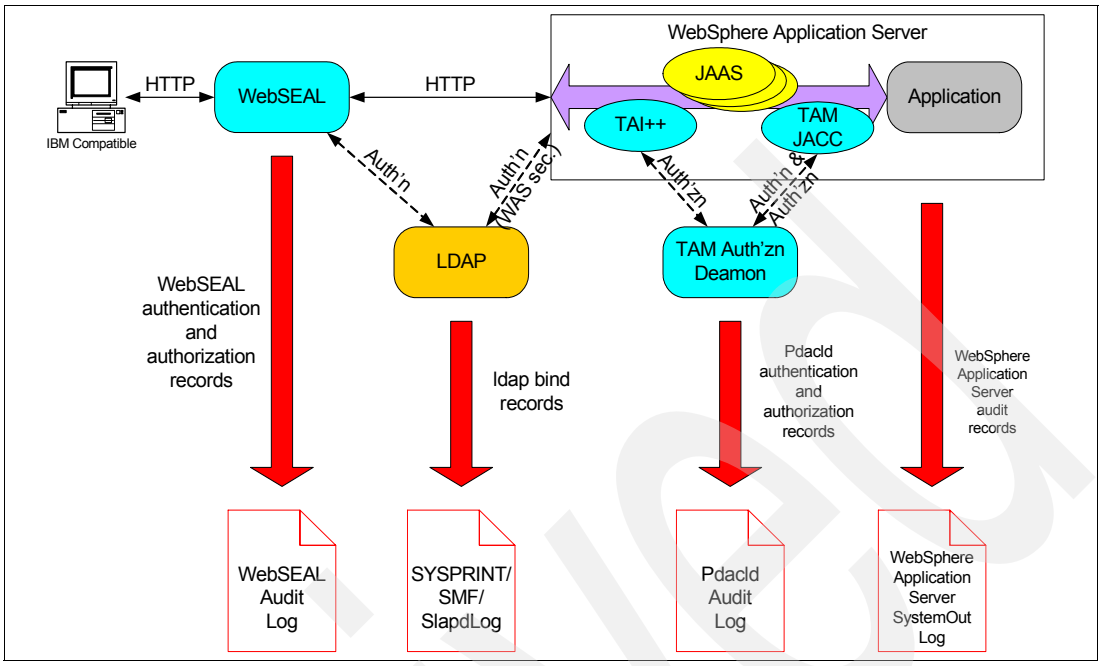


Figure 3-10 Auditing with WebSphere Application Server, TAI++, and the TAM JACC Provider

The key audit logs depicted here are the WebSEAL audit log showing who authenticated to WebSEAL and was authorized to access the relevant URL, and the TAM authorization deamon (pdacld) log showing the TAI++ user validation. The WebSphere Application Server audit log would have limited use and the LDAP audit records (for bind operations) may supplement the TAM audit logs.

Audit consolidation

There are no mechanisms to centralize the WebSphere Application Server SystemOut.log containing audit data. A mechanism would have to be built to parse the relevant records from the log, possibly normalize them, and forward them to a central location.

Consolidation of the LDAP audit records depends on the LDAP server:

- If the LDAP audit records are from the z/OS ISS LDAP, there would need to be a custom mechanism to forward them to some other location.
- The z/OS ITDS records are written to SMF. In one respect they are already consolidated in SMF and could be reported on there. However, the records are Type 83 Subtype 3, and currently there are no products that gather these and centralize them (such as the TCIM Enablers). Custom coding would be required to gather them from SMF and forward to some other consolidated repository.
- For the distributed ITDS LDAP, the audit records are written to the slapd.log file and are among other (non-audit) records. There is no mechanism to forward these to a consolidated repository. A mechanism would have to be written to do this.

The TAM audit logs are the same as used by other TAM deployments and can be consolidated in the same way as discussed in 3.2.1, “Policy, auditing, and compliance aspects of TAMeb” on page 44.

Audit reporting

There are no reporting mechanisms available for the audit records written to the WebSphere Application Server SystemOut.log. One would need to be written to process the records, but this would have limited value based on our observations above.

There is no reporting provided for the different LDAP audit trails listed above. You could run reporting against the z/OS ITDS LDAP SMF records using normal SMF reporting tools. Currently, z/Secure Audit does not process SMF Type 83 Subtype 3, so custom CARLa reports would have to be written.

For reporting against the TAM audit logs see 3.2.1, “Policy, auditing, and compliance aspects of TAMeb” on page 44.

Compliance reporting

This is the same as for TAMeb. See 3.2.1, “Policy, auditing, and compliance aspects of TAMeb” on page 44.

3.4 WebSphere Application Server using z/OS SAF

This section looks at the second deployment model of WebSphere Application Server, which uses z/OS SAF for authentication and authorization.

3.4.1 Overview of WebSphere Application Server using z/OS SAF authentication and authorization

In 3.3, “WebSphere Application Server and TAM JACC Provider” on page 46 we presented an overview of WebSphere Application Server, including general WebSphere Application Server and Java security concepts, and WebSphere Application Server using Tivoli Access Manager components to provide authentication and authorization functions. This section adds to that by describing WebSphere Application Server in a z/OS environment using the z/OS security engine, such as RACF, for authentication and authorization.

Introduction to WebSphere Application Server using z/OS SAF

In a z/OS environment WebSphere Application Server can use operating system security to provide both the authentication and the authorization functions.

WebSphere Application Server on z/OS is at a different level of functionality from its distributed counterpart. The security setting options available with the current version of WebSphere Application Server on z/OS (6.1) are different from WebSphere Application Server on the distributed systems.

The z/OS WebSphere Application Server authentication can be set to use the LocalOS registry, LDAP or custom. To use RACF or other z/OS security engines, you need to use LocalOSUserRegistry. As with the distributed WebSphere Application Server, JAAS and TAI can be used to supplement authentication.

The z/OS WebSphere Application Server supports default and JACC for authorization. There is also special z/OS-only option called System Authorization Facility (SAF) authorization. When this is enabled authorization calls are passed out to SAF (and thus RACF or whatever the security system is).

To implement this in RACF there is a resource class EJBROLE. This is the role object shown in Figure 3-7 on page 50. Example 3-7 shows an example of creating EJB roles.

Example 3-7 Define EJBoles

```
RDEFINE EJBROLE Employee UACC(NONE)
RDEFINE EJBROLE Worker UACC(NONE)
RDEFINE EJBROLE Manager UACC(NONE) APPLDATA('ROLEMGR')
RDEFINE EJBROLE CEO UACC(NONE) APPLDATA('ROLECEO')
```

The value specified for APPLDATA is the RACF user ID that comes into effect if you use the RunAs role approach. If a method is configured to run as an EJBROLE, the user ID specified in the APPLDATA field is the user ID that the process runs as in WebSphere.

There is also a grouping EJB role class called GEJBROLE.

The users and groups shown in Figure 3-7 on page 50 are implemented as RACF users and groups. The principal to role mapping is achieved by permitting users or groups to have read access to the EJBROLE class, as shown in Example 3-8.

Example 3-8 Permit access to EJBRoles

```
PERMIT Employee CLASS(EJBROLE) ID(USREMP) ACCESS(READ)
PERMIT Employee CLASS(EJBROLE) ID(USRWORK) ACCESS(READ)
PERMIT Employee CLASS(EJBROLE) ID(GRPMGRS) ACCESS(READ)
```

In this case USREMP and USRWORK are RACF users, and GRPMGRS is a RACF group.

Architecture of WebSphere Application Server using z/OS SAF

The architecture of this implementation is trivial. z/OS WebSphere Application Server will call out to the local OS security module (such as RACF) for all authentication when local security is enabled. It will use the SAF authorization mechanism when a user attempts to execute a method.

If TAMeb WebSEAL is being used as a reverse proxy security server in front of WebSphere Application Server, there needs to be some mechanism to access the WebSphere Application Server resources as a RACF identify. This may involve:

- Authenticating in WebSEAL with the RACF identity and passing it through the WebSEAL junction through a trust relationship. The standard WebSphere Application Server TAI++ module will do this. Additionally, it may require a custom JAAS login module to authenticate the user to RACF.
- Authenticating in WebSEAL with a TAM identity and mapping this to a RACF identity. This may involve WebSEAL getting the RACF identity and passing it to WebSphere Application Server, which would use a custom TAI module to add it to the PDPrincipal object. It may also involve WebSEAL only passing its TAM identity back and having a TAI module determine the RACF identity. In both cases a custom JAAS login module may be required to authenticate the user to RACF.

The first option may be achievable by using the z/OS LDAP interface to RACF.

3.4.2 Policy, auditing, and compliance aspects of WebSphere Application Server using z/OS SAF

This section discusses how WebSphere Application Server using z/OS SAF fits into a broad audit solution.

Policy definition and implementation

All users, groups, and EJBRoles are defined in RACF. This means that the WebSphere Application Server application security can be managed in the same place, and using the same tools, as other security.

Theoretically, user groups could be implemented as job roles that manage WebSphere Application Server resource access in the same way as access control to other, related, resources. For example, a job role of branch teller may involve WebSphere Application Server EJBRole access as well as CICS and DB2 access.

Audit capture

As RACF is the common point for WebSphere Application Server authentication and authorization, then SMF Type 80 audit records are being cut against WebSphere Application Server application access.

Audit consolidation

In one respect the audit records are already consolidated. They are in a central SMF repository and may be reported against along with the other RACF SMF records.

The TCIM z/OS RACF Enabler can be used to gather these records and forwards them to the central TCIM server for consolidated reporting.

If there is a requirement to tie WebSEAL authentication and coarse-grained authorization to the fine-grained RACF authorization, then both the TAM audit records and the SMF records must be consolidated in one place and in the same format. TCIM can read and process TAM audit records, so consolidating all of the records on the TCIM server is a logical choice. It may also be possible to parse and forward the TAM audit records into SMF, but this would require some customization.

Audit reporting

As all WebSphere Application Server and other RACF audit records are in SMF, then the zSecure Audit tool can be used for reporting.

The TCIM auditing can also be used if the SMF records are exported to TCIM. This is the preferred solution if other, distributed, audit records are also to be reported on.

Compliance reporting

As for the other audit reporting, compliance reporting can be performed using the zSecure Audit tool or TCIM's compliance reporting capability. Neither products provide out-of-the-box compliance reports that use the WebSphere Application Server SMF records.

4

z/OS

This chapter discusses the tools that can be used to perform auditing on the z/OS system. As mentioned earlier, we have limited the scope of this book to the IBM-platform tools and IBM software products. (There are many more tools and products available on the market that may fill niche auditing requirements.)

4.1 RACF

This section covers the Resource Access Management Facility Product (RACF) security product of z/OS. RACF performs the security functions for the z/OS operating system. RACF provides the security administration interfaces for the user and group definitions, and it allows for the implementation of the security policy by letting the security administrator build the access control rules for the system resources. RACF also provides the authentication of users as they enter the z/OS systems and, likewise, the authorization as these users try to access the z/OS resources. Lastly, RACF logs all this activity in the System Management Facility (SMF), which is the z/OS auditing and logging facility.

RACF is integrated into the z/OS operating system when the z/OS system is built by the system programmer using the System Modification Program/Extended (SMP/E) feature of z/OS. This integration allows all the system resource managers to have one basic interface into the z/OS security environment. This interface is called the System Authorization Facility (SAF). When a user tries to access a resource, the resource manager (or owner of that resource) issues a security call (the RACROUTE macro) to SAF, and SAF works with RACF and its access control rules to ensure that the user has access to the resource at the appropriate level.

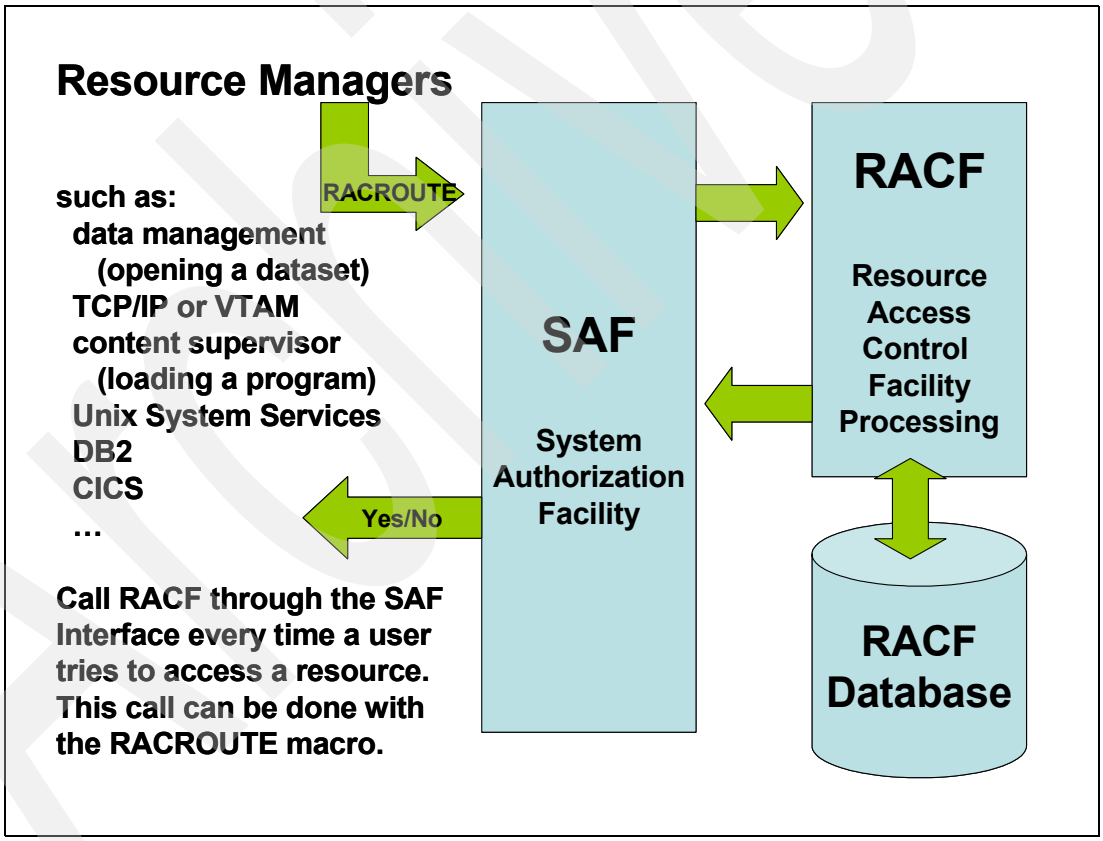


Figure 4-1 Interaction between the z/OS Resource Managers and RACF

4.1.1 Policy, auditing, and compliance aspects

RACF is used to define users to the z/OS system and provides validation for these user IDs with passwords, digital certificates, kerberos tickets, and so on. The group definitions for these users are also defined with RACF commands or interfaces. RACF is also used to define and protect the z/OS resources and authorize users access to these resources.

When RACF defines the users to the z/OS system, there are four different types of users. In addition to a *normal* user, RACF has three privileged attributes that can be assigned to the user ID that is being defined. A normal RACF user is a RACF user ID that is defined without any of the privileged attributes (the majority of the user IDs defined on a z/OS system). These privilege attributes are:

- Special - This attribute allows the user to issue RACF commands, such as define users and change access control rules. It is given to the RACF security administrators.
- Operations - This attribute allows the user access to z/OS data sets so that they can perform activities such as backup and recovery operations. It is given to storage administrators and selected operational personnel.
- Auditor - This attribute allows the user to set, use, and create audit commands and records for resources and users. It is given to z/OS auditors.

All user IDs having either the special or operations privilege attribute assigned to it should be audited for all their activities. This done by using RACF to set AUDIT on the actual user IDs or within the RACF system configuration. This provides a check-and-balance type of environment and helps make the z/OS system compliant with several of the most recent legal requirements.

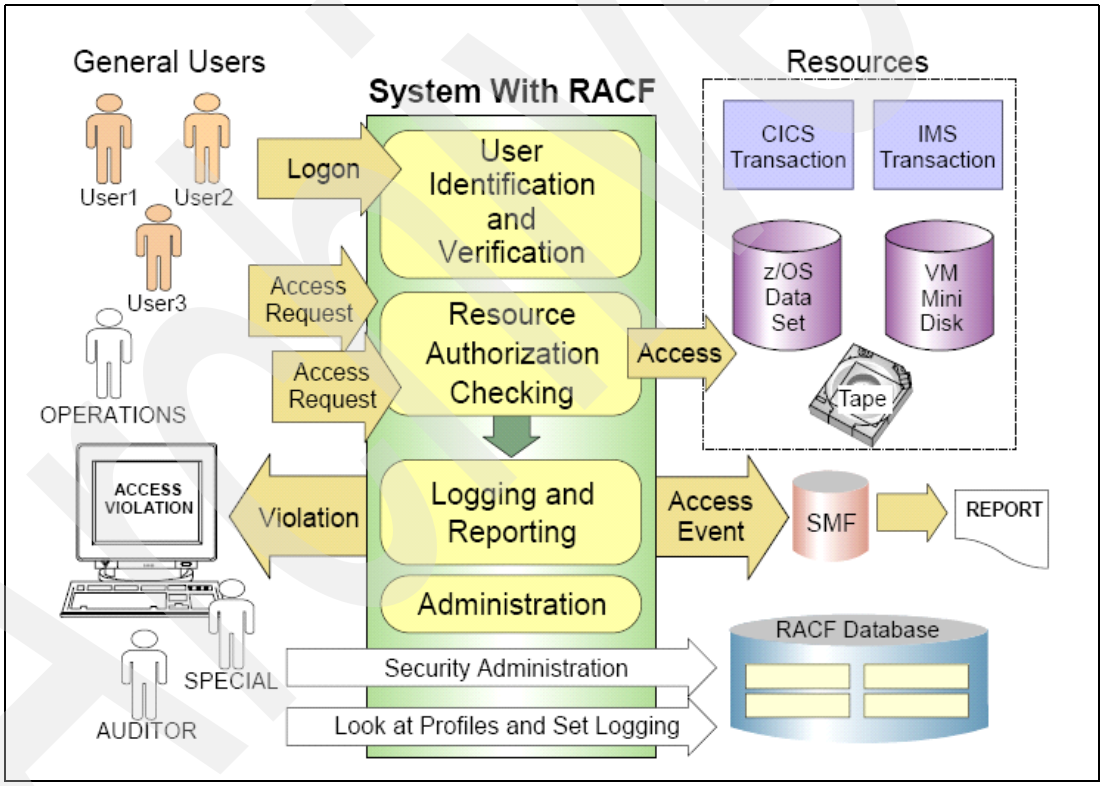


Figure 4-2 Major functions and components of RACF

There are other users who have special privileges on a z/OS system and who need to be audited as privileged users. These are users who have access to critical resources at high levels. Examples of these users are DBAs, system programmers, and administrators who configure critical applications or access highly sensitive data. All user IDs having any form of privilege on system infrastructure resources, software resources, or application resources should be audited. Most companies have identified what they consider critical resources and have policies on how they must be audited. Auditing privilege user IDs and access to critical

resources are particularly important for compliance with Sarbanes Oxley and other regulatory laws.

All access to z/OS resources defined in RACF can be logged and audited. The security policy should define which ones need to be audited at all times, and the others that can be dynamically audited by the RACF AUDITOR as needed. RACF will write its audit records to SMF type 80 to 83 records. The SMF data set is one of the critical system resources that should be protected and isolated.

All resources, especially critical system and data resources, on a z/OS system should be protected via RACF. Resources that are not defined in RACF will not be protected unless they use internal security within the application products itself. Then other tools need to be used for auditing. Access to RACF protected resources can be defined at different levels, and auditing of these resources can be set for successful or failed access at these different levels.

Tip: For more details refer to:

- *z/OS V1R9.0 Security Server RACF Security Administrator's Guide*, SA22-7683-11
- *z/OS V1R9.0 Security Server RACF Auditor's Guide*, SA22-7684-09

RACF has a set of predefined classes of resources and the flexibility so that the security administrators can define new resource classes that need to be protected. These classes are defined within the RACF class descriptor table (CDT). This flexibility allows software that does not use the predefined classes to define and protect their own resource classes within the RACF environment. The RACF AUDITOR can audit RACF resources by the RACF class or by the individual protected resource.

RACF audit records

RACF writes its audit records in SMF types 80, 81, 82, and 83. These SMF audit records can be used for RACF reporting. RACF can also notify the system or the user in real time by either writing a message to the RACF user ID or sending a message to the z/OS syslog. Audit record logs are to be kept according to your companies policy and regulatory laws.

RACF auditing reporting

RACF has two utilities that can be used to assist with auditing and reporting on the RACF configuration and installation. The IRRADU00 utility is used to reformat the SMF security records, the types 80, 81, 82, and 83, for a reporting tool. The same can be said for the IRRDBU00 utility. This tool will rewrite information from the RACF database in a similar format for a selected reporting tool. Using the data from one or both of these RACF utilities, the RACF AUDITOR can build reports about the configuration of its RACF environment, as well as audit reports on who is accessing the system's critical resources.

There are several reporting tools that can use the data from the IRRADU00 and IRRDBU00 utilities. RACF provides samples for DB2 and ICETOOL for building sample RACF reports using this data. If auditors do not want to create their own reports and want to use reports from utilities that are bought off the shelf, then there are tools such as IBM zSecure Audit and Vanguard's VRA tools. By using IBM Tivoli Compliance InSight Manager, the auditors can get much more detailed reporting for more than just the RACF resources and users.

Infrastructure and integrity reporting

As demonstrated in the previous sections, z/OS is built on a structure of checks and balances, so the actions of people with RACF SPECIAL can be monitored and, in some cases, controlled. With the RACF AUDITOR and UAUDIT capabilities, z/OS and RACF provide a method of monitoring all users including the privileged users, as needed. And with a

protected centralized logging facility, SMF, z/OS provides an easy way to allow reports to be generated for the auditors. As discussed in previous sections, within RACF, there are the IRRDBU00 and IRRADU00 unload utilities that can provide RACF database and SMF information to report generators such as ICETOOL or DB2. Examples of these can be found in the *RACF Auditor's Guide*, SA22-7684, and *RACF System Administrator's Guide,* SA22-7683, with samples found in SYS1.SAMPLIB.

One of the major features that is critical to z/OS security is the integrity of the z/OS operating system. It is essential that this integrity is configured and maintained appropriately. The integrity of the z/OS operating system is protected in large part by the System z hardware and microcode. Concepts such as address spaces, storage protect keys, page and frame tables, PSW, and so on, are all features that help protect the integrity of the z/OS operating system. Some of these features that are mentioned are also protected by RACF, the integrated z/OS security product. There are certain critical z/OS resources that must be protected and monitored correctly by RACF to insure that the system integrity is maintained. Examples of the critical resources are the master catalog, the SMP/E environment, APF-authorized programs and data sets, and system configuration files. Not only do these critical resources need to be identified and monitored, but there also needs to be an easy, periodic method of reporting on who has access to them and at what level. For this discussion, APF-authorized data sets are used to demonstrate some examples of how to audit and monitor critical resources that only privileged users should special access to.

As with all security, protecting the APF-authorized data sets is a mixture of process and technology. First, there must be a documented method of what type of data sets will be allowed to be in the APF-authorized programs. The system programmer who is responsible for configuring the z/OS system will build and maintain a list of data sets that are allowed to have APF-authorized programs. This list is controlled through PARMLIB members, but the easier method of finding out which data sets are in the current APF-authorized list is via the operator command D PROG,APF. The output for this display command (Figure 4-3 on page 66) will list all the APF-authorized data sets that are currently running on the system. It is a consideration to protect this command as well as the SETPROG and SET commands with the OPERCMD class in RACF.

When the output of the display is returned, if there are hundreds of data sets in the list, then there is probably a problem with the maintenance process of the APF list. Part of the process of protecting the APF list is the documentation and maintenance of how many data sets should be allowed into the list, what type of data sets, and how to remove data sets. Data sets that are uncataloged and have been deleted from the system or renamed (for example) should not be allowed in the APF list. But the display command only provides the data set names and none of the fine details about the protection of this critical resource. Nor is it an easy method for an auditor to monitor a resource. For more information and a better method of checking the protection of the APF list, other tools need to be used.

```
-D PROG,APF
 FORMAT=DYNAMIC
 ENTRY VOLUME DSNAME
    1  Z19RB1 SYS1.LINKLIB
    2  Z19RB1 SYS1.SVCLIB
    3  Z19RB1 SYS1.VTAMLIB
    4  Z19RB1 IOE.SIOELMOD
    5  Z19RB1 SYS1.SISTCLIB
    6  Z19RB1 SYS1.SERBLINK
    7  Z19RB2 SYS1.SHASLNKE
    8  Z19RB1 SYS1.SICELINK
    9  Z19RB1 SYS1.SORTLIB
   10  Z19RB1 SYS1.SORTPV
   11  Z19RB1 SYS1.SICELPA
   12  Z19RB1 SYS1.SORTLPA
   13  Z19RB1 SYS1.CMDLIB
   14  Z19RB1 SYS1.SIEALNKE
   15  Z19RB1 SYS1.MIGLIB
   16  Z19RB1 SYS1.CSSLIB
   17  Z19RB1 ISF.SISFLOAD
   18  Z19RB1 ISF.SISFLPA
```

Figure 4-3 Sample output from the display command for the APF list

RACF has the DSMON utility, which provides the facility to list out the APF data sets and provide the auditor with the required information. Figure 4-4 is a sample of the JCL and Figure 4-5 on page 67 and example of the output. It is a separate background job and generates a separate report, but it is part of RACF and reports on several items about RACF and system integrity.

```
//JACKCK JOB ‘D5202P,?’,MSHLEVEL=(1,1),MSGCLASS=A
//GO2    EXEC PGM=ICHDSM00,PARM=JACK
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  FUNCTION ALL
  USEROPT USRDSN CEE.SCEERUN
//SYSUT2 DD SYSOUT=*
```

Figure 4-4 Sample DSMON job

```
                 VOLUME   SELECTION   RACF        RACF
DATA SET NAME    SERIAL   CRITERION   INDICATED   PROTECTED    UACC
-----------------------------------------------------------------------
CEE.SCEERUN      SHRESA   APF         NO          YES          READ
```

Figure 4-5 Sample DSMON output

The RACF Data Security Monitor (DSMON) is a good quick method for checking that selected critical resources have the appropriate protection, but it might not provide enough information for a compliance audit. Some of this audit information is event driven, such as when were the APF data sets last changed by USERA who as access because of this RACF profile. Other bits of the audit information is access information, such as these RACF GROUPs have access at the UPDATE level and these people are in that GROUP. To get this type of information, other tools need to be used.

Most of this information can be produced with the use of the RACF database unload utility (IRRDBU00) with a mixture of the SMF audit logs. SMF will provide the event-driven information (USERA changed the APF data set because of this RACF profile), and the IRRDBU00 can provide the information about the GENERAL RESOURCE, USER, and GROUP PROFILEs. Using a combination of the RACF utilities, IRRDBU00 and IRRADU00, all of this information can be provided into either a data set or a DB2 database for processing. A report can be generated using a tool such as ICETOOL from DFSORT™ or DB2. Samples are provided in SYS1.SAMPLIB for several different types of reports.

There are some tools that will provide ad hoc reports and alerts. In 4.3, “zSecure” on page 68 you can find a testing sample using the Tivoli zSecure Audit to generate reports for monitoring APF data sets.

Off-the-shelf auditing tools

There are several off-the-shelf tools that be purchased to enhance the auditing and reporting of a RACF environment. Some of them are:

- IBM zSecure Audit and Alert - These are the ones that we use in this book. The zSecure Audit tool provides predefined audit reports using SMF records and other active system control blocks and configuration parameters.
- CA Examine - This tool reports on the active security and system environment that is currently running. It uses several of the system configuration parameters.
- Vanguard’s VRA - This provides several predefined audit reports. It uses SMF and the RACF information to provide these reports.

4.2 SMF

For most of these tools and for a complete auditing of a z/OS system, the auditor needs to understand what is being logged in the System Management Facility (SMF) data set. The z/OS system writes many types of records into the SMF data set (performance, security, encryption, and so on, information). All this and many other bits of information are written into the SMF data set. The RACF security records are type 80, 81, 82, and 83 records. There are many other SMF records that can be very important for the z/OS auditor, so it is very important that the auditor understands what is being recorded in SMF and how the SMF data set is protected.

The SMFPRMxx parameter in SYS1.PARMLIB defines what SMF records are being recorded. Normally, the SMF records are dumped to a generation data set at appropriate times during the day. The SMFDUMP JCL defines what SMF records will be dumped to your generation data set. The number of generation data sets and for how long a time SMF data are being stored depends on your company's policy and regulatory laws. Figure 4-6 is an example of the SMFPRMxx member of SYS1.PARMLIB and the SMFDUMP JCL used to dump the SMF data.

```
BROWSE     SYS1.PARMLIB(SMFPRMZZ)    01.21                Li
Command ===>
*********************************** Top of Data **********
     ACTIVE                         /*ACTIVE SMF RECORDING
     DSNAME(SYS1.&SYSNAME..MAN1,    /*SMF DATA SET NAMES*/
            SYS1.&SYSNAME..MAN2)    /*SMF DATA SET NAMES*/
     NOPROMPT                       /* NO PROMPT  JUST DO
     REC(PERM)                      /*TYPE 17 PERM RECORDS
     INTVAL(01)
     SYNCVAL(01)
     MAXDORM(3000)                 /* WRITE AN IDLE BUFFER
     STATUS(010000)                /* WRITE SMF STATS AFTE
     JWT(0010)                     /* 522 AFTER 10 MINUTES
     SID(&SYSNAME(1:4))            /* SYSTEM ID IS &SYSNAM
     LISTDSN                       /* LIST DATA SET STATUS
     LASTDS(MSG)                   /*DEFAULT TO MESSAGE
     NOBUFFS(MSG)                  /*DEFAULT TO MESSAGE
     SYS(TYPE(30,70:79,80:83,100:102,110,120),
         EXITS(IEFU83,IEFU84,IEFU85,IEFACTRT,
         IEFUJV,IEFUSI,IEFUJP,IEFUSO,IEFUTL,IEFUAV),
         INTERVAL(SMF,SYNC),NODETAIL)
```

Figure 4-6 Example of SMFPRMxx to capture the type 80, 81, 82, and 83 SMF records

To capture the SMF 80, 81, 82, and 83 records, you need to specify them as part of the SYS parameter. For DB2, you need to add the 102 type to collect the Db2 audit records.

Tip: For more details refer to *z/OS V1R9.0 MVS System Management Facilities (SMF)*, SA22-7630-15

4.3 zSecure

The IBM Tivoli zSecure suite of mainframe security tools provides enhanced administration, compliance, and auditing facilities. The zSecure suite components allow for administration and policy enforcement of RACF security that is simpler and more diverse than the tools that are supplied with the base product. The zSecure suite also provides monitoring, auditing, and alerting features for RACF, ACF2, and top secret-based mainframes and allows this data to be presented to IBM Tivoli Compliance InSight Manager.

The IBM Tivoli zSecure suite is made of six different products:

- IBM Tivoli zSecure Admin: enables more efficient and effective RACF administration, using significantly less resources.
- IBM Tivoli zSecure Audit: compliance and audit solution that enables you to automatically analyze and report on security events and detect security exposures on the z/OS platform.

This is the main zSecure product that we used in this document, especially to monitor the privileged users on z/OS.

- IBM Tivoli zSecure Alert: real-time mainframe threat monitoring allowing you to monitor intruders and identify misconfigurations that could hamper your compliance efforts.
- IBM Tivoli zSecure Command Verifier: policy enforcement solution that enforces compliance to company and regulatory policies by preventing erroneous commands.
- IBM Tivoli zSecure Visual: reduces the need for scarce, RACF-trained expertise through a Microsoft Windows-based GUI for RACF administration.
- IBM Tivoli zSecure CICS Toolkit: allows you to perform mainframe administrative tasks from a CICS environment, freeing up native-RACF resources.

IBM zSecure Audit

Other than parts of the CICS Toolkit and Visual products, all of the zSecure suite runs within z/OS and uses ISPF panels for interaction. It is common to see both zSecure Admin and zSecure Audit deployed together, as they share components and the ISPF interface.

The zSecure Admin and Audit products have a full and sophisticated interactive interface, implemented completely in ISPF functions. ISPF is the program being executed during most of an interactive session. ISPF uses *panel*, *skeleton*, and *message* libraries supplied with zSecure. The interactive panels call the zSecure application program (CKRCARLA) as needed.

For this book we use the zSecure Audit tool to generate some reports on privileged users of the z/OS system. The zSecure Audit tool can also automatically send security information from the z/OS system into Tivoli Compliance Insight Manager. We used predefined reports, but the zSecure Audit product is based upon a custom reporting language called CARLa, which allows the auditor to build specialized reports if the needed.

The zSecure Audit has a full interactive interface that implemented completely in ISPF. The interactive panels and programs call the zSecure Audit program (CKRCARLA) as needed to handle the user's requests.

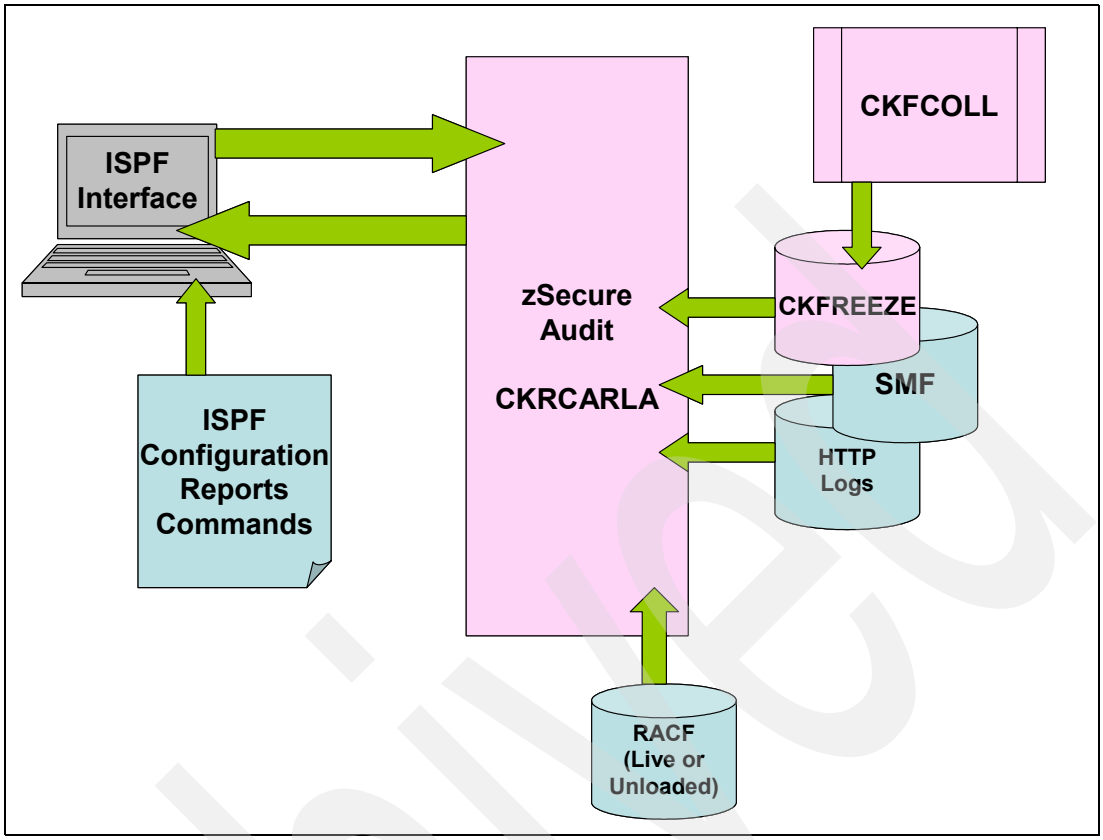


Figure 4-7 Components for the zSecure Audit environment

The zSecure Audit program can operate on a live or an unloaded RACF database. Other data that can be used to generate audit reports are the SMF data sets, the HTTP logs from the IBM HTTP Server, and data generated in the zSecure CKFREEZE data set. The CKFREEZE data is one of the major enhancements to the auditing completeness on z/OS systems. This CKFREEZE data is generated by a batch job called zSecure Collect, or CKRCOLL. This batch job collects data from online VTOCs, VVDSs, catalogs, and active z/OS control blocks and configuration information, so that zSecure Audit can build a more complete picture of the current operating system. The CKRCARLA program (zSecure Audit) runs the CARLa reporting programs to interact with RACF, gather data for and format the reports, and build the display panels.

IBM zSecure Command Verifier

Other zSecure programs can interact with zSecure Audit to enhance the auditing of a z/OS operating system. One of them is IBM Tivoli zSecure Command Verifier, which was mentioned previously. The zSecure Command Verifier is implemented as an exit to the RACF commands. It is the standard RACF command exit, IRREVX01, which is invoked during processing of all RACF commands. With zSecure Command Verifier, the RACF commands can be controlled and protected at a very granular level. Who has the ability to issue the RACF commands or to use certain parameters within certain RACF commands is controlled by RACF profiles and the customer's security policy. Whenever a RACF command is executed, the IRREVX01 exit will check the RACF profiles to see whether this person has the authority to execute this RACF command, and audit records will be written to the SMF data set. In other words, RACF commands and the RACF profiles that protect them become nothing more than another z/OS resource that RACF protects and audits. A good introduction to zSecure Command Verifier can be found in the *zSecure Command Verifier Users Guide*, SC23-6550.

IBM zSecure Alert

Another potential important auditing program is zSecure Alert. This is the real-time monitor for the zSecure products. It issues alerts for security events against critical system resources that need to be addressed in real-time instead of waiting for audit reports to be generated on historical data. The drivers for these alerts are events in SMF and messages written to the system console. zSecure Alert uses SMF exits to capture real-time SMF data and an Enhanced Console Service to intercept write to operator (WTO) messages. This allows zSecure Alert to react in real time to system events. It provides the real-time alerting mechanisms that include smtp (e-mail), WTO, and snmp alerts that IBM Tivoli Security Operation Manager (TSOM) and IBM Tivoli Compliance Insight Manager (TCIM) can process.

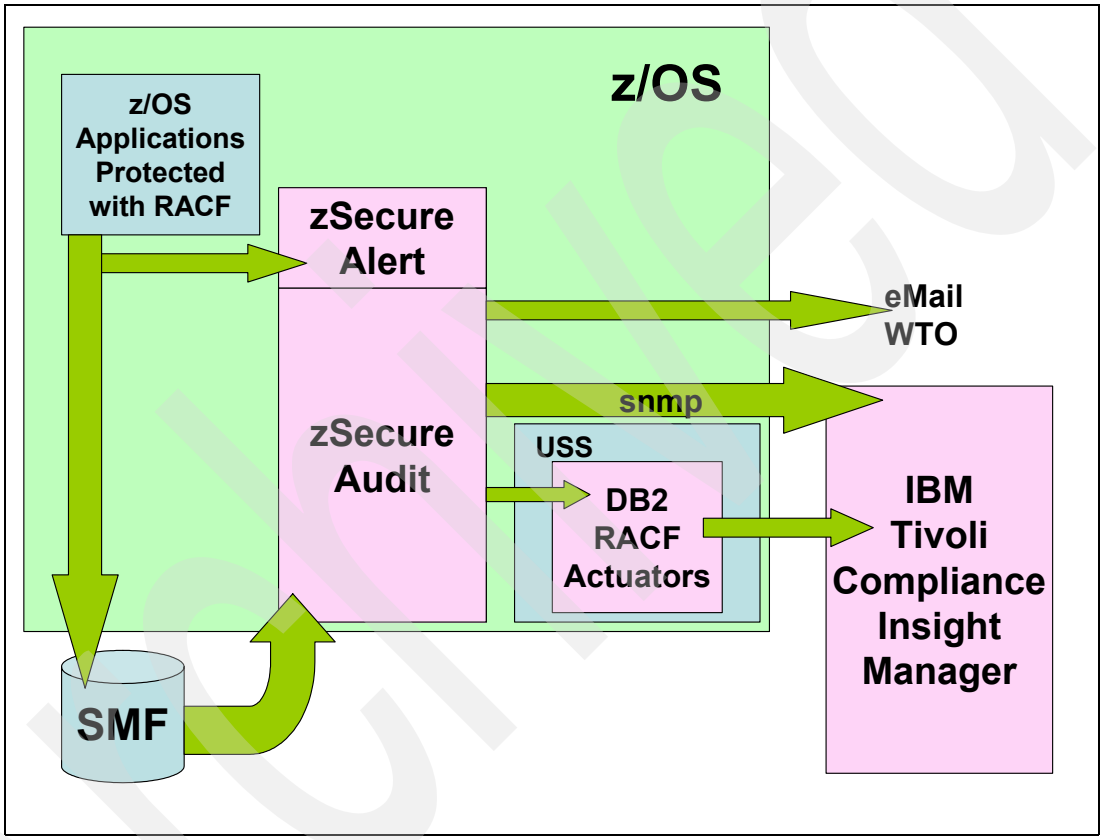


Figure 4-8 Component for zSecure Audit, CKRCARLA, and zSecure Alert, CKRPOLIC

Events can also flow to TCIM via the TCIM z/OS RACF, DB2, or UNIX System Services actuators. These are batch processing of events and not real-time alerting mechanisms. These are what we configured for use in our book.

4.3.1 Policy, auditing, and compliance aspects of the zSecure suite

Auditing, particularly audit reporting, is one of the strengths of the zSecure suite. This section looks at the auditing capabilities of the products.

Policy definition and enforcement with zSecure

The zSecure suite provides two major capabilities in a System z security environment: administration and auditing. The administration tools allow security administrators to manage objects in the existing External Security Manager (ESM) that they have implemented, such as

RACF or the Computer Associates ACF/2 and top secret products. The security policy (in this case the identities and the resource access rights for system z resources) is defined in the ESM, not in the zSecure products.

The zSecure products provide additional ESM resource types to control administration through the zSecure administration products. So security policy related to ESM administration can be extended to include administration through the zSecure products.

The zSecure Command Verifier product can also assist policy enforcement by restricting administration functions to sets of users. Let us look at an example. Assume that a company has naming standards applied to RACF user IDs and the first two characters dictate the department that a user is in. Only RACF administrators for a department can create user IDs for that department (that is, starting with their unique two-character identifier). The zSecure Command Verifier product can implement a filter to restrict account creation in this way. Command Verifier can also be used to enforce policy by supplying, or overriding, command attribute settings such as password expiry interval. A good introduction to zSecure Command Verifier can be found in Chapter 1, "Introduction," and Chapter 2, "Overview," in the *zSecure Command Verifier Users Guide*, SC23-6550.

Audit capture with zSecure

This section looks at the zSecure products and how they generate audit records.

zSecure Admin, Visual, CICS Toolkit, and Audit

The three administrative products (zSecure Admin, zSecure Visual, and zSecure CICS Toolkit) all operate on RACF resources and are controlled by RACF profiles. Audit records for these activities are written by RACF to SMF in the same way that records would be cut for command-line administration of RACF.

The zSecure products add a number of XFACILIT class profiles to control activity performed through the zSecure products:

- The CKR.* profiles control access to the ISPF panel functions and actions run through the CKRCARLA program. For example, there is a CKR.READALL profile that allows administrators read access to zSecure functions. There are also CKR.OPTION.* profiles that can restrict and allow access to specific zSecure RACF panel options.
- Use of the zSecure Collect utility, CKFCOLL, to build the CKFREEZE data set is controlled by CKF.* profiles.
- Use of the CKGRACF utility is controlled by CKG.* profiles.
- Use of zSecure Visual (the Windows-based administrative interface) is controlled by C2R.* profiles.

As these are RACF profiles, they can have audit settings attached, which results in SMF records being cut. In summary, these products do not write records to SMF directly. All activity will be logged by RACF as SMF Type 80 records.

zSecure Alert

The zSecure Alert product can create its own alerts. This generates records, such as additional audit records, to the chosen output destination, such as SMTP going to TCIM. This allows the administrator to supplement SMF-based audit records with custom ones.

Access to the zSecure Alert configuration panels is controlled by RACF profiles in the same way as the other zSecure products. So access to alert settings can be audited to SMF.

zSecure Command Verifier auditing

The zSecure Command Verifier product provides various functions for auditing both the commands as issued and the effects of the implemented policies:

- The *command audit trail* function records information about the RACF commands issued in the affected profiles themselves. This means that using easy RACF list commands it is possible to obtain information about who last changed a particular part of a profile (like the OWNER, the UACC, or the access list).
- The *policy profile effect* function records information about the effect of the Command Verifier policy profiles via SMF records. The RACF command before and after the processing of the policy profiles is recorded via the LOGSTRING information for access to special policy profile effect recording profiles. Of course, it is also possible to use regular SMF access recording for the policy profiles themselves.

The command audit trail function involves writing a history of changes to a profile into that profile's USRDATA field. This gives a quick check, without extracting SMF records, of the changes made to the profile (Figure 4-9).

```
Command Audit Trail for USER IBMUSER

Segment:   CICS       Added on 05.241/03:19 by C4RTEST
                      Changed on 05.241/03:20 by C4RTEST
           TSO        Changed on 05.241/03:19 by C4RTEST
Attrib:    PASSWRD    Removed on 05.238/14:24 by C4RTEST
           INTERV     Changed on 05.241/04:42 by C4RTEST
           RESTR      Added on 05.238/14:24 by C4RTEST
Connect:              BCSC Added on 05.238/14:24 by IBMUSER
GrpAttr:   ADSP       BCSC Removed on 05.238/14:24 by IBMUSER
```

Figure 4-9 Sample output of command audit trail for IBMUSER

The C4RCATMN command is used to view and modify the audit trail data. The trail is controlled by RACF profiles.

The policy profile effect function involves writing SMF records. Via the definition of specific C4R profiles in the XFACILIT resource class, an auditor may specify the events that are to be logged to SMF. For every RACF command, the command as entered by the terminal-user and the command as finally executed may be logged. Of course, it is also possible to specify that both should be logged. The auditor may indicate that auditing via SMF should be done only for selected users, or for all users.

Of each command, only the first 255 characters will be available via the generated SMF records. Also, the error message as issued to the terminal user is available as logstring in the SMF record via the C4R.ERRMSG.command profile.

An example of a profile being used for audit specification is shown in Figure 4-10, which indicates that the ADDUSER command should be audited for all users before being examined by the installation exit. When ADDUSER is issued by user IBMUSER, it should not be audited.

```
C4R.PREAUD.ADDUSER UACC(READ) AUDIT(SUCCESS(READ))
      ACL: IBMUSER NONE
```

Figure 4-10 zSecure Command Verifier profile and audit setting

This auditing is applied irrespective of whether Command Verifier disallows or modifies the command parameters before passing to RACF for processing. This means that it is another possible administrative audit point.

Aside from auditing the entire command, Command Verifier also provides the possibility to audit individual keywords. Command Verifier will create successful and failed access records for the profile that was used in the decision process. For example, if the profile C4R.USER.ID.CRMB* was used to allow the definition of the new user CRMBTST, a successful access event will be recorded against this profile. If a policy profile denied setting a field to the specified value, a violation event will be recorded against the policy profile.

For more details see Chapter 4, "Auditing RACF Commands," in the *zSecure Command Verifier Users Guide*, SC23-6550.

Command Verifier can control which administrative commands a user may run. It can audit these commands in the USRDATA field of the resource and in SMF.

Audit consolidation with zSecure

The zSecure products are operating against the ESM (such as RACF), so all audit records are written to SMF by the ESM. All auditable activity performed through the zSecure products will result in SMF Type 80 records. In one respect, these records are consolidated into a single repository for auditing—the SMF sub-system and its data sets.

As the records written to SMF are standard Type 80 records, they will be processed by the TCIM RACF Enabler and consolidated with the other TCIM data for audit and compliance reporting.

Audit reporting with zSecure

The three zSecure products that are associated with reporting are Admin, Audit, and Alert. This section describes their reporting capabilities.

Reporting with zSecure Admin

While the main function of the zSecure Admin tool is to administer ESM resources, it does have a reporting facility that has components shared with zSecure Audit. In this section we are talking about the reporting function available under the ISPF zSecure RA (for RACF Administration) menu. It is similar under the AA (for ACF2 Administration) menu.

The admin reports can be used to investigate the current system state. They can show how the system is defined, but have no knowledge of any system policy and so do not highlight exposures, nor do they investigate the audit trail in SMF. The admin reports can be used for forensic investigation along with other tools. The reports available under the RA.3 zSecure menu are shown in Figure 4-11.

```
zSecure Admin+Audit for RACF - RACF - Reports
 Option ===>

 1    Profiles        Any profiles fitting mask/qualifier with their data sets
 2    Non redundant   Data set profiles different from less specific profiles
 3    Redundant       All data set profiles, mark if non-redundant (as in 2)
 4    Permit/scope    User/group on access list, or access by any means
 5    Out of group    Group data sets accessible to users outside the group
 6    Non default     Data sets with more in access list than 'owner has alter'
 7    Match           Find profiles that cover a data set or resource
 8    Group tree      Group tree display
 9    USERDATA        Display and action on profiles with USERDATA
 A    Tapevol         Tapevol profile overview
 B    RACFvars        RACF variable profiles
 C    APPL            Application profiles
 D    JES/328X        Jes/328X definitions and log data sets
 E    SDSF            SDSF command and display authorities
 F    JES2            Access to JES2 resources
 G    Compare users   Compare access and/or connect
```

Figure 4-11 zSecure Command Verifier profile and audit setting

These reports come in different categories:

- Special reports to combine data from several profiles, and from profiles and resource information (RA.3.1 through RA.3.6, RA.3.D)
- Special function panels (RA.3.7 through RA.3.9, and RA.3.G)
- Class-specific profile query panels (RA.3.A through RA.3.C)
- Application-specific overview panels (RA.3.E and RA.3.F)

Figure 4-12 shows the RACF permit/scope report showing the highest level of access for user DAVIDED for each resource class. This is one of the special reports that has combined data from several profiles.

```
zSecure Admin+Audit for RACF Authorization for DAVIDED           Line 1 of 52
Command ===>                                                   Scroll===> CSR
                                                   29 Nov 2007 00:59
   Complex  Scope of Profiles HighAcc
   SC60     DAVIDED      4562 OWNER
   Class    Profiles HighAcc
   @LINUX          1 CKGLIST
   ACCTNUM         4 READ
   ACICSPCT        1 READ
   APPCLU          1 CKGLIST
   APPL            4 READ
   CACHECLS        3 CKGLIST
   CBIND          78 READ
   CDT             1 CKGLIST
   CRYPTOZ         6 CONTROL
   CSFKEYS         2 READ
   CSFSERV         2 ALTER
   data set      141 OWNER
   DIGTCERT      120 ???????
   DIGTRING       35 CKGLIST
   DSNADM        166 ALTER
```

Figure 4-12 zSecure admin permit/scope report

These reports are presented as interactive ISPF panels. In most you can drill down and expand where relevant. For example, in the report shown in Figure 4-12 you can select one of the classes and view the profiles and access level for that user. They are most useful for investigative analysis, such as when a user is showing up as accessing a resource that they should not be able to. In this case you could use the zSecure Admin reports to discover why and correct the user access rights.

Full details of the reports can be found in Section 2.12, "RA.3 REPORTS - Reports with profiles and resources," in the *IBM Tivoli zSecure Suite: Admin and Audit for RACF*, LC23-6554.

Audit reporting with zSecure Audit

The main function of zSecure Audit is audit reporting of the z/OS environment (and z/VM if you have the Tivoli zSecure Manager for RACF z/VM). An audit can report on both the current system state and on historical information. It uses a combination of RACF data sets (active or unloaded copies) and the current system information held in the CKFREEZE data set for audit reporting against the current system state, and SMF for historical audit reporting. Many of the zSecure Audit reports are similar to reports available from the DSMON program that is shipped with RACF.

Unlike the zSecure Admin reporting, the zSecure Audit reporting is performed against a security policy. This may be the default supplied one, which is based on the common vulnerabilities of z/OS, or it may be a custom one based on a specific customers policy. You can also select to audit against three levels of the old US Department of Defense (DoD) *orange book* (Trusted Computer System Evaluation Criteria); C1, C2, and B1.

All audit concerns associate a priority number indicating the severity of the concerns identified. Values in the range 0 to 10 are informational. Values from 10 to 20 indicate that a review is desired, if time permits. Values from 20 to 40 indicate that a review is required. Values 40 and higher indicate a serious exposure.

The audit features are found under option AU in the zSecure Main menu, as shown in Example 4-1.

Example 4-1 zSecure Admin+Audit for RACF - main menu, audit options

```
zSecure Admin+Audit for RACF - Main menu
 Option ===>
                                                                More:     +
 SE   Setup           Options and input data sets

 RA   RACF            RACF Administration

 AU   Audit           Audit security and system resources
   C    Change track    Track changes to the system
   L    Libraries       Library status and update analysis
   S    Status          Status auditing of security and system tables/options
   V    Verify          Verify and cleanup security database

 EV   Events          Event reporting from SMF and other logs

 CO   Commands        Run commands from library

 IN   Information     Information and documentation

 LO   Local           Locally defined options
```

The four audit options are:

C (Change track) Change tracking allows the creation of a verified base of sensitive RACF profiles, attributes, and other system settings and reports any changes to this verified base.

L (Libraries) Admin and audit for RACF provides a library update report that is used to find and display changes to members of partitioned data sets, as well as changes to a select few sequential data sets and VSAM data sets that you have designated as critical.

S (Status) Status auditing of security and system tables/options. These are the reports comparing the current system to the defined policy.

V (Verify) This combines profile data with resource descriptions, and reports on inconsistencies. VERIFY also generates commands to fix some inconsistencies.

The most useful audit reports are found under option AU.S (Status). See Figure 4-13.

```
                          zSecure Admin+Audit for RACF - Audit - Status
 Command ===>

 Enter / to select report categories
    MVS tables           MVS oriented tables (reads first part of CKFREEZE)
    MVS extended         MVS oriented tables (reads whole CKFREEZE)
    RACF control         RACF oriented tables
    RACF user            User oriented RACF tables and reports
    RACF resource        Resource oriented RACF tables and reports




 Select options for reports:                                  Audit policy
    Select specific reports from selected categories          /  zSecure
    Concise (short) report                                       C1
    Output in print format                                       C2
    Run in background                                            B1
    Include audit concern overview, higher priorities only
```

Figure 4-13 zSecure Admin+Audit for RACF - audit status options

As shown in Figure 4-13, there are a number of report categories. The first two operate purely on MVS™ system information contained in the CKFREEZE data set. The RACF CONTROL category contains reports on the global system protection, classes, and database templates (all information derived from RACF). The RACF USER category contains (often detailed) reports on the RACF users. They generally generate a lot of output, and the TRUSTED report requires a full CKFREEZE read. The RACF RESOURCE category contains reports on the protection of resources (that is, data sets, UNIX files, and so on), resource profile audit concerns, and database resource usage (that is, size).

There are options to control the report output, including the option to run in the background, output in print format (versus ISPF format), and to include an audit concern overview. There is an overview report for each of the five report categories shown in Figure 4-13.

Example 4-2 shows the audit concerns overview report for the RACF control report category.

Example 4-2 zSecure Audit - audit concern overview report

```
Audit concern overview by priority (higher priorities only)      Line 1 of 134
 Command ===>                                                   Scroll===> PAGE
                                                   29 Nov 2007 19:23
    Pri Complex  Syst Area Key         Audit concern
     35 SC60     SC60 SETR PROTECTAL N The security system is not even invoked f
     30 SC60     SC60 SETR BATCHALLR N Allowing unidentified batch work makes ha
     30 SC60     SC60 SETR REVOKE    N Too many password violations allowed
     25 SC60     SC60 SETR TAPEVOL   N Tape volumes are unprotected / not C1 com
     23 SC60     SC60 CLAS APPCSERV    No protection against hackers masqueradin
     22 SC60     SC60 CLAS DEVICES     Devices like ESCON directors, FEPs, local
     20 SC60     SC60 CLAS data set    Profile changes in class are not audited,
     20 SC60     SC60 CLAS TEMPDSN     Temporary datasets resident after failure
     19 SC60     SC60 CLAS OPERCMDS    Profile changes in class are not audited
     18 SC60     SC60 CLAS VTAMAPPL    Profile changes in class are not audited
     16 SC60     SC60 CLAS FACILITY    Profile changes in class are not audited
```

```
 15 SC60     SC60 SETR AUDIT_GRO N Profile changes in GROUP class are not au
 15 SC60     SC60 SETR AUDIT_USE N Profile changes in USER class are not aud
 15 SC60     SC60 SETR ERASEONSC N Disk scavenging threat not countered / no
 15 SC60     SC60 SETR HISTORY   N Users can use same passwords over and ove
 15 SC60     SC60 SETR INACTIVE  N Apparently unused userids increase risk o
 12 SC60     SC60 CLAS JESSPOOL    Spooled output not subject to RACF contro
```

This report shows all of the high-priority audit concerns, sorted highest concern to lowest concern with information about the concern, including text describing the concern. You can scroll right to see the full concern text or select the specific audit concern to see all the details on one page.

Example 4-3 zSecure Audit - audit concern detail report

```
Audit concern overview by priority (higher priorities only)       Line 1 of 13
 Command ===>                                                   Scroll===> PAGE
                                                   29 Nov 2007 19:23

 System
 Complex name                  SC60
 System name                   SC60

 Audit concern
 Relative audit priority       35
 Audit concern                 The security system is not even invoked for each
 Audit concern                 data set/ not C2 compliant

 SETROPTS setting
 Parameter name                PROTECTALL
 Parameter value               No
 ****************************** Bottom of Data ******************************
```

In this case the report says that the PROTECTALL setting is set to no, which is considered worth reviewing and may be a system exposure (priority = 35).

As the reports are, by default, presented as ISPF panels, there may be options available to review more information or correct the concern. For example, on many reports you can view the RACF user, group, or resource definition from within the reports. On others you can submit a job to correct the condition.

There are many other useful reports available in zSecure Audit under the AU.S menu option. For a full list and more details see Chapter 3, “RACF Admin Guide” in the *IBM Tivoli zSecure Suite: Admin and Audit for RACF*, LC23-6554.

SMF analysis reporting with zSecure Audit

The zSecure Main menu also has the EV (Events) option (see Figure 4-14 on page 81), which contains all of the SMF analysis reports. The SMF displays can work with the live SMF data sets or with sequential SMF data that has been produced by IBM IFASMFDP program. IBM Tivoli zSecure Audit for RACF supports about 100 different SMF record types. For performance reasons, the SMF records might be filtered, but zSecure recommends that at least the following SMF records are used (Table 4-1).

Table 4-1 SMF records for zSecure Audit

Record type	Description
14	INPUT or RDBACK Data Set Activity
15	OUTPUT, UPDATE, INOUT, or OUTIN Data Set Activity
17	Scratch Data Set Status
18	Rename Data Set Status
30	Common Address Space Work
60	VSAM Volume Data Set Updated
61	ICF Define Activity
62	VSAM Component or Cluster Opened
63	VSAM Catalog Entry Defined
64	VSAM Component or Cluster Status
65	ICF Delete Activity
66	ICF Alter Activity
67	VSAM Catalog Entry Delete
68	VSAM Catalog Entry Renamed
69	VSAM Data Space, Defined, Extended or Deleted
80	RACF Processing
81	RACF Initialization
83	RACF Processing Record for Auditing Data Sets
90	System Status
92	UNIX Hierarchical File System
102	DB2 Performance and Audit
109	Firewall
118	TCP/IP Telnet and FTP
119	TCP UDP and IP
120	WebSphere Application Server

```
                          zSecure Admin+Audit for RACF - Main menu
Option ===>
                                                                    More:     +
SE   Setup           Options and input data sets

RA   RACF            RACF Administration

AU   Audit           Audit security and system resources

EV   Events          Event reporting from SMF and other logs
  U    User            User events from SMF
  G    Group           Group events from SMF
  D    Data set        Data set events from SMF
  R    Resource        General resource events from SMF
  F    Filesystem      Unix filesystem events from SMF and other logs
  I    IP              IP events from SMF and other logs
  1    SMF reports     Predefined analysis reports
  2    RACF events     RACF logging for specific events
  4    DB2             DB2 events from SMF
  C    Custom          Custom report
```

Figure 4-14 zSecure Admin+Audit for RACF - main menu, events options

The options available for SMF event reporting are:

U USER Select and list SMF records concerning specific users.

G GROUP Select and list SMF records concerning specific groups.

D DATA SET Select and list SMF records concerning specific data sets.

R RESOURCE Select and list SMF records concerning specific resources.

F FILESYSTEM Select and list log records (SMF and others) concerning UNIX System Services files.

I IP Select and list log records (SMF and others) concerning IP addresses.

1 SMF REPORTS Run predefined analysis reports.

2 RACF EVENTS Select and list RACF processing records to trace specific event types, such as access violations or use of RACF commands.

4 DB2 Select and list SMF records concerning DB2.

C CUSTOM Specify custom queries that do not fit the standard criteria of the other options. This allows you to create custom report displays and to create summaries. You can also make use of the predefined report layouts that are used by the other menu options.

The other input to some of these audit reports is a special collect of DASD and system information that is built by a zSecure process. This collection can be run with an APF authorized option, which is needed to gather protected system data. This data is requested to provide reports about some of the system integrity and system configuration reports that auditors will need. With this information, reports can be generated about IPL parameters, SVCs, and other critical system resources.

Under the U (USER reports), one of the most important reports from the auditor's point of view is TRUSTUSR (an overview of all trusted users in your system). This report shows the

access level of users to sensitive resources, grouped by user. This report will also identify why the users are considered trusted and whether there are reasons to be concerned about these users.

As any auditor would expect, there are the standard SMF event reporting capabilities. So events that concern an APF authorized library can be reported on both from the resource itself or from the user who performed the action.

```
Description
  RACF ACCESS success for DAVIDED: (READ,READ) on data setCEE.SCEERUN

   Record identification
   Jobname + id:  DAVIDED  TSU20045
   SMF date/time: Thu 29 Nov 2007 20:38:29.97
   SMF system:    SC60       record type:  80   record no: CKR2SM01 1489

   Event identification
   RACF event description        Resource access (Success:Successful access)
   RACF event qualifier          0
   RACF descriptor for event     Success
   RACF reason for logging       User
   SAF authority used            Normal
   Access intent                 READ
   Access allowed                READ
```

Figure 4-15 Example of user: DAVIDED accessing the APF authorized library: CEE.SCEERUN

There is also an option within the event reporting where an auditor can track changes to sensitive RACF definitions and critical system resources. This is within the Change Tracking option, which allows an auditor to identify other RACF or system definitions and changes as sensitive if they have not already been marked sensitive by the zSecure product. Examples of things that the zSecure product marks sensitive are changes to:

- SYSTEM-WIDE RACF SPECIAL or OPERATIONS users
- Profiles that protect system-critical data sets such as the RACF data sets
- System configurations such as changes to the list of APF authorized data sets
- RACF CLASS DESCRIPTOR table

Custom reporting with zSecure Audit

Custom reports can also be run through zSecure. This is one of the most powerful auditing tools in the zSecure product. These can be run in batch or ISPF. They can be written completely the CARLa programming language or in a simplified way using the custom command ISPF panels, found under option EV.C in the zSecure main menu.

Below are two simple examples of using CARLa, the zSecure programming language, to customize a few reports to identify all APF authorized libraries and certain activities against them. These are simple examples being run on an educational system. More examples are presented in the *IBM Tivoli zSecure Suite: Admin and Audit for RACF,* LC23-6554, and the *IBM Tivoli zSecure Audit for RACF*, LC23-6553.

In the first example, a report is desired of all the update and higher accesses on an APF authorized library and why the users were allowed to have access. The CARLa code is written into a member of a PDS and then can be driven by JCL or via an ISPF panel if this is a standard report that is to be run on a regular basis.

```
suppress ckfreeze
newlist type=SMF name=smfdata,
  Title="zSecure Audit Updates to Datasets  "
 select event=access dsn=CEE.SCEERUN   intent>READ
  sortlist dsn(25) userid userid:name date(5) ,
  intent(1) | '/' | access(1) reason(explode,15)
```

Figure 4-16 CARLa to list a data set and its accesses

This is a simple set of CARLa code that indicates that only SMF data is to be used. From the SMF data, the event records for the data set, CEE.SCEERUN, have been identified as the APF authorized library that we want to investigate. The report lists all the access to this data set, and identifies the user who accessed the data set and why they were allowed access. Most, if not all, of this data can be obtained from standard zSecure Audit reports, but this is an example of customizing audit reports with CARLa coding.

```
zSecure Audit user Access to data sets

data set                    User     Name          Date  I A Reason code
CEE.SCEERUN                 JJONES   Jack Jones    4Dec  U/C User
CEE.SCEERUN               MHAHN    Mark Hahn     4Dec  U/U Resource
CEE.SCEERUN                 TOMHAC   Tom Hackett   4Dec  A/A Special
CEE.SCEERUN                 MCONWAY  Monique Conway 4Dec  U/N Special
CEE.SCEERUN                 TOMHAC   Tom Hackett   4Dec  U/A Special
```

Figure 4-17 Output from our sample code

Figure 4-18 on page 84 is the output from the first CARLa example. It lists the data set name, the user ID and the associated user's name, the date, the intended access (I), the access allowed (A), and the reason why this audit record was recorded. *Special* indicates that the user is either SPECIAL or OPERATIONS. *User* indicates that the user has the UAUDIT set on their user ID. *Resource* indicates that the audit is either because of the AUDIT option on the resource profile or from the RACHECK code.

The second example is more interesting and complex (Figure 4-18). It involves two stages of CARLa code. This final output will be a list of APF authorized data sets. It will identify all the data sets that do not have a RACF profile. And for those data sets that do have a RACF profile, the output will indicate the UACC level, the audit level for successful access, the audit level for failures, and the profile that is used to protect this data set.

```
print dd=cnrout nopage

newlist type=system outlim=1
  sortlist ,
    "newlist type=system outlim=1, " /,
    "    title='APF data setnames and RACF profile info'" /,
    "  sortlist 'APF data setname'(44) 'UACC    ID(*)   S/F Profile'" /

/* second stage of CARLa code */

newlist type=sensdsn name=apfdsns
  select apf=yes or apflist=yes
  sortlist ,
    "newlist type=racf nopage " ,
    "    emptylist='** No profile for" dsn(0) "'" /,
    "  define idstar(aclaccess,7,'ID(*)') subselect acl(id('*'))" /,
    "  s class=data setbestmatch='" | dsn(0) | "'"   /,
    "  sortlist '" | dsn | "'(44) uacc idstar audits(1) | '/' | " ,
    "    auditf(1) profile "  /
```

Figure 4-18 Example of 2 pass CARLa code

The first stage of CARLa code reads the CKFREEZE data set to identify the APF data sets. The second stage processes the output from stage 1 and uses the RACF information to provide all the profile information about the individual APF authorized data sets.

```
DSNAME
UACC         S/F    PROFILE
 CEE.SCEERUN                                    READ    R/R  CEE.**
** No profile for REXX.SEAGALT
** No profile for REXX.SFANLMD
 SYS1.CMDLIB                                    ALTER    /R  SYS1.*
 SYS1.CSSLIB                                    ALTER    /R  SYS1.*
 SYS1.DGTLLIB                                   ALTER    /R  SYS1.*
 SYS1.LINKLIB                                   ALTER    /R  SYS1.*
** No profile for SYS1.LOCAL.VTAMLIB
 SYS1.LPALIB                                    ALTER    /R  SYS1.*
```

Figure 4-19 Output from the 2 pass CARLa code

The customization of these audit reports can be a powerful tool in the hands of a knowledgeable auditor. The standard reports in zSecure Audit can provide a wealth of information about the z/OS system and RACF information. With the ability to customize these reports, the auditor can tailor reports to a particular situation or event for the z/OS environment.

Reporting with zSecure Alert

The zSecure Alert product is not a reporting tool. However, it is designed to raise an alert when alert conditions are met. These alert conditions define a violation of policy and can contribute to the overall security policy management.

Alert comes with an extensive set of predefined alerts, which include:

- Logon with emergency user ID
- Logon of a user ID with UID(0) (UNIX superuser)
- Highly authorized user revoked for password
- Non-OPERATIONS user accessed data set with OPERATIONS
- Data set added to APF list
- Global write specified when altering file access
- Global security countermeasure deactivated
- SMF data loss started

Based on security policy, these alerts can be enabled and modified, such as changing the severity. Custom alerts can also be defined using the CARLa language. For more information about zSecure Alert, see the *IBM Tivoli zSecure Alert User Reference Manual*, SC23-6547.

Compliance reporting with zSecure

Both zSecure Audit and zSecure Alert are reporting (or alerting) based on the policy defined to zSecure. This may be:

- The default zSecure policy based on known z/OS system exposures
- One of the US Department of Defense (DoD) Orange Book (Trusted Computing System Evaluation Criteria) standards (C1, C2, or B1)
- A custom policy

Note that the Orange Book classifications have been replaced by Common Criteria. There are no zSecure policies for Common Criteria. Also, there are no policy modules for the industry standards such as Sarbanes-Oxley or HIPAA. One could be developed using the existing conditions and custom reports built in CARLa, but this would be a significant exercise.

TCIM and TSIEM have compliance modules for the major industry standards and SMF records can be sent to TCIM via the TCIM z/OS Actuator.

5

UNIX/Linux auditing tools

This chapter covers aspects of Linux OS auditing using the Linux Audit daemon and related tools (referred to hereafter by the term *Linux Audit subsystem*) and describes the TAMOS component.

5.1 Linux auditing subsystem

The Linux Audit subsystem was designed to meet or exceed commonly used security standards such as the Common Criteria CAPP, LSPP, and RBACPP, as well as NISPOM, FISMA, PCI, and DCID 6/3. It has been certified to CAPP on assurance level 4 (EAL4) for Red Hat Enterprise Linux 4 and Novel SuSE Linux Enterprise Server 10.1 and to CAPP, RBACPP, and LSPP on EAL4 for Red Hat Enterprise Linux 5.

The Linux Audit daemon is shipped as part of Red Hat Enterprise Linux (release 4 with update 2 or later), Novel SuSE Linux Enterprise Server (release 10 or later), and many other Linux distributions. The Linux Audit subsystem is Open Source Software and can also be downloaded directly from:

http://people.redhat.com/sgrubb/audit

5.1.1 Architecture

From a high-level point of view, the Linux Audit Subsystem is composed of Kernel components and userland components. In the Kernel, auditing is done at the system call level, meaning that audit data can be collected every time a userland program asks the Kernel for an operation (open file, read, write, create network connection, and so on). In the userland, key security-critical applications (such as those used for authentication, authorization, and auditing) must use a programming interface (API) in order to request for auditing themselves. Those applications are called *trusted programs*.

The Linux Audit subsystem was designed with simplicity in mind to provide system call auditing while keeping as low of an overhead as possible, and avoiding duplicating functionality that is already provided by Linux Security Infrastructure (SELinux or AppArmor, for example). So that the Audit subsystem can work stand-alone, but for a more complete coverage, a set of trusted programs allied with a Linux Security Infrastructure is desirable.

The current Linux Audit subsystem can be detailed further more into components, as shown in Figure 5-1.

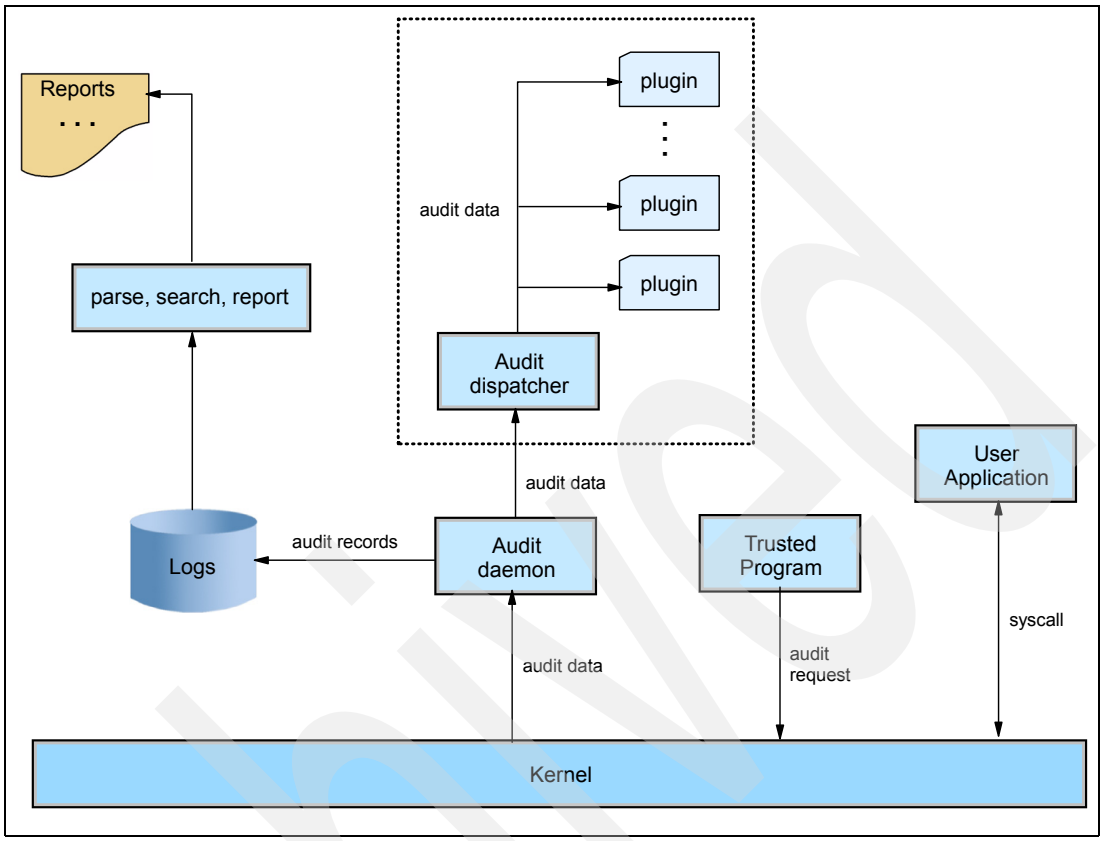


Figure 5-1 Linux Audit subsystem architecture

The components depicted in Figure 5-1 are described in the following sections.

Kernel infrastructure

The Linux Kernel is the core of the Linux operating system. It is responsible for providing a stable infrastructure layer between the hardware and the software (userland). Communication between useland programs and the Kernel are always made through system calls (syscalls for short). Effectively, every state change performed in the operating system can be tracked down to a set of syscalls.

The Linux Audit subsystem can audit syscalls through the presence of *hooks* in every syscall entry point. These hooks can collect audit data or not depending on a set of predefined *rules*. The set of audit data collected is highly dependant of the syscall itself (such as specific parameters or specific objects involved), but a default set of fields is always present, including the time stamp, event identifier, user ID, syscall number, syscall's first fours parameters and outcome (success or not), among others.

Trusted programs

Trusted programs is the term used after security-critical applications, usually responsible for authentication, changing user information or system configuration, or performing auditing. Trusted programs must call a library function (library is included in the Audit package) to perform auditing of the specific security function being performed.

In current Linux Enterprise versions, trusted programs include the following:

- login, su, sudo - used for identification, authentication
- OpenSSH server, gdm - remote and graphical logins
- useradd, usermod, userdel, passwd, chsh, chage - for changing user information
- groupadd, groupmod, groupdel, newgrp - for changing group information
- newrole - for changing security context information
- auditd, auditctl - for performing audit logging and control
- cups, vixie-cron, amtu, pam-modules - other security-related activity

Audit daemon

The audit daemon (also known by its executable name *auditd*) is the userland program responsible for gathering the audit data as provided by the Kernel and writing them into a log file. The auditd can also rotate the log files and perform actions when there are errors writing to the log or the disk is full (like suspending the logging, sending an e-mail, or putting the system in single-user mode).

Audit dispatcher and plug-ins

The audit dispatcher (executable name is *audispd*) provides a framework for multiplexing incoming audit events in real-time and forwarding them to configured audit plug-ins.

An audit plug-in can be designed to perform actions based on real-time audit data, like forwarding events to other systems, inputting events into IDS or IPS systems, or performing remote logging.

IBM recently developed an audit plug-in for remote z/OS RACF auditing that can be used to consolidate Linux Audit data into SMF records in a remote z/OS system. This plug-in is discussed further in “The z/OS Remote-Services Audit dispatcher plug-in” on page 96.

Logs

Logs are written by the audit daemon, and the log file is usually located in the /var/log/audit directory. The current (active) audit log filename is audit.log, and rotated log files are named audit.log.*n*, where *n* is the number of logs kept (lowest *n* means newer).

The Linux Audit log format

Despite the Linux Audit log being pure text, it is not advisable to rely on its format for reading *raw* audit data (see “Parse, search, and report infrastructure” on page 91 for more information about accessing audit data), but it is important to keep in mind the fundamental concepts presented below.

The Linux Audit subsystem organizes audit data in a three-level hierarchical structure, as follows:

1. The basic construction block for the Audit data structure is the audit field, a *key*=value pair that effectively carries information about the parameters of the operation being audited, where *key* is the parameter name specifying the type of data being audited. For example:
   - msg=audit(1193086796.415:6816) - time stamp, milliseconds, and event serial number
   - pid=22300 - operation audited was issued by process with ID 22300
   - uid=0 - operation audited was issued by user with ID 0
   - auid=500 - operation audited was issued by user with original ID 500
   - terminal=ssh - operation audited was performed in an ssh terminal
   - res=success - operation audited resulted in success
2. A set of audit fields describing information logically grouped for an audited operation forms an audit record. The audit daemon logs them one per line, and they are always started by a *type*= field describing the audit record type, followed by a *msg*= field with the time

stamp. Which fields are present in each record and their meanings is greatly dependant on the audit record type. Example 5-1 shows three audit records: a DAEMON_START record showing that the audit daemon started, a failed USER_LOGIN, and a successful USER_LOGIN, all using remote logins via ssh.

Example 5-1 Audit records sample

```
type=DAEMON_START msg=audit(1193086796.538:137): auditd start, ver=1.6.2,
format=raw, auid=0 pid=22357 res=success, auditd pid=22357
type=USER_LOGIN msg=audit(1193086815.055:2401): user pid=22368 uid=0 auid=0
msg='acct=root: exe="/usr/sbin/sshd" (hostname=?, addr=9.57.139.29, terminal=sshd
res=failed)'
type=USER_LOGIN msg=audit(1193086817.235:2408): user pid=22370 uid=0 auid=0
msg='uid=0: exe="/usr/sbin/sshd" (hostname=klausk.pok.ibm.com, addr=9.57.139.29,
terminal=/dev/pts/2 res=success)'
```

3. Finally, an audit event is a set of records logically grouped to describe an operation. When the userspace asks the Kernel for an operation, any auditable event involved generates an audit record that shares a unique event identifier (time stamp, milliseconds, and event serial) through all records within the same event. Example 5-2 shows a single audit event composed of two audit records (note how the msg=audit() fields are equal), describing an Ethernet device that has entered in promiscuous mode. The first record brings the operation itself, while the second record shows the associated system call.

Example 5-2 Single audit event sample

```
type=ANOM_PROMISCUOUS msg=audit(1193086853.185:2410): dev=eth0 prom=256 old_prom=0
auid=0
type=SYSCALL msg=audit(1193086853.185:2410): arch=80000016 syscall=102 success=yes
exit=0 a0=e a1=3ffff88a2b8 a2=1 a3=3ffff88a5a8 items=0 ppid=22300 pid=22400 auid=0
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts1 comm="tcpdump"
exe="/usr/sbin/tcpdump" key=(null)
```

Note that when an excursion to the Kernel is requested by a syscall, different parts of the Kernel involved in the operation may provide supplemental records. An SELinux Mandatory Access Control (AVC) denial to open a file may be audited, for example, by three different records in the same event: an Access Vector Cache (AVC) denial record, with associated SYSCALL record showing the open() syscall being used, together with a PATH record pointing to the file actually involved.

Also note that a single record may not contain the complete information to audit an operation. A PATH type record will not bring information like user ID or host name. That is already present in the SYSCALL record associated, and the association itself is done via the shared event identifier.

Parse, search, and report infrastructure

In order to enable access to audit data stored in the logs, the Linux Audit subsystem provides a set of tools and libraries to present a standard, stable interface for searching, reporting, and parsing events.

The tools are aimed at system administrators and auditors wanting to use the audit data to report on compliance, perform forensics analysis, or simply for house-keeping. The two tools shipped with the Linux Audit subsystem are:

- ausearch - a command-line tool to query audit daemon logs, where the user can specify different search criteria and use the output in other tools or back to the ausearch tool itself. Ausearch can also interpret the meaning of certain numeric fields and present them in a

more human-readable way (that is, convert userid=500 to userid=klausk, or syscall=2 to syscall=close).

- aureport - a command-line tool to produce summary reports based on audit daemon logs. With aureport, the user can query reports about authentication, files, account modifications, mandatory access control (MAC) events, users, and nodes, among others. An event summarized by aureport can be further detailed by using ausearch on this specific event.

Besides the tools aimed at auditors and system administrators, the Linux Audit subsystem provides an Application Programming Interface (API) in the form of a C library or a Python module for use by developers or system administrators wanting to build their own customized tools. This subject is not covered in this book. For more information refer to the Linux Audit subsystem manual pages.

5.1.2 Defining and enforcing policy

In the Linux Audit subsystem, there are basically three different entities types that can request for auditing:

- A *Trusted program*, using the audit API, may generate audit records for the specific operation being processed. For example, the OpenSSH server (and related PAM modules) will generate pertinent audit records whenever a user tries to log into the system using ssh—be it successful or not. The system administrator often does not have control over what is being logged because the audit request is usually hard-coded within the trusted program itself, but it is safe to just keep in mind that every security-pertinent operation in the platform level will be logged.
- The enabled *Linux Security Module* (if any), such as SELinux or AppArmor, may generate audit records for failed or successful operations, depending on the specific policy in place. For example, he Red Hat 5 *targeted* policy instructs SELinux to log audit records whenever the HTTP server process tries to create an outbound network connection. In the SELinux case, the system administrator can control the situations where logging is enforced or suppressed by redefining the SELinux policy itself: either through the definition of a policy module or, in specific cases, through manipulation of the SELinux booleans. Linux Security Module specifics are not covered in this book. For more information, refer to the SELinux (Red Hat) or AppArmor (Novel SuSE) documentation.
- A set of *Audit rules* can be defined in the audit daemon configuration or at run time, instructing the Linux Audit subsystem to enforce logging. An audit rule can perform logging on any system call (syscall) or by tracking access to any *existing* file or directory in the system (this is called a *watch*). The audit rules are explained below in more detail.

Defining audit rules

Defining what rules are needed in order to audit a given situation can be a complex task. In the Linux Audit subsystem context this task usually requires some knowledge of the system architecture and sometimes the application architecture as well. This is because every situation requiring auditing needs to be mapped in terms of *syscalls* usage, files access, or both (not counting situations like authentication, where that is already covered by the use of trusted programs).

Audit rules are defined either by a plain-text configuration file (usually placed at /etc/audit/audit.rules) that is read and parsed upon the audit daemon initialization, or by using the **`auditctl`** command to create rules at run rime. (Note though that these are not persistent across reboots. For persistent rules the system administrator should use the configuration file instead.)

The format for defining rules is the same for both defining the configuration file and the command-line version. In the configuration file, rules are added one per line, in the same format that they would take if they were passed as a parameter for the **auditctl** command. Example 5-3 shows a sample /etc/audit/audit.rules file, while Example 5-4 shows the same set of rules added through the **auditctl** command.

Example 5-3 Audit rules sample with audit.rules

```
## This is /etc/audit/audit.rules file
## Blank lines and lines starting with the '#' signal are ignored

# Remove any existing rules:
-D

# Audit write or attribute chage access to login and password database:
-w /etc/passwd -p wa
-w /etc/shadow -p wa

# Audit any directory creation or removal
-a entry,always -S mkdir -S mkdirat -S rmdir
```

Example 5-4 Audit rules sample with auditctl

```
[prompt] auditctl -D
[prompt] auditctl -w /etc/passwd -p wa
[prompt] auditctl -w /etc/shadow -p wa
[prompt] auditctl -a entry,always -S mkdir -S mkdirat -S rmdir
```

The Audit rules format is summarized below in a simplistic approach (for more information, refer to the auditctl manual page in your Linux installation) and divided in two categories:

- Rules involving file system watches - A *file watch* is an internal Linux Audit subsystem mechanism designed to track every access to its inode, meaning that the file or directory must exist to have a watch associated. If a file or directory is watched, every syscall dealing with this file or directory entry is audited, excluding the read and write syscalls (excluded by default to not overwhelm the logs). If the file name is changed or removed and a new file is created in its place, the Linux Audit Subsystem will automatically update its internal structures to track the new file. The watch rule is specified through the -w filename construct, where filename must bring the full path to the desired filename to be watched and only one filename can be specified per rule. No wild cards are allowed. By default, every access is audited, but the system administrator can narrow matches by specifying a permissions filter with the -p *perm* construct, where *perm* is any combination of the following:
  - r - audit syscalls trying to read the file or directory
  - w - audit syscalls trying to write to the file or directory
  - x - audit syscalls trying to execute the file or directory
  - a - audit syscalls trying to change the file or directory attributes
- Rules involving syscalls - The Linux Audit subsystem can audit *any* syscall happening in the system with great granularity over where and when to audit. The general format to add syscall is using the *-a list,action -S syscall* construct, where:
  - *list* can be either *entry* or *exit*. The difference lies at the moment the Kernel checks whether the this syscall is to be audited. Checking at syscall's *entry* is done before any execution takes place (and thus some fields are not yet available for filtering, such as inode, exit value success), while at the syscall's *exit* the action already took place and all the fields are available.

– *action* can be either *always* or *never*. You can combine broader scope rules being filtered by smaller scoped and vice-versa by using *always* to instruct the Kernel to perform auditing on this rule and use *never* to instruct the Kernel to prevent auditing.

– *syscall* can be the *syscall name* or *number*. Every syscall has a name and a number associated with it. The system administrator can specify either one.

Note: If the system in question is a *bi-architecture* system (that is, it is capable of running in two different bit modes (say a 64-bit native mode and a 32-bit compatibility mode)), it is important to note that the Linux Kernel provides the compatibility layer by making two separate sets of syscalls available. The system administrator can specify which bit mode is to be audited adding a -F arch=b32 for 32-bit (or 31-bit, in the s390 case) mode or -F arch=b64 for 64-bit mode. If nothing is specified, the Kernel uses the native (usually, 64-bit) mode by default, without doing any further filtering. This could result in *ghost* syscall records being logged by applications running in different bit modes, but calling a syscall that has the same number (syscall numbers may differ between platforms and bit modes). The best practice in this case is to *always* specify the bit mode when adding rules to a bi-architecture system, even if that means having two rules for each syscall being audited (covering both bit modes). See Example 5-5.

Both watch and syscall audit rules can be complemented with filtering on the available record fields by appending one or more -F field=value statements to the rule. Note that more comparison operators are supported besides the equal sign (=). Also note that field availability for filtering may vary depending on whether the rule is a watch rule, a syscall at entry level, or a syscall at exit level. Refer to the auditctl manual page for a detailed description of the filter operators and availability.

Example 5-5 shows a more typical set of Linux Audit subsystem rules for a bi-architecture system.

Example 5-5 Typical Linux Audit subsystem rules

```
## /etc/audit/audit.rules file
##

# Remove any existing rules
-D

# 1 - Audit time changes
-w /etc/localtime -p wa -k TimeChange
-a entry,always -F arch=b32 -S settimeofday -S adjtimex -k TimeChange
-a entry,always -F arch=b64 -S settimeofday -S adjtimex -k TimeChange
-a entry,alwats -F arch=b32 -S clock_settime -k TimeChange
-a entry,alwats -F arch=b64 -S clock_settime -k TimeChange

# 2 - Track unsuccessful creation of files and directories
-a exit,always -F arch=b32 -S creat -S mkdir -F exit=-13 -k UnsuccNewObj
-a exit,always -F arch=b64 -S creat -S mkdir -F exit=-13 -k UnsuccNewObj
-a exit,always -F arch=b32 -S mkdirat -S mknodeat -F exit=-13 -k UnsuccNewObj
-a exit,always -F arch=b64 -S mkdirat -S mknodeat -F exit=-13 -k UnsuccNewObj
-a exit,always -F arch=b32 -S link -S symlink -F exit=-13 -k UnsuccNewObj
-a exit,always -F arch=b64 -S link -S symlink -F exit=-13 -k UnsuccNewObj
-a exit,always -F arch=b32 -S linkat -S symlinkat -F exit=-13 -k UnsuccNewObj
-a exit,always -F arch=b64 -S linkat -S symlinkat -F exit=-13 -k UnsuccNewObj

# 3 - Audit for HTTP server failures in opening files
```

```
-a exit,always -F arch=b32 -S open -F uid=apache -F success=0 -k ApacheOpen
-a exit,always -F arch=b64 -S open -F uid=apache -F success=0 -k ApacheOpen
-a exit,always -F arch=b32 -S openat -F uid=apache -F success=0 -k ApacheOpen
-a exit,always -F arch=b64 -S openat -F uid=apache -F success=0 -k ApacheOpen
```

Note how the example above uses the *-k keyword* statement to identify the records pertaining to a rule set. Audit records generated using a certain keyword string can be easily identified by issuing the *ausearch -k* string.

The first set of rules, aimed at tracking time changes to the system, is watching for the /etc/localtime file writes or attribute changes as well as any call (successful or not) to the Linux syscalls responsible for changing the system time.

The second set of rules is aiming for tracking permission failures (*exit=-13*, permission denied; see the errno manual page) in syscalls that create new objects in the file system.

The third set of rules aims at tracking failures in opening files within the HTTP server process. Since in this example the HTTP server runs under the apache user, a rule filtering any failed open or openat syscall (filter: success=0) is combined to a clause to catch only events caused by the apache user (filter: uid=apache). Filters in the same sentence are combined with a logical AND, while different sentences are combined with a logical OR.

Also note that the order in which the rules are entered is important. The Linux Audit subsystem evaluates rules from first to last, and event triggers on the first matching rule.

5.1.3 Audit logs capture and storage

The entity responsible for storing audit data and managing the logs in the Linux Audit subsystem is the Audit daemon. Logs are usually written to a file named audit.log in the /var/log/audit directory.

The daemon configuration is a plain-text file usually located at /etc/audit/auditd.conf with a format of one configuration keyword per line, followed by an equal sign, followed by the desired keyword value: keyword=value.

The ideal configuration can greatly depend on the environment and auditing policy in effect for each system. The most important configuration aspects (for more information, refer to the auditd.conf manual page) are:

- flush - The flush keyword determines what strategy to use for flushing audit data into disk. This parameter dictates the balance between performance (hold audit data in daemon's memory and do buffered writes) and audit record freshness in case of exceptions (do not hold audit data—write everything before moving on). If the policy instructs that no audit data can be lost in exceptional cases (system errors, power outage, and so on), this setting should be flush=data or flush=sync, instructing the daemon to keep audit data synchronized at all times.
- max_log_file, max_log_file_actions, and num_logs can control the maximum audit log size, what to do when this size is reached, and the number of logs to keep if the *rotate* strategy is chosen. Most auditing policies require the system administrators to keep logs for a certain amount of time (policies usually do not care for the data size). At the same time, system administrators probably want to keep the log files as the same size. In this case, the system administrator should set max_log_file=100 to set the maximum log file size to 10 0MB (for example), while setting num_logs=99 to keep the maximum number possible of logs and max_log_file_action=keep to keep previous logs in case the log file size limit is reached. The system administrator must schedule regular unloads of the audit

data to archive, to allow more data to come in and keep the log file count low. Having a high count can determine an event loss in a heavy-loaded system, because the deamon must rename all kept files before starting a new one.

- space_left and space_left_actions, admin_space_left and admin_space_left_action, and disk_full_action and disk_error_action can configure two levels of warning for running short on disk space (space_left and admin_space_left) and their actions (including sending e-mail, suspending collection, executing a command, among others). Actions for when the disk is finally full or a hardware error occurred while writing audit data are also configurable.

We usually recommend keeping the /var/log/audit directory mounted in a separate disk or partition so that available space and resource consumption can be more easily managed.

5.1.4 Audit records consolidation and standardization

The Linux Audit subsystem does not come with a built-in mechanism for audit data consolidation (in the terms that it will not perform remote logging directly). Despite that, the system administrator can always move the audit logs for archiving in another box. Problems with this approach are:

- The log format is unique to the Linux Audit subsystem, and people must not rely on its format for parsing directly (use the provided API instead).
- Audit data's numeric fields (like user IDs) cannot be easily interpreted outside the system in which they were generated.

Recent developments, though, enabled the Linux Audit subsystem to provide a plug-in infrastructure. This way, plug-ins can be built to implement remote logging or as audit data gateways to Intrusion Detection Systems (IDS) or Intrusion Prevention Systems (IPS).

In order to enable Linux audit data consolidation to a centralized system, IBM has developed a plug-in for the Linux Audit subsystem capable of forwarding events to a z/OS system with IBM Tivoli Directory Server (ITDS) installed—the Linux Audit subsystem RACFplug-in.

The z/OS Remote-Services Audit dispatcher plug-in

The z/OS Remote-Services Audit dispatcher plug-in is included with recent versions of the Linux Audit subsystem package (starting from audit-1.6.3 onwards). The plug-in is designed to forward Linux Audit events, as they happen, to the System Management Facility (SMF) located at a z/OS v1R8 (or later). To do such, the plug-in issues a remote audit request through ITDS for z/OS, that will cut SMF records type 83, subtype 4—one SMF record for each Linux Audit record processed.

The SMF record format is determined by the parameters ITDS uses to call the R_AUDITX SAF service, explained below:

- LinkValue - will carry the Linux Audit record's serial number. This is a unique number shared across records pertaining to the same event, so it can be used to link records of the same event again.
- Violation - always zero (false).
- Event Code - always number two (authorization).
- Event Code Qualifier - zero (success) if audit operation performed successfully, three (fail) if the audited operation performed unsuccessfully, or one (info) in the case that there is no information about the operation outcome (no res= or success= fields in the event).

- Class - the RACF CDT class, fixed to @LINUX. In order to be able to process Linux remote audit requests, the z/OS system programmer must make sure that a @LINUX RACF CDT class is configured. (Check the audisp-racf manual page for details.)
- Resource - will carry the Linux record event type (for example, SYSCALL, CWD, PATH, USER_ACCT, LOGIN, and so on; see Appendix A, “Linux Audit record description” on page 293, for a complete list). The z/OS system programer must make sure that at least a generic @LINUX profile is configured to audit these resources. The system programmer can selectively enforce logging or ignore these records based on the resource’s name. (Check the audisp-racf manual page for details.)
- LogString - will carry a string in the following format: remote audit request from *USER*. Linux (hostname.domain):*RECORD_TYPE*, where *USER* is the RACF user ID configured to perform remote auditing (in the plug-in configuration; see racf.conf manual page), *hostname.domain* is the Linux box’s host name, and *RECORD_TYPE* is the Linux Audit record type.
- DataFieldList - All the fields present in a Linux Audit record are formatted as a variable number of SMF relocates of type 114 (application-specific data), in a fieldname=value format. An additional relocate type 113 (date and time security event occurred) is placed carrying the time stamp of when the event occurred, following the Tue Sep 25 16:06:01 2007 format.

Once the Linux Audit events are in the SMF database, the system programmer can dump them to XML using a SMFUNLOAD job. The XML data can then be easily parsed to produce powerful reports. Example 5-6 shows a stripped-down XML dump with only one SYSCALL record. Note the Class field carrying @LINUX value, the ResName field with the Linux record type (LINUX), and all other fields distributed as relocates (d:dateTime and d:otherData). Also note how all fields contain human-readable fields for convenience.

Example 5-6 Extract of XML dump

```
<?xml version='1.0' encoding='ebcdic-cp-us' ?>
<securityEventLog xmlns='http://www.ibm.com/xmlns/zOS/IRRSchema'>

  <rdf:Description rdf:about=''
                   xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
                   xmlns:dc='http://purl.org/dc/elements/1.1/'>
    <dc:creator>z/OS Security Server RACF SMF Unload (HRF7740)</dc:creator>
    <dc:subject>RACF Security Event Log 2007-11-09 15:18:58</dc:subject>
    <dc:language>en</dc:language>
  </rdf:Description>

  <event>
    <eventType>*SAFAUTZ</eventType>
    <eventQual>SUCCESS</eventQual>
    <timeWritten>15:15:23.88</timeWritten>
    <dateWritten>2007-11-09</dateWritten>
    <systemSmfid>SC60</systemSmfid>
    <prodFmid>HIT7730</prodFmid>
    <prodName>LDAPICTX</prodName>
    <details xmlns:d="http://www.ibm.com/xmlns/zOS/SAFSchema">
      <violation>N</violation>
      <userNdfnd>N</userNdfnd>
      <userWarning>N</userWarning>
      <evtUserId>GLDSRV</evtUserId>
      <evtGrpId>SYS1</evtGrpId>
      <authNormal>N</authNormal>
```

```
<authSpecial>N</authSpecial>
<authOper>N</authOper>
<authAudit>N</authAudit>
<authExit>N</authExit>
<authFailsft>N</authFailsft>
<authBypass>N</authBypass>
<authTrusted>N</authTrusted>
<logClass>N</logClass>
<logUser>N</logUser>
<logSpecial>N</logSpecial>
<logAccess>Y</logAccess>
<logRacinit>N</logRacinit>
<logAlways>N</logAlways>
<logCmdviol>N</logCmdviol>
<logGlobal>N</logGlobal>
<termLevel>000</termLevel>
<backoutFail>N</backoutFail>
<profSame>N</profSame>
<jobName>TDSSRV1</jobName>
<readTime>18:06:25</readTime>
<readDate>2007-10-22</readDate>
<logLevel>N</logLevel>
<logLogopt>N</logLogopt>
<logSecl>N</logSecl>
<logCompatm>N</logCompatm>
<logApplaud>N</logApplaud>
<logVmevent>N</logVmevent>
<logNonomvs>N</logNonomvs>
<logOmvsnprv>N</logOmvsnprv>
<authOmvssu>N</authOmvssu>
<authOmvssys>N</authOmvssys>
<racfVersion>7740</racfVersion>
<srvrUserId>GLDSRV</srvrUserId>
<srvrGrpId>SYS1</srvrGrpId>
<prodId>LDAPICTX</prodId>
<logRauditx>N</logRauditx>
<resName>SYSCALL</resName>
<class>@LINUX</class>
<profileName>*</profileName>
<logstr>Remote audit request from AUDITX. Linux (linuz2z.itso.ibm.com):
type: SYSCALL</logstr>
<eventLink>0000000000000E415</eventLink>
<d:dateTime>Fri Nov  9 15:11:41 2007</d:dateTime>
<d:otherData>type=SYSCALL</d:otherData>
<d:otherData>arch=s390x</d:otherData>
<d:otherData>syscall=open</d:otherData>
<d:otherData>success=yes</d:otherData>
<d:otherData>exit=3</d:otherData>
<d:otherData>a0=800fbb00</d:otherData>
<d:otherData>a1=O_WRONLY|O_CREAT|O_TRUNC</d:otherData>
<d:otherData>a2=1b6</d:otherData>
<d:otherData>a3=0</d:otherData>
<d:otherData>items=1</d:otherData>
<d:otherData>ppid=14670</d:otherData>
<d:otherData>pid=14672</d:otherData>
```

```
      <d:otherData>auid=root</d:otherData>
      <d:otherData>uid=root</d:otherData>
      <d:otherData>gid=root</d:otherData>
      <d:otherData>euid=root</d:otherData>
      <d:otherData>suid=root</d:otherData>
      <d:otherData>fsuid=root</d:otherData>
      <d:otherData>egid=root</d:otherData>
      <d:otherData>sgid=root</d:otherData>
      <d:otherData>fsgid=root</d:otherData>
      <d:otherData>tty=pts1</d:otherData>
      <d:otherData>comm=bash</d:otherData>
      <d:otherData>exe=/bin/bash</d:otherData>
      <d:otherData>subj=root:system_r:unconfined_t:s0-s0:c0.c1023</d:otherData>
      <d:otherData>key=(null)</d:otherData>
    </details>
  </event>
</securityEventLog>
```

Having the Linux Audit records in SMF format can bring advantages like the easier handling of centralized audit data (using well-established SMF management procedures), possible added security by placing audit data out of the audited system (leaving it out of the *sphere of influence* of local system administration), and the possibility of having reports covering multiple systems. There are also some potential disadvantages, like the increased network load, increased resource usage by the SMF database, the possible audit data lost due to network outages, and the fact that there are still no preconfigured reports or tools to deal with this type of record. Section 5.1.5, “Reporting and compliance checks” on page 99, shows an example of how to report on Linux Audit data using zSecure Audit and the CARLa reporting language.

5.1.5 Reporting and compliance checks

With the Linux Audit subsystem, reporting effectively involves two steps of equal importance: having a set of rules that will audit the desired situation accordingly, and having the correct set of commands to produce the desired report.

In the subject of having a set of rules to audit the desired situation, usually the system administrator does not need to worry about operations being done through trusted programs (that is, authentication, login, user account modifications, security policy modifications, and so on) or outstanding security events that the system logs by default (that is, anomalies, MAC violation attempts, security label change or override, and so on). But the system administrator *does* need to have a custom set of rules to audit a specific operation or to detect changes to the system not being made through the use of trusted programs, and this might be a challenge for system administrators not very well-versed in what files are pertinent or what syscalls can be used to perform such tasks.

Example 5-7 shows a set of rules designed to audit changes to the local user and group authentication. Changes being made through the trusted programs (such as useradd, usermod, groupadd, and groupmod, among others) are audited by default, but the set of rules shown enable the system administrator to audit direct changes made to sensitive files, such as replacing them.

Example 5-7 Sample set of rules to audit changes for local user and group authentication

```
-w /etc/group -p wa -k AuthMod
-w /etc/passwd -p wa -k AuthMod
-w /etc/gshadow -p wa -k AuthMod
-w /etc/shadow -p wa -k AuthMod
-w /etc/security/opasswd -p wa -k AuthMod
```

Note the AuthMode key added to the rules, used to easily identify records generated by these rules when reporting.

Example 5-8 shows a account modifications for the current day, using the **aureport** command. Reports generated by **aureport** are summarized and do not carry any details. The system administrator can use the event number in another search using **ausearch** to inspect details about the operation (as seen in Example 5-9).

Example 5-8 Account modification sample

```
[root@linux2z residency]# aureport --interpret --start today --mods

Account Modifications Report
===========================================
# date time auid addr term exe success event
===========================================
2. 11/09/2007 17:44:34 root ? pts/1 /usr/bin/passwd yes 395
3. 11/09/2007 17:44:34 root ? pts/1 /usr/bin/passwd yes 396
[root@linux2z residency]#
```

Example 5-9 shows the details of events 395 and 396, though two separate calls to the **ausearch** command. Note that both events were triggered by the **/usr/bin/passwd** command in the same second, but the first PAM (Linux Pluggable Authentication Module) accounting the authorization token change (chauthtok) for user ID klausk, while the second is the **passwd** command itself auditing a password change for the same user ID. Also note that the user who actually requested the change was audited (uid=root and auid=root), as well as the operation's outcome (res=sucess).

Example 5-9 Sample for event 395 and 396

```
[root@linux2z residency]# ausearch --interpret --event 395 --start today
----
type=USER_CHAUTHTOK msg=audit(11/09/2007 17:44:34.501:395) : user pid=7865
uid=root auid=root subj=root:system_r:unconfined_t:s0-s0:c0.c1023 msg='PAM:
chauthtok acct=klausk : exe=/usr/bin/passwd (hostname=?, addr=?, terminal=pts/1
res=success)'
[root@linux2z residency]# ausearch --interpret --event 396 --start today
----
```

```
type=USER_CHAUTHTOK msg=audit(11/09/2007 17:44:34.501:396) : user pid=7865
uid=root auid=root subj=root:system_r:unconfined_t:s0-s0:c0.c1023 msg='op=change
password id=klausk exe=/usr/bin/passwd (hostname=?, addr=?, terminal=pts/1
res=success)'
[root@linux2z residency]#
```

Example 5-10 shows how the password change operation was logged by the set of rules in Example 5-7 on page 100. Note that the actual operation behind *pas*swd execution is a rename syscall, indicating that the file has been replaced. Also note that a second operation was logged—this one not using any of the trusted programs. The last event in the example (event serial 403) shows that user ID *root* used the command **/bin/cp** to overwrite the /etc/shadow file.

Example 5-10 Password change sample logging

```
[root@linux2z residency]# ausearch --interpret --start today --key AuthMod
----
type=PATH msg=audit(11/09/2007 17:44:34.501:394) : item=4 name=/etc/shadow
inode=357283 dev=5e:05 mode=file,400 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0
type=PATH msg=audit(11/09/2007 17:44:34.501:394) : item=3 name=/etc/shadow
inode=357377 dev=5e:05 mode=file,400 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0
type=PATH msg=audit(11/09/2007 17:44:34.501:394) : item=2 name=/etc/nshadow
inode=357283 dev=5e:05 mode=file,400 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0
type=PATH msg=audit(11/09/2007 17:44:34.501:394) : item=1 name=/etc/ inode=357281
dev=5e:05 mode=dir,755 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0
type=PATH msg=audit(11/09/2007 17:44:34.501:394) : item=0 name=/etc/ inode=357281
dev=5e:05 mode=dir,755 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0
type=CWD msg=audit(11/09/2007 17:44:34.501:394) :  cwd=/root/residency
type=SYSCALL msg=audit(11/09/2007 17:44:34.501:394) : arch=s390x syscall=rename
success=yes exit=0 a0=2000003753c a1=20000037530 a2=20000031fe4 a3=0 items=5
ppid=1838 pid=7865 auid=root uid=root gid=root euid=root suid=root fsuid=root
egid=root sgid=root fsgid=root tty=pts1 comm=passwd exe=/usr/bin/passwd
subj=root:system_r:unconfined_t:s0-s0:c0.c1023 key="AuthMod"
----
type=PATH msg=audit(11/09/2007 18:17:55.571:403) : item=0 name=/etc/shadow
inode=357283 dev=5e:05 mode=file,400 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0
type=CWD msg=audit(11/09/2007 18:17:55.571:403) :  cwd=/root/residency
type=SYSCALL msg=audit(11/09/2007 18:17:55.571:403) : arch=s390x syscall=open
success=yes exit=4 a0=3ffffc79c00 a1=201 a2=0 a3=3ffffc79c00 items=1 ppid=1838
pid=8032 auid=root uid=root gid=root euid=root suid=root fsuid=root egid=root
sgid=root fsgid=root tty=pts1 comm=cp exe=/bin/cp
subj=root:system_r:unconfined_t:s0-s0:c0.c1023 key="AuthMod"
[root@linux2z residency]#
```

Like almost all UNIX command-line programs, aureport and ausearch are intentionally kept simple in concept, but can be powerful when combined with other commonly found utilities like awk, perl, and sed, among others. The shell script found in Example 5-11 can be used to report all events involving account modifications, plus all events involving writes to sensitive authentication files, as defined by the rules in Example 5-7 on page 100.

Example 5-11 Shell script used to report events for account modifications and writes to sensitive files

```
#!/bin/bash
for i in $(aureport --start today --mods | \
           awk '/^[0-9]+\./{print $9}' |\
           sort | uniq)
do
  ausearch --interpret --start today --event $i
done
ausearch --interpret --start today --key AuthMod
```

Besides the account modification events, aureport can produce a variety of summarized logs including login events, Mandatory Access Control (MAC) and Access Vector Cache (AVC) messages, and events involving syscalls or file objects, among others. See Appendix A, “Linux Audit record description” on page 293, for more information about the relation between record types and reports.

ausearch can be used to search for events based on a variety of fields, including file names and commands involved, user ID, group ID, event type (called *message type* in the ausearch manual page), syscall, and so on.

For more examples about reporting and searching Linux Audit events, refer to 9.4, “Linux privileged (uid=0)” on page 235.

Reporting on Linux Audit events using zSecure Audit

The IBM Tivoli zSecure is a powerful suite designed to enable security management and reporting in z/OS. Included in the suite is the zSecure Audit and the CARLa definition language, two powerful reporting and data-extraction tools that can query the z/OS SMF database for specific data.

When using the RACF audit dispatcher plug-in, records are sent to the SMF database in the SMF type 83, subtype 4, but the record format itself is not sufficient to understand the Linux audit event, since it has its own meanings for each field.

It is possible to define reports that are specialized for these Linux events in SMF format using CARLa programs. Example 5-12 shows a basic example of how this can be achieved.

Example 5-12 Sample of SMF using CARLa

```
newlist type=smf
define type=smf linuxtod(30) as racf_section(113)
define type=smf linuxOther(50) as racf_section(114)
define type=smf link(8) as racf_section(9)
select type=(83(4))
sortlist userid link  resource(0) linuxtod linuxOther
```

Example 5-12 on page 102 shows a simple report definition with the event time stamp (linuxtod defined as racf_section 113) and the relevant fields (linuxOther defined as racf_section 114), and also reports the Linux record type (resource field in SMF). A sample report is shown in Example 5-13.

Example 5-13 Sample report

```
GLDSRV   @LINUX   CRED_ACQ Wed Oct 24 13:06:49 2007       node=linux2z.itso.ibm.
                                                  type=CRED_ACQ
                                                  pid=30268
                                                  uid=root
                                                  auid=root
                                                  acct=root
                                                  exe="/usr/sbin/sshd"
                                                  hostname=9.12.5.200
                                                  addr=9.12.5.200
                                                  terminal=ssh
                                                  res=success
GLDSRV   @LINUX   CRED_ACQ Wed Oct 24 13:11:55 2007       node=linux2z.itso.ibm.
                                                  type=CRED_ACQ
                                                  pid=30568
                                                  uid=root
                                                  auid=root
                                                  acct=root
                                                  exe="/usr/sbin/sshd"
                                                  hostname=9.12.5.200
                                                  addr=9.12.5.200
                                                  terminal=ssh
                                                  res=success
GLDSRV   @LINUX   CRED_ACQ Wed Oct 24 13:24:22 2007       node=linux2z.itso.ibm.
                                                  type=CRED_ACQ
                                                  pid=30882
                                                  uid=root
                                                  auid=root
                                                  acct=root
                                                  exe="/usr/sbin/sshd"
                                                  hostname=9.12.5.200
                                                  addr=9.12.5.200
                                                  terminal=ssh
                                                  res=success
```

The report shown in Example 5-13 serves as a baseline for building more powerful reports using CARLa. For more information refer to IBM zSecure documentation and Appendix A, “Linux Audit record description” on page 293.

5.2 Tivoli Access Manager for Operating Systems (TAMOS)

This section provides an overview of TAMOS, looking at its features and high-level architecture.

UNIX has several inherent problems with security when it is examined from an enterprise point of view. These include:

- There is no inherent *enterprise-strength* security infrastructure in the earlier UNIX and Linux operating systems. Each vendor has addressed this issue by implementing their own forms of enhanced or hardened UNIX security, such as the Enhanced Security Option in AIX® or Security-Enhanced Linux (SELinux). While these implementations often resolve the issue of low security, they lead to different processes and tools to implement policy on disparate systems.
- Another problem centers around the concept of group users. In UNIX, when a group user account is used, such as *root*, all auditing is based on the group user account, not an individual user account. By its nature, this makes auditing events on the host system extremely difficult.
- Many functions in UNIX/Linux require root (or UID=0) authority to perform day-to-day tasks, so in order to perform normal functions, there is significant risk of malicious or accidental damage to the system due to the user having too much access. No access control rules apply to root, which means that not only can they access any file, they can wipe their tracks from any audit logs.

TAMOS addresses these concerns by providing a layer of authorization policy enforcement in addition to that provided by a native UNIX/Linux operating system. It applies fine-grained access controls that restrict or permit access to key system resources.

Controls are based on user identity, group membership, the type of operation, time of the day or day of the week, and the accessing application. An administrator can control access to specific file resources, login and network services, and changes of identity. These controls can also be used to manage the execution of administrative procedures and to limit administrative capabilities on a per-user basis.

When protected resources are accessed, TAMOS performs an authorization check based on the accessing user's identity, the action, and the resource's access controls to determine whether access should be permitted or denied. This means that even if the user assumes the root identity (for example, by using su), but the TAMOS policy does not allow that user to access that resource, then the access will be denied by TAMOS before it gets to the OS.

In addition to authorization policy enforcement, mechanisms are provided to verify defined policy and audit authorization decisions. This auditing can be set on a global basis, against specific users or against specific resources. For example, you could audit all access by root (or any UID=0 users) and against a set of high-risk files.

Access controls are stored in a policy database that is centrally maintained in the IBM Tivoli Access Manager (TAM) environment. As TAMOS on multiple servers can share the same policy database, common policy can be built centrally and distributed. This is better than creating policy for each system and manually applying it. You can define a set of policies based on different attributes of the systems and join them to suit specific machines. For example, you may have a set of OS-specific policies (for example, one for SUSE Linux, one for AIX 5.3, one for Solaris™ 9), another set for middleware (for example, one for WebSphere Application Server, one for Tivoli Directory Server, and one for DB2), and another set for applications (for example, one for the human resources software, one for the accounts receivable software, and one for the POS systems). If you have three SUSE Linux machines running WebSphere Application Server and the POS software, they can each share the same policy centrally built from three sets of definitions. This is easier to maintain than having a unique policy for each server that contains parallel definitions that must be kept in synch over time.

The UNIX/Linux user definitions are stored in a user registry that is also centrally maintained in the environment. TAMOS does not perform user authentication. It trusts the operating system authentication mechanism. However, it can apply additional login policy, such as checking for password expiration and enforcing a consistent password strength check.

The next section details how these functions are implemented.

5.2.1 TAMOS architecture

TAMOS is implemented as a series of UNIX/Linux daemons, the kernel extension, and some control files. It uses the TAM framework for policy management and the TAM user registry (such as an LDAP) for managing its users and groups. The high-level architecture for TAMOS is shown in Figure 5-2.

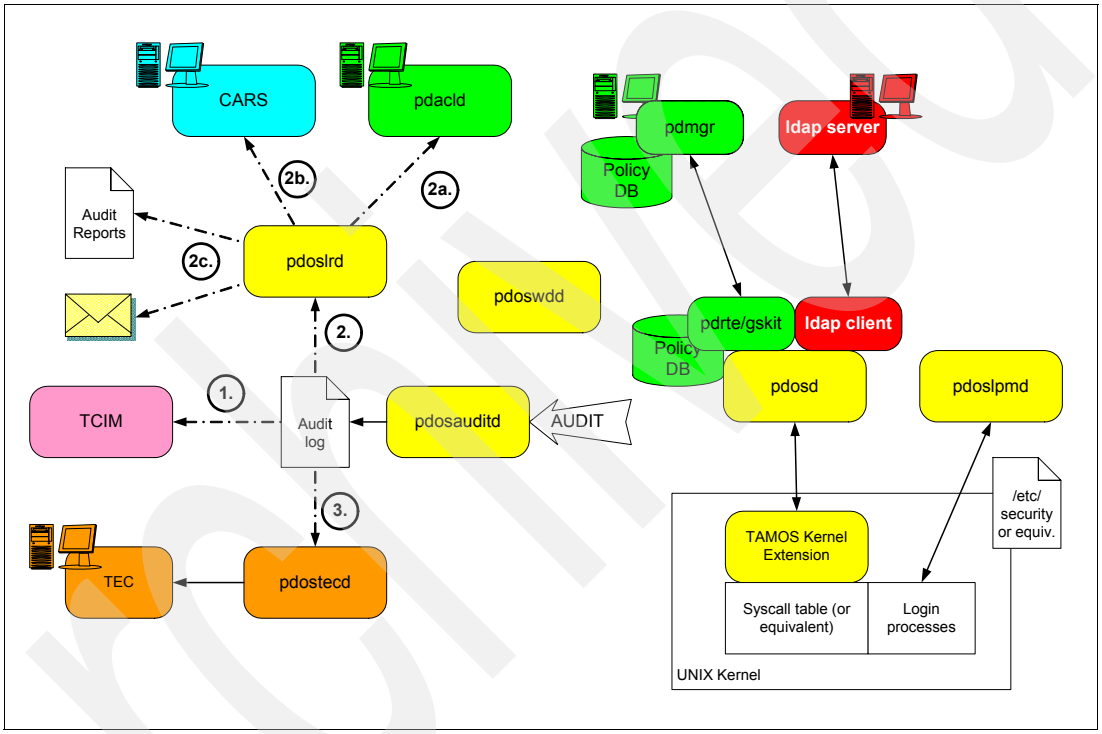


Figure 5-2 TAMOS components

The product has the following processes, or daemons:

pdosd This is the central authorization deamon for TAMOS. It is called by the TAMOS Kernel Extensions when a system call is made and performs authorization decisions based on the user credentials and the policy defined in the local copy of the policy database. It uses the TAM run time (pdrte) to receive copies of the policy database from the TAM policy server (pdmgr) and the ldap client to retrieve user credentials from the user repository.

pdoslpmd This is the login process management deamon and intercepts calls from the login processes and verifies login-related policy (such as password expiration and password strength rules).

pdoswdd This is the watchdog daemon. It monitors to make sure that the other daemons are running and if it detects that they are not running it restarts them. This is one of the TAMOS internal security measures.

pdosauditd This daemon captures the binary audit data generated by the other daemons (primarily pdosd) and generates audit logs. These audit logs can be viewed with the **`pdosaudview command`** (not shown), gathered by TCIM (1), distributed by the log router daemon (2), or gathered by the Tivoli Enterprise Console® (TEC) daemon (3).

pdoslrd This is the log router daemon that can route audit records on to a TAM authorization daemon for consolidation with other TAMOS system audit files (2a), the Tivoli Common Audit and Reporting Server (CARS) (2b), or other file formats such as SMTP (2c).

TAMOS is able to implement this security layer as it hooks into the system calls at the kernel level, as shown in Figure 5-3.

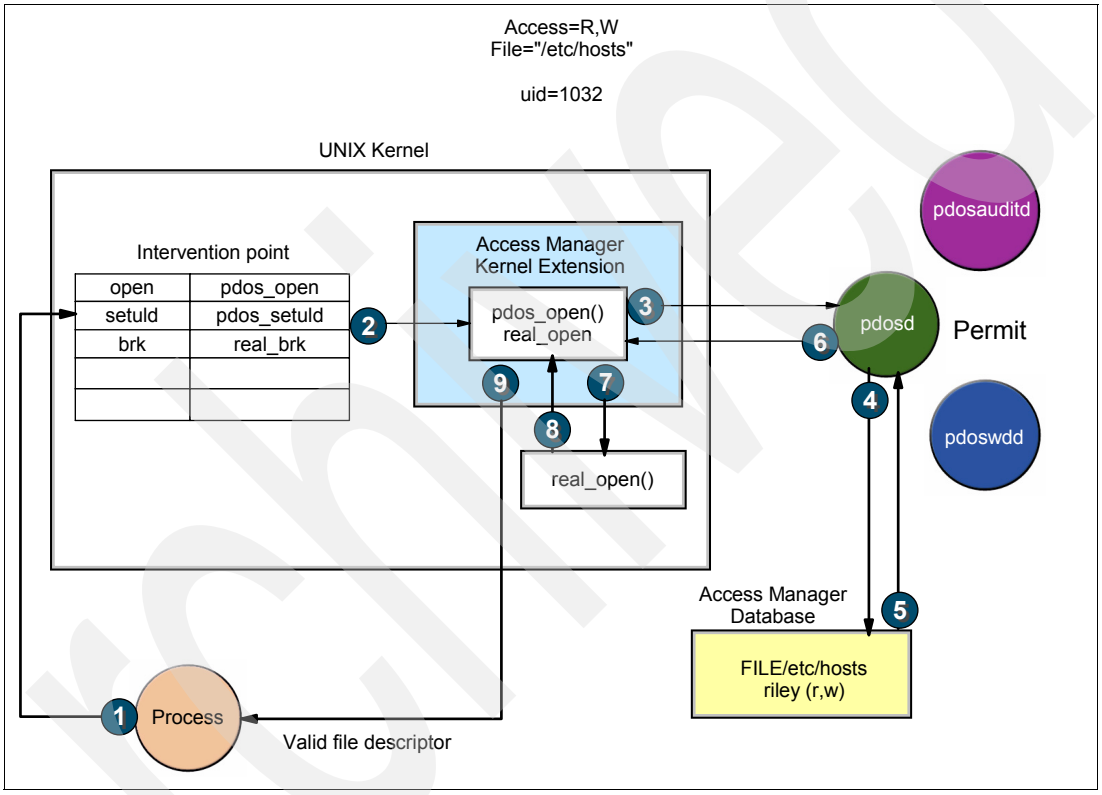


Figure 5-3 TAMOS kernel interception

Figure 5-3 shows the flow of a process attempting to access the /etc/hosts file with TAMOS installed. The steps are:

1. The process attempts to open the file, resulting in an open() call to the kernel for /etc/hosts.
2. The kernel intervention point (such as the syscall table) has been modified at startup so that the addresses for the kernel functions have been replaced by the equivalent TAMOS function. In this case, the open() is mapped to the pdos_open() function in the TAMOS Kernel Extension.
3. The TAMOS kernel extension calls the TAMOS authorization daemon (pdosd), passing it the identity of the user/process, the resource being accessed, and the type of access.
4. The pdosd deamon requests the relevant policy definitions from the policy database.
5. These are processed by the pdosd daemon, which in this case allows the file access.

6. This response is returned to the TAMOS kernel extension.
7. The kernel extension calls the real kernel open function (shown as real_open() in Figure 5-3 on page 106).
8. The real open() function returns a file descriptor.
9. This is passed back to the application.

This approach allows TAMOS to apply access control to all requests to the kernel and apply consistent auditing.

For further information about TAMOS, see *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-60144.

5.2.2 Policy, auditing, and compliance Aspects of TAMOS

This section discusses how TAMOS fits into a broad audit solution.

Policy definition and implementation

TAMOS implements policy through the centralized TAM policy framework. It can define user roles and map them to access rights that apply to UNIX/Linux systems. As the TAM policy database may be shared across various TAM-based products, such as TAM for e-business, TAM for Business Integration, and the WebSphere Application Server Access Manager JACC Provider, it may be possible to implement enterprise-wide roles that apply access rights across multiple systems to cross-system users.

Roles are implemented in the form of TAM groups. Users or groups are mapped to resources in the protected object space via *access control lists* (ACLs) and *protected object policies* (POPs). ACLs dictate what access a user or group has on a resource. POPs dictate a level of service on that access, such as time-of-day restrictions or auditing levels.

The policy implemented through TAMOS covers both resource access control and login policy. The types of policy that can be implemented include:

File Access to files, directories, soft links, hard links, and device files.

TCB Files can also be defined as part of a trusted computing base (TCB), which is monitored for changes and reacts when changes are detected (such as disabling all access to the file).

Network Access to connect to specific ports or services on the system, and access for this machine to connect to other machines in the network.

Surrogate Policy can be applied to all users or processes who perform any form of user switching. TAMOS also provides its own SUDO implementation that is part of the policy.

Login Control when and from where a user can log in to a system. For example, you could restrict root login to a specific terminal.

Password Policy relating to password strength and aging.

This policy can be defined using the TAM Web Portal Manager browser-based interface, or the **`pdadmin`** command-line utility. These are the same tools that would be used to define TAMeb or other TAM-related policy.

There are a set of pre-built policy definitions, calls Fast Start Policy Modules, available from the TAMOS support site:

http://www.ibm.com/software/sysmgmt/support/IBMTivoliAccessManagerforOperatingSystems.html

At the time of writing these modules provided policy definitions for DB2, UNIX, Oracle®, WebSphere Application Server, IBM HTTP Server (IHS), and TAMeb (WebSEAL). As of TAMOS 6.0 these modules are not officially supported but can still form the basis of a policy baseline.

More details on TAMOS policy and policy definition can be found in the *TAMOS 6.0 Administration Guide,* SG32-1709.

Audit capture

Auditing activity on a UNIX/Linux system is one of the strengths of TAMOS. You can apply audit policy consistently across a large set of disparate systems to ensure that you are capturing the same activity.

TAMOS auditing operates on three levels: auditing of *authorization activity*, auditing of process execution or file access (referred to as *auditing trace events*), and auditing for *system health*. We do not discuss the system health auditing, but the following paragraphs summarize the auditing available for the first two.

The authorization auditing is driven by the TAMOS policy enforcement. When an authorization decision is made by TAMOS and where the auditing levels match, an audit record is cut. The types of authorization auditing are:

Global Where a level of auditing is enabled for a specific machine. It can be set to permit or deny.

Resource-level Where a level of auditing is enabled for a specific resource. These definitions can be set to permit, deny, or both, so you can capture allowed access to a resource, denied access to a resource, or all access to a resource.

There is another level of granularity on resource-level auditing where specific access types can be audited. Auditing for resources can be further filtered. For example, you could specify that only create, rename, and delete operations against a file resource are audited.

User-level Where a level of auditing is enabled for a specific user. Supported audit levels for user-level audit authorization policy are permit, deny, loginpermit, logindeny, all, and none. The loginpermit and logindeny settings control audit record generation for login authorization decisions, which are also included in the permit and deny settings.

Authorization auditing levels are generally cumulative, so if the global level is deny and there is a specific resource-level permit set, then both permit and deny audit records will be cut for that resource. There are some exceptions to this.

You would use the authorization auditing where you have a well-defined security policy implemented through TAMOS. It is explicit. You are only auditing against decisions made on the policy that you have defined. The other type of activity auditing is the auditing of trace events.

Trace event auditing can be specified at a global level or at a user level. There are four settings for global-level trace event auditing:

trace_exec Track program invocations initiated by the exec() system call. If this is set at the global level or for a very active user, this could generate a large volume of audit data.

trace_exec_l Track program invocations initiated by the exec() system call where the accessing user's identity and effective UNIX identity do not match. For example, a user has performed a su and then executed a program.

trace_exec_root Track program invocations initiated by the exec() system call by the root user (UID=0). This is only for users who have logged in as a uid=0 user, not someone who has assumed root identity after login.

trace_file Track access to file system resources that are protected by TAMOS (that is, defined in the TAMOS policy that applies to the system).

You can specify equivalent settings at a user levels exec, exec_l, file, all, and none. Use of *none* overrides any global settings.

Note: Trace audit events are only generated for processes that descend from a login event that was detected by TAMOS. If there has been system access using a login mechanism that is *not* defined to TAMOS, no audit records will be produced.

As the trace_exec (user exec), trace_exec_l (exec_l), and trace_exec_root trace settings operate on all resources, not just those defined in the TAMOS policy for a system, they can be used to monitor for unexpected behavior and tune the policy.

An effective policy management strategy would be to define the standard policy and enable auditing against that policy to track violations. Then you would enable one or more trace audit settings to monitor for unexpected activity and review against the standard policy, either modifying the policy or agreeing and documenting the exceptions.

Another feature of the TAMOS policy system is the *warning mode* option. You can set a resource authorization definition to warning mode. When TAMOS detects a resource access that is denied, it allows the access but writes out an audit record indicating that it has been allowed under the warning mode. This is another effective tool to test and tune policy applied to a system. It is common to deploy new policy definitions in warning mode for a period, then review the allowed violations.

For more information see Chapter 7, "Auditing," of the *TAMOS 6.0 Administration Guide*, SG32-1709.

Audit consolidation

TAMOS has its own mechanism for audit log consolidation. It can use the TAMOS log router daemon (pdoslrd) to forward audit logs to a central location. This is shown in Figure 5-2 on page 105. This mechanism does not merge the audit logs. It creates a depot of different system logs on one machine that can be used for reporting.

The log router daemon can also forward events to the Tivoli Common Audit and Reporting Service (CARS), although use of CARS is being replaced by Tivoli Compliance Insight Manager (TCIM). The TAMOS Tivoli Enterprise Console (TEC) daemon can forward TAMOS audit records to TEC, although this is normally used for real-time event reporting, not historical audit and compliance reporting. These components are also shown in Figure 5-2 on page 105.

TCIM (see 2.1, "Tivoli Compliance Insight Manager (TCIM)" on page 14) has two mechanisms to pull TAMOS audit log records to the TCIM server: a TAMOS actuator that runs on the TAMOS server and a TAMOS SSH actuator that runs on another server and connects to the TAMOS server via SSH. TCIM is configured to read the TAMOS binary audit logs and perform the W7 mapping for the TAMOS audit record structure.

Audit reporting

If TAMOS-only mechanisms are used for reporting there are TAMOS commands to access the binary audit log on the local system and to access the consolidated log files.

The **pdosaudview** command is used to view the local TAMOS audit logs. It has an extensive argument list to allow filtering of data. By default, it uses a keyvalue format (-F keyvalue), as shown in Example 5-14. It can also produce a comma-separate format (-F concise) and a readable display with an attribute and its description on each line (-F verbose).

Example 5-14 shows a series of audit records relating to the root user creating a user. There is a trace exec record (A=Trace) showing the command entered and its arguments, the authorization call record (A=Check Access), followed by more trace exec records for the subsequent exec() system calls associated with the useradd.

Example 5-14 pdosaudview example

```
TS=Tue 30 Oct 2007 02:05:30 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/usr/sbin/useradd
(useradd -m -u 1100 -s /bin/sh insight),APID=7406,RPSN=/bin/bash,UQ=0
TS=Tue 30 Oct 2007 02:05:30 PM EST,E=7,V=P,R=5,RT=File,AN=root,AEN=root,A=Check
Access,P=wr,Q=34,PBN=zlinux,PON=File/dev/kazndrv,SRN=/dev/kazndrv,APID=7406,RPSN=/
usr/sbin/useradd,O=S,UQ=1
TS=Tue 30 Oct 2007 02:05:30 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/usr/sbin/useradd.local
(/usr/sbin/useradd.local insight 1100 100
/home/insight),APID=7407,RPSN=/usr/sbin/useradd,UQ=2
TS=Tue 30 Oct 2007 02:05:30 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/bash
(/usr/sbin/useradd.local insight 1100 100
/home/insight),APID=7407,RPSN=/usr/sbin/useradd.local,UQ=3
TS=Tue 30 Oct 2007 02:05:30 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/usr/sbin/useradd.local
(/bin/bash /usr/sbin/useradd.local insight 1100 100
/home/insight),APID=7407,RPSN=/usr/sbin/useradd.local,UQ=4
TS=Tue 30 Oct 2007 02:05:30 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/bash (/bin/bash
/usr/sbin/useradd.local insight 1100 100
/home/insight),APID=7407,RPSN=/usr/sbin/useradd.local,UQ=5
```

The pdosaudview keyvalue report format is great for an experienced user and very useful for forensic investigations. But it is not a format that would be usable by auditors and other non-technical people. The verbose report format may be usable, but can be very lengthy. For more usable reports some scripting would be required.

The **pdoscollview** command is used to view the consolidated audit log files. The arguments and report formats are basically the same as for pdosaudview.

Compliance reporting

TAMOS does not do compliance reporting. You can use the auditing settings and reporting commands to report on where access by users against resources was permitted or denied by TAMOS, but there is no mechanism to measure this against a level of compliance. Using just the audit facilities provided by TAMOS requires extensive customization.

For compliance reporting, we recommend using TCIM.

5.3 UNIX auditing

In the previous sections we discussed the Linux auditing subsystem and the Tivoli Access Manager for Operating Systems product. To round out the discussion on UNIX/Linux auditing tools, we provide a summary of the auditing mechanisms available in the major UNIX implementations available: Sun™ Solaris, Hewlett-Packard HP-UX, and IBM AIX. Most UNIX implementations have some form of auditing capability, such as logging failed password attempts in the syslog file, but most vendors now provide enhanced auditing features. This section will looks at the Solaris Basic Security Module and Trusted Solaris™ audit system, the HP-UX audit subsystem, and the AIX system accounting system and auditing subsystem.

5.3.1 Auditing on Sun Solaris systems

The Sun Microsystem Solaris UNIX operating system has been very popular for many years and is widely deployed. For a number of years there have been two versions of Solaris available—the Solaris Operating Environment and the Trusted Solaris Operating Environment. Solaris 8 and Solaris 9 provided the Basic Security Module (BSM), sometimes called the SunSHIELD Basic Security Module. With the move to Solaris 10 this module has been renamed Solaris Auditing. Trusted Solaris, currently at Version 8, has its own embedded audit mechanism. To following sections summarize these audit mechanisms.

Solaris Basic Security Module or Solaris Auditing

The Basic Security Module provides the security auditing feature. It supports system-wide and per-user auditing.

It is disabled by default and must be enabled for the functions to work. It consists of a number of components, including:

- The bsmconv and bsmunconv utilities enable and disable the BSM.
- The audit daemon, /usr/bin/auditd, controls the generation and location of audit trail files. The **`audit`** command controls the behavior of the audit daemon.
- The /etc/security/audit_data file contains information about the currently running audit process, such as the PID and the location of the current audit log file.
- The /etc/security/audit_event file defines the audit events and their definitions. This includes a mapping of each event to relevant event classes. There is also an /etc/security/audit_class file that defines the classes.
- The /etc/security/audit_control file contains audit control information used by the audit deamon, such as the location of audit files and classes to audit.
- The /etc/security/audit_user file controls per-user auditing. It contains the user names, event classes to always audit, and event classes to never audit.
- The audit.log files that contain the audit records and will contain a date and time stamp in their name. Their location is defined in the audit_control file. These files can be local or on an audit server (locally mounted).

BSM comes with a set of audit policies that determine the characteristics of the audit records for the local host. These are disabled by default and can be enabled by the **`auditconfig`** command. Policies include:

arge When enabled, this policy adds the environment variables of an executed program script to the exec audit record. The resulting audit records contain much more detail than when this policy is disabled.

argv When enabled, this policy adds the arguments of an executed program script to the exec audit record. The resulting audit records contain much more detail than when this policy is disabled.

path When enabled, this policy records every path that is used in conjunction with an audit event to every audit record.

With Solaris 8 and 9, the audit logs contain the audit records in a binary format. Each record is made up of multiple tokens, with each token being on a separate line, tied together with the same token ID. The token contents vary depending on the audit event class. The **`auditreduce`** and **`praudit`** commands can be used together to list out a consolidated audit trail from multiple audit log files. With Solaris 10, records can be written in binary to the audit files or in cleartext to the syslog. Also, the **`praudit`** command in Solaris 10 can output XML formatted records.

TCIM supports the wtmpx log that keeps track of logins and logouts, and the BSM audit logs for Solaris running on SPARC systems (not x86). The events in the log files are mapped by TCIM into the GEM data format.

Tip: For further information see the following:

- Chapter 19, "Configuring auditing for Solaris," in the *IBM Tivoli Compliance Insight Manager: Installation Guide*, GI11-8176
- Chapter 21, "Auditing Topics," in the Solaris 9 *System Administration Guide: Security Services*

  http://docs.sun.com/app/docs/doc/806-4078/6jd6cjs67?a=view
- Part VII, "Solaris Auditing," in the Solaris 10 *System Administration Guide: Security Services*

  http://docs.sun.com/app/docs/doc/816-4557/auditplan-6?a=view

Trusted Solaris Auditing

In the Trusted Solaris Operating Environment auditing is enabled by default. It can be configured by system and security administrators. Auditing can work across a network of Trusted Solaris systems, referred to as a Trusted Solaris network, and can consolidate all audit records into a single audit trail. This allows record selection based on systems, users, events types, time of day, or a combination of these.

As with the Solaris BSM, auditing in Trusted Solaris consists of audit classes, and these can be enabled by workstation or user.

The implementation of Trusted Solaris Auditing is the same as for Solaris 8 BSM (as discussed in the previous section). It uses the same daemons, control files, and logging format as BSM.

While there is no specific mention of Trusted Solaris in the TCIM documentation, as the log files are in the same format as the supported Solaris BSM log files, there should be no reason why TCIM could not collect them.

Tip: For further information see:

- Chapter 19, "Configuring auditing for Solaris," in the *IBM Tivoli Compliance Insight Manager: Installation Guide*, GI11-8176
- The Trusted Solaris 8 *Audit Administration* guide

  http://docs.sun.com/app/docs/doc/805-8121

5.3.2 Auditing on HP-UX systems

To audit an HP-UX system, it must be a trusted system. An HP-UX system is trusted if the file /tcb/files/auth/system/default is present. If that file is not present, then the system is not a trusted system and you need to convert it using the instructions provided in the HP-UX documentation. Once converted, the audit subsystem must be started.

The HP-UX Trusted System auditing includes:

- The audit daemon, /sbin/init.d/auditing, that monitors the system and writes audit events.
- The **`audsys`** command that controls the auditing system. It uses a /.secure/etc/audnames file to hold audit file names and details.
- The **`audusr`** command is used to define users to be audited.
- The **`audevent`** command changes or displays event of system call status.
- The **`audisp`** command is used to display audit records.

Auditing can be defined for specific users or all users using the **`audusr`** command. There are no levels of auditing. All activity for the user is audited or none is.

You can also define which system components generate audit events using the **`audevent`** command. HP-UX uses the term events to refer to audit classes. These include:

create Object creation (creat(), mkdir(), mknod(), msgget(), pipe(), semget(), shmat(), and shmget()).

moddac Discretionary access control (DAC) modification (chmod(), chown(), fchmod(), fchown(), fsetacl(), lchmod(), lchown(), putpmsg(), semop(), setacl(), and umask()).

process Process operations (exit(), fork(), kill(), mlock(), mlockall(), munlock(), munlockall(), nsp_init(), plock(), rtprio(), setcontext(), setrlimit64(), sigqueue(), ulimit64(), and vfork()).

login Log all logins and logouts, including login() and init().

The audit logs contain audit records, each consisting of an audit record header and a record body. The header contains the time stamp, user, error, and event type. The body is the variable-length component containing more information about the audited activity. This information is compressed and the **`audisp`** command is used to view it. The **`audisp`** command can take arguments to restrict the audit data by user, event name, terminal, time, successes only, failures only, or a combination of these.

TCIM supports the HP-UX trusted system audit log formats. The events in the log files are mapped by TCIM into the GEM data format.

Tip: For further information see:

- Chapter 20, "Configuring auditing for HP-UX," in the *IBM Tivoli Compliance Insight Manager: Installation Guide*, GI11-8176
- The "Auditing Trusted Systems" in the *Administering Your HP-UX Trusted System* manual

  `http://docs.hp.com/en/B2355-90121/ch02s05.html`

5.3.3 Auditing on AIX systems

Auditing on AIX consists of the system accounting subsystem and the auditing subsystem. The system accounting subsystem was designed for usage-based charging but does record login/logout. The auditing subsystem is similar to the auditing systems on Solaris and HP-UX.

The AIX auditing subsystem includes:

- The audit subsystem daemon. It is controlled by the **audit** command. There is an associated daemon, auditbin, that handles writing the data to the binary log files and toggling them.
- Some back-end commands that handle the processing of the audit data, but are driven from the daemons, such as auditcat.
- The /etc/security/audit/config is the auditing subsystem configuration file. It includes the audit mode, file locations, class definitions, user-to-class mappings, and other settings.
- The /etc/security/audit/object file defines the object-specific auditing, such as which file system objects and what access against these objects to audit.
- The /audit/trail/* files. These are from the two temporary binary files: /audit/bin1 and /audit/bin2.

Auditing can be specified on a per-user basis, called *event auditing,* or on a per-object basis, called *object auditing*. Event auditing involves defining event classes from the existing events defined to the auditing subsystem. The event definitions are in the /etc/security/audit/events file. Auditing events include:

FILE_Write	We want to know about file writes to configuration files, so this event is used with all files in the /etc tree.
PROC_SetUserIDs	All changes of user IDs.
AUD_Bin_Def	Audit bin configuration.
USER_SU	The **su** command.
PASSWORD_Change	The **passwd** command.
AUD_Lost_Rec	Notification in case there where lost records.
CRON_JobAdd	New cron jobs.
AT_JobAdd	New at jobs.
USER_Login	All logins.
PORT_Locked	All locks on terminals because of too many invalid attempts.

These event types are mapped to classes and the classes mapped to users in the /etc/security/audit/config file. For example, to define two event classes of general and files the following would be added to the config file:

```
general=USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename,FS_Chdir,FS_Chro
ot,PORT_Locked,PORT_Change,FS_Mkdir,FS_Rmdir
files=FILE_Open,FILE_Read,FILE_Write,FILE_Close,FILE_Link,FILE_Unlink,FILE_Rename,
FILE_Owner,FILE_Mode,FILE_Acl,FILE_Privilege,DEV_Create
```

The /etc/security/audit/object file is used to map file system objects to a level of auditing specified by an audit event. For example, to set the read and write auditing for /etc/security/passwd, the following would be added to the file:

```
/etc/security/passwd:
   r = "S_PASSWD_READ"
   w = "S_PASSWD_WRITE"
```

Audit records can be written out in two forms: BIN mode and STREAM mode. Bin mode involves using two live binary files. When one is full the audit process swaps to the other and writes out the data from the full one to the audit trail files. In-stream mode audit records are written out synchronously via an audit psuedo-device. Bin mode is recommended for production deployments. The audit records are written out in two parts: the audit header and the audit tail. The header contains information common to all events. The tail contains event-specific information (and thus varies in format and length from record to record).

The **auditselect**, **auditpr**, and **auditmerge** commands are available to process BIN or STREAM mode audit records:

auditselect This can be used to select only specific audit records with SQL-like statements.

auditpr This is used to convert the binary audit records into a human-readable form. The amount of information displayed depends on the flags specified on the command line.

auditmerge This is used to merge binary audit trails. This is especially useful if there are audit trails from several systems that need to be combined. The **auditmerge** command takes the names of the trails on the command line and sends the merged binary trail to standard output, so you still need to use the **auditpr** command to make it readable.

TCIM supports the AIX audit subsystem log formats. There are some audit subsystem settings required for the TCIM adapter to retrieve the events and forward them to the TCIM server. The events in the log files are mapped by TCIM into the GEM data format.

Tip: For further information see:

- Chapter 21, “Configuring auditing for AIX,” in the *IBM Tivoli Compliance Insight Manager: Installation Guide*, GI11-8176
- Chapter 2, “Auditing on AIX,” in the IBM Redbooks publication *Accounting and Auditing on AIX 5L™*, SG24-6396
- The "Auditing overview" section in the *AIX 5L Version 5.3 Security* guide, SC23-4907

6

DB2 auditing tools

This chapter discusses DB2 on z/OS subsystem auditing. Data has always been the lifeblood of any company. With the fast growth of e-business applications, a database system is more and more critical by its ever-greater demand of data security. In the auditing point of view, Data Disclosure and Privileged User Activity are key areas of database subsystem audit.

6.1 Introduction to DB2 on z/OS auditing

The following sections introduce the DB2 concepts that are required to approach a DB2 auditing discussion.

DB2 privileges and administrative authorities

Each DB2 object type (for example, table, plan, view) has a set of privileges. For example, the db2 table has the privileges of select, insert, alter, update, delete, index, and so on. A privilege allows a specific function to be performed, often on a specific object. An explicit privilege has a name and is held as a result of an SQL GRANT statement. An administrative authority is a set of privileges, often covering a related set of objects. Authorities often include privileges that are not explicit, have no name, and cannot be specifically granted. For example, the ability to terminate any utility job is included in the SYSOPR authority. DB2 has a set of system authorities (SYSADM, SYSCTRL, SYSOPR) and database authorities (DBADM, DBCTRL, DBMAINT).

Multiple paths for DB2 auditing

The fundamental auditing questions for data auditing may be who is privileged to access what objects and who has actually accessed the data. To answer the first question, we can check the DB2 catalog, which hold records of every granted privilege or authority. As for the second question, we may find answers from a DB2 auditing trace. Based on a different scenario, we can automatically start audit trace whenever DB2 is started, or active DB2 trace manually, specifying the audit classes and the destination for trace records.

The default destination for DB2 auditing trace is a System Management Facility (SMF). SMF is a component providing a standardized method for writing out records of activity to a file in System z terms. So when the business requirements call for basic auditing of multiple databases as part of a broader compliance and audit management requirement (possibly including other operating systems, applications, security, and network devices), we may think of using SMF to audit information collection. Tivoli Compliance InSight Manager as a centralized tool is the best way by which we can get DB2 auditing data from an SMF record and generate compliant auditing report consistence with other system auditing.

When a given business requirement asks for comprehensive auditing of the DB2 subsystem, DB2 Audit Management Expert (AME) should be the best solution. AME also includes a DB2 Log Analysis function to give centralized auditing information, greatly simplified access for auditors, and data integrity through segregation of duties on DB2. For DB2 for Linux, UNIX, and Windows auditing, DB2 Audit Management Expert for Multiplatforms offers the integrated subsystem auditing solution.

In a centralized security management point of view, we recommend using RACF to control access to the data sets. This is not only for auditing, but to fully protect DB2 data at the system level, and the system administrator will get an integrated security picture for all subsystems on the mainframe. We also explain how to convert DB2 security to RACF in this chapter.

6.2 Auditing data from SMF

SMF provides full instrumentation of all baseline activities running on System z, including I/O, network activity, software usage, error conditions, processor utilization, and the DB2 subsystem. SMF forms the basis for many monitoring and automation utilities. That is why we use SMF records for auditing multiple subsystems on z/OS, the z/OS operating system itself,

the Security Server (RACF), and so on. The question for an installation is how we can initialize SMF recording for DB2 on System z and how we can obtain compliance auditing reports once we have the auditing data.

The following sections provide an answer to these questions. Following is a list of tasks that can be used to set up the SMF environment and also some details about how to use Tivoli Compliance InSight Manager (TCIM) that provided the compliance solution.

SMF customization

To configure SMF for DB2:

1. Start the DB2 auditing trace with the following command:

   ```
   -start trace(audit) class(1,3,4,5,7,9) dest(smf) location(db8l)
   ```

   where the CLASS specifications depend upon the granularity of the auditing requirement. See Table 6-1 on page 125 for further details.

2. Update the parmlib member SMFPRMxx to include SMF records for DB2. Changing the parmlib member takes effect at the next IPL. Example 6-1 show the content of our parmlib member SMFPRMZZ.

   Example 6-1 SMFPRMZZ sample member

   ```
   ACTIVE                        /*ACTIVE SMF RECORDING*/
    DSNAME(SYS1.&SYSNAME..MAN1,  /*SMF DATA SET NAMES*/
           SYS1.&SYSNAME..MAN2)  /*SMF DATA SET NAMES*/
    NOPROMPT                     /* NO PROMPT  JUST DO IT  */
    REC(PERM)                    /*TYPE 17 PERM RECORDS ONLY*/
    INTVAL(01)
    SYNCVAL(01)
    MAXDORM(3000)                /* WRITE AN IDLE BUFFER AFTER 30 MIN*/
    STATUS(010000)               /* WRITE SMF STATS AFTER 1 HOUR*/
    JWT(0010)                    /* 522 AFTER 10 MINUTES*/
    SID(&SYSNAME(1:4))           /* SYSTEM ID IS &SYSNAME  */
    LISTDSN                      /* LIST DATA SET STATUS AT IPL*/
    LASTDS(MSG)                  /*DEFAULT TO MESSAGE          */
    NOBUFFS(MSG)                 /*DEFAULT TO MESSAGE          */
    SYS(TYPE(30,70:79,80:83,100:102,110,120),
          EXITS(IEFU83,IEFU84,IEFU85,IEFACTRT,
   IEFUJV,IEFUSI,IEFUJP,IEFUSO,IEFUTL,IEFUAV),
   INTERVAL(SMF,SYNC),NODETAIL)
   ```

   You can also change the SMF configuration dynamically by updating the SMF recording using the following command:

   ```
   SETSMF SYS(TYPE(100:102))
   ```

   At this point, the system records DB2 statistics (record type 100), accounting (record type 101), and performance (record type 102) will be collected by SMF.

3. Once you collect the SMF records, you might want to filter and extract only the DB2 portion from the SMF data set into a user data set containing only the DB2 records. Example 6-2 shows the IFASMFDP input parameter we used to filter the DB2 SMF data.

   Example 6-2 IFASMFDP input statements to extract DB2 records

   ```
   INDD(DUMPIN,OPTIONS(ALL))
   OUTDD(DUMPR60,TYPE(70:83,100:102,120))
   ```

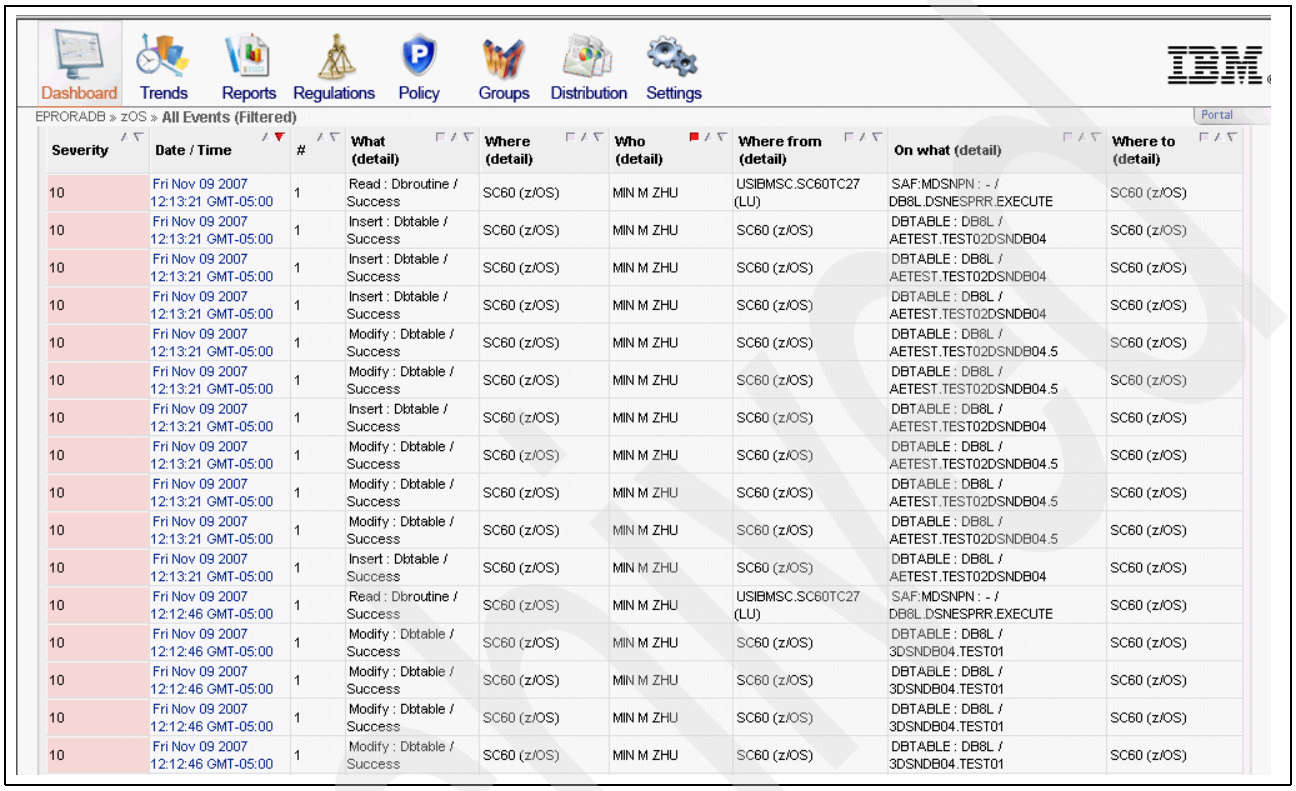
Using TCIM for DB2 compliance

You can use the extracted DB2 SMF records as input to TCIM if you need to analyze and provide reporting to understand whether your DB2 installation is compliance. To do so, we loaded the SMF records into the TCIM and mapped to W7 format for reporting. Example 6-3 shows an SQL script sample for creating DB2 table AETST.TEST01 in the DB2 subsystem:

Example 6-3 SQL script sample

```
CREATE DATABASE AETEST;
CREATE TABLE AETEST.TEST02
(NAME  VARCHAR(10), TELEPHONE INTEGER, LOCATION  VARCHAR(20), F1 INTEGER);
COMMIT
INSERT INTO AETEST.TEST02 (NAME,TELEPHONE,LOCATION,F1) VALUES
('AUDT1',2234,'POK',22);
COMMIT;
INSERT INTO AETEST.TEST02 (NAME,TELEPHONE,LOCATION,F1) VALUES
('AUDT2',2235,'POK',24);
COMMIT;
DELETE  FROM AETEST.TEST02 WHERE NAME='AUDT1';
COMMIT;
SELECT * FROM AETEST.TEST02;
COMMIT;
GRANT SELECT, INSERT, UPDATE, DELETE ON AETEST.TEST02 TO ZHU;
COMMIT;
```

Figure 6-1 shows an example of what you can see from the auditing report in “Events on Database for z/OS on Server EPRORADB” from the TCIM portal. With multiple filter conditions in each item, you can drill down to the auditing record for more detailed information. TCIM is designed by W7 policy describing who did what, when, where, from where, where to, and on what. By using W7 formatted information, auditors can interpret audit information and apply an auditing report.



EPRORADB » zOS » **All Events (Filtered)**

Severity	Date / Time	#	What (detail)	Where (detail)	Who (detail)	Where from (detail)	On what (detail)	Where to (detail)
10	Fri Nov 09 2007 12:13:21 GMT-05:00	1	Read : Dbroutine / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC27 (LU)	SAF:MDSNPN : - / DB8L.DSNESPRR.EXECUTE	SC60 (z/OS)
10	Fri Nov 09 2007 12:13:21 GMT-05:00	1	Insert : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / AETEST.TEST02DSNDB04	SC60 (z/OS)
10	Fri Nov 09 2007 12:13:21 GMT-05:00	1	Insert : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / AETEST.TEST02DSNDB04	SC60 (z/OS)
10	Fri Nov 09 2007 12:13:21 GMT-05:00	1	Insert : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / AETEST.TEST02DSNDB04	SC60 (z/OS)
10	Fri Nov 09 2007 12:13:21 GMT-05:00	1	Modify : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / AETEST.TEST02DSNDB04.5	SC60 (z/OS)
10	Fri Nov 09 2007 12:13:21 GMT-05:00	1	Modify : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / AETEST.TEST02DSNDB04.5	SC60 (z/OS)
10	Fri Nov 09 2007 12:13:21 GMT-05:00	1	Insert : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / AETEST.TEST02DSNDB04	SC60 (z/OS)
10	Fri Nov 09 2007 12:13:21 GMT-05:00	1	Modify : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / AETEST.TEST02DSNDB04.5	SC60 (z/OS)
10	Fri Nov 09 2007 12:13:21 GMT-05:00	1	Modify : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / AETEST.TEST02DSNDB04.5	SC60 (z/OS)
10	Fri Nov 09 2007 12:13:21 GMT-05:00	1	Modify : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / AETEST.TEST02DSNDB04.5	SC60 (z/OS)
10	Fri Nov 09 2007 12:13:21 GMT-05:00	1	Insert : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / AETEST.TEST02DSNDB04	SC60 (z/OS)
10	Fri Nov 09 2007 12:12:46 GMT-05:00	1	Read : Dbroutine / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC27 (LU)	SAF:MDSNPN : - / DB8L.DSNESPRR.EXECUTE	SC60 (z/OS)
10	Fri Nov 09 2007 12:12:46 GMT-05:00	1	Modify : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / 3DSNDB04.TEST01	SC60 (z/OS)
10	Fri Nov 09 2007 12:12:46 GMT-05:00	1	Modify : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / 3DSNDB04.TEST01	SC60 (z/OS)
10	Fri Nov 09 2007 12:12:46 GMT-05:00	1	Modify : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / 3DSNDB04.TEST01	SC60 (z/OS)
10	Fri Nov 09 2007 12:12:46 GMT-05:00	1	Modify : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / 3DSNDB04.TEST01	SC60 (z/OS)

Figure 6-1 Sample event record summary in TCIM

For example, auditors can see that the ZHU user inserted a record in table “AETEST.TEST02“. They can then drill down to check the detail information to audit the operation. Figure 6-2 shows an example of the detailed TCIM auditing record.

EPRORADB » zOS » All Events (Filtered) » **Event Detail** Portal

Severity	10	This is a policy exception.	
When	Tue Nov 13 2007 14:33:39 GMT-05:00	Non business hours (10)	
What	Insert : Dbtable / Success	Other Events (10)	
Where	SC60 (z/OS)	z/OS platform (10)	
Who	MIN M ZHU (SC60.ZHU)	ZHU (10) SC60.RACF Special - Production (10)	More...
From Where	SC60 (z/OS)	z/OS platform (10)	
On What	DBTABLE : DB8L / 6DSNDB04	Other Objects (10)	
Where To	SC60 (z/OS)	z/OS platform (10)	

▸ **Incident Tracking**

▾ **Additional information**

Aspect	Value
Event :: logrecordtype	SMF,102-145
Event :: recorddesc	DB2 DB8L ZHU DML insert table 6 in db DSNDB04 RR by DB8L DSNESPRR.DSNESM68 SQL "INSERT INTO AETEST.TEST02 (NAME,TELEPHONE,LOCATION,F1) VALUES ('AUDT5',2234,'POK',26)"
Who :: originator	SC60.ZHU

Figure 6-2 TCIM detailed record report

TCIM provides other auditing views. For example, you can select **Reports** on TCIM portal and select **Detail Investigation- Type Check**, which lists the SMF log record type and record description (Figure 6-3).

Severity	Date / Time	#	What (detail)	Where (detail)	Who (detail)	Where from (detail)	On what (detail)	Where to (detail)	logrecordtype (event aspect)	recorddesc (event aspect)
10	Tue Nov 13 2007 14:33:39 GMT-05:00	1	Insert : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / 6DSNDB04	SC60 (z/OS)		
10	Tue Nov 13 2007 14:33:39 GMT-05:00	1	Insert : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / 6DSNDB04	SC60 (z/OS)	SMF,102-145	
10	Tue Nov 13 2007 14:33:39 GMT-05:00	1	Insert : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / 6DSNDB04	SC60 (z/OS)		DB2 DB8L ZHU DML insert table 6 in db DSNDB04 RR by DB8L DSNESPRR.DSNESM68 SQL "INSERT INTO AETEST.TEST02 (NAME,TELEPHONE,LOCATION,F1) VALUES ('AUDT2',2234,'POK',22)"
10	Tue Nov 13 2007 14:33:39 GMT-05:00	1	Insert : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / 6DSNDB04	SC60 (z/OS)	SMF,102-145	DB2 DB8L ZHU DML insert table 6 in db DSNDB04 RR by DB8L DSNESPRR.DSNESM68 SQL "INSERT INTO AETEST.TEST02 (NAME,TELEPHONE,LOCATION,F1) VALUES ('AUDT2',2234,'POK',22)"
10	Tue Nov 13 2007 14:33:39 GMT-05:00	1	Insert : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / 6DSNDB04	SC60 (z/OS)		
10	Tue Nov 13 2007 14:33:39 GMT-05:00	1	Insert : Dbtable / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	DBTABLE : DB8L / 6DSNDB04	SC60 (z/OS)	SMF,102-145	

Figure 6-3 TCIM detail investigation sample report

6.3 Auditing Management Expert for DB2

There are tremendous regulatory compliance pressures to demonstrate adequate institutional controls including audit reporting. DBAs are responsible for generating audit data with which they are in turn audited, which constitutes a significant security risk and exposure. In this

section we focus on DB2 on z/OS Subsystem auditing, using the DB2 Audit Management Expert for DB2 on z/OS tool (DB2 AME).

6.3.1 Architecture

DB2 AME Architecture is typically a client/server connection. In our scenario an AME server running on a mainframe is the central control point for the AME network. The AME agent acts as a container to run the various collectors that are appropriate to the specific type of system on which it operates. The server transmits data to the audit repository that is collected from the AME agents that are running data collectors on z/OS systems. A single audit server can support data collection from multiple agents on multiple z/OS systems, including multiple DB2 instances. Besides, AME provides two user interfaces: administration and reporting.

AME tracks, correlates, and analyzes database activities from the Instrumental Trace Interface (IFI) and DB2 log, and deposits audit data into the AME repository.

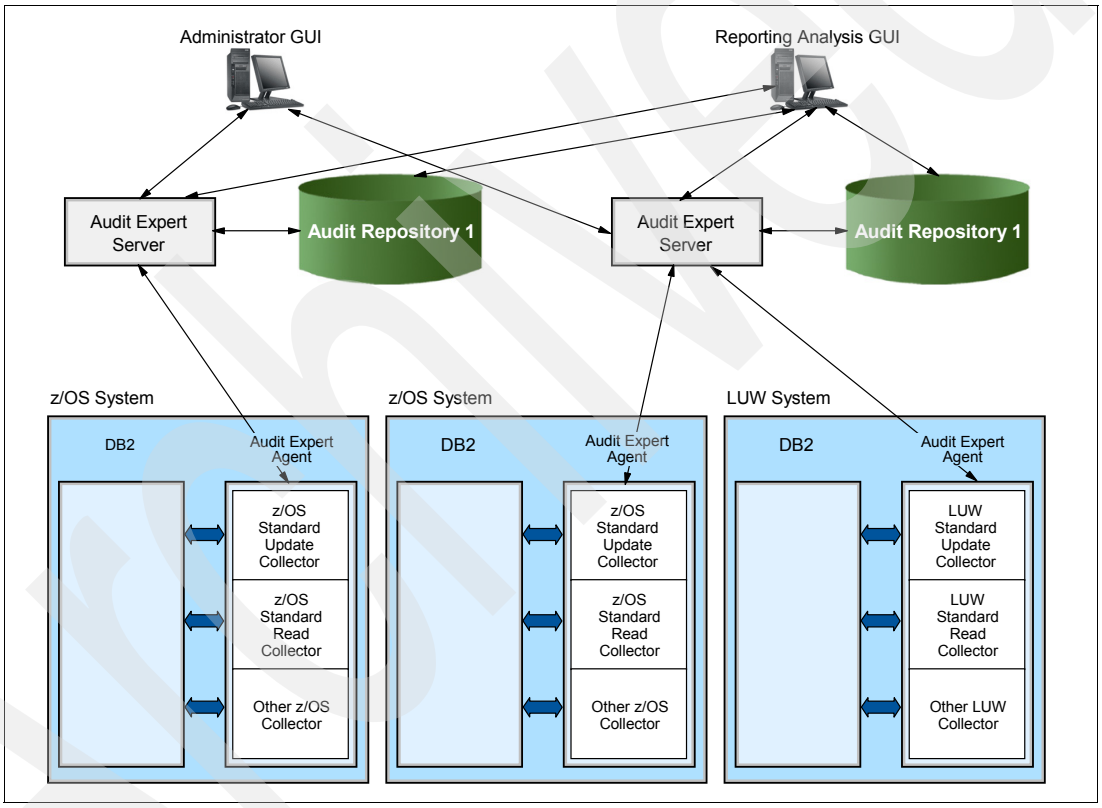


Figure 6-4 DB2 AME architecture

The DB2 AME architecture components are:

- Audit server

  The audit server is the central control point for an AME network to provide service to:

  – Support the administration user interface.
  – Support the reporting user interface.
  – Access the audit repository.
  – Set up monitoring criteria.
  – Populate audit log tables.
  – Extract data from audit log tables for reporting.
  – Perform audit repository administration functions.

- Agent

  The AME agent runs as a started task on z/OS to allow DB2 auditing trace to be started automatically at IPL time. The AME agent, working as a container, runs different collectors and maintains the necessary communications link to send information back and forth to the AME server.

- Audit repository

  The AME audit repository stores audit data collected by the AME agent. It can be deployed in DB2 on mainframe or distributed platforms. A centralized repository creates consistency of views as a single source for reporting, which is available both online and in batch. Since AME repository consists of data from the DB2 trace, it can isolate every possible security thread to system resource and auditors can only read records in the repository instead of updating them. The access to the repository is customized in the AME admininstration interface.

- Data collectors

  AME uses data collectors to collect data from the system and store the data in an AME repository.

- Collections

  A collection associates a specific collection profile with a specific DB2 subsystem. By collection profile, data collector will capture auditing data from DB2 trace and store it in a repository. The AME administrator can specify a collection as active (to collect audit data) or inactive (to stop collecting audit data). One active collection is permitted in one DB2 subsystem.

- Profiles

  AME has two types of profiles: collection profiles and authorization profiles. These profiles describe to AME what and how to collect auditing data and who can access these data.

  – Collection profiles

    A collection profile determines the amount of data to be collected based a selection of tables, events, users, DB2 plans to be audited, and the tracing frequency. The combination of these items determines the subset of traced activity. Collection profiles are written in XML format. We recommend applying collection profiles to a test environment first and then promoting them to the production environment. This is a simple and doable way for testing and security. The items for collection profiles include:

    - Source subsystem and time
    - General audit rule: all failed authorizations, successful and failed AuthID changes, successful grants and revokes, IBM DB2 utility, and DB2 commands
    - DB2 objects (target tables): with first read or first change record in trace
    - Events: successful first read and first change
    - Identity (Auth IDs, WSTrans, or WSNames) and plans

  – Authorization profiles

    Authorization profiles describe which audit data, once collected, can be viewed by associated users or groups. Once an authorization profile has been created, subject to permission, a user can:

    - View the data collected by the collection profile by means of the reporting user interface (read).
    - Edit the authorization profile (write).

- Grant permissions on the authorization profile to other AME users (grant).

► Log analysis

AME includes the log analysis function, which can be used in the reporting interface to get log analysis reports for insert, delete, and update operations on target tables. We do not recommend using a wide log interval for this analysis since this might use system resources and cause overhead. Use AME to see that the interested log range can save system overhead.

► User interfaces

AME has two clients: AME admininstration and AME reporter. AME admininstration is used by the audit administrator to manage auditors (user ID/group), define auditing rules, authorize user or group, enable the AME agent, and set up a repository connection for AME reporter to communicate with DB2 on mainframe.

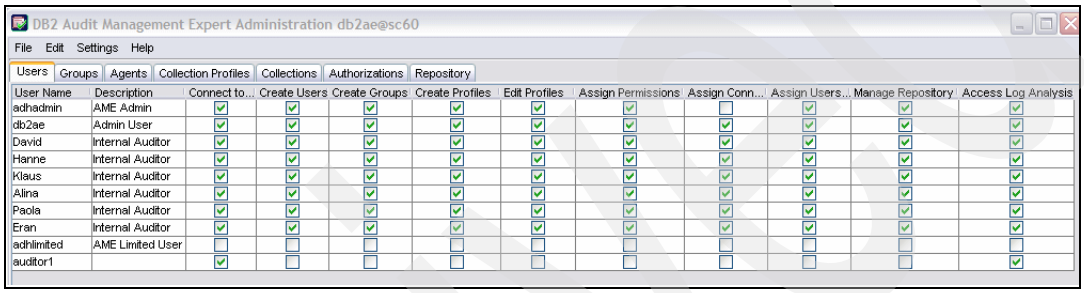


Figure 6-5 AME admininstration Client

With the user ID and password were initialized by the administrator, the auditor logged on the AME reporter to check audited activities and generate related auditing reports.

6.3.2 Policy, auditing, and compliance aspects

How do you approach and design the audit process and what do you audit? The auditing policy can vary depending on industry and regulations, although there are some general policies for auditing that we discuss in this section. In particular, what we describe is how to provide data that will be useful for an auditing task.

Policy

DB2 subsystem records auditing data in the auditing trace. The DB2 auditing trace separates different types of access activities into auditing trace classes. In this way, auditors can control the auditing data amount by specifying auditing trace classes when starting DB2 trace depending on their target goal. In most production systems, all db2 auditing trace classes are rarely active all the time to avoid unnecessary system overhead. Performance is also an element in defining the auditing policy.

Table 6-1 DB2 auditing trace classes description

Class	Audit events	Explanation
1	Access attempts	Access attempts that DB2 denies because of inadequate authorization. This class is the default.
2	Explicit GRANT and REVOKE	Explicit GRANT and REVOKE statements and their results. This class does not trace implicit grants and revokes.

Class	Audit events	Explanation
3	CREATE, ALTER, and DROP	CREATE, ALTER, and DROP statements that affect audited tables, and the results of these statements. This class traces the dropping of a table that is caused by DROP TABLESPACE or DROP DATABASE and the creation of a table with AUDIT CHANGES or AUDIT ALL. ALTER TABLE statements are audited only when they change the AUDIT option for the table.
4	FIRST CHANGE OF AUDIT OBJECT	Changes to audited tables. Only the first attempt to change a table, within a unit of recovery (UOR), is recorded. Access to a dependent table that are caused by attempted deletions from a parent table is also audited.
5	FIRST READ OF AUDIT OBJECT	All read accesses to tables that are identified with the AUDIT ALL clause.
7	ASSIGHMENT OR CHANGE OF AUTH. ID	► Changes through an exit routine ► Changes through a SET CURRENT SQLID statement ► An outbound or inbound authorization ID translation ► An ID that is being mapped to a RACF ID from a Kerberos security ticket
8	DB2 UTILITY	The start of a utility job and the end of each phase of the utility.

From an AME perspective, you can use the administration interface to define the auditing policy and the edit collection profiles. Collection profiles can be edited based on auditing policy. Data can then be filtered based on any combination of DB2 objects, DB2 user IDs, application connections to DB2, or time range. Several profiles can be switched when auditing policy change. The profile is written in XML format and can be imported or exported to other audit servers or assigned to another DB2 system within the same audit server. Example 6-4 shows a sample profile summary.

Example 6-4 Profile summary

```
Profile1 for z/OS
Rules
Schedule
Month Day of the week Hour
start  ALL   Monday    18
stop  ALL   Monday    24
General Audits
All failed authorizations
Successful authid changes
Failed authid changes
Successful grants and revokes
IBM DB2 utilities
DB2 commands
Audited Tables
Schema   Name   Audit successful first read   Audit first change
AETEST   TEST01 true   true
Included AuthIDs
BERNHAR
ZHU
Included Plans
DSN8EPU
DSNACCC
```

Once the collection profile is active, the AME agent tells the collector to capture related auditing data from the DB2 auditing trace and stores them in the AME repository automatically. The refresh interval time can be specified either at product customization or at a later time.

Auditing and compliance aspects

Auditing and compliance aspects are:

- Setting and filter options

  Multiple AME servers can be configured on the client setting tab. When the collection profile is enabled in the AME administration interface, the AME agent starts the DB2 audit trace and data will be loaded and stored into the repository by collectors. AME reporter shows the auditing picture by combining items such as data range, subsystem, users, activity result, time period, and so on, which can be changed by auditors.

- Overview auditing and warning

  AME reporter shows auditing views at three levels: normal status, warning status, or critical status. This is for each auditing event in different DB2 subsystems.

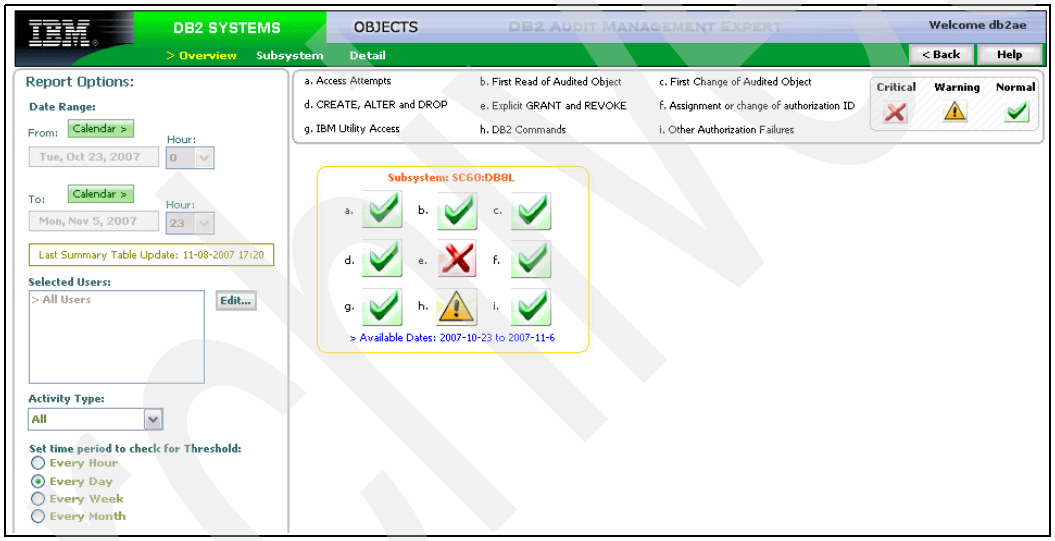


Figure 6-6 Auditing views

Audit can modify threshold value by experience or check auditing data collection history, as shown in Figure 6-7.

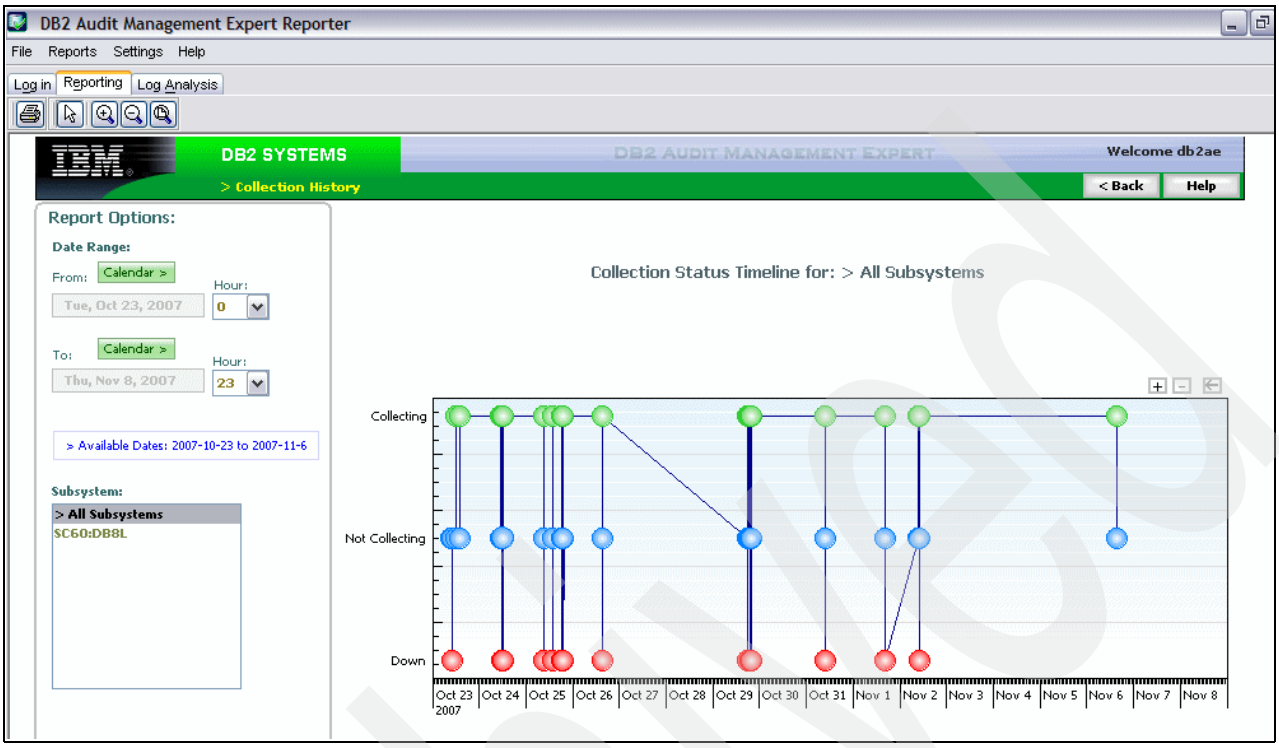


Figure 6-7 Collection history

- Subsystem auditing

  Drilling down from overview, auditors can get subsystem auditing information, as shown in Figure 6-8. On this panel, you can see different auditing events depending on the choices selected on the bottom of the panel. The auditing events are separated into nine groups (from a to i choices). By default, AME reporter maps these auditing events into three auditing pictures:

  – Absolute count of activity
  – Threshold summary by hour
  – Absolute count of activity by hour

  Reporter sorts auditing activities by successful ones and failed ones, which makes it easier for the auditor to notify the doubtful events and pay attention to the detailed information. If the auditor clicks the graph or select a detail item on the top, the detail auditing picture is displayed.

Figure 6-8 Auditing reporting view

- Events Auditing and reporting

  From the detailed auditing picture, you can see the top five objects, top five users, and count for each kind of auditing event. Selecting **Explicit Grant and Revoke**, for example, we can see that explicit grant and revoke for package, table/view, user auth, appli-plan, and database are the top five objects by activity counting. Besides, there are several failed and successful grants and revokes for USERAUTH, which may draw the auditor's attention for further checking. Similarly, auditors can use the top five users who issued explicit grant or revoke commands in the past. For example, DB2AE submitted 68 times during the specified time, which seems unusual.

Figure 6-9 shows the record statistics of activity count by minute, hour, or day, from which auditors can understand DB2 operation trends. Usually, they check the peak time period or those unsafe ones.

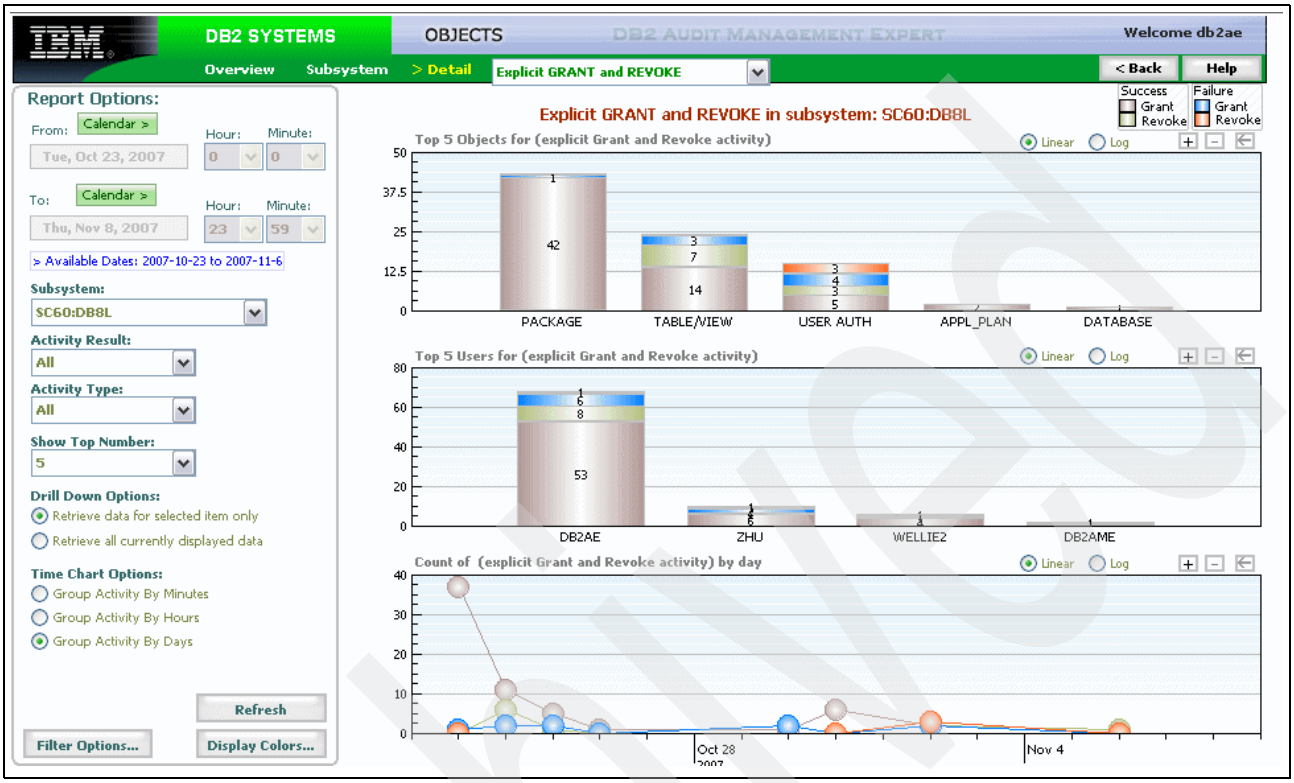


Figure 6-9 Events auditing and reporting

Suppose that the auditors now focus on explicit grant and revoke operation on USERAUTH and need to get the detail auditing report with all possible information regarding when, who, where, and what have been done on USERAUTH in the DB2 subsystem. By clicking the object **USERAUTH** on the top picture, AME drilled down to generate a detailed auditing report and display it on the panel. The auditing report listed the TIMESTAMP, RETURN CODE, IFICODE, CORRELATION_ID, SCHEMA, NAME(for DB2 Object), SQLID, AUTHORIZATION ID, ORIGINAL_OP_ID, DB2 PLAN, DB2 SUBSYSTEM, LOCATION, NETWORK_ID, LUNAME, CONNECTION_NAME, STATEMENT_TXT, ACCESS_ATTEMPTED, and so on. Columns of records vary according to different types of activity reports. Furthermore, the auditing report can be unloaded to Excel® by clicking **Export** on the bottom for user-defined customization or other application.

- Log analysis

  When we audit all operations (insert, delete, update) on target tables, the auditing report only records the first successful read or change within one UOR. So if auditors need to collect all DB2 object operations or the record before and after each update, AME offers an extra log analysis function to generate a DB2 log analysis report. To perform this operation, auditors require no mainframe technical skill. AME generates all the JCL needed to satisfy the request. Auditors only require authorization within AME and a TSO user ID on z/OS to submit the job.

Example 6-5 is a sample of Log Analysis JCL, which reports on insert, update, and delete on table AETEST.TEST01 during a specified period.

Example 6-5 Log analysis JCL sample

```
//ADHJOB JOB ,'DB2 AME',MSGCLASS=H,REGION=0M
//STEP2    EXEC PGM=ADHGEN1,REGION=0M,COND=(4,LT)
//STEPLIB  DD DISP=SHR,DSN=ADH.V1R1M0.SADHLOAD
//         DD DISP=SHR,DSN=DB8L8.SDSNEXIT
//         DD DISP=SHR,DSN=DB8L8.SDSNLOAD
//DB2PARMS DD DISP=SHR,DSN=ADH.V1R1M0.CONTROL
//MODEFILE DD   DSN=DB2AE.ADHLAT.MODE.R0000000,
//         DISP=OLD
//SYSOUT   DD   SYSOUT=*
//CFILES   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//GENRPT   DD   SYSOUT=*
//EXTREP   DD   SYSOUT=*
//SUMREPT  DD   SYSOUT=*
//XDREPT   DD   SYSOUT=*
//QTRPT    DD   SYSOUT=*
//WARNINGS DD   SYSOUT=*
//MESSAGES DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//DATAIN   DD   *
SSID              =DB8L
START DATE   =2007/11/07
START TIME    =11:15:52
END DATE       =2007/11/07
END TIME        =11:40:52
/*
//FILTERS  DD *
GENERAL FILTERS...........
SHOW UPDATES   =Y
SHOW INSERTS   =Y
SHOW DELETES   =Y
SHOW ROLLBACKS =N
CATALOG DATA   =N
SHOW UNCOMMITS =N
OBJECT FILTERS-BY NAME....
TABLE OWNER=AETEST
TABLE NAME =TEST01
PARTITION  =0000
INC/EXC    =I
/*
//STEP3    EXEC PGM=ADHDTL1,REGION=0M,COND=(4,LT)
//STEPLIB  DD DISP=SHR,DSN=ADH.V1R1M0.SADHLOAD
//         DD DISP=SHR,DSN=DB8L8.SDSNEXIT
//         DD DISP=SHR,DSN=DB8L8.SDSNLOAD
//DB2PARMS DD DISP=SHR,DSN=ADH.V1R1M0.CONTROL
//MODEFILE DD   DSN=DB2AE.ADHLAT.MODE.R0000000,
// DISP=OLD
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//WARNINGS DD   SYSOUT=*
//MESSAGES DD   SYSOUT=*
```

```
//SYSUDUMP DD   SYSOUT=*
//CFILES   DD   SYSOUT=*
//TICSPECS DD   *
D,SYSDA   ,C,00050,00050
/*
//DATAIN   DD   *
PROCESS ALL    =Y
/*
//STEP4    EXEC PGM=ADHDTL2,REGION=0M,COND=(4,LT)
//STEPLIB  DD DISP=SHR,DSN=ADH.V1R1M0.SADHLOAD
//         DD DISP=SHR,DSN=DB8L8.SDSNEXIT
//         DD DISP=SHR,DSN=DB8L8.SDSNLOAD
//DB2PARMS DD DISP=SHR,DSN=ADH.V1R1M0.CONTROL
//MODEFILE DD   DSN=DB2AE.ADHLAT.MODE.R0000000,
// DISP=OLD
//WARNINGS DD   SYSOUT=*
//MESSAGES DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//CFILES   DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//*
//STEP5    EXEC PGM=ADHDTL3,REGION=0M,COND=(4,LT)
//STEPLIB  DD DISP=SHR,DSN=ADH.V1R1M0.SADHLOAD
//         DD DISP=SHR,DSN=DB8L8.SDSNEXIT
//         DD DISP=SHR,DSN=DB8L8.SDSNLOAD
//DB2PARMS DD DISP=SHR,DSN=ADH.V1R1M0.CONTROL
//MODEFILE DD   DSN=DB2AE.ADHLAT.MODE.R0000000,
// DISP=OLD
//SYSOUT   DD   SYSOUT=*
//DTLRPT   DD   SYSOUT=*
//XDREPT   DD   SYSOUT=*
//WARNINGS DD   SYSOUT=*
//MESSAGES DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//CFILES   DD   SYSOUT=*
//*
//*
//AMSC0001 EXEC PGM=IDCAMS,COND=(4,LT)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *

  DELETE DB2AE.ADHLAT.EXTFILE.R0000000
  DELETE DB2AE.ADHLAT.MODE.R0000000
```

After submitting the job ADHJOB through the AME reporter interface, auditors can check for the job status by selecting the OUTPUT key. When the status shows completed, select **View Report** to see the log analysis report (summary/detail) displayed on the panel, which can be saved in text format with time stamp added by AME automatically and can be retrieved at any time. Example 6-6 shows a sample of the log analysis detail report from DB8L.

Example 6-6 Log analysis report sample from DB8L

```
DB2 LOG ANALYSIS - DETAILS REPORT: DB8L
***************************************
 ACTION DATE       TIME     TABLE OWNER  TABLE NAME                URID
 ------ ---------- -------- ------------ ------------------------- ------------
 INSERT 2007-10-30 11.16.19 AETEST       TEST01                    00011A3B6736
DATABASE TABLESPACE DBID  PSID  OBID  AUTHID   PLAN     CONNTYPE LRSN
 -------- ---------- ----- ----- ----- -------- -------- -------- ------------
 DSNDB04  TEST01     00004 00002 00003 ZHU      DSNESPRR BATCH    C16C7954BA25
MEMID CORRID       CONNID   LUW=NETID/LUNAME/UNIQUE/COMMIT      PAGE/RID
 ----- ------------ -------- ---------------------------------- -----------
 00000 ZHU          TSO      USIBMSC /SC60DB2 /C16C7954AFCD/0001 00000002/0F
ROW STATUS  NAME  TELEPHONE LOCATION F1
 ----------- ----- --------- -------- ---
 CURRENT     AUDT5 +2234     POK      +26
 POST-CHANGE AUDT5 +2234     POK      +26
 PRE-CHANGE  -     -         -        -
```

6.4 Use of RACF for DB2 security

DB2 has its own protection mechanisms for controlling access to DB2 objects, but we recommend using RACF for DB2 security for the following reasons:

- Single point of control for administration and auditing
- Ability to define security rules before a DB2 object is created
- Allows security rules to persist when a DB2 object is dropped
- Ability to protect multiple DB2 objects with a single security rule using generic profiles or member/grouping profiles
- Eliminates DB2 cascading revoke
- Preserves DB2 privileges and administrative authorities
- Flexibility for multiple DB2 subsystems:
  - One set of RACF classes for multiple DB2 subsystems
  - One set of RACF classes for each DB2 subsystem
- Selectable on an object-by-object basis

DB2 security mechanisms consist of several constructs:

- Objects, such as tables, tablespaces, databases, user-defined functions, and so on
- Privileges, such as insert, update, select, and so on
- Administrative authorities, such as DBADM, SYSADM, and so on

6.4.1 How to migrate DB2 security into RACF

Before converting DB2 security to RACF, we briefly summarize DB2 object mapping to RACF class and its scope.

- DB2 objects (table, database, view, user-defined function, and so on) correspond to RACF general resource classes, as shown in Table 6-2.

Table 6-2 Racf mapping to DB2 objects

DB2 object type	RACF class name
Jar	MDSNJR
Schema	MDSNSC
Stored Procedure	MDSNSP
User Defined Function	MDSNUF
User Defined Type	MDSNUT
Table Space	MDSNTS
Table/Index/View	MDSNTB
System	MDSNSM
Storage Group	MDSNSG
Plan	MDSNPN
Package	MDSNPK
Database	MDSNDB
Collection	MDSNCL
Bufferpool	MDSNBP
Sequence	MDSNSQ

- DB2 privileges are a part of RACF profile names.
- DB2 administrative authorities are profiles within RACF general resource classes.

The scope of RACF classes is:

- Multi-subsystem scope (default)
  - One set of general resource classes that protect multiple subsystems.
  - General resource names are prefixed with a DB2 subsystem name.
  - Classes provided in the IBM-supplied CDT are multi-system scope.
  - Protect multiple subsystems with a single set of resource profiles.
  - There are fewer classes overall.
- Single subsystem scope (an option)
  - One set of general resources classes is dedicated to one subsystem.
  - General resource names are not prefixed with the DB2 subsystem name.
  - Classes must be defined by the installation.
  - Segregates resources by subsystem.
  - There are fewer profiles per class.

Using the DB2 to RACF Migration tool, you can easily convert contents of SYSIBM.SYSxxxAUTH tables to RACF profiles. To do this:

1. Run the DB2 security convert to RACF utility.

   This utility converts SYSIBM.SYSxxxAUTH tables to equivalent RACF profiles, which are used in conjunction with the RACF-provided DB2 Access Control Exit.

The utility requires the DB2 Rexx function be installed. Example 6-7 shows a sample JCL to invoke the tool.

Example 6-7 Sample JCL to invoke the DB2 security to RACF tool

```
//DB2AER1 JOB (999,POK),'AME',CLASS=A,REGION=0K,
//            MSGCLASS=X,TIME=10,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//STEP1  EXEC  PGM=IKJEFT01
//SYSTSPRT DD  SYSOUT=*
//SYSPROC  DD  DISP=SHR,DSN=DB2AE.RACFDB2
//SYSEXEC  DD  DISP=SHR,DSN=DB2AE.RACFDB2
//*CLIST    DD  DISP=SHR,DSN=RCF.RACFDB2.CONVCLST
//CLIST    DD  DISP=(NEW,CATLG),DSN=DB2AE.RACFDB2.CONVCLST.NEW,
//   UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VB,LRECL=255)
//*OPTCLST  DD  DISP=SHR,DSN=RCF.RACFDB2.OPTCLST
//OPTCLST  DD  DISP=(NEW,CATLG),DSN=DB2AE.RACFDB2.OPTCLST.NEW,
//   UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VB,LRECL=255)
//SYSTSIN  DD  *
  EXECUTIL SEARCHDD(YES)
  %RACFDB2 DB2 DB8L  DSN      2
//*        OWNER  SSID CLASSMNT MODEL CHAROPT
```

2. Issue an RACF command to define the RACF user, group, profile, and access list.

6.4.2 DB2 using RACF authorization checks

When a DB2 object is accessed, the RACF-supplied DSNX@XAC module performs the needed RACF authorization checks to verify that the user is allowed to access the resource. If the user is not allowed to access the resource, other DB2 privilege checks (such as DBADM and SYSADM) are performed. Figure 6-10 shows the logical authorization flow. After converting DB2 security to RACF, there will be some other changes for auditing for the SMF recording.

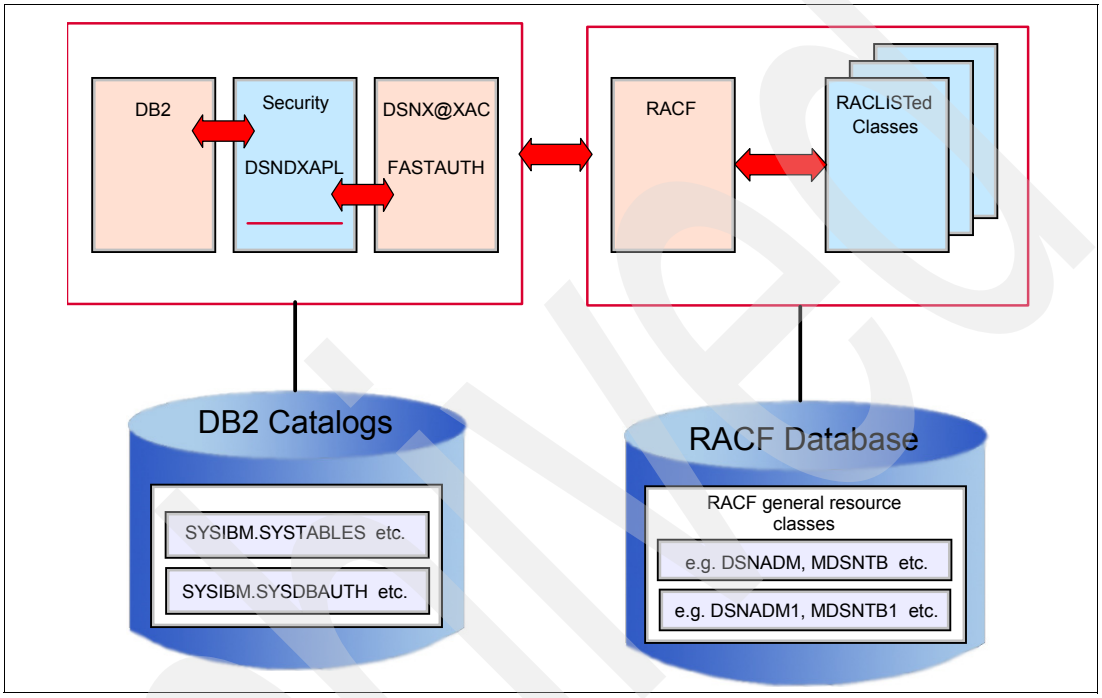


Figure 6-10 DB2 and RACF interactions

Other considerations for using RACF for DB2 security can be:

- The DB2 application plan is not invalidated when a security change is made to an RACF-protected resource.
- Be sure to RACLIST REFRESH general resource classes after defining, changing, or deleting a resource profile.
- Note that after the application of OW52799/PQ68177, ownership of a view is not sufficient to grant access.
- DB2 does not call RACF for any requests made by the INSTALLSYSADM and INSTALLSYSOPR user IDs.

With RACF performing the authorization decisions for DB2, auditing data is written to SMF along with other RACF authorization decisions. This means that data can be analyzed using the same zSecure Audit reporting tools and could be forwarded to TCIM along with the other RACF authorization audit records.

6.5 DB2 Universal Database auditing

In the previous sections we described in detail the z/OS DB2 auditing facilities and the DB2 Audit Management Expert. For completeness, this section summarizes the auditing available with the DB2 Universal Database (UDB), which is widely used in distributed systems.

DB2 UDB has its own auditing facility that is not dependent on the audit logging features of its host operating system. It is described in detail in Chapter 6, “Auditing DB2 Activities,” in the *IBM DB2 Universal Database Administration Guide: Implementation*, SC09-4820.

The DB2 audit facility generates, and allows you to maintain, an audit trail for a series of predefined database events. The records generated from this facility are kept in an audit log file. The analysis of these records can reveal usage patterns that would identify system misuse. Once identified, actions can be taken to reduce or eliminate such system misuse. The audit facility acts at an instance level, recording all instance-level activities and database-level activities.

The audit facility must be stopped and started explicitly. When starting, the audit facility uses existing audit configuration information. Since the audit facility is independent of the DB2 server, it remains active even if the instance is stopped. In fact, when the instance is stopped, an audit record may be generated in the audit log.

The audit facility uses a single command-line program to configure and manage auditing. It uses operating system files external to the database for storing the records and can extract the records into different formats depending on the intended use.

6.5.1 Configuring and managing the DB2 UDB audit facility

The audit facility is configured and managed through the db2audit tool. It allows an authorized administrator to:

- Start recording auditable events within the DB2 instance.
- Stop recording auditable events within the DB2 instance.
- Configure the behavior of the audit facility, including selecting the categories of the auditable events to be recorded.
- Request a description of the current audit configuration.
- Flush any pending audit records from the instance and write them to the audit log.
- Extract audit records by formatting and copying them from the audit log to a flat file or ASCII delimited files. Extraction is done for one of two reasons: in preparation for analysis of log records or in preparation for pruning log records.
- Prune audit records from the current audit log.

Example 6-8 shows an example of using the db2audit tool.

Example 6-8 Using the db2audit tool

```
ldapdb2@tameb:~/sqllib/security> db2audit start
AUD0000I  Operation succeeded.
ldapdb2@tameb:~/sqllib/security> db2audit configure scope all status both

AUD0000I  Operation succeeded.
ldapdb2@tameb:~/sqllib/security> db2audit describe
DB2 AUDIT SETTINGS:
```

```
Audit active: "TRUE "
Log errors: "TRUE "
Log success: "TRUE "
Log audit events: "TRUE "
Log checking events: "TRUE "
Log object maintenance events: "TRUE "
Log security maintenance events: "TRUE "
Log system administrator events: "TRUE "
Log validate events: "TRUE "
Log context events: "TRUE "
Return SQLCA on audit error: "FALSE "

AUD0000I  Operation succeeded.
ldapdb2@tameb:~/sqllib/security>
```

In this example we started the audit facility, set all auditing on (for both successful and failed activities), and then displayed the settings. The next section looks at the audit records cut by the facility.

6.5.2 Audit information written by the DB2 UDB audit facility

There are different categories of audit records that may be generated. In the description of the categories of events available for auditing (below), you should notice that following the name of each category is a one-word keyword used to identify the category type. The categories of events available for auditing are:

- Audit (AUDIT) - generates records when audit settings are changed or when the audit log is accessed.
- Authorization Checking (CHECKING) - generates records during authorization checking of attempts to access or manipulate DB2 objects or functions.
- Object Maintenance (OBJMAINT) - generates records when creating or dropping data objects.
- Security Maintenance (SECMAINT) - generates records when granting or revoking object or database privileges, or DBADM authority. Records are also generated when the database manager security configuration parameters SYSADM_GROUP, SYSCTRL_GROUP, or SYSMAINT_GROUP are modified.
- System Administration (SYSADMIN) - generates records when operations requiring SYSADM, SYSMAINT, or SYSCTRL authority are performed.
- User Validation (VALIDATE) - generates records when authenticating users or retrieving system security information.
- Operation Context (CONTEXT) - generates records to show the operation context when a database operation is performed. This category allows for better interpretation of the audit log file. When used with the log's event correlator field, a group of events can be associated back to a single database operation. For example, an SQL statement for dynamic SQL, a package identifier for static SQL, or an indicator of the type of operation being performed, such as CONNECT, can provide needed context when analyzing audit results.

You can audit successes, failures, or both, for each audit category. Any operation on the database may generate several records. The actual number of records generated and moved to the audit log depends on the number of categories of events to be recorded as specified by the audit facility configuration.

We use a simple example to show the audit records generated for some activity. Example 6-9 shows a series of DB2 commands being run by a database owner.

Example 6-9 A sample set of DB2 commands to be audited

```
tameb:/home/ldapdb2/idsslapd-ldapdb2/etc # su ldapdb2
ldapdb2@tameb:~/idsslapd-ldapdb2/etc> db2 list database directory

 System Database Directory

 Number of entries in the directory = 4

Database 1 entry:

 Database alias                       = EVENT
 Database name                        = EVENT
 Local database directory             = /home/ldapdb2
 Database release level               = a.00
 Comment                              =
 Directory entry type                 = Indirect
 Catalog database partition number    = 0
 Alternate server hostname            =
 Alternate server port number         =

Database 2 entry:

 Database alias                       = AMDB
 Database name                        = AMDB
 Local database directory             = /home/ldapdb2
 Database release level               = a.00
 Comment                              =
 Directory entry type                 = Indirect
 Catalog database partition number    = 0
 Alternate server hostname            =
 Alternate server port number         =
...
ldapdb2@tameb:~/idsslapd-ldapdb2/etc> db2 connect to amdb

   Database Connection Information

 Database server        = DB2/LINUX 8.2.1
 SQL authorization ID   = LDAPDB2
 Local database alias   = AMDB
...
ldapdb2@tameb:~/idsslapd-ldapdb2/etc> db2 list tables

Table/View                      Schema          Type  Creation time
------------------------------- --------------- ----- --------------------------
ACLINHERIT                      LDAPDB2         T     2006-02-16-13.50.25.377137
ACLPERM                         LDAPDB2         T     2006-02-16-13.50.25.660439
ACLPROP                         LDAPDB2         T     2006-02-16-13.50.25.291268
ALIASEDOBJECT                   LDAPDB2         T     2006-02-16-13.51.26.003953
BUSINESSCATEGORY                LDAPDB2         T     2007-08-03-21.53.50.248068
C                               LDAPDB2         T     2006-02-16-13.50.39.821328
CN                              LDAPDB2         T     2006-02-16-13.50.46.968218
CRYPTOSALT                      LDAPDB2         T     2006-02-16-13.52.05.375780
```

```
...
ldapdb2@tameb:~/idsslapd-ldapdb2/etc> db2 'describe select * from cn'

SQLDA Information

 sqldaid : SQLDA     sqldabc: 896  sqln: 20  sqld: 4

 Column Information

 sqltype               sqllen  sqlname.data                   sqlname.length
 --------------------  ------  ------------------------------ --------------
 496   INTEGER              4  EID                                         3
 448   VARCHAR            256  CN                                          2
 448   VARCHAR            240  CN_T                                        4
 448   VARCHAR            240  RCN_T                                       5
...
ldapdb2@tameb:~/idsslapd-ldapdb2/etc> db2 'select eid,cn from cn'

EID         CN


----------- ------------------------------------------------------------
          2 LOCALHOST
          3 IBMPOLICIES
          4 CRYPTO
          6 PWDPOLICY
...
```

The commands run are:

- db2 list database directory - shows the databases in this instance
- db2 connect to amdb - connects to one of the database (As they are the owner, they do not need to authenticate.)
- db2 list tables - lists the tables in the database
- db2 describe select * from cn - lists the columns in the cn (LDAP common name) table
- db2 select eid,cn from cn - shows the eid and cn values for all records in the cn table

These activities are logged in the db2audit.log file as with TAMOS the audit log, called db2audit.log. It contains binary-format records and cannot be browsed. The **db2audit extract** command can be used to extract some or all of the records from the db2audit.log file, as shown in Example 6-10.

Example 6-10 Use of the db2audit tool to extract audit records

```
ldapdb2@tameb:~/sqllib/security> db2audit extract file db2audit.log.txt

AUD0000I  Operation succeeded.
```

This has extracted all records from the db2audit.log file and written them to the db2audit.log.txt file. The audit facility record layouts can be found in the aforementioned manual.

Example 6-11 shows some of the records extracted from a DB2 UDB system that relate to the connect command.

Example 6-11 Extracted db2 audit records for db2 connect command

```
timestamp=2007-11-22-15.11.20.650545;category=VALIDATE;audit event=GET_USERID;
  event correlator=0;event status=0;
  userid=ldapdb2;authid=LDAPDB2;execution id=ldapdb2;
  origin node=0;coordinator node=0;
  application id=*LOCAL_APPLICATION;application name=db2bp;
  auth type=CLIENT;plugin name=IBMOSauthclient;

timestamp=2007-11-22-15.11.20.664444;category=CONTEXT;audit event=CONNECT;
  event correlator=2;
  database=AMDB;userid=ldapdb2;authid=LDAPDB2;
  origin node=0;coordinator node=0;
  application id=*LOCAL.ldapdb2.071122041120;application name=db2bp;

timestamp=2007-11-22-15.11.20.675449;category=VALIDATE;audit event=AUTHENTICATION;
  event correlator=2;event status=0;
  database=AMDB;userid=ldapdb2;authid=LDAPDB2;execution id=ldapdb2;
  origin node=0;coordinator node=0;
  application id=*LOCAL.ldapdb2.071122041120;application name=db2bp;
  auth type=SERVER;plugin name=IBMOSauthserver;
```

These records show the ldapdb2 user connecting to the database and being authenticated (in this case against an operating system user ID ldapdb2). There were additional records cut associated with this connect operation, such as determining group membership of the user.

Note that the records contain an event correlator field. This allows multiple records to be correlated into a single activity.

Example 6-12 shows some of the audit records generated from the **db2 list tables** command.

Example 6-12 Extracted db2 audit records for db2 list tables command

```
timestamp=2007-11-22-15.11.36.299258;category=CHECKING;audit
event=CHECKING_OBJECT;
  event correlator=3;event status=0;
  database=AMDB;userid=ldapdb2;authid=LDAPDB2;
  origin node=0;coordinator node=0;
  application id=*LOCAL.ldapdb2.071122041120;application name=db2bp;
  package schema=NULLID;package name=SQLC2E06;
  package section=0;object schema=NULLID;object name=SQLC2E06;object type=PACKAGE;
  access approval reason=OBJECT;access attempted=EXECUTE;

timestamp=2007-11-22-15.11.36.551570;category=CONTEXT;audit event=PREPARE;
  event correlator=3;
  database=AMDB;userid=ldapdb2;authid=LDAPDB2;
  origin node=0;coordinator node=0;
  application id=*LOCAL.ldapdb2.071122041120;application name=db2bp;
  package schema=NULLID;package name=SQLC2E06;
  package section=201;text=SELECT NAME, CREATOR, TYPE, CTIME FROM SYSIBM.SYSTABLES
WHERE CREATOR = USER ORDER BY CREATOR, NAME;
```

```
timestamp=2007-11-22-15.11.36.559920;category=CHECKING;audit
event=CHECKING_OBJECT;
  event correlator=3;event status=0;
  database=AMDB;userid=ldapdb2;authid=LDAPDB2;
  origin node=0;coordinator node=0;
  application id=*LOCAL.ldapdb2.071122041120;application name=db2bp;
  package schema=NULLID;package name=SQLC2E06;
  package section=201;object type=ALL_OBJECTS;
  access approval reason=DBADM;access attempted=ALL;

timestamp=2007-11-22-15.11.36.704012;category=CONTEXT;audit event=OPEN_CURSOR;
  event correlator=4;
  database=AMDB;userid=ldapdb2;authid=LDAPDB2;
  origin node=0;coordinator node=0;
  application id=*LOCAL.ldapdb2.071122041120;application name=db2bp;
  package schema=NULLID;package name=SQLC2E06;
  package section=201;text=SQLCUR201;

timestamp=2007-11-22-15.11.36.821692;category=CONTEXT;audit event=CLOSE_CURSOR;
  event correlator=5;
  database=AMDB;userid=ldapdb2;authid=LDAPDB2;
  origin node=0;coordinator node=0;
  application id=*LOCAL.ldapdb2.071122041120;application name=db2bp;
  package schema=NULLID;package name=SQLC2E06;
  package section=201;text=SQLCUR201;
```

Much of the information in these records would only be of use to a DB2 system administrator or DBA, as the records do not seem to relate directly to the activity on the command line. The second record does show the SQL being run, where a number of fields are being extracted from the SYSIBM.SYSTABLES table, which is the master list of tables in the database. But an auditor is unlikely to know this.

The CHECKING_OBJECT record shown above is the result of the authorization. It does not explicitly state what database data was being accessed, just that ldapdb2 attempted to access ALL_OBJECTS, with access attempted ALL and was granted because of DBADM. (Access is approved. The application/user has DBADM authority.)

Example 6-13 shows some of the audit records generated from the **db2 'select eid,cn from cn'** command.

Example 6-13 Extracted db2 audit records for db2 select command

```
timestamp=2007-11-22-15.12.23.288628;category=CONTEXT;audit event=PREPARE;
  event correlator=7;
  database=AMDB;userid=ldapdb2;authid=LDAPDB2;
  origin node=0;coordinator node=0;
  application id=*LOCAL.ldapdb2.071122041120;application name=db2bp;
  package schema=NULLID;package name=SQLC2E06;
  package section=201;text=select eid,cn from cn;

timestamp=2007-11-22-15.12.23.299539;category=CHECKING;audit
event=CHECKING_OBJECT;
  event correlator=7;event status=0;
  database=AMDB;userid=ldapdb2;authid=LDAPDB2;
  origin node=0;coordinator node=0;
  application id=*LOCAL.ldapdb2.071122041120;application name=db2bp;
```

```
  package schema=NULLID;package name=SQLC2E06;
  package section=201;object type=ALL_OBJECTS;
  access approval reason=DBADM;access attempted=ALL;

timestamp=2007-11-22-15.12.23.310645;category=CONTEXT;audit event=OPEN_CURSOR;
  event correlator=8;
  database=AMDB;userid=ldapdb2;authid=LDAPDB2;
  origin node=0;coordinator node=0;
  application id=*LOCAL.ldapdb2.071122041120;application name=db2bp;
  package schema=NULLID;package name=SQLC2E06;
  package section=201;text=SQLCUR201;

timestamp=2007-11-22-15.12.23.324498;category=CONTEXT;audit event=DESCRIBE;
  event correlator=9;
  database=AMDB;userid=ldapdb2;authid=LDAPDB2;
  origin node=0;coordinator node=0;
  application id=*LOCAL.ldapdb2.071122041120;application name=db2bp;
  package schema=NULLID;package name=SQLC2E06;
  package section=201;text=select eid,cn from cn;
```

The first two records show DB2 preparing the query and checking the user's authority to run the query. The last two records show further DB2 activity relating to the command. As before, much of the information is of little use to anyone other than a DBA. However, you can identify which user has accessed which DB2 object and why they were granted or denied access.

Reporting on the DB2 UDB audit records

The format produced by the **db2audit extract file** command, shown previously, is only useful for visual checking of records. There is another extract option, delasc, that will produce delimited ascii records. The default delimeter is suitable for loading the records into a db2 database. You can specify different delimiters to suit how you intend to use the data, such as:

```
db2audit extract delasc
db2audit extract delasc delimiter !
db2audit extractdelasc delimiter 0xff
```

These commands produce a series of files with a .del suffix, as shown in Example 6-14.

Example 6-14 Extracting db2 audit records into a ascii delimited form

```
ldapdb2@tameb:~/sqllib/security> db2audit extract delasc delimiter '"'

AUD0000I  Operation succeeded.
ldapdb2@tameb:~/sqllib/security> ls -l *.del
-rw-rw-rw-    1 ldapdb2  idsldap       434 Nov 22 16:00 audit.del
-rw-rw-rw-    1 ldapdb2  idsldap     40911 Nov 22 16:00 checking.del
-rw-rw-rw-    1 ldapdb2  idsldap    279340 Nov 22 16:00 context.del
-rw-rw-rw-    1 ldapdb2  idsldap         0 Nov 22 16:00 objmaint.del
-rw-rw-rw-    1 ldapdb2  idsldap         0 Nov 22 16:00 secmaint.del
-rw-rw-rw-    1 ldapdb2  idsldap      8954 Nov 22 16:00 sysadmin.del
-rw-rw-rw-    1 ldapdb2  idsldap     83852 Nov 22 16:00 validate.del
ldapdb2@tameb:~/sqllib/security> cat audit.del
"2007-11-22-14.39.12.587642","AUDIT","EXTRACT",0,0,"ldapdb2","LDAPDB2"
"2007-11-22-14.56.26.755597","AUDIT","EXTRACT",0,-1042,"ldapdb2","LDAPDB2"
"2007-11-22-14.56.58.657663","AUDIT","EXTRACT",0,0,"ldapdb2","LDAPDB2"
"2007-11-22-15.14.35.703937","AUDIT","EXTRACT",0,0,"ldapdb2","LDAPDB2"
"2007-11-22-15.49.31.575182","AUDIT","EXTRACT",0,0,"ldapdb2","LDAPDB2"
```

```
"2007-11-22-16.00.27.298467","AUDIT","EXTRACT",0,-1042,"ldapdb2","LDAPDB2"
"2007-11-22-16.00.50.606510","AUDIT","EXTRACT",0,0,"ldapdb2","LDAPDB2"
"2007-11-22-16.11.15.134789","AUDIT","EXTRACT",0,0,"ldapdb2","LDAPDB2"
"2007-11-22-16.12.25.995274","AUDIT","EXTRACT",0,-1042,"ldapdb2","LDAPDB2"
```

In this example all of the records are extracted into the ascii delimited files with a quotation mark (“) as the delimiter.

The resulting audit.del file is shown. Note that there are actually commas (,) between the fields and the quotation mark is providing the end marks for each of the discrete fields. Some of the output records will contain commas (such as the SQL statements used previously), so you cannot rely on the commas alone to delimit the fields.

The structure of the records is not held in any of the files. Whatever tool is going to use the data needs to know the structure. Also, there is no option to merge the records together. Each file is structured according to the audit record category. Merging the records would require the addition of a category flag in each record.

For audit reporting you could develop an application that reads the extracted and formatted db2audit records. It may also be possible to load the records into DB2 tables and use the DB2 Audit Management Expert.

TCIM has an actuator that supports the DB2 UDB. Full details of installation and configuration of the actuator can be found in Chapter 43, “Configuring auditing for the IBM DB2 Universal Database,” of the *Tivoli Compliance Insight Manager Installation Guide*, GI11-8176. TCIM accepts and processes audit records of the following categories: AUDIT, CHECKING, SECMAINT, SYSADMIN, and VALIDATE. With the DB2 audit records, the TCIM policy could be defined against the records in isolation or combined with other related audit activity, such as applications accessing DB2 and DBAs working at the operating system level.

Part 2

Scenarios

This part describes the auditing scenarios used as a sample for auditing an enterprise and shows how to collect the auditing data and how to use the tool to interpret the data. Your installation may require different tools, reports, or compliance associations based upon your security policies and compliance requirements. Refer to “Tools” on page 11 for a description of the various tools that might be used for auditing.

7

Introduction to scenarios

This chapter introduces the infrastructure of our enterprise with a description of how the different servers on the different platforms interface with each other, where the different logs, audit information, and security information are stored on each of these platforms and servers, and the flow of the users through the systems and when the audit and log records are created.

The infrastructure and security information that is provided in this chapter is elaborated upon and tested in the following chapters. This chapter provides an introduction to the scenarios.

7.1 ITSO configuration

Figure 7-1 illustrates the sample configuration that we build to run the scenarios cases. Our configuration does not cover all the possible configurations an installation can have, but we aim to provide a base sample that can be applicable in most enterprises.

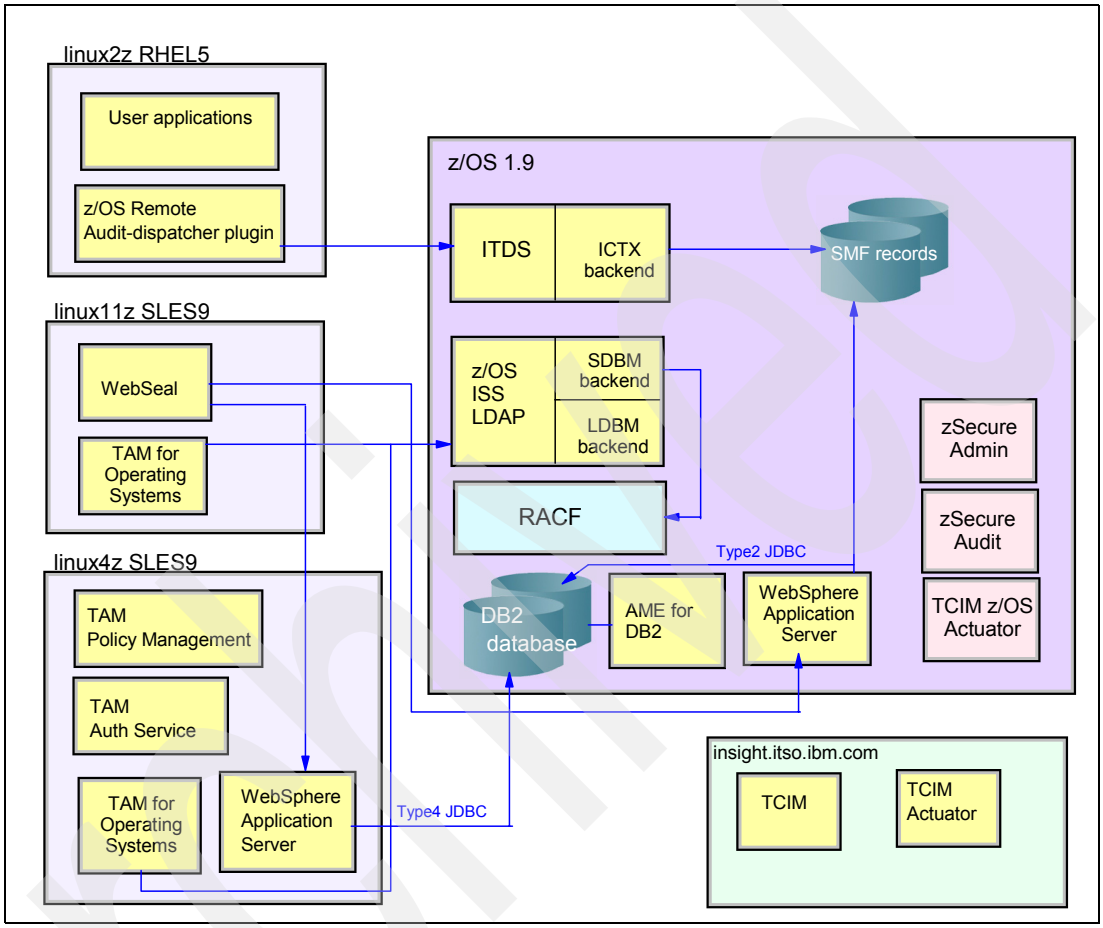


Figure 7-1 Architecture layout

In our configuration we would like to summarize the elements that can be common in an enterprise: a z/OS system where most of the data, handled by DB2, are located; a Linux for System z to host fewer traditional applications; and a transaction manager such as WebSphere Application Server. We do not cover CICS/TS and IMS/TM as part of our configuration.

As you can see in Figure 7-1, the major elements on the z/OS system are ITDS and ISS, since we are still in a migration phase from the LDAP exploiters and they are not all capable of interfacing with the ITDS component, the WebSphere Application Server with the application business logic, and the data stored in the DB2.

On the Linux for System z we have user applications running and, in order to trace events, we installed the z/OS Remote-Services Audit dispatcher plug-in, which is making use of the ITDS LADP on z/OS through the ICTX back end.

All the SMF auditing data were collected on z/OS and processed through TCIM.

We also installed the zAudit for additional auditing functionality on z/OS and the TCIM actuator to unload z/OS data to the TCIM local database.

To expand the WebSphere Application Server configuration to something more similar to a user installation, we also installed it on a Linux for System z server.

Additional Linux for System z servers were installed to host the WebSeal and TAMOS to provide authentication functionalities. These elements can be installed on different platforms, but Linux for System z seemed to be a good choice for our installation.

Figure 7-2 shows an high-level view of the auditing process divided by phases and outlines the tools that we used during the process of accumulating, analyzing, and reporting the auditing records.

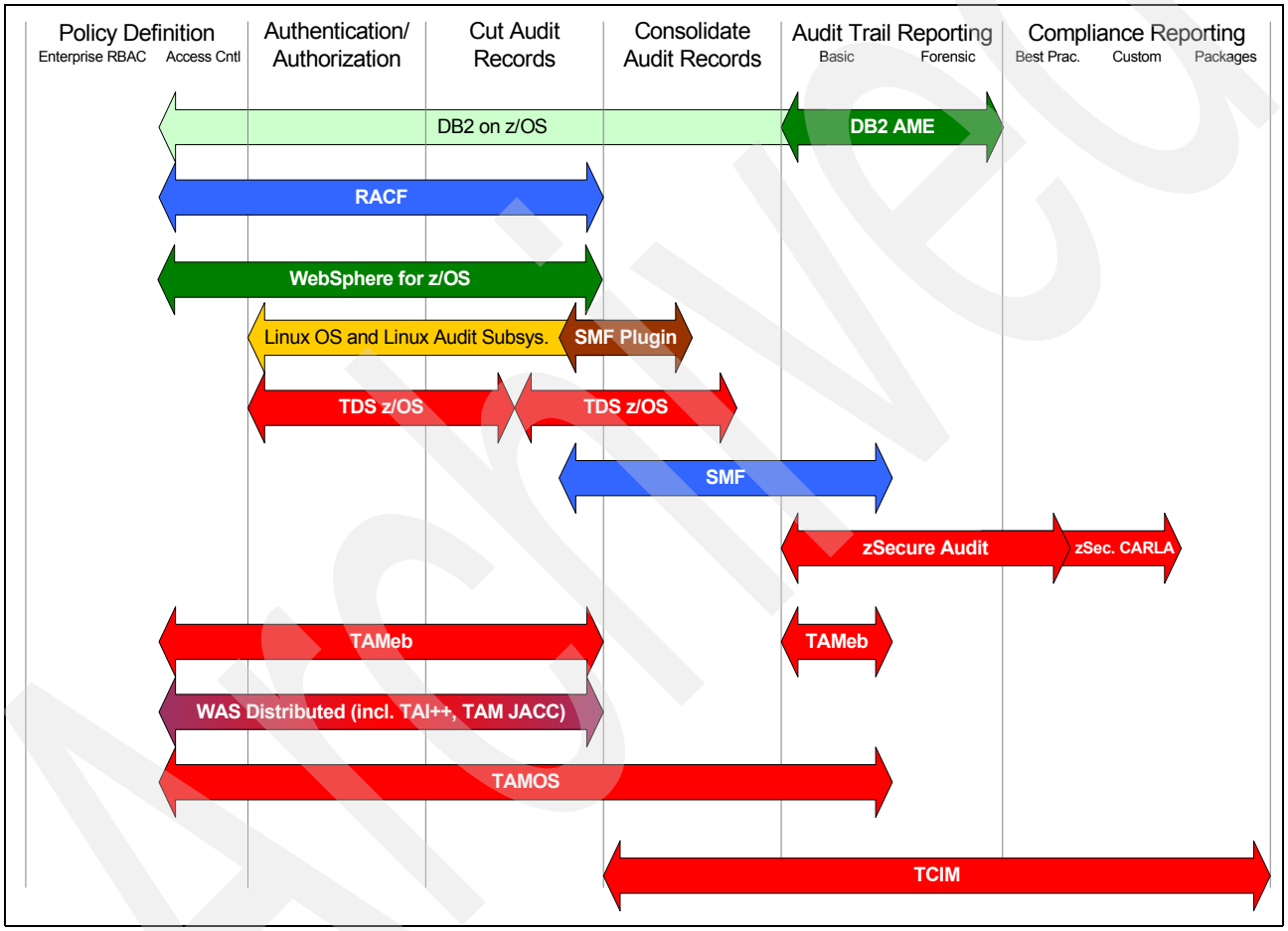


Figure 7-2 Auditing approach and tools

Based on this configuration and on the auditing flow, we ran a few scenarios to demonstrate how and where the auditing can take place and how an installation can easily relate and aggregate the auditing process. For more information about the audit data flow see Chapter 8, “e-business scenarios” on page 159, and Chapter 9, “Privileged access” on page 199.

7.2 Scenarios description

In our environment we describe and test two sets of auditing scenarios:

- e-business scenarios, described more fully in Chapter 8, “e-business scenarios” on page 159, intended to simulate e-business access to a company’s core business applications and data. We show that we can determine, for audit purposes, which Web users accessed the applications and data.

  For this scenario, our environment includes two multi-tiered application streams from the user (a Web browser) through the business application to the business data (stored in DB2 tables on z/OS). One application flow runs in WebSphere Application Server on Linux, and the other runs in WebSphere Application Server on z/OS.

  For our illustrations, we used an existing sample application, SWIPEDB2. With this application, which consists of a Web application and a stateless session Enterprise Java Bean (EJB), after authenticating, the user receives a form that allows the specification of a DB2 query and other parameters. Submitting the form invokes a servlet, which calls on an EJB to invoke DB2 and return the results of the query.

  These users can be divided into two groups:

  – External users: simulating a customer authenticating and getting authorized for transactions with the company

  – Internal users: simulating an employee who is also a customer of the company and is authorized for transactions

- The privilege user scenarios (see “Privileged access” on page 199 for more details) - These scenarios cover IT personnel using privileges to operate and administer the IT environment that we use for this project. They show that we can determine who has taken actions in the environment that requires privileged access.

7.2.1 External users: a customer of the company

An external user comes in to check his account and do some updates. We assume that the company has intrusion detection systems.

Browse scenario

The user authenticates with the uid of Usera and the pw assigned to Usera. The assumption is that the customers (external users) are not defined with a user ID in the mainframe security system, which in our case is RACF.

The user is authenticating the ID Usera through the Tivoli Access Manager ID (which is using LDAP as the user registry) on a system deployed on Linux towards an LDAP placed in the secure zone. Technically, the user comes in via a reverse proxy, which allows the user onto the system after authentication. In our case we use TAM, which has two parts, a Webproxy and TAM user registry.

The following audit records are cut for authentication in:

- TCP/IP/intrusion detection systems (We do not go into the detail of intrusion detection systems or a managed security services.)
- WebSEAL - for authentication of the user
- LDAP - for the validation of the user's password

The user can now start to use the application. In this case the user wants to query (read) a DB2 table (modeling a situation in which a user wants to check his account balance). Usera

uid is mapped in WebSphere Application Server to a RACFuser ID by either of the following or by both:

- Trusted Association Interceptor (TAI)
- JAAS login module

We can differentiate by using IDs for read-only transactions and a more limited set of IDs for updates of the DB2 data for stronger control.

The following audit records are cut for authorization to the actual data:

- In RACF: authorization to the application components by the user ID mapped to Usera.
- In DB2: The actual read of the data is also being audited and recorded in the DB2 log since this particular user ID mapped to Usera is being audited (and recorded).

Update scenario

Our user now wants to update a DB2 table (modeling the situation where the user is making a deposit).

Since the user is still authenticated, the user can continue in his current session, although the policy might trigger 2-phase authentication. This can, for example, be authenticating again with his digital certificate, fingerprint, or secure ID.

The same scenario as reading data apples, although the security policy for updates might trigger other mapping mechanisms to another user ID used for updating the DB2 data.

7.2.2 Internal user: an employee of the company

A standard employee of the company wants to check her account. The assumption is that in this scenario, the employee of the company is defined with a user ID on the mainframe security system, which in this case is RACF.

The user authenticates with the uid of Testu1 and the pw assigned to the RACF user ID belonging to this user. The authentication occurs in WebSEAL deployed on Linux, which has as its registry an LDAP placed in the secure zone. In this case, technically, the user comes in via a reverse proxy and allows her onto the system after authentication.

In this case audit records are cut in the:

- TCP/IP/intrusion detection systems (We do not go into the detail about intrusion detection systems or a managed security services.)
- WebSEAL - for authentication of the user
- LDAP - for the validation of the uid

Up to this point the process was similar to the external user scenario.

When WebSEAL looks up the uid LDAP, the internal user will be using Native Authentication with her RACF ID. The difference from the external scenario is that authentication will take place using her RACF password rather than a password stored in LDAP. We expect an additional audit record in RACF for the password authentication.

7.2.3 Privileged user scenarios

Since the administrators and privileged users are the most powerful roles, we show how the administrator's actions can be monitored through the middlewares. The privileged user test

cases are detailed in the relevant sections in Chapter 9, “Privileged access” on page 199. Before exploring the scenarios, we summarize which privilege users are usually defined in each middleware. The following privileges exists on the different systems.

Z/OS

On z/OS, RACF has three defined privileges categories:

- SYSTEM SPECIAL
- OPERATIONS
- AUDIT

The SYSTEM SPECIAL attribute

SYSTEM SPECIAL is a privilege given to a user through the TSO ID to administer the security environment on z/OS for the installations using RACF.

A user with SYSTEM SPECIAL can define all resources to be protected by RACF on a z/OS system, and will give access to them according to the company’s policies and processes. A user with SYSTEM SPECIAL can also define other users with privileges on other subsystems if they are protected and defined in RACF using external security.

Important: All activity done by a user, either via his TSO ID or for a started task, with SYSTEM SPECIAL or GROUP SPECIAL assigned to it should be audited.

The OPERATIONS attribute

OPERATION is a privilege given to access any data with any access without a specific defined access list. If however, a resource is defined with access lists and the ID with the OPERATIONS attribute is defined with access = NONE, no access to this particular resource is given.

The OPERATIONS attribute is mostly used for specific started tasks or IDs belonging to operations planners who use their IDs for submitting batch jobs for production.

Important: The OPERATIONS attribute should be very carefully used, and needs full auditing. We do **not** recommend usage of the OPERATIONS attribute. If it is used within your environment, start to monitor/audit the users who have OPERATIONS and give them access via resource access lists instead.

Started tasks IDs should *not* be defined with the TSO Segment.

The AUDIT attribute

AUDIT is a privilege given to users performing audit on z/OS systems.

An auditor’s responsibility is to verify that RACF is meeting your installation's security policies and processes. As an auditor, you can check RACF controls on a user or on a system-wide level or the group-AUDITOR attribute (with responsibility for checking RACF controls for a group and its subgroups). A user with the group-AUDITOR attribute can only monitor the users and resources owned by a specific group and its subgroups.

Logging and recording data about specific events is the key to auditing the use of RACF at your installation. You must ensure that RACF logs the information that you need. RACF uses the system management facilities (SMF) to log data about various RACF events.

There are also specific utilities that require the audit attribute by the submitter. Certain events are always logged in RACF, for example, the RVARY command. See the RACF Auditors Guide for more information.

DB2

If DB2 is customized for RACF, all authorities can be administered via RACF instead of using internal security within DB2. Customizing DB2 for RACF is described in 6.4, “Use of RACF for DB2 security” on page 133.

Following is a brief description of the different levels of authorization that you can have in a DB2 subsystem. For more details on the DB2 system authorities see the DB2 Administrator Guide.

- SYSOPR: system operator. This can issue most commands.
- Installation SYSOPR: used for installation of DB2. The authority is not recorded in the DB2 catalog. Therefore, the catalog does not need to be available to check the installation SYSOPR authority. No ID can revoke the authority. It can be removed only by changing the module that contains the subsystem initialization parameters (typically DSNZPARM). IDs with Installation SYSOPR authority can also:
  - Access DB2 when the subsystem is started with ACCESS(ACCESS(MAINT).
  - Run all allowable utilities on the directory and catalog databases (DSNDB01 and DSNDB06).
  - Run the REPAIR utility with the DBD statement.
  - Start and stop the database that contains the application registration table (ART) and the object registration table (ORT).
- PACKADM: package administrator. Has all package privileges on all packages in specific collections. Has the CREATE IN privilege on those specific collections.
- DBMAINT: Database maintenance authority allows privileges over a specific database to an ID. DBMAINT can perform the following actions within a specific database:
  - Create objects.
  - Run utilities that do not change **`data v Issue`** commands.
  - Terminate all utilities on the database except DIAGNOSE, REPORT, and STOSPACE.
- DBCTRL: Database controller authority includes DBMAINT privileges over a specific database.
- DBADM: Database administrator authority includes DBCTRL privileges over a specific database. Additionally, DBADM has privileges to access any tables in a specific database by using SQL statements. DBADM can also perform the following actions:
  - Drop or alter any tablespace, table, or index in the database.
  - Issue a COMMENT, LABEL, or LOCK TABLE statement for any table in the database.
  - Issue a COMMENT statement for any index in the database.
- SYSCTRL: The system controller authority is designed for administering a system that contains sensitive data. The system controller has nearly complete control of the DB2 subsystem. However, the system controller cannot access user data directly unless the privilege to do so is explicitly granted. SYSCTRL can:
  - Act as installation SYSOPR (when the catalog is available) or DBCTRL over any database.
  - Run any allowable utility on any database.
  - Issue a COMMENT ON, LABEL ON, or LOCK TABLE statement for any table.

– Create a view on any catalog table for itself or for other IDs v Create tables and aliases for itself or for others IDs.

Important: SYSCTRL authority is intended to separate system control functions from administrative functions. However, SYSCTRL is not a complete solution for a high-security system. If any plans have their EXECUTE privilege granted to PUBLIC, an ID with SYSCTRL authority can grant itself SYSADM authority. The only control over such actions is to audit the activity of IDs with high levels of authority.

- SYSADM: System administrator authority includes *all* SYSCTRL, PACKADM, and DBADM privileges, including access to all data. SYSADM can perform the following actions and grant other IDs the privilege to perform these actions:
  - Use all the privileges of DBADM over any database.
  - Use EXECUTE and BIND on any plan or package and COPY on any package.
  - Use privileges over views that are owned by others.
  - Set the current SQL ID to any valid value.
  - Create and drop synonyms and views for other IDs on any table.
  - Use any valid value for OWNER in BIND or REBIND.
  - Drop database DSNDB07.

Important: The SYSADM privilege can grant privileges to other IDs by granting them SYSADM authority.

TAM administrators

The Tivoli Access Manager suite of products, such as Tivoli Access Manager for e-business (TAMeb) and Tivoli Access Manager for Operating Systems (TAMOS), contain accounts and groups that are used to administer the system and are considered privileged accounts.

The TAM products provide for a single *super* administrator account that is called "sec_master". This account can perform all actions within a TAM environment, such as modifying policy, changing user access rights, and resetting passwords. Thus, this account carries a risk of accidental or malicious damage to the TAM resources and could give users more access to systems than they need, or could hijack ordinary accounts to hide unwarranted access. However, the audit trail is held outside of TAM in operating system files, so a separate operating system account is required to tamper with the audit trail. Additional accounts can be made administrators by adding them to the iv-admin group. This is described in Chapter 4, “Default Security Policy,” in the *TAMeb V6.0 Administration Guide*, SC32-1686.

Within TAMOS there is an administrative group called osseal-admin, which is a group within TAM (UNIX group osseal). All UNIX/Linux accounts that are to be subject to TAMOS policy are defined within TAM as TAM users, as well as being operating system users. To perform TAMOS administrative functions, the user account must be a member of this TAM group. This does not give the user the rights to access the TAMOS audit files and commands. To do this a user must be a member of the osseal-auditors group (UNIX group ossaudit). Care needs to be taken when users are members of both groups, as that would mean that an administrator could perform a malicious change and cover his tracks by altering the audit trail. These are defined in the “Users and Groups” section of Chapter 3, “Runtime,” in the *TAMOS 6.0 Administration Guide*, SC32-1709.

In addition, there will be operating system accounts to run the services or daemons that should be subject to the normal system account monitoring, such safer database-owning accounts or application-owning accounts.

The privileged user scenarios for TAM as for are detailed in 9.3 TAM sysadmin on p189. The test cases covered are:

- Listing the TAM administrators by listing the members of the iv-admin group
- Creating a new administrator and adding it to the iv-admin group
- Changing TAM policy
- Changing TAM auditing
- Changing TAMOS auditing
- Administration by another iv-admin user

The audit records for these activities were checked in the TAM audit logs.

ITDS (LDAP) administrators

The different flavors of Tivoli Directory Server (see 2.2, “Tivoli Directory Server (TDS)” on page 29) have different administrative users.

The distributed ITDS allows the installer to specify an administrative account, such as cn=root or cn=administrator. This is usually referred to as the Tivoli Directory Server *administrator*, *server administrator*, or *Primary directory administrator*. The primary directory administrator is associated with a specific user account. There is only one primary directory administrator account for the LDAP server. The primary directory administrator has full rights to manage the LDAP server.

The administrator can also define administrative groups and grant the relevant administrative access to these groups to provide for distributed or delegated administration. If the AuditAdmin role is granted to the administrative groups, they have unrestricted access to the – Audit log, Admin Audit log, All other server logs, Audit log settings (cn=Audit, cn=Log Management, cn=Configuration), Admin Audit log settings (cn=Admin Audit, cn=Log Management, cn=Configuration), and Default log management settings (cn=Default, cn=Log Management, cn=Configuration). Any member of an administrative group with this role could modify the audit trail to hide their tracks and should be monitored and audited. Further information about creating administrative groups can be found in the section “Creating the administrative group” in Chapter 12, “Securing directory access,” of the *Tivoli Directory Server V6.1 Administration Guide*, SC32-1564. Note that both the administrator accounts and administrative groups are entries within LDAP and do not relate to operating system users or the database owner account.

The z/OS ITDS has a single administrator specified in the ADMINDN keyword in the ds.profile configuration file. A RACF user ID can be mapped to the ADMINDN. Normally, this DN has unrestricted access to all entries in the directory except for entries in back ends that are read-only replicas. When the LDAP server is in maintenance mode, the LDAP administrator has unrestricted access to all entries in the directory. There will also be an operating system account (such as a RACF account) for running the started tasks and owning the ITDS data sets. The z/OS ITDS does not have administrative groups.

The privileged user scenarios for the z/OS ITDS administrator are detailed in 9.2, “z/OS ITDS LDAP administrator DN” on page 213. We did not have a distributed ITDS in our environment to test auditing. The test cases covered are:

- Modifying the schema to add new attribute types and object classes
- Adding a new user
- Modifying the new user object to add additional attributes
- The audit records for these test cases were checked in SMF

UNIX/Linux

All UNIX/Linux systems have the root user—a user with UID=0. This user is the system administrator of the system and has unrestricted and often un-audited access to every resource on a UNIX/Linux system. The exposure with this account is that it is often required to perform user-level tasks, so ordinary users are often given access to root, and there is a risk of malicious or accidental damage to the system due to this high level of access. To compound the problem, root is often exempt from auditing, and even if the actions of root are audited, the account can go and modify the audit files.

The privileged user scenarios for the Linux root account are detailed in 9.4, “Linux privileged (uid=0)” on page 235. The test cases covered are:

- Auditing root logins
- Tracking changes to the user database

Many products, such as Tivoli Access Manager for Operating Systems, address this exposure by removing the need to give users root access. There is also a root group (GID=0) that can give ordinary users too much access.

The privileged user scenarios for the UNIX/Linux root account on a TAMOS-controlled system are detailed in 9.5, “UNIX/Linux privileged user with TAMOS (uid=0)” on page 245. The test cases covered are:

- Creating another privileged user
- Logging in as a new user and performing suspect activity
- Root attempts to hide an audit trail
- Tracking su use
- Tracking sudo use

The audit records for these test cases were checked in the TAMOS audit facility and TCIM.

WebSphere Application Server

Every WebSphere Application Server installation that is enabled for security defines one or more users with privileges. The privileges are assigned according to predefined roles. (WebSphere Application Server has two supported interfaces for performing administrative functions. One is the administrative console, and the other is a command and scripting interface known as wsadmin.) In Version 6.1, these roles include:

Monitor Users in this role can view the WebSphere Application Server configuration and the current state of the application servers. Depending on the sensitivity of the configuration information, this role may not be considered privileged at all.

Operator An operator has, in addition to monitor privileges, the ability to change the runtime state, such as by starting and stopping application servers.

Configurator In addition to monitor privileges, a user with the configurator role can make most routine changes to the WebSphere Application Server configuration. For example, a configurator can create resources, install and uninstall applications, and customize certain aspects of the security configuration, such as Secure Sockets Layer or Common Secure Interoperability Version 2. Actions by a configurator should be audited for compliance with installation policy.

Administrator This is the most powerful WebSphere Application Server role. The user belonging to this role has all of the privileges of an operator and a configurator. In addition, this person is authorized to perform

global administrative tasks such as configuring authentication and authorization mechanisms, enabling or disabling security within WebSphere Application Server, and generating cryptographic keys. In some installations, one of the users belonging to the administrator role will be the owner of the configuration file tree. Clearly, persons with administrator user IDs must be deserving of a high degree of trust, and their actions have to be subject to audit.

Deployer Users in this role can perform both configuration and operation actions, similar to operators and configurators, but are limited to performing such operations on applications. The deployer role is limited to the wsadmin interface. The actions of the deployer should be subject to audit.

Adminsecuritymanager This role is useful when an installation wants to control administrative roles in a fine-grained manner, such as at an individual application level. An Adminsecuritymanager can create administrative authorization groups and map users to administrative roles within authorization groups.

8

e-business scenarios

This chapter contains a description of the e-business scenarios that we exercised to show capabilities for auditing the activities of external users and company employees accessing core applications and data hosted in WebSphere Application Server and DB2, respectively.

We used two scenarios, reflecting two common situations found in many enterprises. The first, referred to as *external user scenario*, shows the situation where a remote user accesses the application through a proxy. Since the remote user is usually not an employee of the enterprise, there may need to be a mapping of the remote user's identity to an identity that is meaningful to the application, and especially to DB2. One of the security challenges we faced was when and where to perform the mapping and how to audit the mapping that had taken place (and hence which remote user ID may have been responsible for a given DB2 query or update). The second, referred to as *internal user scenario*, illustrates the case where an internal user, often an employee of the enterprise, accesses the application locally.

Because there are often significant differences in how z/OS handles certain security and identity processing from the way in which it is managed by distributed platforms, each scenario is illustrated in two ways: with the WebSphere Application Server on z/OS and with the WebSphere Application Server on Linux on System z.

8.1 Scenarios with WebSphere Application Server on z/OS

This chapter describes the different scenarios that we implemented in a WebSphere Application Server for z/OS environment and report how we set up the auditing and reporting infrastructure.

8.1.1 Environment and setup

Figure 8-1 shows the setup that we used for scenarios where WebSphere Application Server is on z/OS. The user, whether internal or remote, connects to WebSEAL, which acts as a reverse proxy and authenticates the user (using a password or other credential supplied by the user). The authentication takes place against an LDAP directory. If the user is internal, there will be an indication (ibm-nativeid) in the LDAP directory entry giving the user's internal user ID in RACF. If that is present, authentication will be performed against the password for that user in the RACF database. This is known as native authentication. Otherwise, the external user is authenticated using a password stored in LDAP.

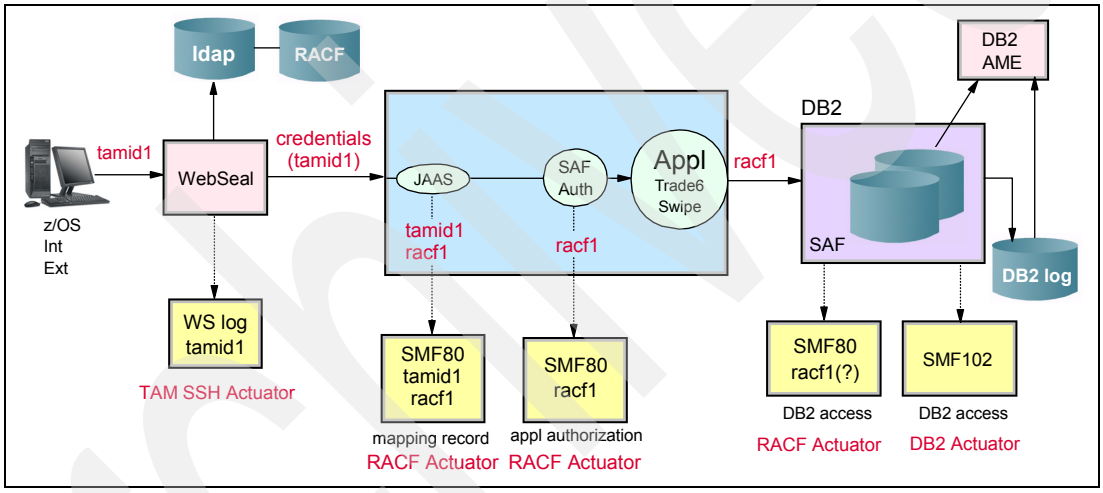


Figure 8-1 e-business scenario with WebSphere Application Server on z/OS

When the user has been authenticated (for example, by providing the correct password), the request is forwarded to WebSphere. In the z/OS case, we chose to perform the mapping of external users to internal, RACF user IDs (sometimes called technical IDs) upon first receiving the request into WebSphere Application Server. We could have done so using either a Trust Association Interceptor (TAI) or a custom mapping module installed into the JAAS login configuration. We chose the latter, which also has the capability of setting a chosen value into the x509name field of the Access Control Environment Element (ACEE). We decided to see whether placing information about the original remote user ID into this field would allow us to use auditing records to make an end-to-end correlation between the remote user ID and the RACF user ID used to access DB2.

The method we chose for performing the mapping was to have the mapping module search LDAP using the remote user's distinguished name and retrieve a special attribute created for this purpose (com-ibm-example-jaas-ExamplePluggable-Map) to hold the RACF user ID to which the remote user should be mapped. For simplicity, we decided that internal users (those with the ibm-nativeid attribute) would be *mapped* to the same RACF user IDs as are used for password native authentication. We could have chosen to map the internal users to technical IDs as well, if we did not want to individually grant internal users access to DB2 data.

Another alternative is to configure WebSphere Application Server with an LDAP registry instead of the RACF registry signified by choosing LocalOS. In that case, the mapping could be delayed until the call to DB2. This is essentially the approach used for the case with WebSphere Application Server on Linux on System z. We chose the LocalOS approach because WebSphere Application Server enforces using the same registry for both application and administration users, and many enterprises require the administration of WebSphere Application Server on z/OS to be secured with RACF. We also noted that the Linux on System z illustration uses LDAP as the configured user registry.

Our DB2 installation uses RACF to secure databases and tables. Many installations use DB2 GRANT statements, but we decided that it was often advantageous for security to be administered in a centralized way by a security function rather than a database administrator. WebSphere Application Server for z/OS V6.1 does not support the protection of DB2 objects through the DSNX@XAC exit,[1] which means that it does not automatically pass the caller's user ID to DB2 to RACF. In order to avoid this restriction, we coded our Enterprise Java Bean components to request SyncToOsThread by creating an env-entry element in the deployment descriptor that sets com.ibm.websphere.security.SyncToOSThread to true (Example 8-1).

Example 8-1 Excerpt from ejb-jar.xml

```
<env-entry>
 <description>Causes this bean to sync the OS thread to the RunAs
caller</description>
 <env-entry-name>com.ibm.websphere.security.SyncToOSThread</env-entry-name>
 <env-entry-type>java.lang.Boolean</env-entry-type>
 <env-entry-value>true</env-entry-value>
</env-entry>
```

In order to allow this setting to be effective (that is, for WebSphere Application Server to create an ACEE on the thread of execution representing the caller's user ID), we needed to set the z/OS security option "Enable application server and z/OS thread identity synchronization" to true through the administration application (Figure 8-2 on page 162). This can be accomplished using the following steps:

1. Log on to the administrative application with a user ID authorized to act as an administrator.
2. Expand **Security** and click **Secure administration, applications and infrastructure**.
3. Click **z/OS security options**.
4. Check the box next to **Enable application server and z/OS thread identity synchronization**.
5. Click **Apply** and then **Save** to save the changes to the master configuration.

[1] See http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/csec_settingracf.html.

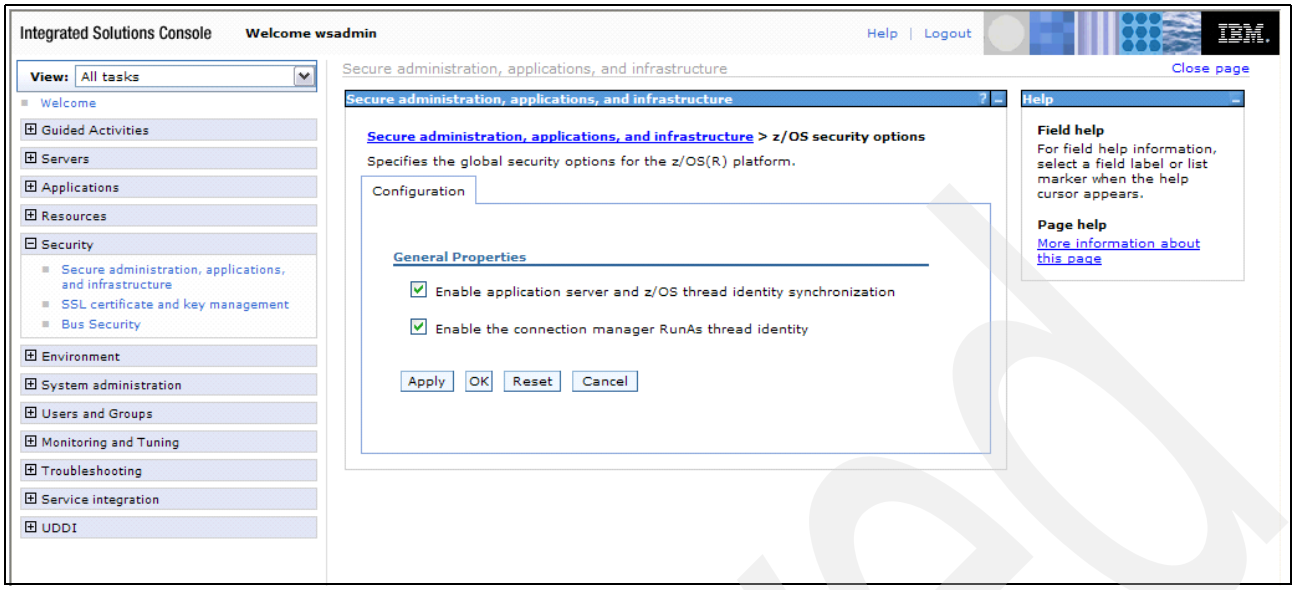


Figure 8-2 z/OS security options

We then had to set up RACF FACILITY class profiles:

```
rdef FACILITY BBO.SYNC.CL6603.CLU6603 UACC(NONE)
rdef FACILITY BBO.TRUSTEDAPPS.CL6603.CLU6603 UACC(NONE)
```

CL6603 is the cell name of our installation, and CLU6603 is the cluster name of our application server.

We then issued the following RACF commands to permit the user ID of the WebSphere Application Server controller to access the profiles:

```
permit BBO.SYNC.CL6603.CLU6603 class(facility) id(ascr1) acc(control)
permit BBO.TRUSTEDAPPS.CL6603.CLU6603 class(facility) id(ascr1) acc(control)
```

This allows WebSphere Application Server to perform the SyncToOsThread function for any application caller. We could have given the controller user ID read instead of control access. In that case, we would have required a profile in the SURROGATE class to cover each caller and to have given the controller user ID access to each such SURROGATE profile.

JAAS login module

We patterned our JAAS login module, which we called ITSOFirstLoginModule, in package com.ibm.websphere.security, after the example SampleSAFMappingModule, which can be found in the WebSphere Application Server for z/OS,Version 6.1 InfoCenter:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/csec_writesafmapmods.html

Design

We introduced some differences in order to meet our objectives of mapping users based on an attribute in the LDAP directory.

In the login() method, we determine whether the request originated at our WebSEAL server. This is necessary for two reasons. One is that we are going to request WebSphere Application Server to build a JAAS Subject and ACEE based on a mapped user ID. Since we are not authenticating that user ID, we need to ensure that the request has been

authenticated previously. We trust WebSEAL to have done that, but we do not trust any other source.

Second, there may be internal users (in our case the WebSphere Application Server administrator is an example) who log on locally and do not authenticate to the WebSEAL server. We want WebSphere Application Server to handle authentication for these users who do not have to be mapped to technical user IDs.

Once we have determined that the source of the request is trustworthy and have extracted the user ID with the help of the WSSecurityPropagationHelper class, we then invoke a separate package (see Appendix C, “Additional material” on page 331) called ExamplePluggableJAAS to look up the com-ibm-example-jaas-ExamplePluggable-Map attribute in the LDAP entry for the extracted user ID.

Finally, in the shared state, a map that is passed as an argument to the initialize() method of our login module, we set ZOS_USERID to the mapped identity (which is used to build the ACEE) and ZOS_AUDIT_STRING to the user ID extracted from the request (the user ID authenticated by WebSEAL). We also included the capability of writing information about the logging to the WebSphere Application Server SYSPRINT.

Installation

Installation of the JAAS login module involved several steps. After compiling ITSOFirstLoginModule and packaging it in a jar file, we installed the jar file in the directory {WAS_INSTALL_ROOT}/classes along with the jar file containing the ExamplePluggableJAAS mapping routines.

We also installed the file ExamplePluggable.properties in the directory {WAS_INSTALL_ROOT}/properties after configuring it to use our LDAP server.

Finally, we configured the WEB_INBOUND system login configuration to include ITSOFirstLoginModule as the first module to be invoked. Normally, SampleSAFMappingModule (the example given in the InfoCenter) is installed between ltpaLoginModule and wsMapDefaultInboundLoginModule. The SampleSAFMappingModule, however, is written on the assumption that the user ID passed by WebSEAL (or some other proxy server) is a user ID in the user registry configured to WebSphere Application Server. For WebSEAL, this would be LDAP. In our case, WebSEAL uses LDAP, and we have configured WebSphere Application Server to use LocalOS (RACF). So, we need to change (map) the incoming user ID before control reaches the IBM-supplied ltpaLoginModule.

To configure the WEB_INBOUND system login configuration, we logged onto the WebSphere Application Server administration application with an administrator user ID and selected **Secure administration, applications and infrastructure** (Figure 8-3).

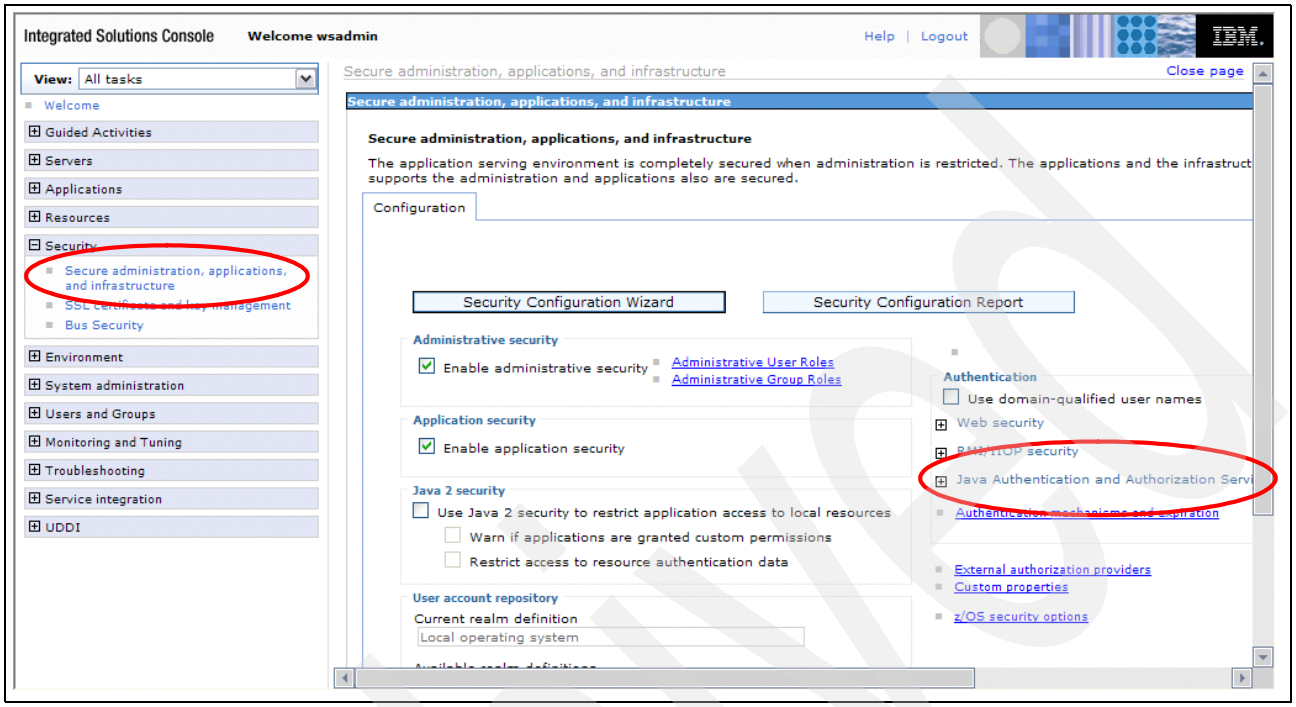


Figure 8-3 WebSphere Application Server security administration

From the Secure administration, applications, and infrastructure page, we expanded the Java Authentication and Authorization section and chose **System logins** (Figure 8-4).

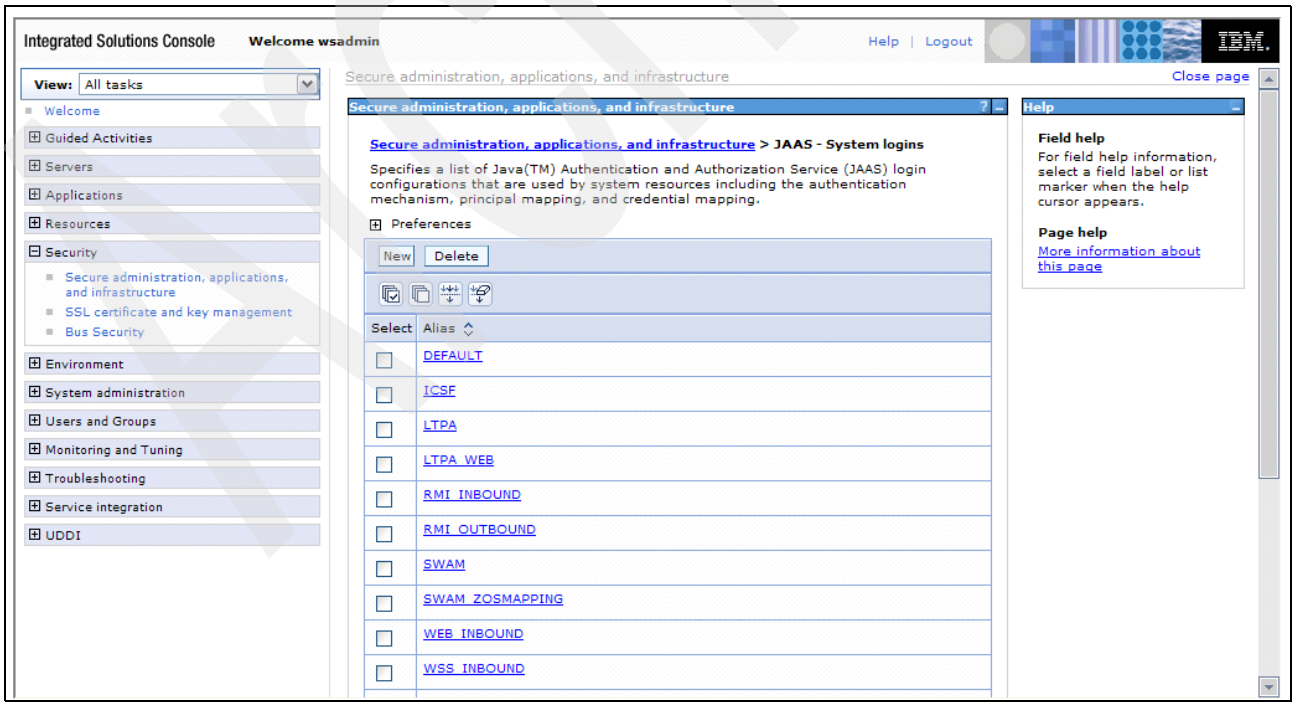


Figure 8-4 JAAS - system logins

Here we see a number of login configurations. For our purposes, we are interested only in requests coming from WebSEAL via HTML, so we decided that we need to configure only the WEB_INBOUND modules. Clicking **WEB_INBOUND** brought us to a property sheet where we selected **JAAS login modules**. We could then see the list of IBM-supplied modules for this configuration (Figure 8-5).

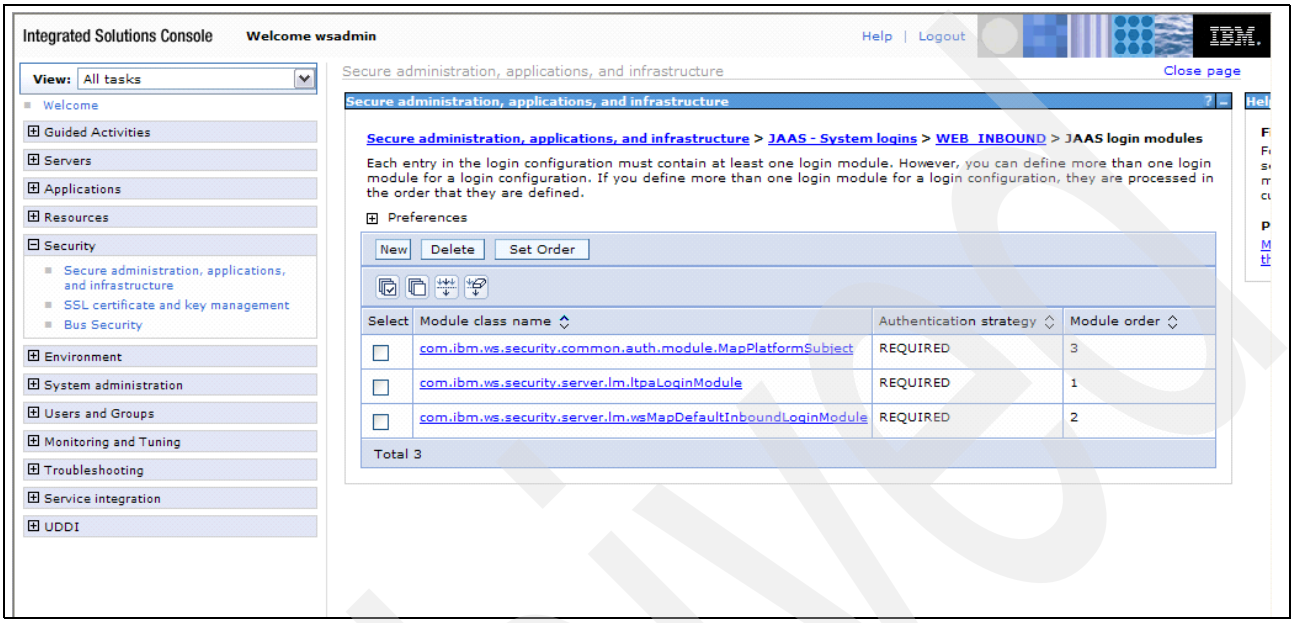


Figure 8-5 WEB_INBOUND login modules

In order to install ITSOFirstLoginModule as the first module in the configuration, we clicked **New** and completed the resulting property sheet by filling in the fully qualified (package.name) module name com.ibm.websphere.security.ITSOFirstLoginModule (Figure 8-6).

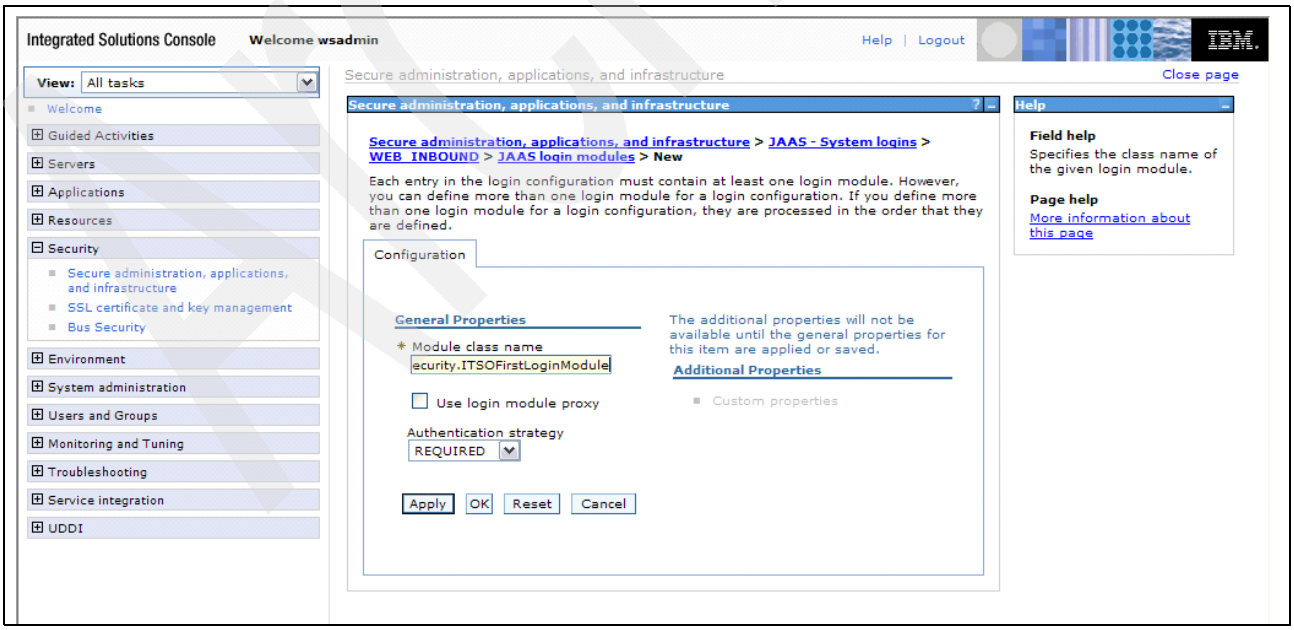


Figure 8-6 Adding a new login module

After clicking **Apply**, we could then click **Custom properties**, which displayed an empty list of properties. We clicked **New**, which brought us to a property sheet on which we entered the name `audit` and the value `true` (Figure 8-7).

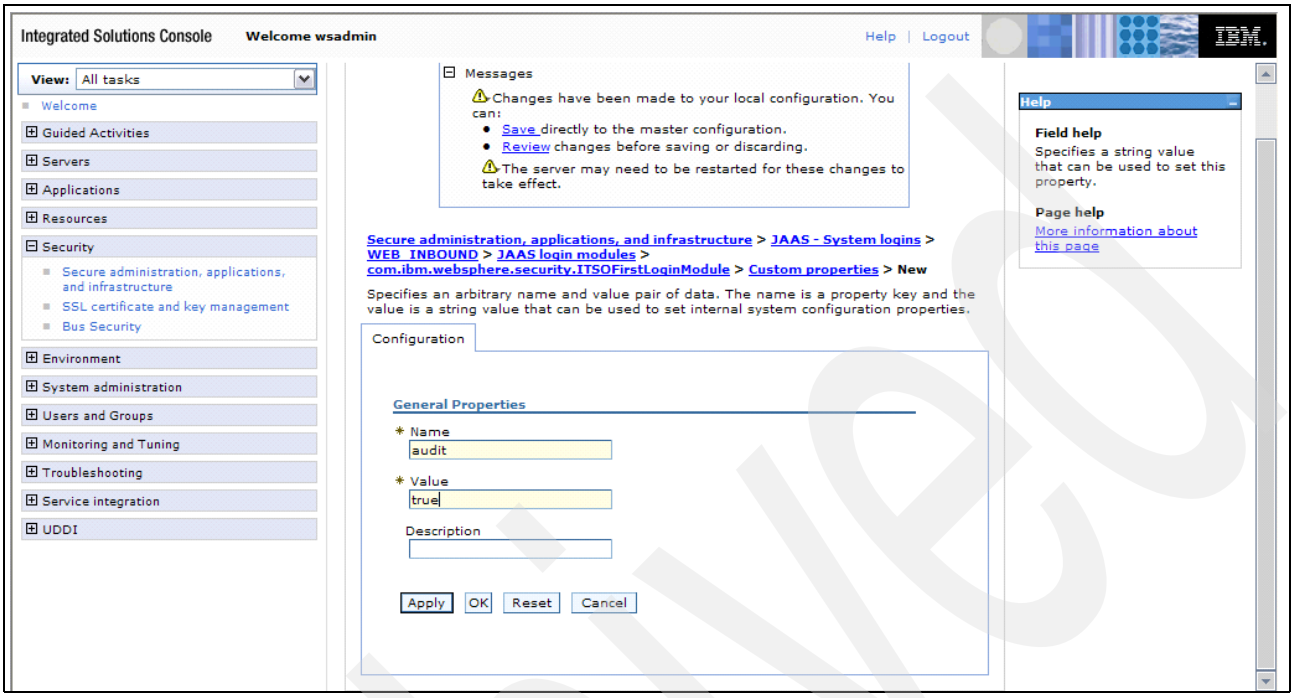


Figure 8-7 Adding custom properties

We repeated the process for the following properties:

debug	false
delegate	com.ibm.websphere.security.ITSOFirstLoginModule
useWSPrincipalName	false

audit is the property that we designated in the design of our module to cause it to write the original and mapped user IDs to the SYSPRINT file. *debug* is the property used (in the SampleSAFMappingModule and in ours) to cause the writing of messages that can be used in examining the internal workings of the module. *delegate* is a property that is required to be the full name of the actual module being installed. useWSPrincipalName is a property checked within the mapping module (again, it is similar to the use in SampleSAFMappingModule) to determine whether mapping is actually needed. A value of true indicates that no mapping is to take place.

Returning to the list of JAAS login modules for the WEB_INBOUND configuration, we could see that the ITSOFirstLoginModule was fourth in the order. We needed to make it first, so we selected it and clicked **Set Order** (Figure 8-8).

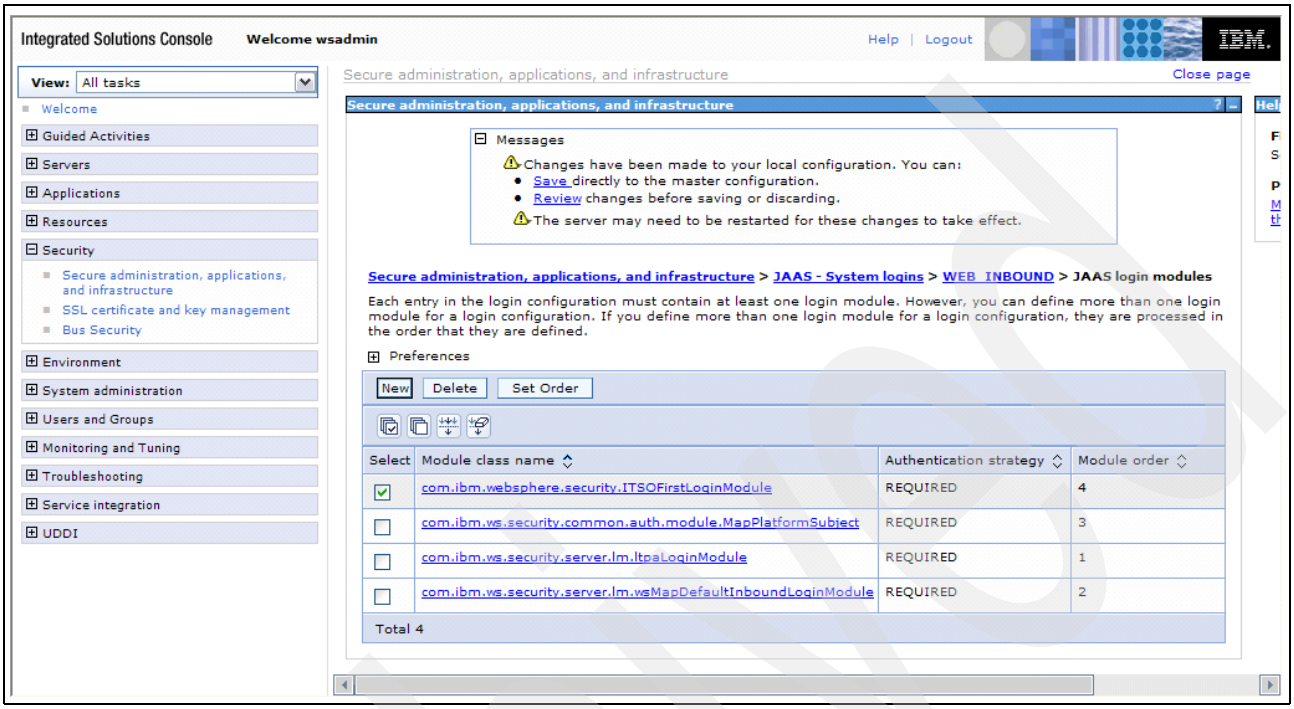


Figure 8-8 Revised module list

On the resulting page, we selected our module and clicked **Move Up** (Figure 8-9).

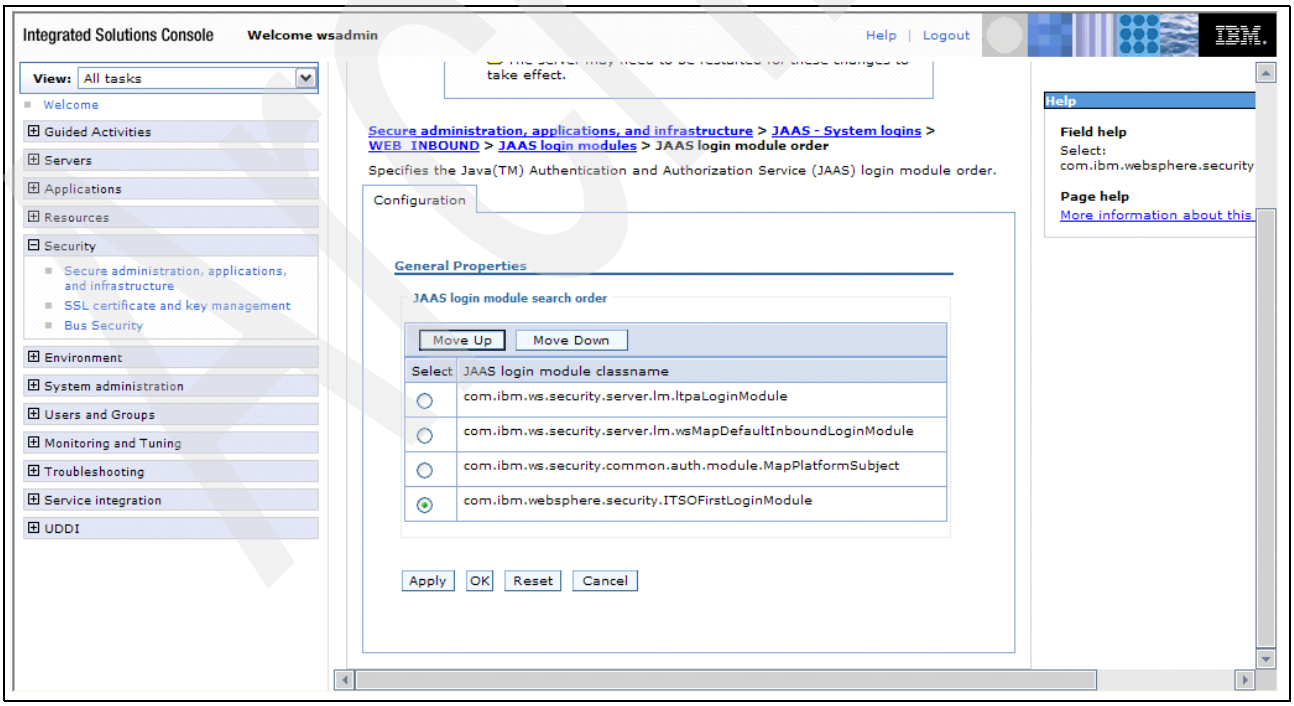


Figure 8-9 Changing module order

This moved the module up one place (Figure 8-10). We repeated the process two more times to bring our module to the top of the list, then clicked **OK**.

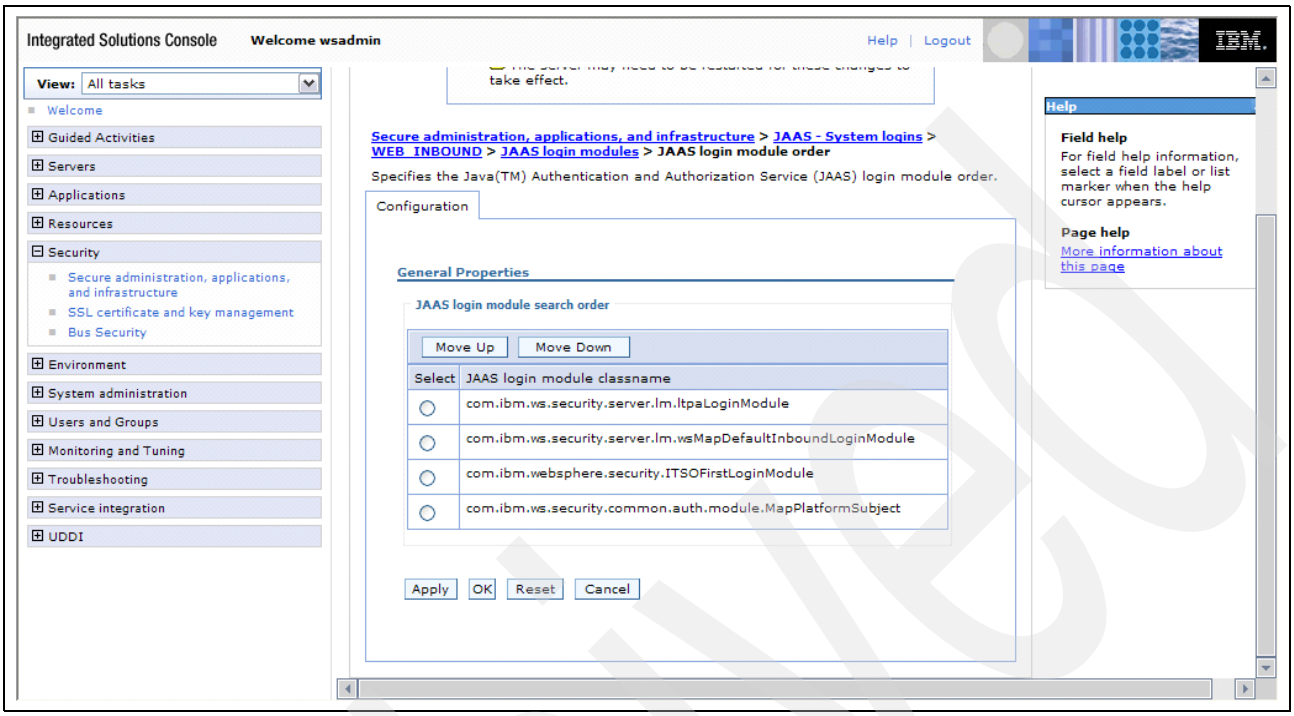


Figure 8-10 Moving up

Finally, we clicked **Save** directly to the master configuration to complete our installation.

8.1.2 External user scenario

In this section we discuss an external user scenario.

Performing the test case

Our first test with WebSphere Application Server on z/OS is to access the enterprise application SWIPEDB2 running in WebSphere Application Server on z/OS using the user ID testu2, which has no corresponding RACF password. This user ID is mapped by our JAAS login module (ITSOFirstLoginModule) to the technical user ID EXTUSRU, which is expected to have read and update access to the required DB2 table. We begin by accessing the following Web site:

`https://linux11z.itso.ibm.com/wtsc60_http/SWIPEDB2Web/`

The host field, linux11z.itso.ibm.com, represents the system on which we have installed WebSEAL. The next portion of the URL, wtsc60_http, is the name of a WebSEAL junction that will proxy authenticated requests to WebSphere Application Server on z/OS. SWIPEDB2Web is the context root of the Web application in WebSphere Application Server.

If this is the first time that we have accessed this single sign-on domain, we will be prompted to enter a user name and password. We enter the user name testu2 and the password defined for testu2 in the LDAP directory (Figure 8-11).

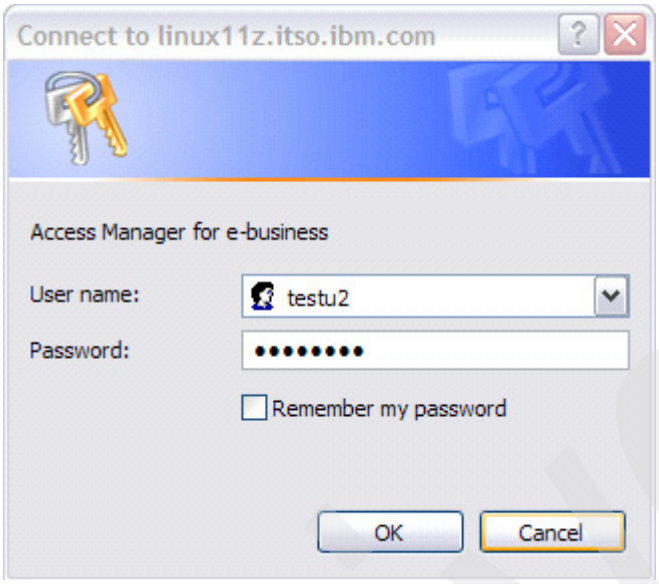


Figure 8-11 WebSEAL sign-on

Upon clicking **OK**, we are brought to the SWIPEDB2Web main page, which is a form. This application allows testing of a number of different DB2 connector configurations. We want to show a DB2 type 2 connection using container-managed authentication and no application-supplied user ID and password so that the user ID of the caller will be passed to DB2 for authorization. We have set up the JNDI name java:comp/env/jdbc/db2contJCA1 to use this kind of connector.

Since we are not expecting the application to supply a user ID or password, we leave these fields blank. In the SQL Query Statement field we enter our query in the DB2 table:

```
SELECT * from mconnol.accountprofileejb;
```

When we click **Submit** (Figure 8-12) the form is posted to a servlet that invokes the SWIPEDB2 EJB. The EJB in turn invokes DB2, passing the caller's user ID and the query that we provided. The results are displayed by a Java server page, a portion of which is shown in Figure 8-13. Note that this application is used for illustrative purposes only. The output is not in an especially readable format. We are emphasizing here only the steps to authorize and audit access to the applications and data, not the principles of good user interface design.

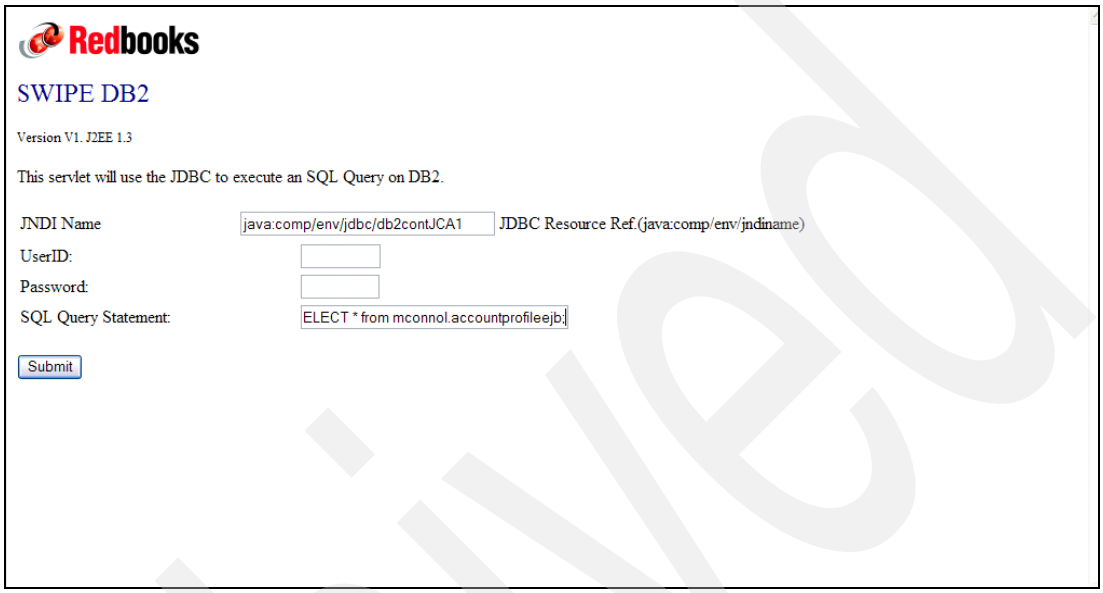


Figure 8-12 SWIPEDB2Web form

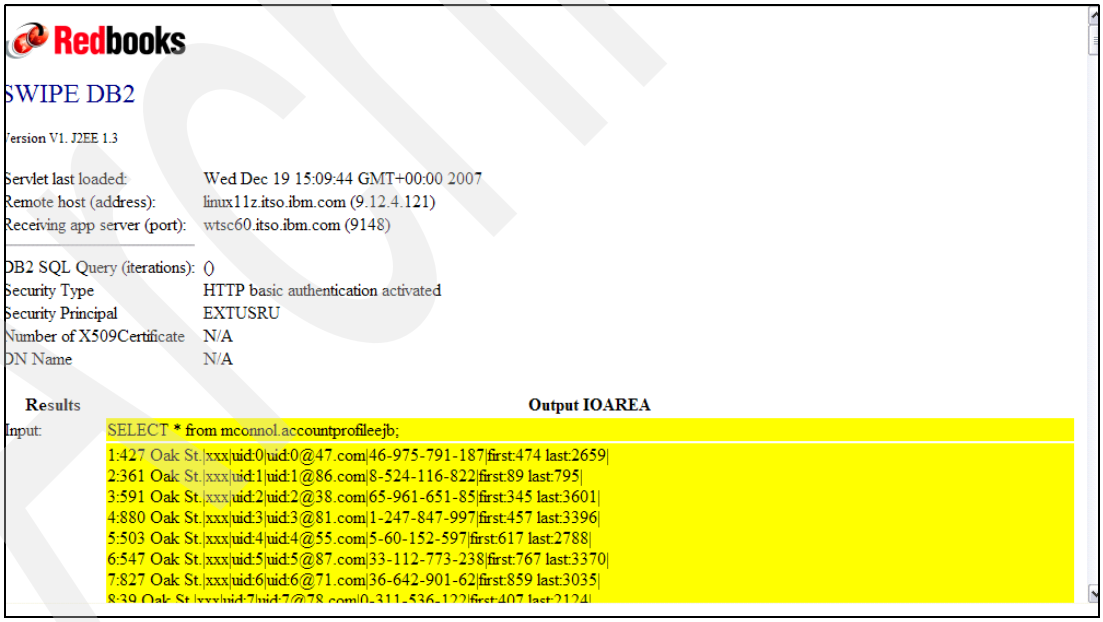


Figure 8-13 SWIPEDB2 results

Examining the audit trail

We expect to be able to use the iview component of IBM Tivoli Compliance Insight Manager (TCIM) to find evidence of this transaction. Since we are likely to be trying to answer the question *who accessed the DB2 table mconnol.accountprofileejb;?*, we might begin by looking for events that include objects looking like *MCONNOL*.

We begin by logging on to our TCIM portal and scrolling down to Database Overview (Figure 8-14).

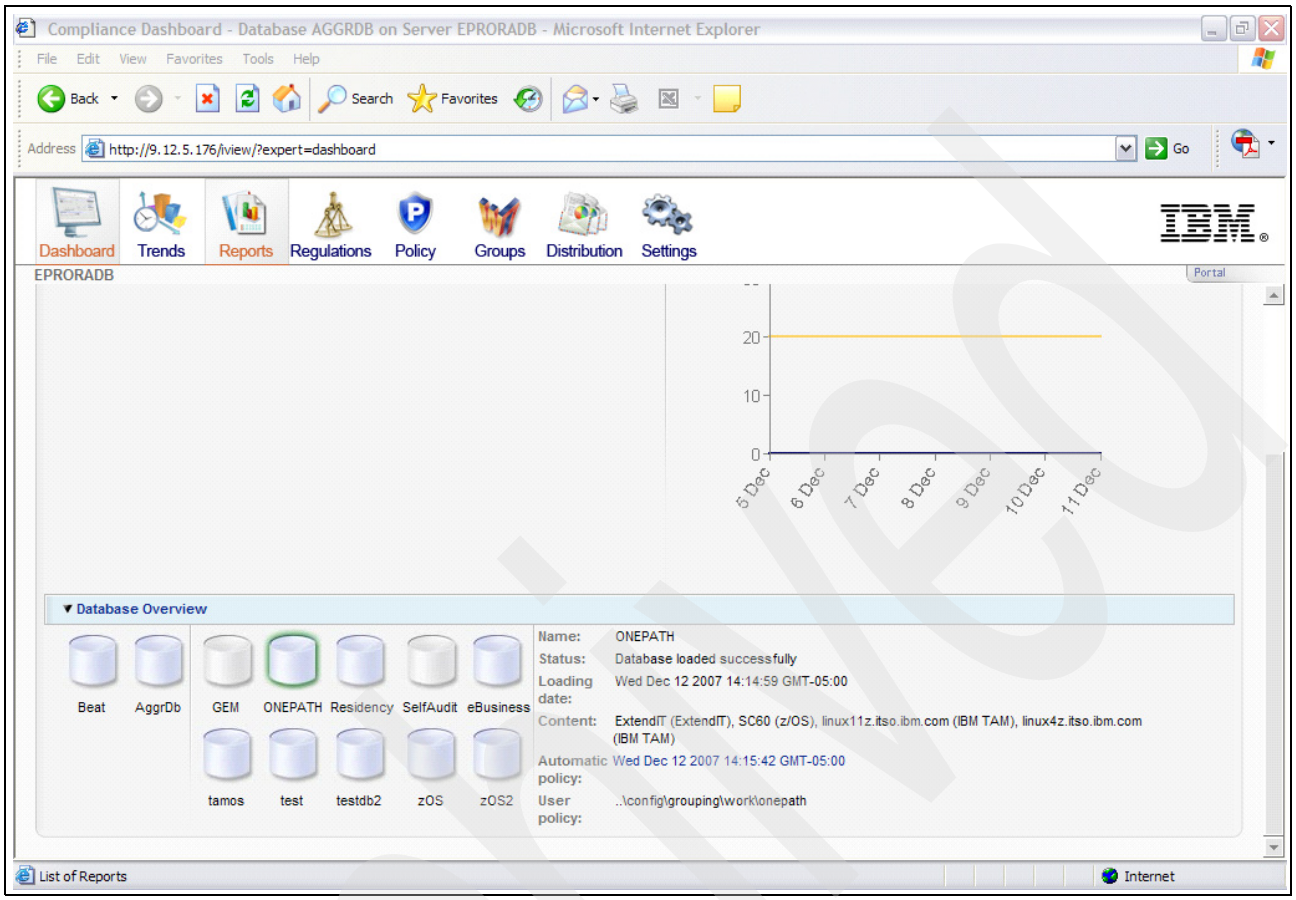


Figure 8-14 TCIM portal dashboard

Next, we click the database where our auditing records are kept, called ONEPATH. Since we have not set up the object group MCONNOL to be included on the summary page, we click the Total event list icon in the Total Events row (Figure 8-15).

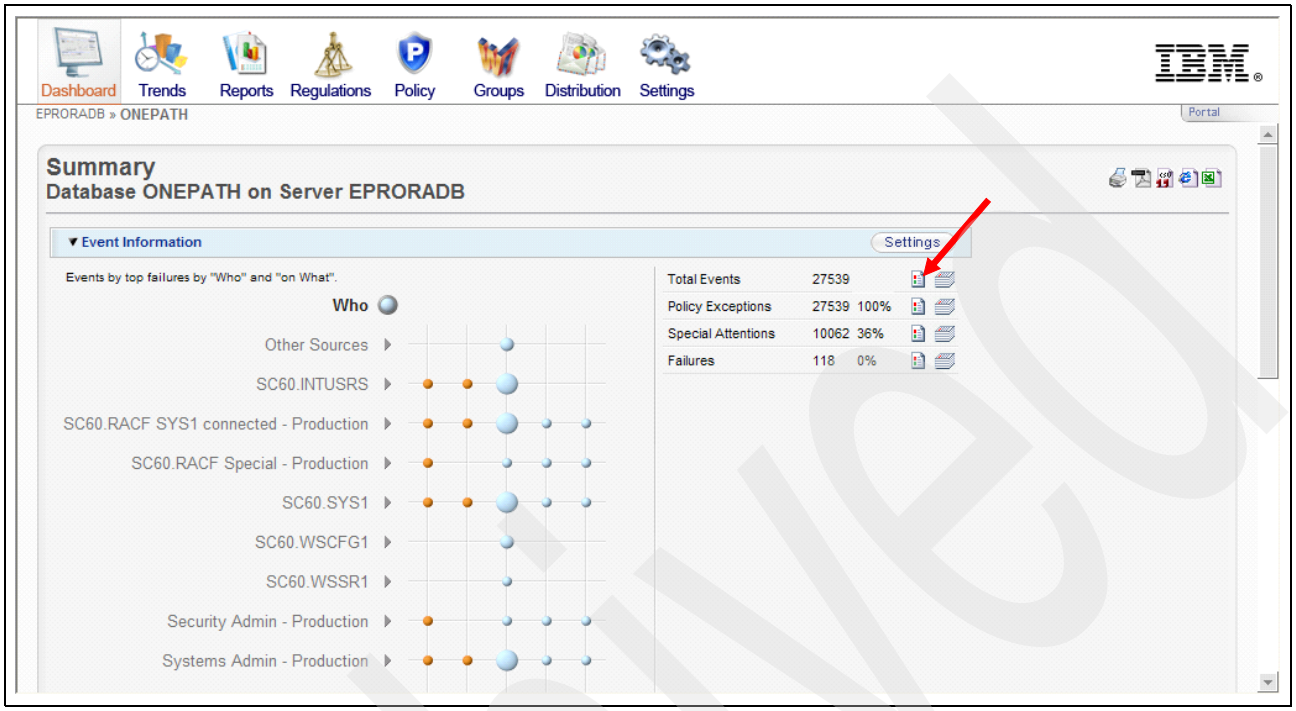


Figure 8-15 Database summary page

By clicking the small grey square at the head of the On What (detail) column, a window opens, in which we can filter on *MCONNOL* to get events on the object group of interest (Figure 8-16).

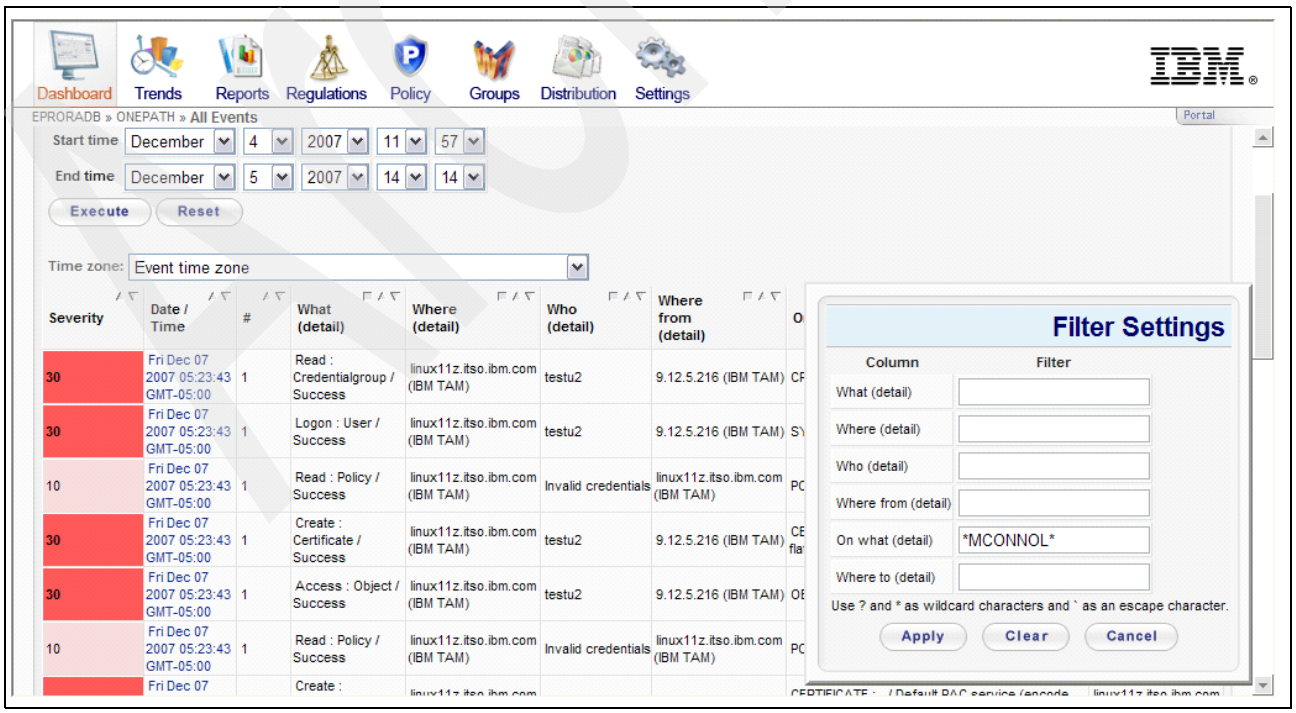


Figure 8-16 Filtering from total events list

The result shows two events of interest (Figure 8-17).

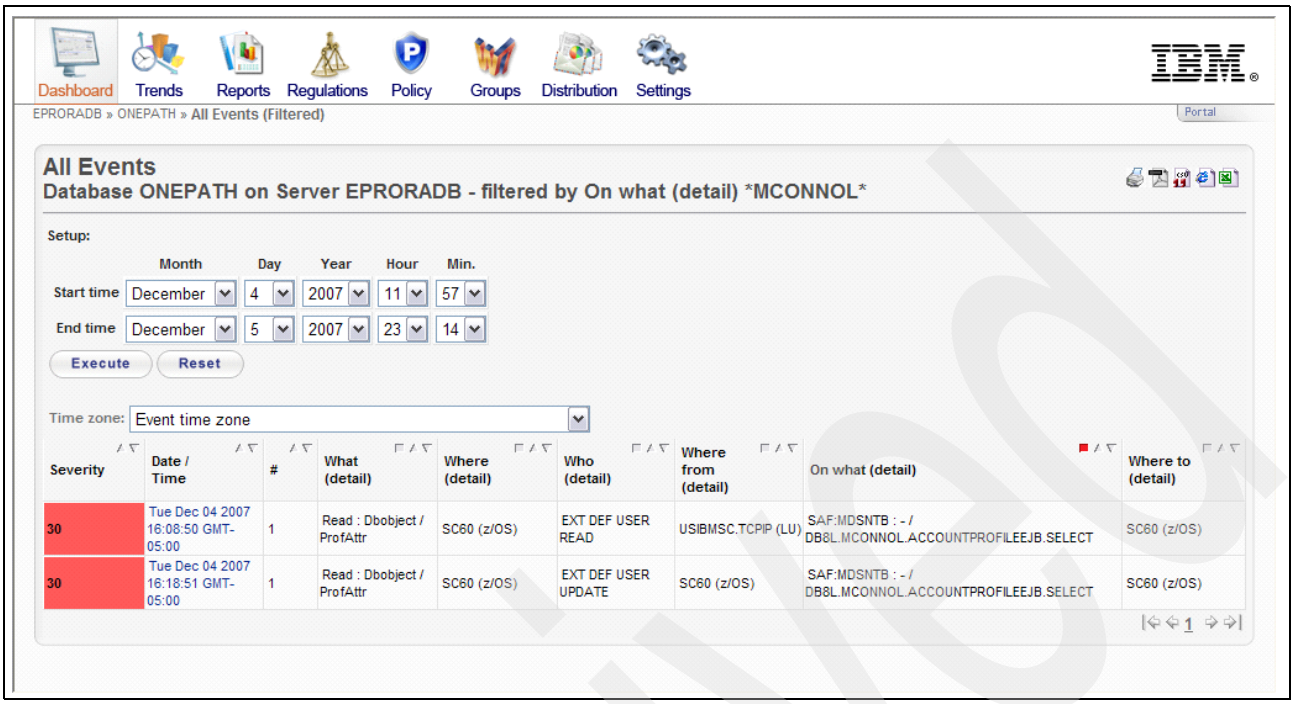


Figure 8-17 Results of filtering on database object

We clicked the second of these events to get a more detailed report (Figure 8-18).

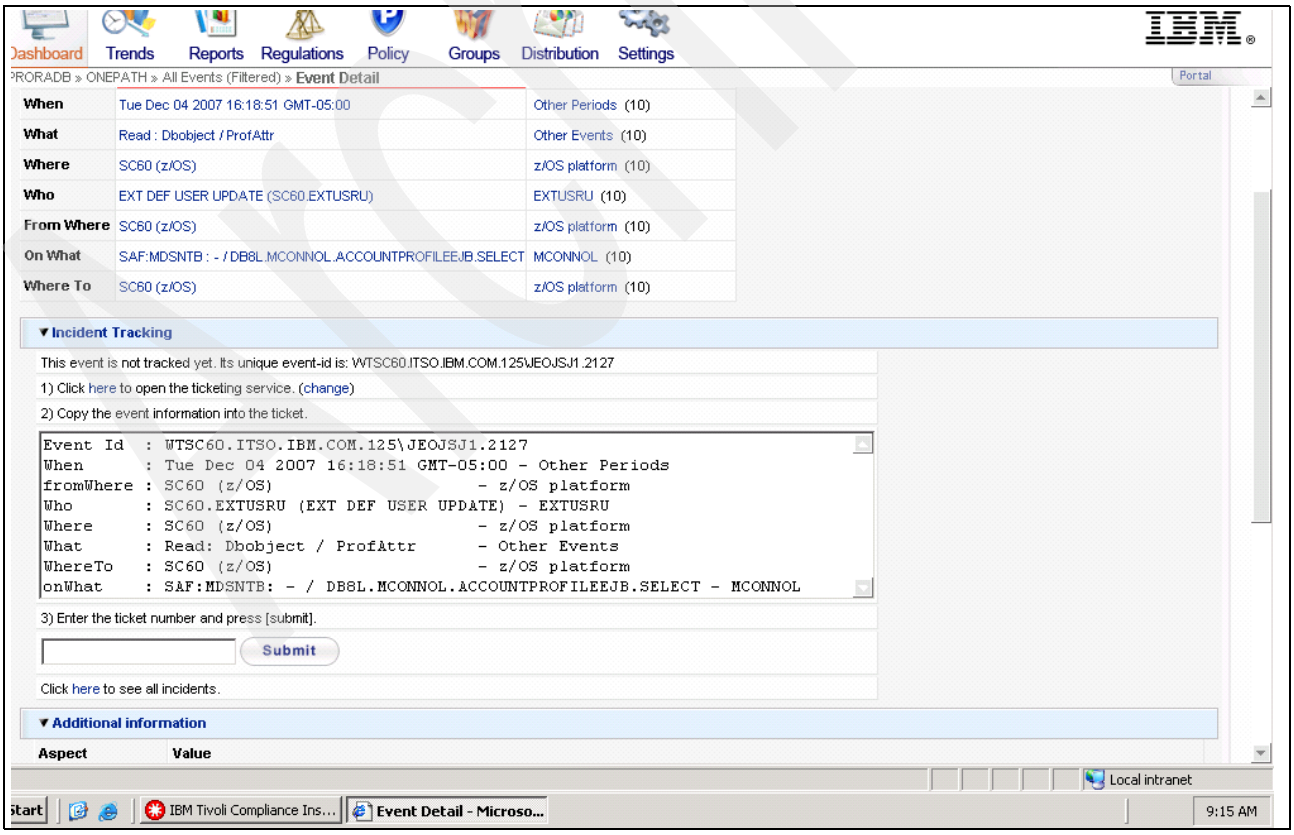


Figure 8-18 Read operation event

This shows quite clearly that at 16:18:51 on Tue Dec 04, 2007 the user ID EXTUSRU performed a read operation on DB8L.MCONNOL.ACCOUNTPROFILEEJGB in the system SC60 (z/OS). Furthermore, the from where entry shows us that the request came from SC60.

Next, we can filter on the user ID EXTUSRU and platform SC60(z/OS) to see whether there is more detail available (Figure 8-19). From the result (Figure 8-20 on page 175), we can tell that around the same time period, this user successfully accessed the EJBROLE profiles DB2ServletManager and DB2EJBManager.

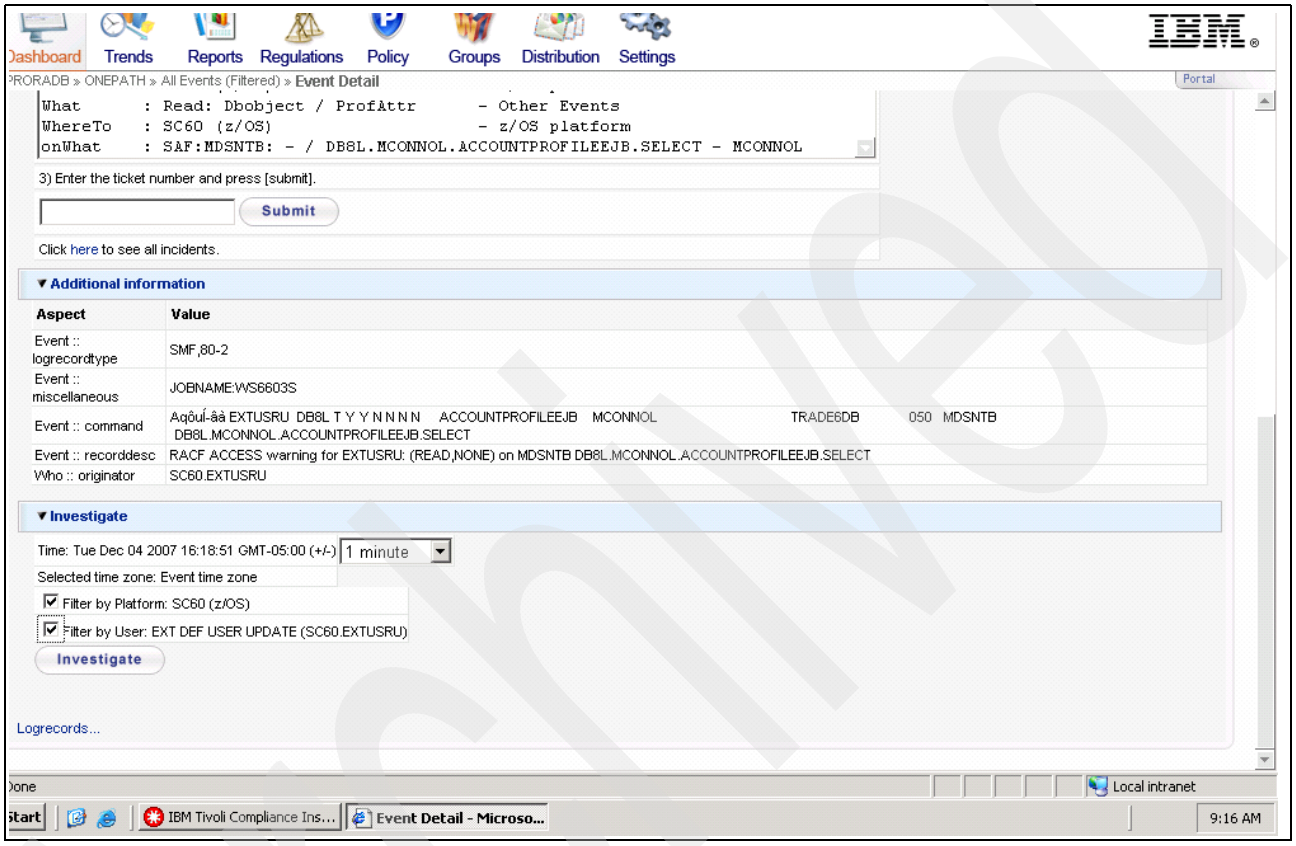


Figure 8-19 Filtering on user ID and platform

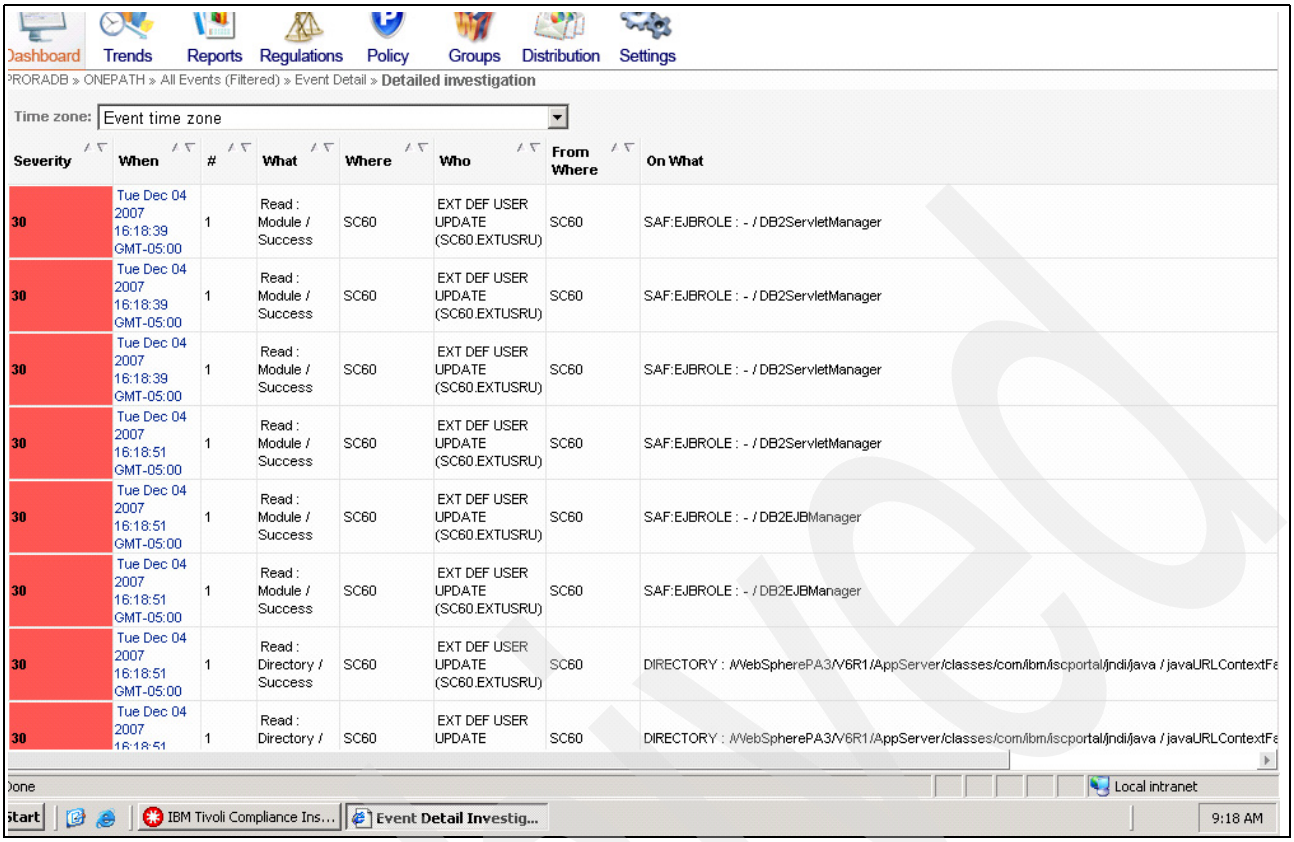


Figure 8-20 Related events

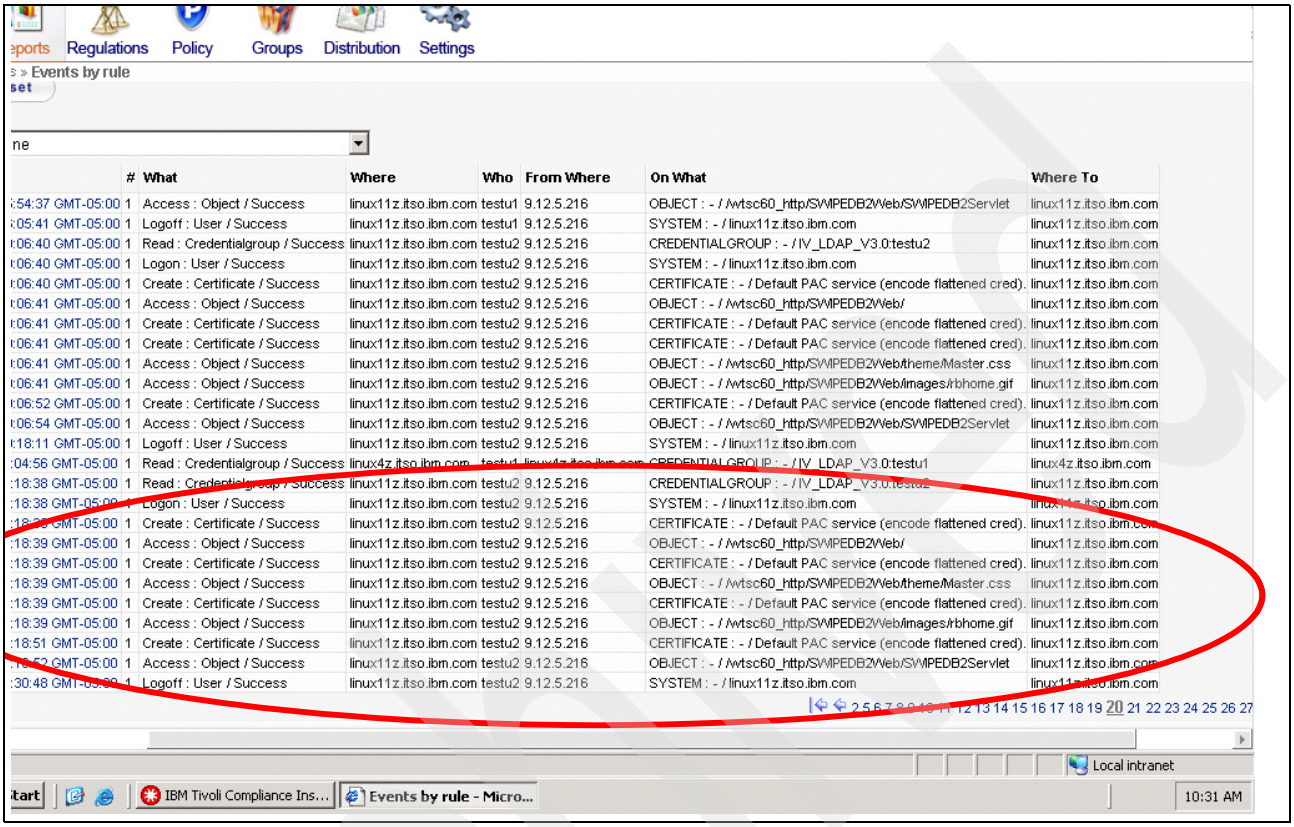
Here we encountered a problem. There does not seem to be anything in iView to tell us that EXTUSRU was the user ID mapped by testu2. We expected that by setting the field ZOS_AUDIT_STRING to the original user ID in our JAAS mapping module, we would see this value recorded in the SMF 80 records. An examination of the SMF 80 records shows that the value that we expected to be set in the ZOS_AUDIT_STRING field (knows as x500Subject) is the EXTUSRU ID.

At this point we relied on the auditing that we built into our JAAS login module (see “Design” on page 162 and “Installation” on page 163). By correlating time stamps shown in iView with records in the WebSphere Application Server servant region SYSPRINT, we were able to find the audit record created by our JAAS login module (see Example 8-2).

Example 8-2 Audit record sample from JAAS login module

```
Audit: com.ibm.websphere.security.ITSOFirstLoginModule
 Values have been stored into the shared state
Audit: com.ibm.websphere.security.ITSOFirstLoginModule
 original User ID testu2
Audit: com.ibm.websphere.security.ITSOFirstLoginModule
 mapped User ID EXTUSRU
Audit: com.ibm.websphere.security.ITSOFirstLoginModule
 principal class com.ibm.websphere.security.auth.WSPrincipal
```

This tells us that the original user mapped to EXTUSRU was testu2. We can now use this information to interrogate the date presented by iView further. Here we see records indicating that user testu2 was authenticated at our WebSEAL server and authorized to wtsc60http/SWIPEDB2Web/ (Figure 8-21).



	#	What	Where	Who	From Where	On What	Where To
:54:37 GMT-05:00	1	Access : Object / Success	linux11z.itso.ibm.com	testu1	9.12.5.216	OBJECT : - / /wtsc60_http/SWIPEDB2Web/SWIPEDB2Servlet	linux11z.itso.ibm.com
:05:41 GMT-05:00	1	Logoff : User / Success	linux11z.itso.ibm.com	testu1	9.12.5.216	SYSTEM : - / linux11z.itso.ibm.com	linux11z.itso.ibm.com
:06:40 GMT-05:00	1	Read : Credentialgroup / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	CREDENTIALGROUP : - / IV_LDAP_V3.0:testu2	linux11z.itso.ibm.com
:06:40 GMT-05:00	1	Logon : User / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	SYSTEM : - / linux11z.itso.ibm.com	linux11z.itso.ibm.com
:06:40 GMT-05:00	1	Create : Certificate / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com
:06:41 GMT-05:00	1	Access : Object / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	OBJECT : - / /wtsc60_http/SWIPEDB2Web/	linux11z.itso.ibm.com
:06:41 GMT-05:00	1	Create : Certificate / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com
:06:41 GMT-05:00	1	Create : Certificate / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com
:06:41 GMT-05:00	1	Access : Object / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	OBJECT : - / /wtsc60_http/SWIPEDB2Web/theme/Master.css	linux11z.itso.ibm.com
:06:41 GMT-05:00	1	Access : Object / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	OBJECT : - / /wtsc60_http/SWIPEDB2Web/images/rbhome.gif	linux11z.itso.ibm.com
:06:52 GMT-05:00	1	Create : Certificate / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com
:06:54 GMT-05:00	1	Access : Object / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	OBJECT : - / /wtsc60_http/SWIPEDB2Web/SWIPEDB2Servlet	linux11z.itso.ibm.com
:18:11 GMT-05:00	1	Logoff : User / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	SYSTEM : - / linux11z.itso.ibm.com	linux11z.itso.ibm.com
:04:56 GMT-05:00	1	Read : Credentialgroup / Success	linux4z.itso.ibm.com	testu1	linux4z.itso.ibm.com	CREDENTIALGROUP : - / IV_LDAP_V3.0:testu1	linux4z.itso.ibm.com
:18:38 GMT-05:00	1	Read : Credentialgroup / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	CREDENTIALGROUP : - / IV_LDAP_V3.0:testu2	linux11z.itso.ibm.com
:18:38 GMT-05:00	1	Logon : User / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	SYSTEM : - / linux11z.itso.ibm.com	linux11z.itso.ibm.com
:18:38 GMT-05:00	1	Create : Certificate / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com
:18:39 GMT-05:00	1	Access : Object / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	OBJECT : - / /wtsc60_http/SWIPEDB2Web/	linux11z.itso.ibm.com
:18:39 GMT-05:00	1	Create : Certificate / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com
:18:39 GMT-05:00	1	Access : Object / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	OBJECT : - / /wtsc60_http/SWIPEDB2Web/theme/Master.css	linux11z.itso.ibm.com
:18:39 GMT-05:00	1	Create : Certificate / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com
:18:39 GMT-05:00	1	Access : Object / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	OBJECT : - / /wtsc60_http/SWIPEDB2Web/images/rbhome.gif	linux11z.itso.ibm.com
:18:51 GMT-05:00	1	Create : Certificate / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com
:19:52 GMT-05:00	1	Access : Object / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	OBJECT : - / /wtsc60_http/SWIPEDB2Web/SWIPEDB2Servlet	linux11z.itso.ibm.com
:30:48 GMT-05:00	1	Logoff : User / Success	linux11z.itso.ibm.com	testu2	9.12.5.216	SYSTEM : - / linux11z.itso.ibm.com	linux11z.itso.ibm.com

Figure 8-21 Related WebSEAL events

Conclusion

How effectively were we able to track this use case from browser to database back end and back? Let us work through the steps in the user case and see how the logging worked:

1. The authentication and coarse-grained access activity (access to the SWIPEDB2 application) in WebSEAL was logged in the WebSEAL audit log against the TAM user (testu2). These records flowed through to TCIM.
2. The mapping of the TAM user (testu2) to RACF user EXTUSRU was performed in the custom JAAS module and logged to the WAS SYSPRINT. These records did not flow through to TCIM, which would require some form of TCIM actuator and some customization/configuration.
3. While the custom JAAS module set the ZOS_AUDIT_STRING to the TAM identity, the resulting SMF records (Type 80) showed this as the RACF user, so something in the chain has overridden the value that we set. So even though these records were written to SMF and forwarded to TCIM, they were of no use to us as they did not show the mapping. We do not know what or why the ZOS_AUDIT_STRING value was overwritten.
4. The EJB-level authorization (fine-grained authorization) was performed by the SAF Authorization module and logged to SMF (Type 80) and forwarded successfully to TCIM. We can see authorization records for this user successfully accessing the EJBROLE profiles DB2ServletManager and DB2EJBManager.

5. The JDBC™-level authorization to access the DB2 resources, as controlled by RACF, is logged to SMF (Type 80) and forwarded successfully to TCIM. We can see authorization records for this user performing a read against DB8L.MCONNOL.ACCOUNTPROFILEEJB.

So that the TAM/WebSEAL components provide useful audit logging that can flow through to TCIM, the WebSphere Application Server SAF Authorization and DB2 components provide useful audit logging through RACF to SMF, which can flow through to TCIM. The challenge is in the identity mapping components. These modules are custom, and some work is required to generate audit data, either through RACF and SMF or through another route, that can tie the TAM identity-based audit records to the RACF identity-based ones. This solution also requires customizing the LDAP identity data used in authentication and authorization, and production use of this would require procedures and tools, like Tivoli Identity Manager, to ensure that the TAM identities were mapped to the appropriate RACF identities.

8.1.3 Internal user scenario

In this section we discuss and internal user scenario.

Performing the test case

Our second WebSphere Application Server for z/OS test is almost identical to the earlier one (described in 8.1.2, “External user scenario” on page 168), except that we use the user ID usera. Usera has the ibm-nativeid attribute in its LDAP entry. This in turn points to a RACF user ID of INTUSR1. Hence, the password for usera is the password for INTUSR1. We use the same URL as in the external case and receive the same logon prompt (Figure 8-22).

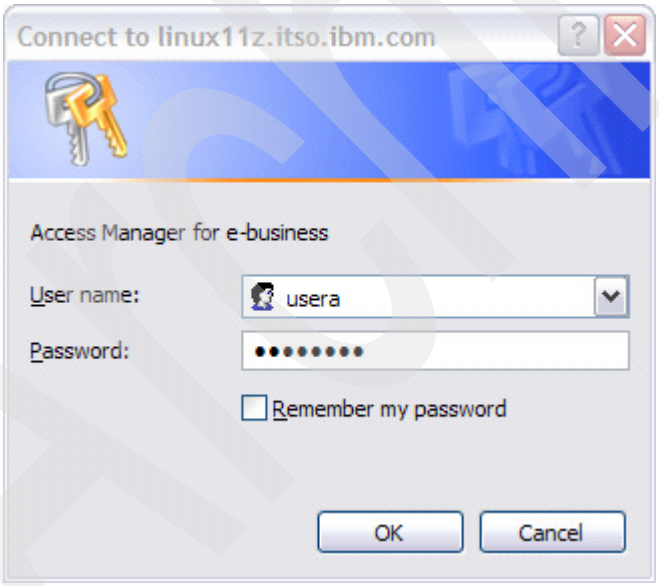


Figure 8-22 WebSeal sign-on for internal user

This user ID is once again mapped by our JAAS login module (ITSOFirstLoginModule), but this time, we chose to map it to the personal RACF user ID corresponding to usera (INTUSR1), which also has read access to the required DB2 table.

As before, we complete the Web form and click **Submit** (Figure 8-23).

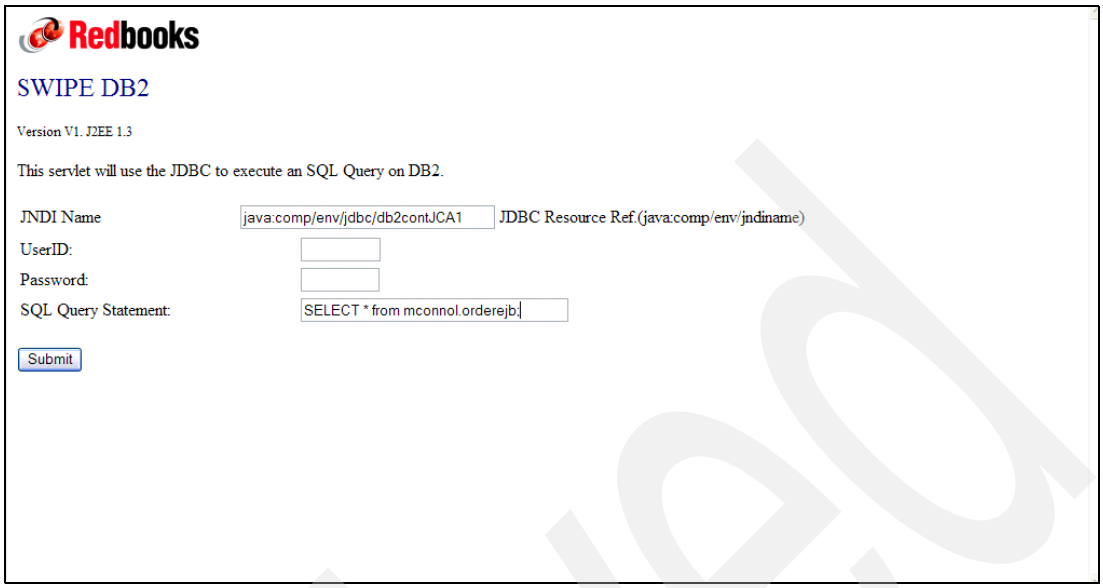


Figure 8-23 SWIPEDB2 Web form

This time, however, we are surprised by the result (Figure 8-24).

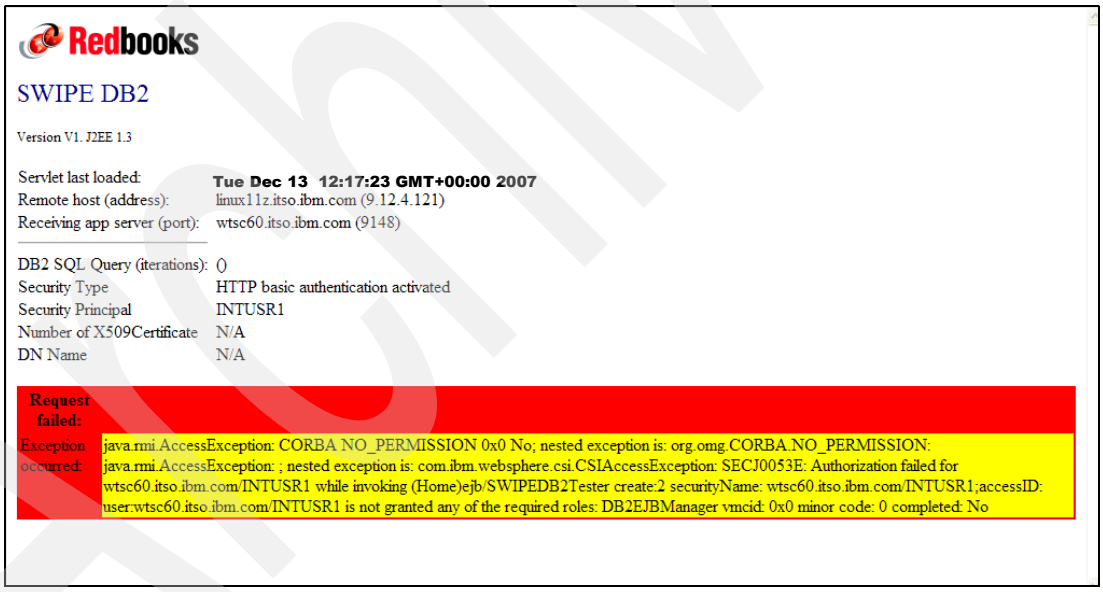


Figure 8-24 SWIPEDB2 failure message

Instead of seeing the results of our DB2 query, we get an error message stating that our user ID, INTUSR1, is not a member of the role DB2EJBManager, which is required to access the EJB. Realizing that this is an oversight, we easily correct the problem by issuing the RACF commands:

```
permit DB2EJBManager class(ejbrole) id(INTUSR1) acc(read)
setr raclist(ejbrole) refresh
```

Once this has been accomplished, a simple refresh of the page from the browser brings us the results we expected (Figure 8-25).

```
Remote host (address):        linux11z.itso.ibm.com (9.12.4.121)
Receiving app server (port):  wtsc60.itso.ibm.com (9148)

DB2 SQL Query (iterations):   ()
Security Type                 HTTP basic authentication activated
Security Principal            INTUSR1
Number of X509Certificate     N/A
DN Name                       N/A

Results                       Output IOAREA
Input:                        SELECT * from mconnol.orderejb;
                              1:24.95|2007-10-16 19:27:04.399|buy|completed|140.00|10.0|2007-10-16 19:27:04.111|0|0|s:284|0|
                              2:24.95|2007-10-16 19:27:06.141|buy|completed|103.00|85.0|2007-10-16 19:27:06.123|1|0|s:367|1|
                              3:24.95|2007-10-16 19:27:06.267|buy|closed|179.00|169.0|2007-10-16 19:27:06.265|2|1|s:169|2|
                              4:24.95|2007-10-16 19:27:06.319|buy|closed|159.00|193.0|2007-10-16 19:27:06.317|3|2|s:841|3|
                              5:24.95|2007-10-16 19:27:06.362|buy|closed|92.00|180.0|2007-10-16 19:27:06.359|4|2|s:804|4|
                              6:24.95|2007-10-16 19:27:06.405|buy|closed|160.00|69.0|2007-10-16 19:27:06.403|5|2|s:823|5|
                              7:24.95|2007-10-16 19:27:06.446|buy|closed|22.00|158.0|2007-10-16 19:27:06.444|6|2|s:244|6|
                              8:24.95|2007-10-16 19:27:06.494|buy|closed|151.00|20.0|2007-10-16 19:27:06.492|7|2|s:611|7|
                              9:24.95|2007-10-16 19:27:06.53|buy|closed|3.00|143.0|2007-10-16 19:27:06.528|8|2|s:253|8|
                              10:24.95|2007-10-16 19:27:06.61|buy|closed|68.00|108.0|2007-10-16 19:27:06.608|9|2|s:726|9|
                              11:24.95|2007-10-16 19:27:06.665|buy|closed|144.00|58.0|2007-10-16 19:27:06.661|10|2|s:505|10|
                              12:24.95|2007-10-16 19:27:06.758|buy|closed|164.00|75.0|2007-10-16 19:27:06.755|11|2|s:584|11|
                              13:24.95|2007-10-16 19:27:06.8|buy|closed|130.00|93.0|2007-10-16 19:27:06.798|12|3|s:380|12|
                              14:24.95|2007-10-16 19:27:06.829|buy|closed|111.00|94.0|2007-10-16 19:27:06.827|13|3|s:424|13|
                              15:24.95|2007-10-16 19:27:06.881|buy|closed|104.00|191.0|2007-10-16 19:27:06.879|14|3|s:199|14|
                              16:24.95|2007-10-16 19:27:06.919|buy|closed|137.00|149.0|2007-10-16 19:27:06.917|15|3|s:666|15|
```

Figure 8-25 SWIPEDB2 results

Examining the audit trail

Once again, we use the iView component of TCIM to find evidence of this transaction. This time we show a different approach by beginning using an *event by rule* report with an On What filter on the object group MCONNOL. Figure 8-26 shows the TCIM reports page. We select **Events by rule**.

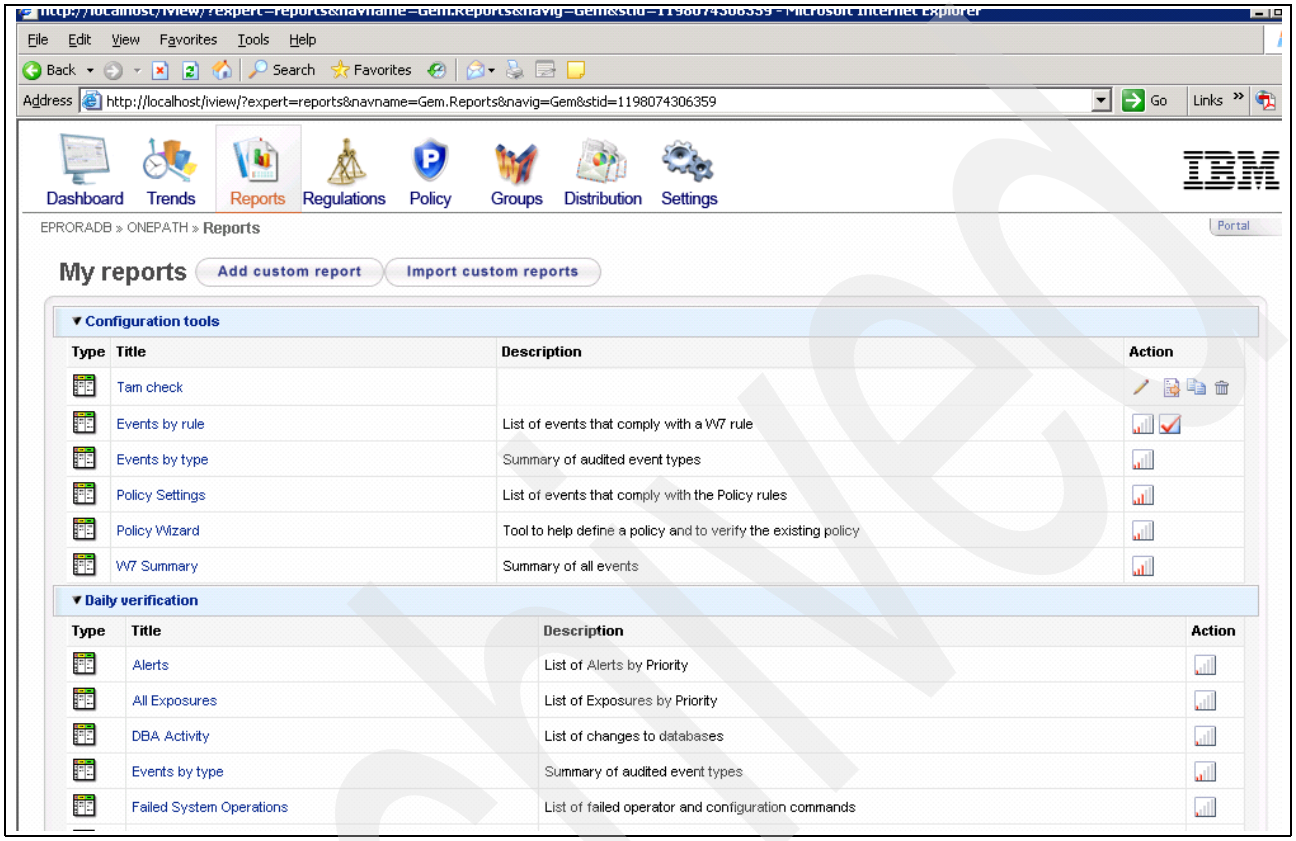
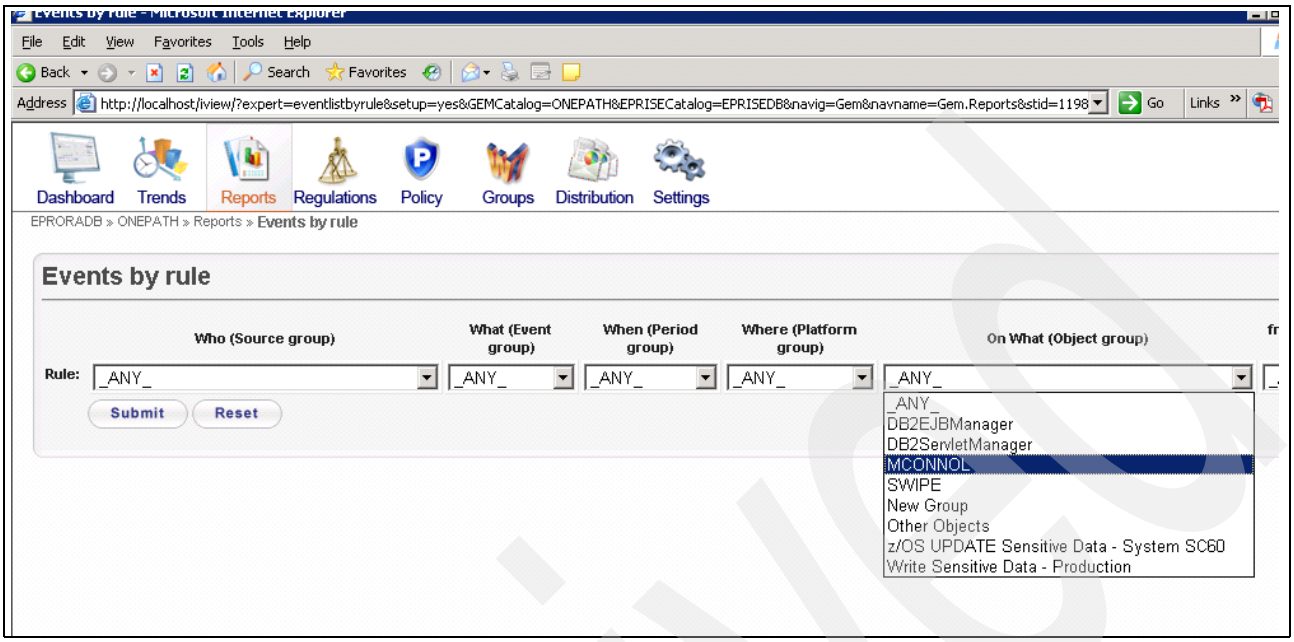


Figure 8-26 TCIM reports page

We had previously defined an On What group to allow the display of any objects including MCONNOL as part of the name (Figure 8-27).



EPRORADB » ONEPATH » Reports » **Events by rule**

Events by rule

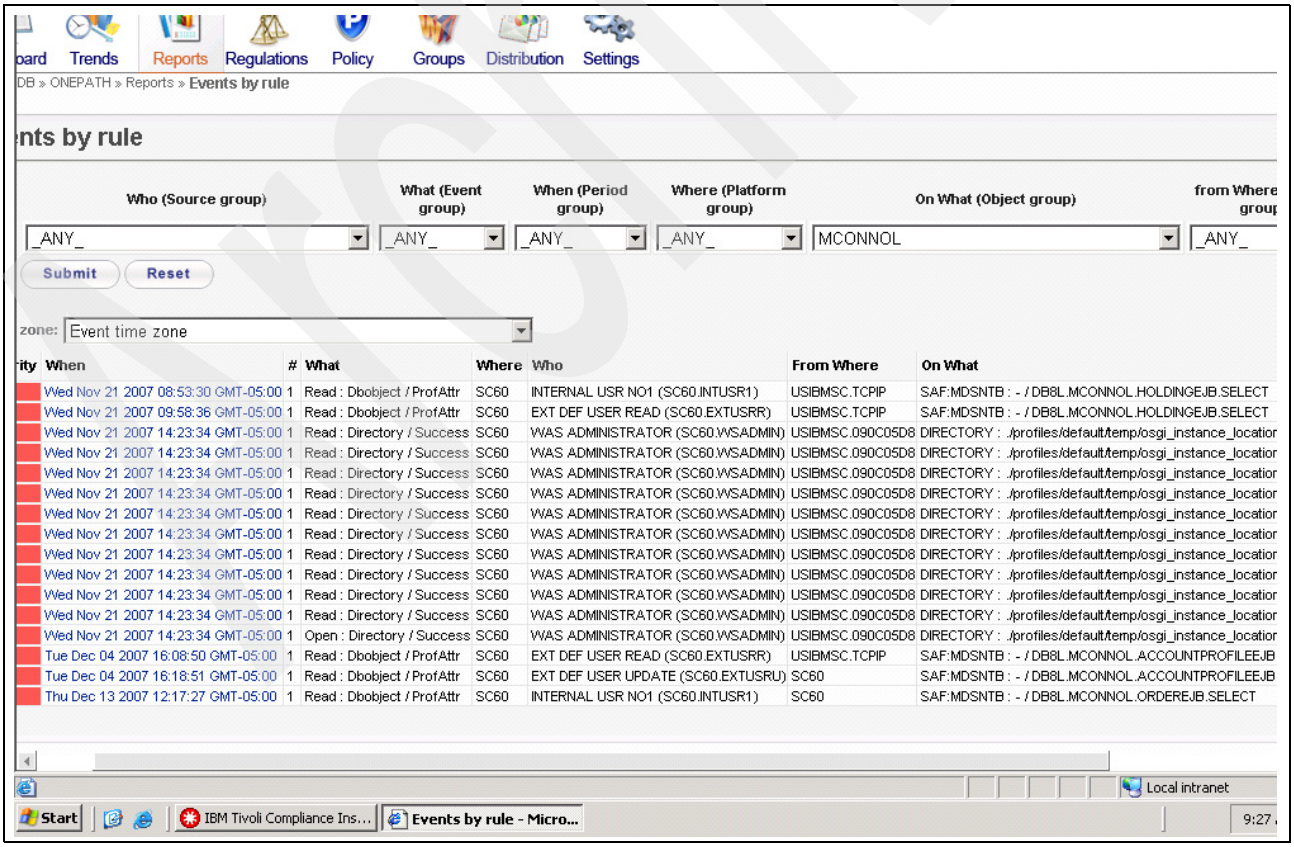
	Who (Source group)	What (Event group)	When (Period group)	Where (Platform group)	On What (Object group)	fr
Rule:	_ANY_	_ANY_	_ANY_	_ANY_	_ANY_	

On What (Object group):

- _ANY_
- DB2EJBManager
- DB2ServletManager
- MCONNOL
- SWIPE
- New Group
- Other Objects
- z/OS UPDATE Sensitive Data - System SC60
- Write Sensitive Data - Production

Figure 8-27 Filtering by object group

The results of the events filtered are shown in Figure 8-28.



DB » ONEPATH » Reports » **Events by rule**

nts by rule

Who (Source group)	What (Event group)	When (Period group)	Where (Platform group)	On What (Object group)	from Where grou
ANY	_ANY_	_ANY_	_ANY_	MCONNOL	_ANY_

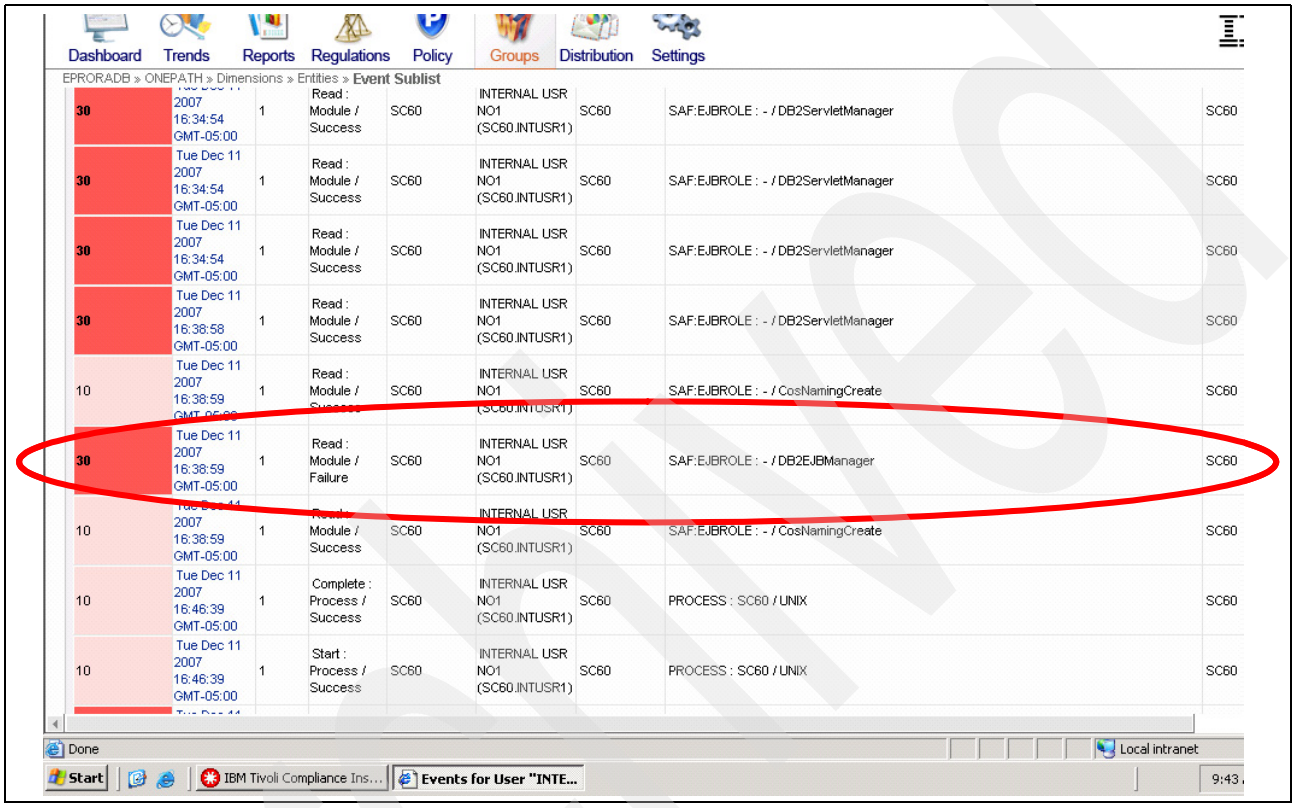
zone: Event time zone

ity	When	#	What	Where	Who	From Where	On What
🟥	Wed Nov 21 2007 08:53:30 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	USIBMSC.TCPIP	SAF:MDSNTB : - / DB8L.MCONNOL.HOLDINGEJB.SELECT
🟥	Wed Nov 21 2007 09:58:36 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60	EXT DEF USER READ (SC60.EXTUSRR)	USIBMSC.TCPIP	SAF:MDSNTB : - / DB8L.MCONNOL.HOLDINGEJB.SELECT
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatior
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatior
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatior
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatior
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatior
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatior
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatior
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatior
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatior
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatior
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Open : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatior
🟥	Tue Dec 04 2007 16:08:50 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60	EXT DEF USER READ (SC60.EXTUSRR)	USIBMSC.TCPIP	SAF:MDSNTB : - / DB8L.MCONNOL.ACCOUNTPROFILEEJB
🟥	Tue Dec 04 2007 16:18:51 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60	EXT DEF USER UPDATE (SC60.EXTUSRU)	SC60	SAF:MDSNTB : - / DB8L.MCONNOL.ACCOUNTPROFILEEJB
🟥	Thu Dec 13 2007 12:17:27 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	SC60	SAF:MDSNTB : - / DB8L.MCONNOL.ORDEREJB.SELECT

Figure 8-28 Filtered results

This shows, among other things, that at 12:17:27 on December 13th the user ID INTUSR1 performed a read operation on DB8L.MCONNOL.ORDEREJB in the system SC60 (z/OS). As expected, the From Where entry shows us that the request came from SC60 as well.

Filtering on the user ID INTUSR1, we see that there is more detail available. In addition to results similar to those we saw for our external user (8.1.2, “External user scenario” on page 168), we also find an earlier failure attempting to access the DB2EJBManager EJBROLE profile (Figure 8-29).



EPRORADB » ONEPATH » Dimensions » Entities » **Event Sublist**

30	2007 16:34:54 GMT-05:00	1	Read : Module / Success	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	SC60	SAF:EJBROLE : - / DB2ServletManager	SC60
30	Tue Dec 11 2007 16:34:54 GMT-05:00	1	Read : Module / Success	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	SC60	SAF:EJBROLE : - / DB2ServletManager	SC60
30	Tue Dec 11 2007 16:34:54 GMT-05:00	1	Read : Module / Success	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	SC60	SAF:EJBROLE : - / DB2ServletManager	SC60
30	Tue Dec 11 2007 16:38:58 GMT-05:00	1	Read : Module / Success	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	SC60	SAF:EJBROLE : - / DB2ServletManager	SC60
10	Tue Dec 11 2007 16:38:59 GMT-05:00	1	Read : Module / Success	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	SC60	SAF:EJBROLE : - / CosNamingCreate	SC60
30	Tue Dec 11 2007 16:38:59 GMT-05:00	1	Read : Module / Failure	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	SC60	SAF:EJBROLE : - / DB2EJBManager	SC60
10	Tue Dec 11 2007 16:38:59 GMT-05:00	1	Read : Module / Success	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	SC60	SAF:EJBROLE : - / CosNamingCreate	SC60
10	Tue Dec 11 2007 16:46:39 GMT-05:00	1	Complete : Process / Success	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	SC60	PROCESS : SC60 / UNIX	SC60
10	Tue Dec 11 2007 16:46:39 GMT-05:00	1	Start : Process / Success	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	SC60	PROCESS : SC60 / UNIX	SC60

Figure 8-29 Failed EJBROLE access attempt

Again, we note that there does not seem to be anything in iView to tell us that INTUSR1 was the user ID mapped by usera. So, as before, we relied on the auditing that we built into our JAAS login module (see “Design” on page 162 and “Installation” on page 163). By correlating time stamps shown in iView with records in the WebSphere Application Server servant region SYSPRINT, we were able to find the audit record created by our JAAS login module (see Example 8-3).

Example 8-3 Audit records created by JAAS login module

```
Audit: com.ibm.websphere.security.ITSOFirstLoginModule
 Values have been stored into the shared state
Audit: com.ibm.websphere.security.ITSOFirstLoginModule
 original User ID usera
Audit: com.ibm.websphere.security.ITSOFirstLoginModule
 mapped User ID INTUSR1
Audit: com.ibm.websphere.security.ITSOFirstLoginModule
 principal class com.ibm.websphere.security.auth.WSPrincipal
```

This tells us that the original user mapped to INTUSR1 was usera. We can now use this information to interrogate the date presented by iView further. Here we see records indicating that user usera was authenticated at our WebSEAL server and authorized to wtsc60http/SWIPEDB2Web/.

> **Note:** If you have the SAF APPL class active, any user ID mapped by the JAAS login module must have at least READ access to your application server's APPL profile (for example, CBS390 or the name of your security domain, if any).

Conclusion

How effectively were we able to track this use case from browser to database back end and back? Even though we used a different user and native authentication to RACF in WebSEAL, the remaining results are the same. WebSEAL was able to log the authentication and application authorization, and RACF was able to log the EJB authorization and DB2 table access, but extensive customization would be required to get the identity mapping information logged into TCIM to tie the records together. The custom LDAP identity data concerns are as in the previous scenario.

8.2 Scenarios with WebSphere Application Server on Linux on System z

This chapter describes the different scenarios we implemented in a WebSphere Application Server on Linux for System z environment and report how we set up the auditing and reporting infrastructure.

8.2.1 Environment and setup

Figure 8-30 shows the setup for WebSphere Application Server on Linux on System z. Compared to the previous scenarios with WebSphere Application Server on z/OS, there are some important differences to notice here. We did not use a customized JAAS login configuration when the request was first received. The standard TAM JACC Provider was used to provide container-level authorization. WebSphere Application Server on Linux on System z is using an LDAP registry, so there is no need to map the external user ID to a RACF user until the call to DB2.

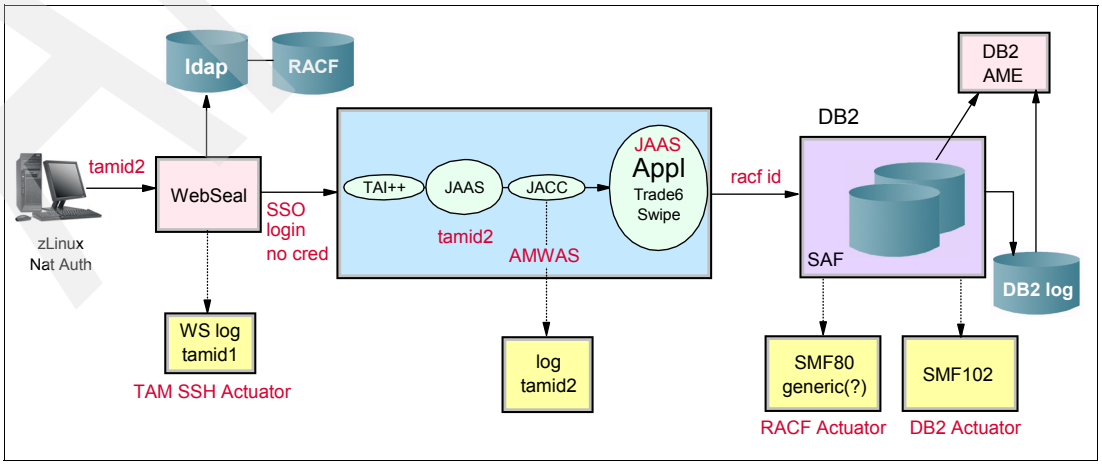


Figure 8-30 e-business scenario with WebSphere Application Server on Linux on System z

Some practitioners recommend that any enhancements to the JAAS Subject be made when the request initially reaches WebSphere Application Server, which in our case would be the WEB_INBOUND login configuration. For the configuration and scenarios that we have chosen, that would mean enhancing the WEB_INBOUND configuration with a module (a subset of the ITSOFirstLoginModule discussed in 8.1.1, “Environment and setup” on page 160) that would map the LDAP user ID to a RACF user ID. We decided that this step was overly complicated for our simple configuration and would add little to our understanding of auditing.

In order to accept the user ID passed by WebSEAL, we did configure a Trust Association Interceptor, known as the TAI ++, that is included as part of WebSphere Application Server specifically for use with WebSEAL. Configuring the TAI++ for use with WebSEAL is described in the WebSphere Application Server (z/OS) V6.1 InfoCenter at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/tsec_ssowsstep4TAIplusplus.html

The SWIPEDB2 application accesses DB2 through a data source. In the application’s deployment descriptor is a resource reference that points to the data source. That resource reference is mapped at installation time to a resource defined in the configuration. Along with that mapping, we can specify an authentication method (a way of providing authentication information to DB2). This is where we defined a mapping module to map the external user ID to a RACF user (technical ID).

JAAS mapping module

As with the z/OS illustration (“JAAS login module” on page 162), we configured a mapping module to search LDAP using the remote user’s distinguished name and retrieve a special attribute created for this purpose (com-ibm-example-jaas-ExamplePluggable-Map) to hold the RACF user ID to which the remote user should be mapped.

Design

The interface for a JAAS mapping module is identical to that for a JAAS login module. The major difference is that a special object, the ManagedConnectionFactory, accessed through the WSManagedConnectionFactoryCallback, is populated with the mapped user ID and password. For simplicity, our implementation simply included the password in the module itself. A more realistic mapping module would retrieve the password from a secure password store.

Installation

Installation of the JAAS mapping module involved several steps. After compiling ITSOCustomMappingLoginModule and packaging it in a jar file, we installed the jar file into the directory {WAS_INSTALL_ROOT}/lib/ext along with the jar file containing the ExamplePluggableJAAS mapping routines.

We also installed the file ExamplePluggable.properties in the directory {WAS_INSTALL_ROOT}/properties after configuring it to use our LDAP server.

Next, we created a mapping configuration for our mapping module to reside in. We logged into the WebSphere Application Server administration application, expanded the **Security** tree, and clicked **Global security** (Figure 8-31).

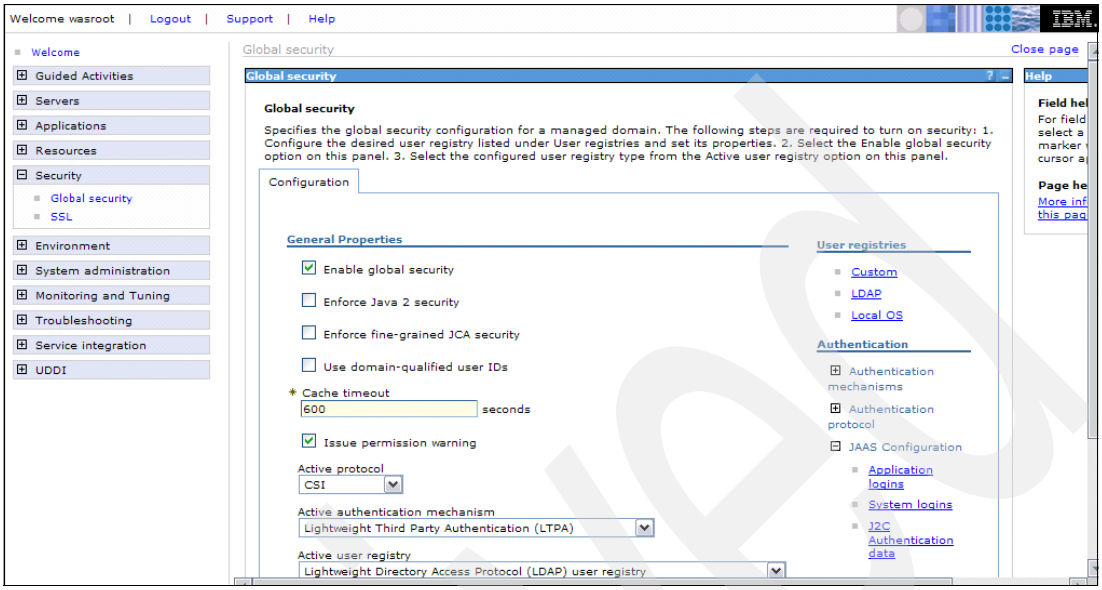


Figure 8-31 Global security page

On the Global security page we expanded **JAAS Configuration** and clicked **Application logins**. This shows us the default IBM-supplied login configurations (Figure 8-32) to which we need to add our own configurations.

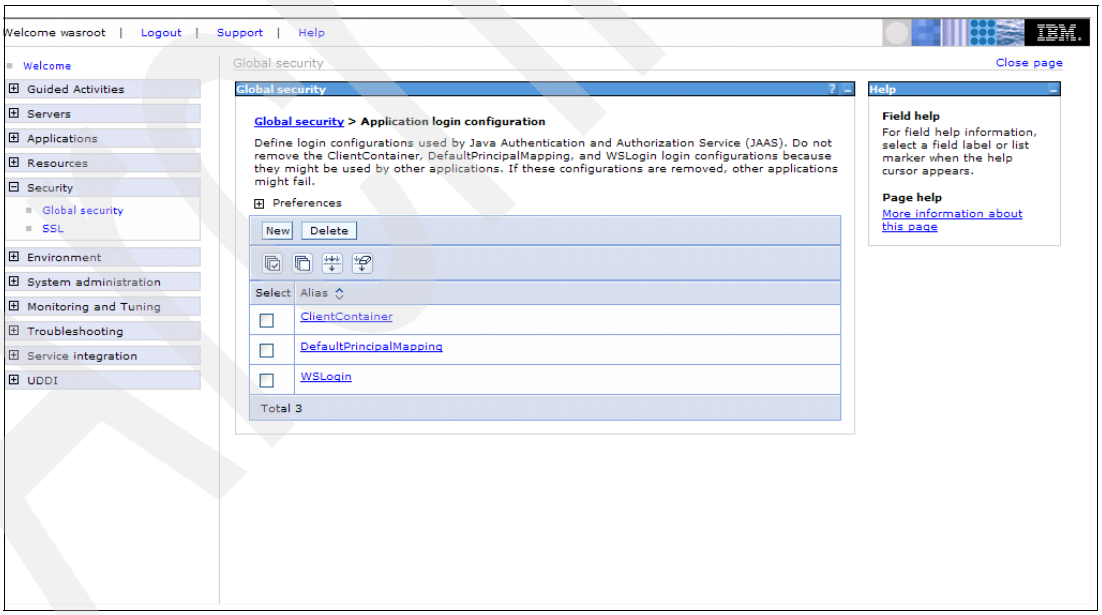


Figure 8-32 Application logins

Clicking **New** brings us to a property sheet where we fill in the name of the configuration that we are creating, ITSOPrincipalMappingConfiguration (Figure 8-33).

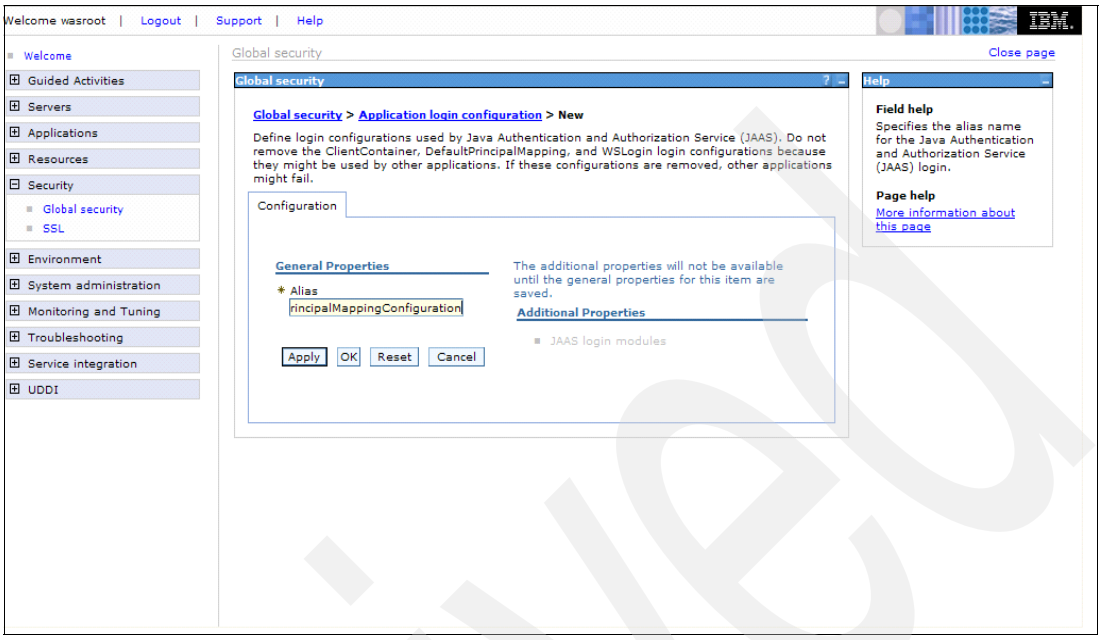


Figure 8-33 Creating a new login configuration

When we click **Apply**, we can then click **JAAS login modules**. This presents an empty list, so we click **New**, giving us a property sheet where we can enter the name of our login module (including the package name), `com.ibm.websphere.security.ITSOCustomMappingLoginModule` (Figure 8-34).

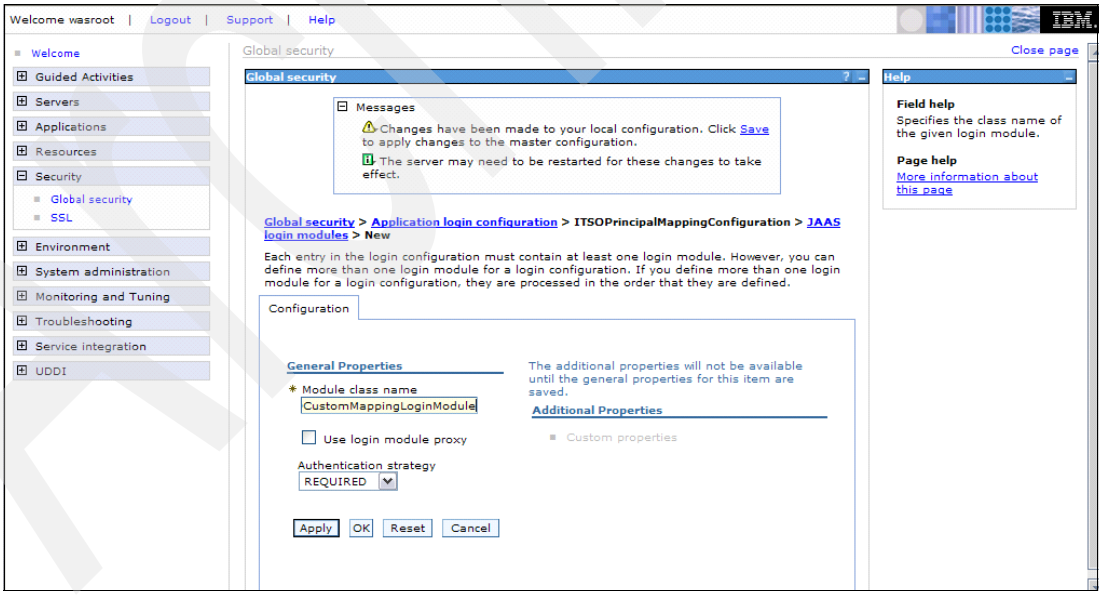


Figure 8-34 Login module property sheet

After we click **Apply**, we can click **Custom properties** and receive a property sheet similar to Figure 8-7 on page 166. Here we add the properties debug(false) and audit(true) in a similar way (Figure 8-35).

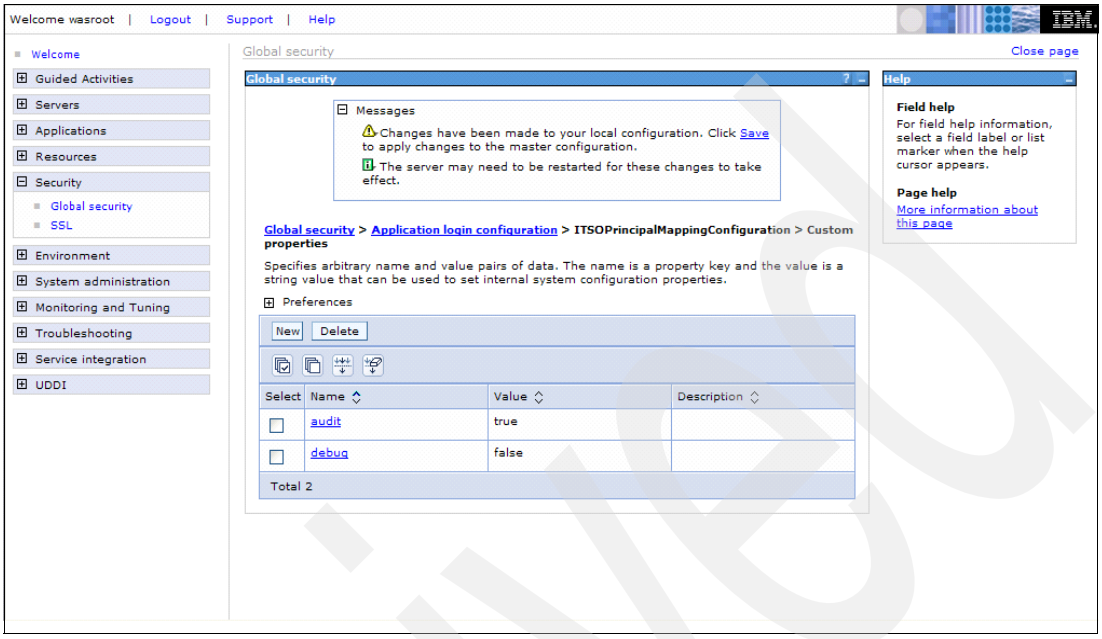


Figure 8-35 Mapping module custom properties

Before continuing to the next step, we save the changes to the master repository.

Finally, we configured the authentication method for the data source. To do this, we expanded the **Applications** tree and clicked **Enterprise Applications** (Figure 8-36).

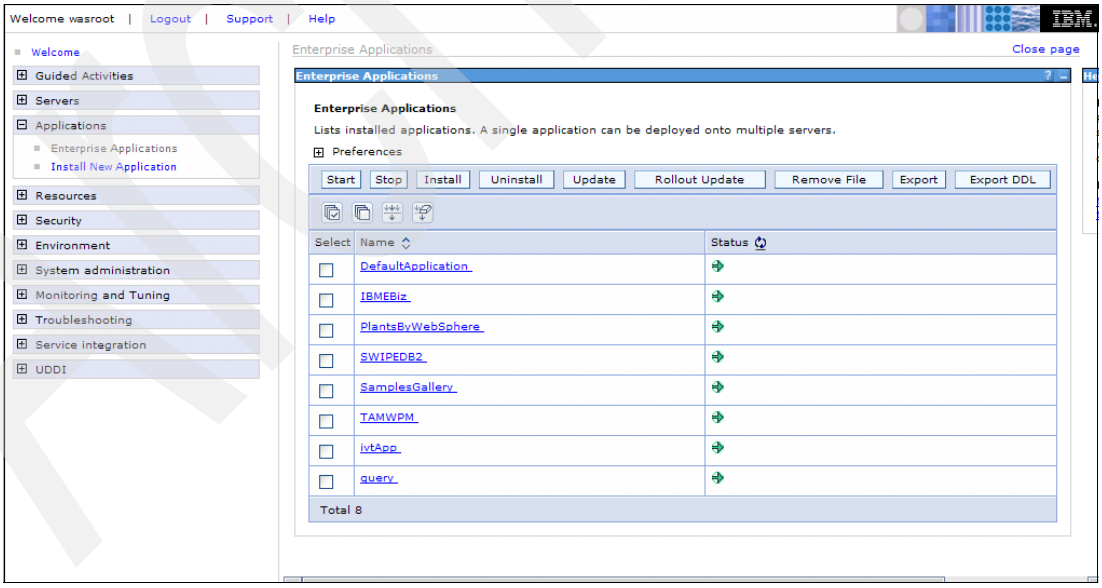


Figure 8-36 Enterprise applications

In this window, we clicked our application name (**SWIPEDB2**) to see the application details (Figure 8-37).

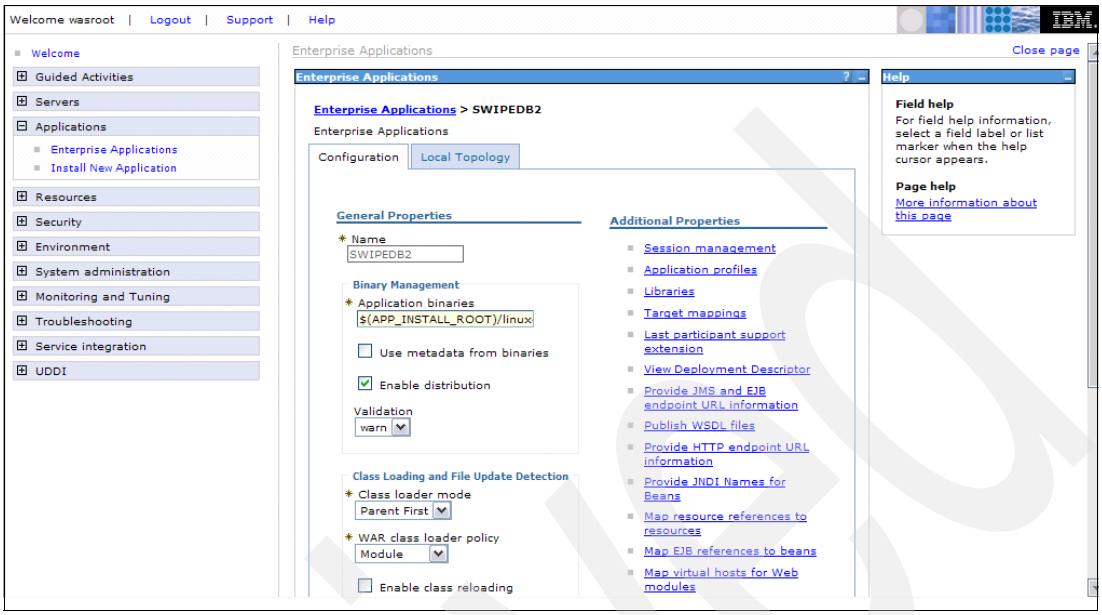


Figure 8-37 SWIPEDB2 details

Here we clicked **Map resource references to resources** in order to see our data source. Here we see a list of data sources. The one of interest to us, which we recognize by its reference binding of jdbc/db2contJCA1, happens to be at the top of the list. It has resource authorization set to container, which is what we need, and authentication method of none, which we want to change to use our mapping module.

Above the list of resources, we find the area that we can use to select a new authentication method. We check the select box next to the jdbc/db2contJCA1 resource, then check **Use custom login method** and select the application login **ITSOPrincipalMappingConfiguration** from the drop-down list (see Figure 8-38).

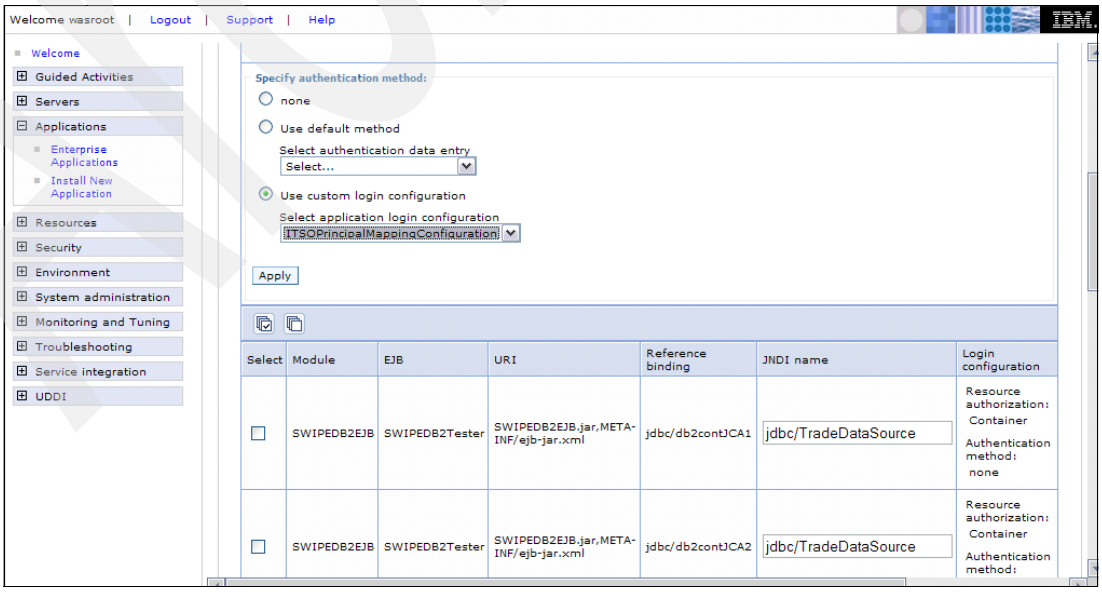


Figure 8-38 Configure authentication method

After clicking **Apply**, the authentication method changes to reflect our selection of ITSOPrincipalMappingConfiguration, and a Mapping Properties button appears (Figure 8-39).

Attention: Before proceeding, scroll to the bottom of this page and click **OK**.

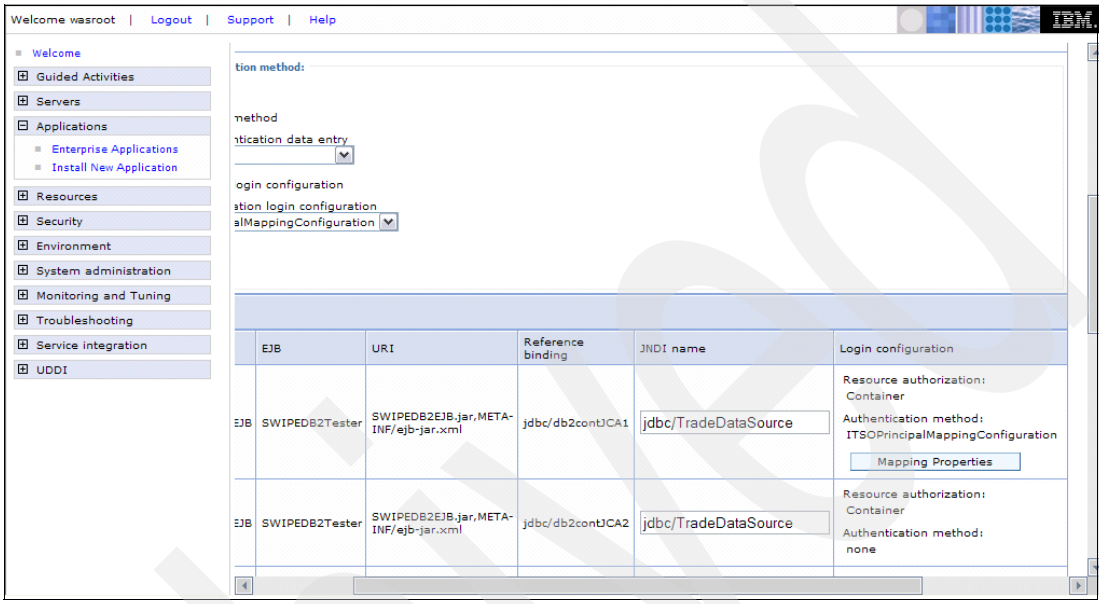


Figure 8-39 Modified authorization method

Finally, be sure to save these changes to the master repository.

Conclusion

How effectively were we able to track this use case from browser to database back end and back? Let us work through the steps in the user case and how the logging worked:

1. The authentication and coarse-grained access activity (access to the SWIPEDB2 application) in WebSEAL was logged in the WebSEAL audit log against the TAM user (testu1). These records flowed through to TCIM.
2. The mapping of TAM user (testu1) to RACF user EXTUSRR was performed in the custom JAAS module and logged to the WAS SYSPRINT. These records did not flow through to TCIM, which would require some form of TCIM actuator and some customization/configuration.
3. The EJB-level authorization (fine-grained authorization) was performed by the TAM JACC provider. The provider itself does not cut any audit records. The TAM authorization service should cut audit records for the authorization, but we did not see any (we have seen TAM authorization deamon (pdacld) records cut in other environments). This may be due to audit configuration of the TAM authorization process or because they were written but not forwarded to TCIM.
4. The JDBC-level authorization to access the DB2 resources, as controlled by RACF, is logged to SMF (Type 80) and forwarded successfully to TCIM. We can see authorization records for this user performing a read against DB8L.MCONNOL.HOLDINGEJB.

The TAM/WebSEAL components provide useful audit logging that can flow through to TCIM, and DB2 provided useful audit logging through RACF to SMF, which can flow through to TCIM. The TAM JACC Provider may have produced EJB authorization information, but we

did not see it. As before, the challenge is in the identity mapping components. These modules are custom and some work is required to generate audit data, either through RACF and SMF or through another route, that can tie the TAM identity-based audit records to the RACF identity-based ones. This solution also required customizing the LDAP identity data used in authentication and authorization, and production use of this would require procedures and tools, like Tivoli Identity Manager, to ensure that the TAM identities were mapped to the appropriate RACF identities.

8.2.2 External user scenario

In this section we provide an external user scenario.

Performing the test case

Our first test with WebSphere Application Server on Linux on System z is to access the enterprise application SWIPEDB2 running in WebSphere Application Server on Linux on System z using the user ID testu1, which has no corresponding RACF password. We begin by pointing our browser to:

`https://linux11z.itso.ibm.com/linux4z_was/SWIPEDB2Web/`

The host field, linux11z.itso.ibm.com, represents the system on which we have installed WebSEAL. The next portion of the URL, linux4z_was, is the name of a WebSEAL junction that will proxy authenticated requests to WebSphere Application Server on Linux on System z. SWIPEDB2Web is the context root of the Web application in WebSphere Application Server.

If this is the first time that we access this single sign-on domain, we are prompted to enter a user name and password. We enter the user name testu1 and the password defined for testu1 in the LDAP directory (Figure 8-40).

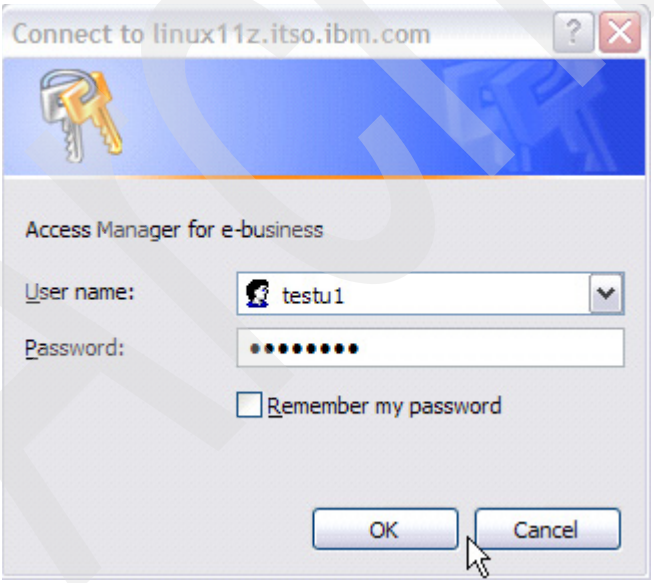


Figure 8-40 WebSEAL sign-on

Upon clicking **OK**, we are brought to the SWIPEDB2Web main page, which is a form. This application allows testing of a number of different DB2 connector configurations. We want to show a DB2 type 4 connection using container-managed authentication and no application-supplied user ID and password so that the user ID passed to DB2 for

authorization can be derived from our JAAS mapping module. We have set up the JNDI name java:comp/env/jdbc/db2contJCA1 to use this kind of connector.

Since we are not expecting the application to supply a user ID or password, we leave these fields blank. In the SQL Query Statement field we enter our query against the DB2 table:

```
SELECT * from mconnol.accountprofileejb;
```

When we click **Submit** (Figure 8-41), the form is posted to a servlet that invokes the SWIPEDB2 EJB. The EJB in turn invokes the DB2 connector, which is configured to call our JAAS mapping module. That module maps the user ID testu1 to the RACF user ID EXTUSRR. The query is then forwarded to DB2 with the mapped user ID. The results are displayed by a Java server page, a portion of which is shown in Figure 8-42 on page 192. Note that this application is used for illustrative purposes only. The output is not in an especially readable format. We are emphasizing here only the steps to authorize and audit access to the applications and data, not the principles of good user interface design.

Figure 8-41 SWIPEDB2 Web form

```
Remote host (address):        linux11z.itso.ibm.com (9.12.4.121)
Receiving app server (port):  linux4z.itso.ibm.com (9080)

DB2 SQL Query (iterations):   ()
Security Type                 HTTP basic authentication activated
Security Principal            testu1
Number of X509Certificate     N/A
DN Name                       N/A

Results                       Output IOAREA
Input:                        SELECT * from mconnol.accountprofileejb;
                              1:427 Oak St.|xxx|uid:0|uid:0@47.com|46-975-791-187|first:474 last:2659|
                              2:361 Oak St.|xxx|uid:1|uid:1@86.com|8-524-116-822|first:89 last:795|
                              3:591 Oak St.|xxx|uid:2|uid:2@38.com|65-961-651-85|first:345 last:3601|
                              4:880 Oak St.|xxx|uid:3|uid:3@81.com|1-247-847-997|first:457 last:3396|
                              5:503 Oak St.|xxx|uid:4|uid:4@55.com|5-60-152-597|first:617 last:2788|
                              6:547 Oak St.|xxx|uid:5|uid:5@87.com|33-112-773-238|first:767 last:3370|
                              7:827 Oak St.|xxx|uid:6|uid:6@71.com|36-642-901-62|first:859 last:3035|
                              8:39 Oak St.|xxx|uid:7|uid:7@78.com|0-311-536-122|first:407 last:2124|
                              9:28 Oak St.|xxx|uid:8|uid:8@82.com|87-160-66-719|first:38 last:2929|
                              10:823 Oak St.|xxx|uid:9|uid:9@97.com|78-967-684-645|first:581 last:877|
                              11:824 Oak St.|xxx|uid:10|uid:10@36.com|43-917-16-79|first:305 last:3713|
                              12:859 Oak St.|xxx|uid:11|uid:11@98.com|6-349-945-600|first:195 last:4419|
                              13:383 Oak St.|xxx|uid:12|uid:12@34.com|85-995-778-509|first:170 last:606|
                              14:891 Oak St.|xxx|uid:13|uid:13@48.com|57-707-861-589|first:971 last:4494|
                              15:480 Oak St.|xxx|uid:14|uid:14@54.com|54-569-32-980|first:0 last:3878|
                              16:408 Oak St.|xxx|uid:15|uid:15@23.com|71-594-533-575|first:578 last:1943|
```

Figure 8-42 SWIPEDB2 results

Examining the audit trail

We expect to be able to use the iView component of IBM Tivoli Compliance Insight Manager (TCIM) to find evidence of this transaction. Since we are likely to be trying to answer the question “who accessed the DB2 table mconnol.accountprofileejb?” we begin by filtering the table of events by *MCONNOL* in the On What column

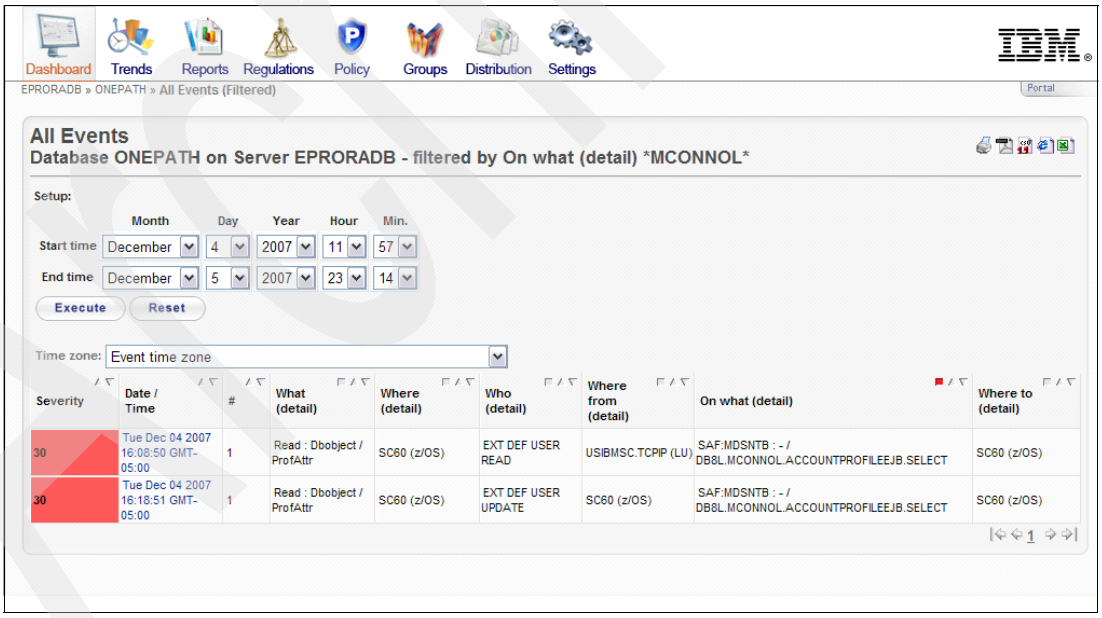


Figure 8-43 Database events for DB8L.MCONNON.ACCOUNTPROFILEEJB

This shows that at 16:08:50 on Tue Dec 04, 2007 the user ID EXTUSRR performed a read operation on DB8L.MCONNOL.HOLDINGEJB in the system SC60 (z/OS). This time, however, in contrast to the events discussed in 8.1, “Scenarios with WebSphere Application Server on z/OS” on page 160, we see that the request originated off the z/OS platform.

Here we encountered a problem, however. There does not seem to be anything in iView to tell us that EXTUSRR was the user ID mapped by testu1. For this, we need to rely on our JAAS mapping module, which has been designed to write audit records. In the case of Linux on System z, these records go to the SystemOut.log file found in {WAS_INSTALL_ROOT}/profiles/default/logs/server1 (Figure 8-44).

```
SystemOut     O CustomMapping => login
SystemOut     O Principal name => cn=testu1,ou=Swipe,ou=Employees,o=ITSO
SystemOut     O Mapped to => EXTUSRR
```

Figure 8-44 Audit records in SystemOut.log

As with the z/OS illustration, we can drill down to the event details and search for events at a similar time. In the Linux on System z case we do not see anything specifically indicating that the user has been checked against the J2EE roles (DB2ServletManager or DB2EJBManager), since the JACC provider does not have extensive auditing logs. We do see, however, that WebSEAL has authorized testu1 to the URL where the Servlet and EJB run (Figure 8-45).

	GMT-05:00							
10	Tue Dec 04 2007 11:08:50 GMT-05:00	1	Create : Certificate / Success	linux11z.itso.ibm.com (IBM TAM)	Testu1	9.12.5.216 (IBM TAM)	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com (IBM TAM)
10	Tue Dec 04 2007 11:08:50 GMT-05:00	1	Access : Object / Success	linux11z.itso.ibm.com (IBM TAM)	Testu1	9.12.5.216 (IBM TAM)	OBJECT : - / /linux4z_was/SWIPEDB2Web/SWIPEDB2Servlet	linux11z.itso.ibm.com (IBM TAM)
	Tue Dec 04		Access : Object /	linux11z.itso.ibm.com				linux11z.itso.ibm.com

Figure 8-45 Event records from Linux

Conclusion

How effectively were we able to track this use case from browser to database back end and back? Let us work through the steps in the user case and see how the logging worked:

1. The authentication and coarse-grained access activity (access to the SWIPEDB2 application) in WebSEAL was logged in the WebSEAL audit log against the TAM user (testu1). These records flowed through to TCIM.
2. The mapping of the TAM user (testu1) to RACF user EXTUSRR was performed in the custom JAAS module and logged to the WAS SYSPRINT. These records did not flow through to TCIM, which would require some form of TCIM actuator and some customization/configuration.
3. The EJB-level authorization (fine-grained authorization) was performed by the TAM JACC provider. The provider itself does not cut any audit records. By default it is configured in local mode. If it was changed to remote mode the authorization decisions would be sent to the TAM authorization service, which would cut audit records. Hoverer, this setup is not documented, may not be supported, and has performance implications.
4. The JDBC-level authorization to access the DB2 resources, as controlled by RACF, is logged to SMF (Type 80) and forwarded successfully to TCIM. We can see authorization records for this user performing a read against DB8L.MCONNOL.HOLDINGEJB.

The TAM/WebSEAL components provide useful audit logging that can flow through to TCIM, and DB2 provided useful audit logging through RACF to SMF, which can flow through to TCIM. The TAM JACC Provider in our configuration did not provide the container-level authorization audit logging.

As before, the challenge is in the identity mapping components. These modules are custom and some work is required to generate audit data, either through RACF and SMF or through another route, that can tie the TAM identity-based audit records to the RACF identity-based ones. This solution also requires customizing the LDAP identity data used in authentication

and authorization, and production use of this would require procedures and tools like Tivoli Identity Manager to ensure that the TAM identities were mapped to the appropriate RACF identities.

8.2.3 Internal user scenario

In this section we provide an internal user scenario.

Performing the test case

Our final test with WebSphere Application Server on Linux on System z is to access the enterprise application SWIPEDB2 running in WebSphere Application Server on Linux on System z using the user ID userb, which has a corresponding RACF password. We begin by pointing out browser to:

`https://linux11z.itso.ibm.com/linux4z_was/SWIPEDB2Web/`

The host field, linux11z.itso.ibm.com, represents the system on which we have installed WebSEAL. The next portion of the URL, linux4z_was, is the name of a WebSEAL junction that will proxy authenticated requests to WebSphere Application Server on Linux on System z. SWIPEDB2Web is the context root of the Web application in WebSphere Application Server.

If this is the first time that we access this single sign-on domain, we are prompted to enter a user name and password. We enter the user name `userb` and the password defined for userb in RACF, since this user has the ibm-nativeid attribute (Figure 8-46).

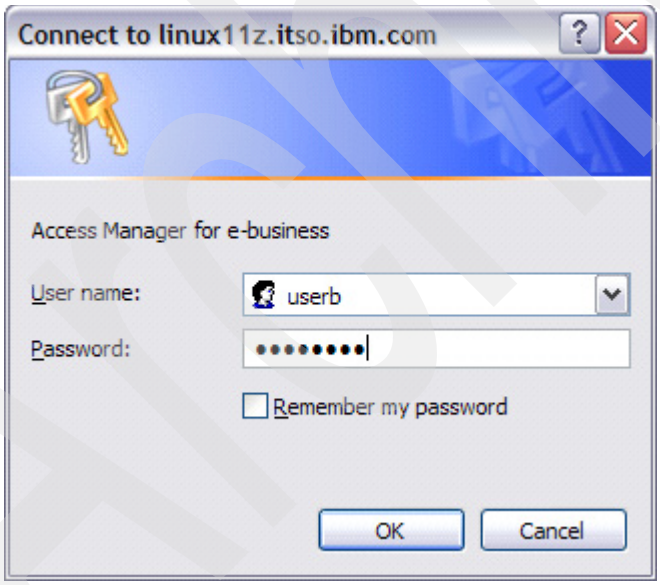


Figure 8-46 WebSEAL sign-on

Upon clicking **OK** we are brought to the SWIPEDB2Web main page, which is a form. This application allows testing of a number of different DB2 connector configurations. We want to show a DB2 type 4 connection using container-managed authentication and no application-supplied user ID and password so that the user ID passed to DB2 for authorization can be derived from our JAAS mapping module. We have set up the JNDI name java:comp/env/jdbc/db2contJCA1 to use this kind of connector.

Since we are not expecting the application to supply a user ID or password, we leave these fields blank. In the SQL Query Statement field we enter our query against the DB2 table:

```
SELECT * from mconnol.holdingejb;
```

When we click **Submit** (Figure 8-47), the form is posted to a servlet that invokes the SWIPEDB2 EJB. The EJB in turn invokes the DB2 connector, which is configured to call our JAAS mapping module. That module maps the user ID userb to the RACF user ID INTUSR1. The query is then forwarded to DB2 with the mapped user ID. The results are displayed by a Java server page, a portion of which is shown in Figure 8-48. Note that this application is used for illustrative purposes only. The output is not in an especially readable format. We are emphasizing here only the steps to authorize and audit access to the applications and data, not the principles of good user interface design.

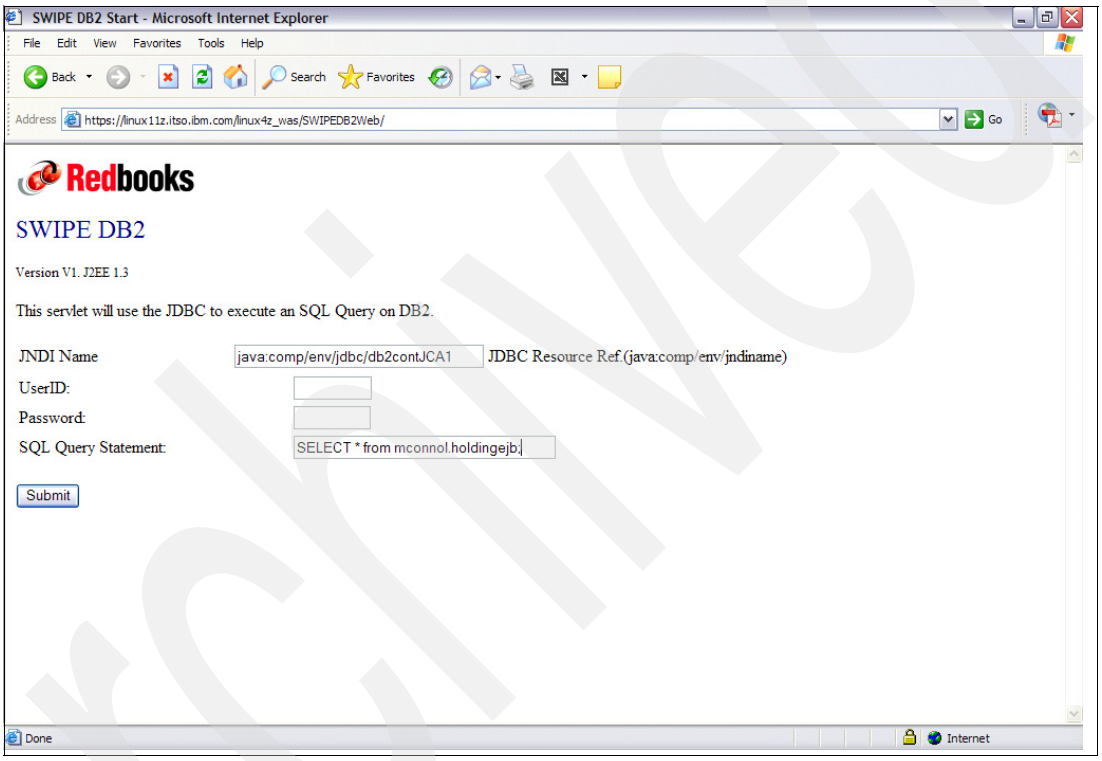


Figure 8-47 SWIPEDB2 Web form

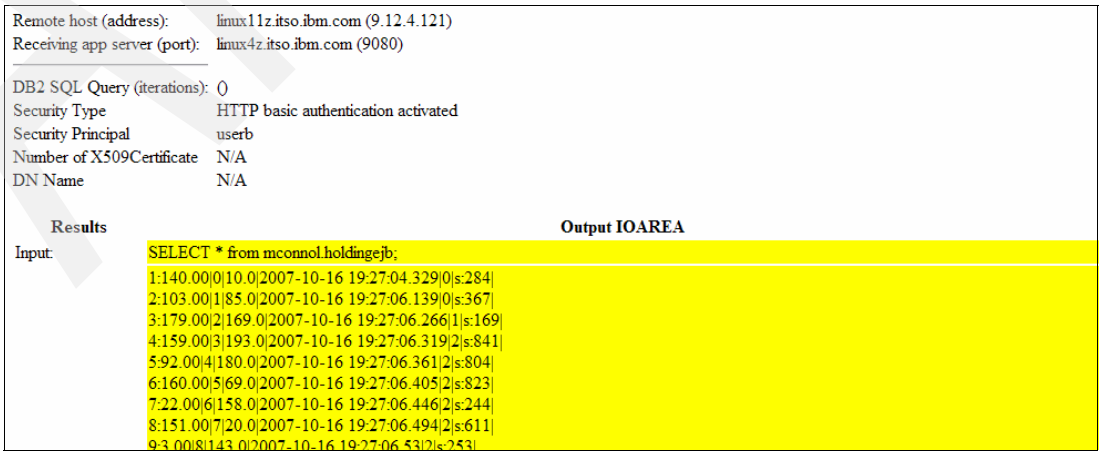
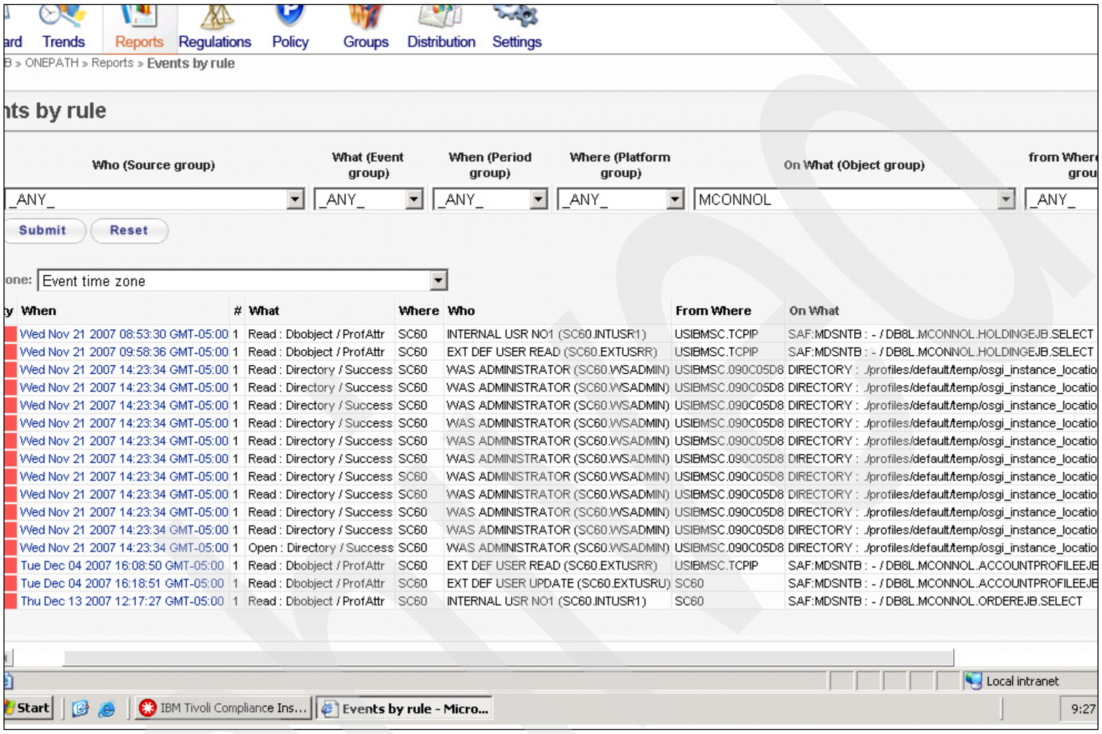


Figure 8-48 SWIPEDB2 results

Examining the audit trail

We expect to be able to use the iView component of IBM Tivoli Compliance Insight Manager (TCIM) to find evidence of this transaction. Since we are likely to be trying to answer the question “who accessed the DB2 table mconnol.holdingejb?” we might begin by using an *event by rule* report that gives us the table as a relevant object. Since we have not defined objects at the individual table level, we instead use an On What filter on the object group MCONNOL (Figure 8-49).



ts by rule

Who (Source group): _ANY_
What (Event group): _ANY_
When (Period group): _ANY_
Where (Platform group): _ANY_
On What (Object group): MCONNOL
from Where grou: _ANY_

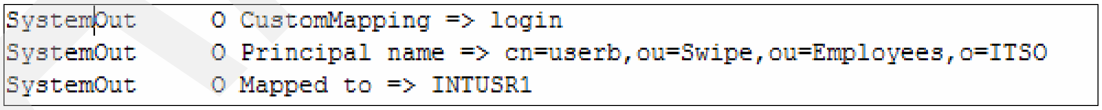
one: Event time zone

y	When	#	What	Where	Who	From Where	On What
🟥	Wed Nov 21 2007 08:53:30 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	USIBMSC.TCPIP	SAF:MDSNTB : - / DB8L.MCONNOL.HOLDINGEJB.SELECT
🟥	Wed Nov 21 2007 09:58:36 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60	EXT DEF USER READ (SC60.EXTUSRR)	USIBMSC.TCPIP	SAF:MDSNTB : - / DB8L.MCONNOL.HOLDINGEJB.SELECT
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatio
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatio
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatio
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatio
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatio
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatio
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatio
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatio
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatio
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Read : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatio
🟥	Wed Nov 21 2007 14:23:34 GMT-05:00	1	Open : Directory / Success	SC60	WAS ADMINISTRATOR (SC60.WSADMIN)	USIBMSC.090C05D8	DIRECTORY : ./profiles/default/temp/osgi_instance_locatio
🟥	Tue Dec 04 2007 16:08:50 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60	EXT DEF USER READ (SC60.EXTUSRR)	USIBMSC.TCPIP	SAF:MDSNTB : - / DB8L.MCONNOL.ACCOUNTPROFILEEJE
🟥	Tue Dec 04 2007 16:18:51 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60	EXT DEF USER UPDATE (SC60.EXTUSRU)	SC60	SAF:MDSNTB : - / DB8L.MCONNOL.ACCOUNTPROFILEEJE
🟥	Thu Dec 13 2007 12:17:27 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60	INTERNAL USR NO1 (SC60.INTUSR1)	SC60	SAF:MDSNTB : - / DB8L.MCONNOL.ORDEREJB.SELECT

Figure 8-49 Database events for DB8L.MCONNON.HOLDINGEJB

This shows that at 08:53:30 on Wed November 21, 2007 the user ID INTUSR1 performed a read operation on DB8L.MCONNOL.HOLDINGEJB in the system SC60 (z/OS). As in 8.2.2, “External user scenario” on page 190, we see that the request originated off the z/OS platform.

Once again, we rely on our JAAS mapping module to tell us what the mapping was (Figure 8-50).



```
SystemOut     O CustomMapping => login
SystemOut     O Principal name => cn=userb,ou=Swipe,ou=Employees,o=ITSO
SystemOut     O Mapped to => INTUSR1
```

Figure 8-50 Audit records in SystemOut.log

As with the z/OS illustration, we can drill down to the event details and search for events at a similar time. In the Linux on System z case we do not see anything specifically indicating that the user has been checked against the J2EE roles (DB2ServletManager or DB2EJBManager), since the JACC provider does not have extensive auditing logs. We do see, however, that WebSEAL has authorized userb to the URL where the Servlet and EJB run (Figure 8-51).

Logon : User / Success	linux11z.itso.ibm.com (IBM TAM)	userb	9.12.5.216 (IBM TAM)	SYSTEM : - / linux11z.itso.ibm.com	linux11z.itso.ibm.com (IBM TAM)
Create : Certificate / Success	linux11z.itso.ibm.com (IBM TAM)	userb	9.12.5.216 (IBM TAM)	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com (IBM TAM)
Access : Object / Success	linux11z.itso.ibm.com (IBM TAM)	userb	9.12.5.216 (IBM TAM)	OBJECT : - / /linux4z_was/SWIPEDB2Web/	linux11z.itso.ibm.com (IBM TAM)
Read : Credentialgroup / Success	linux4z.itso.ibm.com (IBM TAM)	userb	linux4z.itso.ibm.com (IBM TAM)	CREDENTIALGROUP : - / IV_LDAP_V3.0:userb	linux4z.itso.ibm.com (IBM TAM)
Create : Certificate / Success	linux11z.itso.ibm.com (IBM TAM)	userb	9.12.5.216 (IBM TAM)	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com (IBM TAM)
Access : Object / Success	linux11z.itso.ibm.com (IBM TAM)	userb	9.12.5.216 (IBM TAM)	OBJECT : - / /linux4z_was/SWIPEDB2Web/theme/Master.css	linux11z.itso.ibm.com (IBM TAM)
Create : Certificate / Success	linux11z.itso.ibm.com (IBM TAM)	userb	9.12.5.216 (IBM TAM)	CERTIFICATE : - / Default PAC service (encode flattened cred).	linux11z.itso.ibm.com (IBM TAM)
Access : Object / Success	linux11z.itso.ibm.com (IBM TAM)	userb	9.12.5.216 (IBM TAM)	OBJECT : - / /linux4z_was/SWIPEDB2Web/SWIPEDB2Servlet	linux11z.itso.ibm.com (IBM TAM)

Figure 8-51 Events for userb

Conclusion

How effectively were we able to track this use case from browser to database back end and back? Even though we used a different user and native authentication to RACF in WebSEAL, the remaining results are the same. WebSEAL was able to log the authentication and application authorization, and RACF was able to log the DB2 table access. Our configuration of the TAM JACC Provider did not produce audit records. Extensive customization would be required to get the identity mapping information logged into TCIM to tie the records together. The custom LDAP identity data concerns are as in the previous scenario.

9

Privileged access

In this chapter we describe and test user scenarios for different privileges on each system and middleware used for this project. Each scenario is tested with the objective of documenting the audit trail.

9.1 WebSphere Application Server privileged user

As discussed in 7.2.3, “Privileged user scenarios” on page 151, the WebSphere Application Server administrator is the most powerful role. The role name *administrator* is a fixed value in WebSphere Application Server. Installation customization is achieved by mapping this role name to a user ID or group either through WebSphere Application Server itself or, in the case of WebSphere Application Server for z/OS, through the SAF (RACF) EJBROLE class (if the LocalOS user registry is being used).

Note, however, that the user IDs or groups are associated with the administrator role, and they must be valid user IDs in the configured user registry (LocalOS, LDAP, custom or federated). Also, when a JACC provider is employed for authorization, it is used for application authorization, but not for administrative authorization.

This section provides scenarios based on this role and how to audit it.

9.1.1 Environment setup

The following tasks have been executed to set up the environment for the scenarios.

z/OS setup

We chose to use the SAF EJBROLE class to authorize administrators for z/OS. This required a user registry of LocalOS. It also required setting the security custom property com.ibm.security.SAF.authorization to true.

To set up the user registry, log on to the administrative console and click **Secure administration, applications and infrastructure** (Figure 9-1). From the Available realm definitions pull-down, select **Local** operating system.

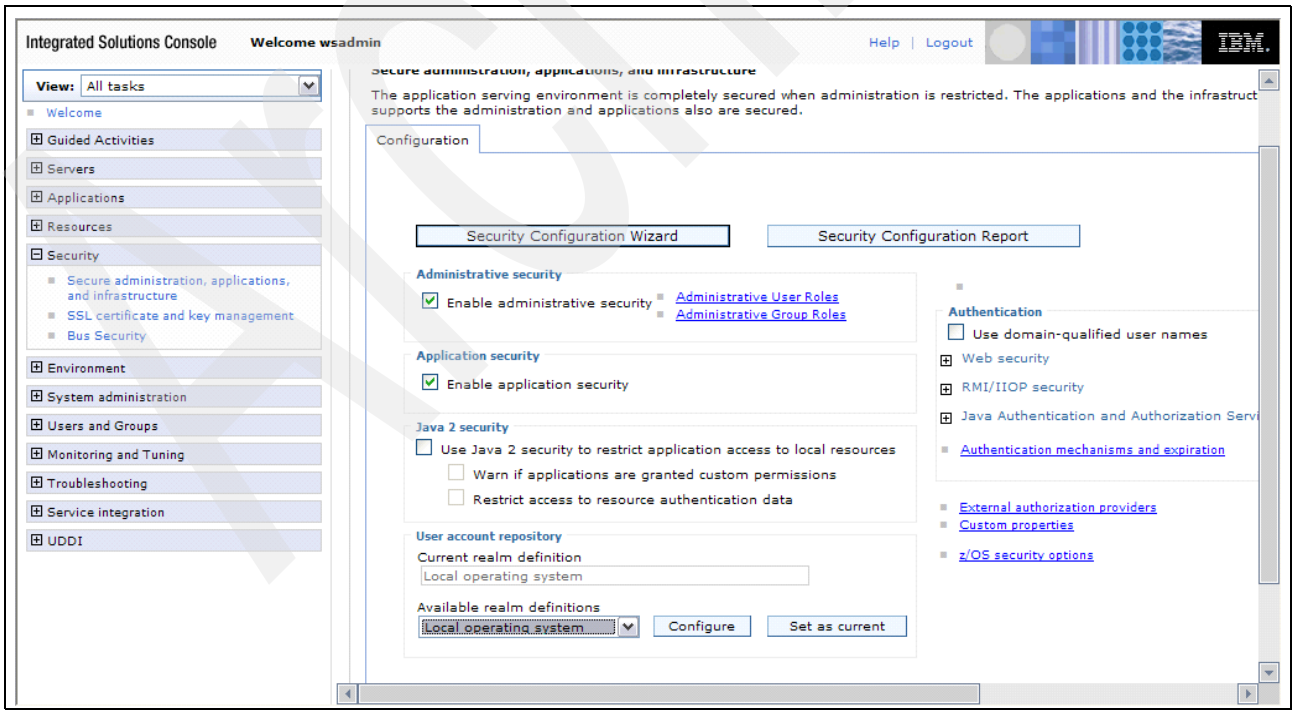


Figure 9-1 LocalOS

Next click **Customer properties** and ensure that com.ibm.security.SAF.authorization is set to true (Figure 9-2).

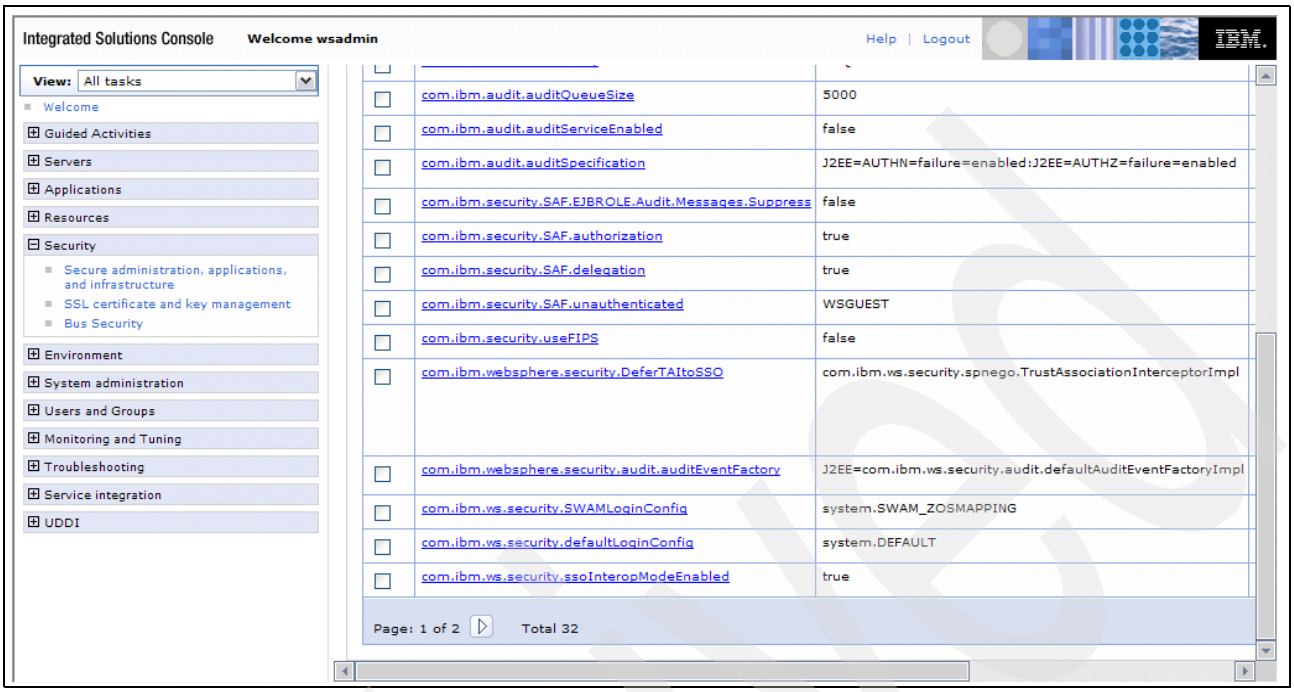


Figure 9-2 SAF authorization

Tip: Although we chose to define our application server with administrative security off, the default with WebSphere Application Server for z/OS V6.1 is to define it with security on. In that case, the values discussed here would be entered in the customization dialog.

With this setup, there is no mapping of users or groups to administrative roles within the administrative application. Instead, we created a RACF profile in the EJBROLE class and permitted the user IDs acting as administrators to them:

```
rdef EJBROLE administrator UACC(NONE)
pe administrator CLASS(EJBROLE) ID(wsadmin) ACCESS(READ)
```

Note: EJBROLE profile names are case sensitive.

Linux setup

In the case of Linux, the only option is to use WebSphere Application Server administration to map users to administrative roles. This should be performed before turning on Global Security.

In the administration console, expand **Console settings** and choose either **Console Users** or **Console Groups** (Figure 9-3).

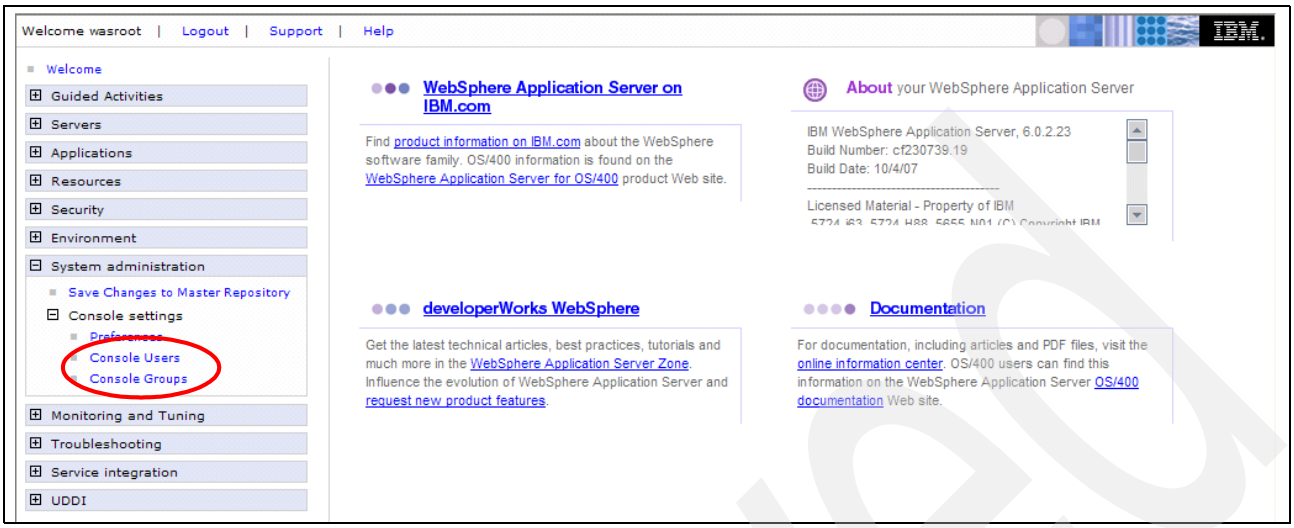


Figure 9-3 Selecting console user setting

In either case, click **Add** to be presented with the page for adding a user or group (Figure 9-4).

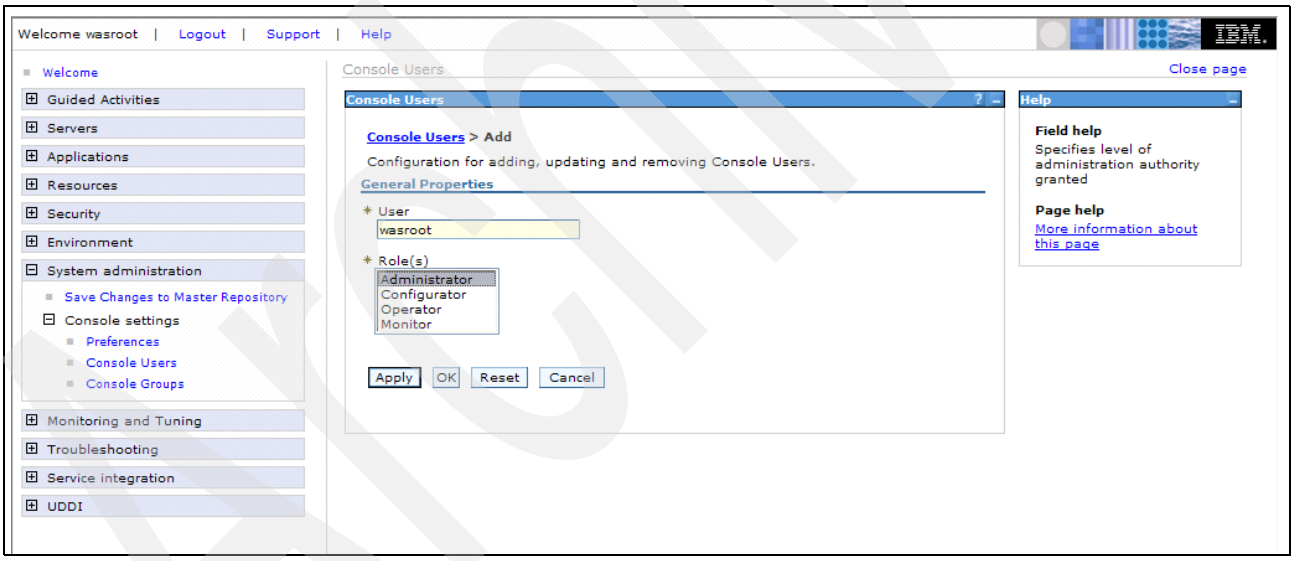


Figure 9-4 Add console user

Fill in the name and highlight one or more of the allowed roles. After clicking **OK**, click **Save** to save the configuration changes to the master repository. Since you will need to restart the entire cell configuration after turning Global Security on, it is not necessary to restart the server if you have not already turned on Global Security.

9.1.2 z/OS test case

Since we discuss SyncToOsThread in 8.1.1, “Environment and setup” on page 160, we thought it would be a good exercise to see what auditing information can be obtained when a

WebSphere Application Server administrator turns on the “Enable application server and z/OS thread identity synchronization” option.

Performing the test case

The test consisted of the following steps:

1. Log on to the WebSphere Application Server administrative console with an administrator user ID (wsadmin).
2. Expand the **Security** tree and click **Secure administration, applications, and infrastructure**.
3. Click **z/OS security options** (Figure 9-5).

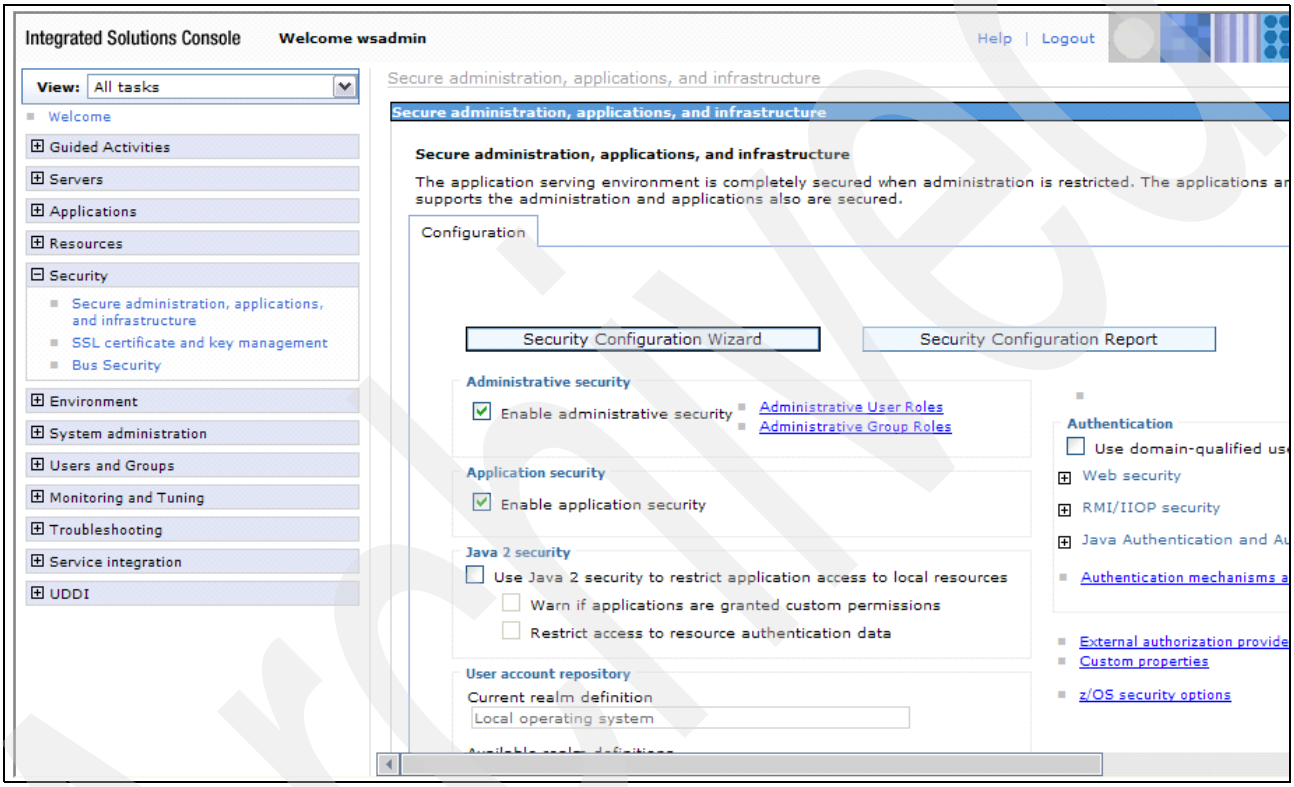


Figure 9-5 Secure administration, applications, and infrastructure

4. Check the box marked **Enable application server and z/OS thread identity synchronization** and click **OK** (Figure 9-6).
5. Click **Save** to save the changes to the master configuration.

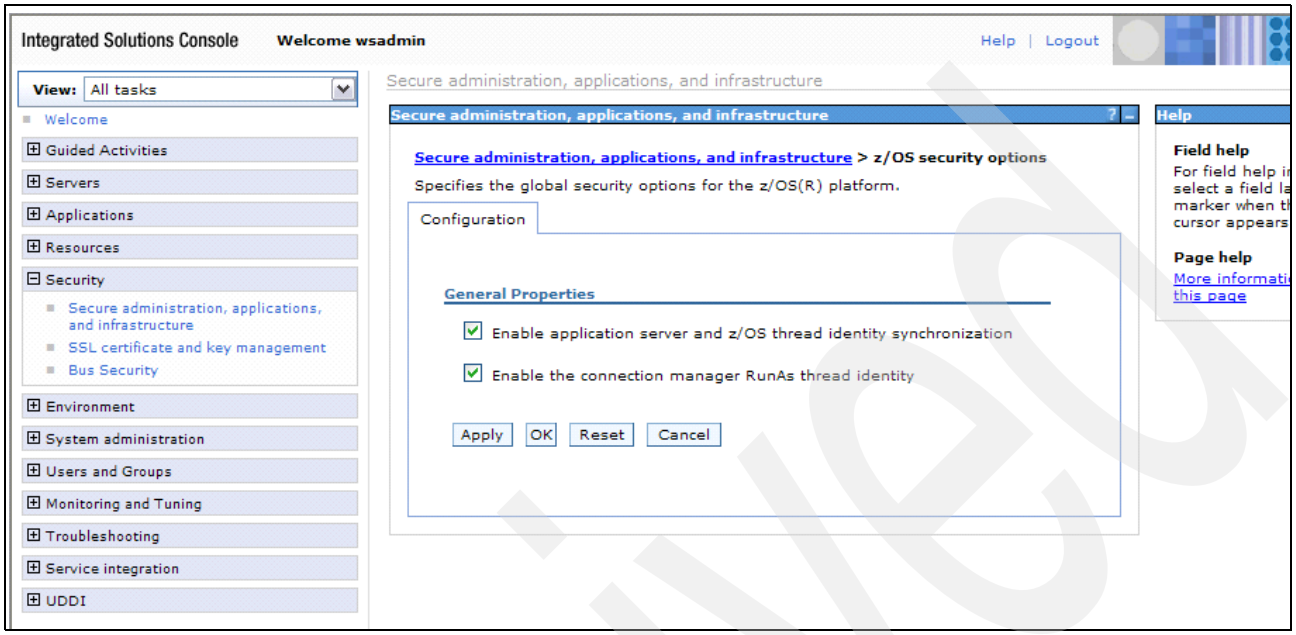


Figure 9-6 z/OS security options

Checking the audit trail

We begin by examining logs captured and formatted by Tivoli Compliance Insight Manager (TCIM) through the iView tool (see 2.1.6, “iView” on page 25).

1. From the dashboard, we choose our database, called Onepath (Figure 9-7).

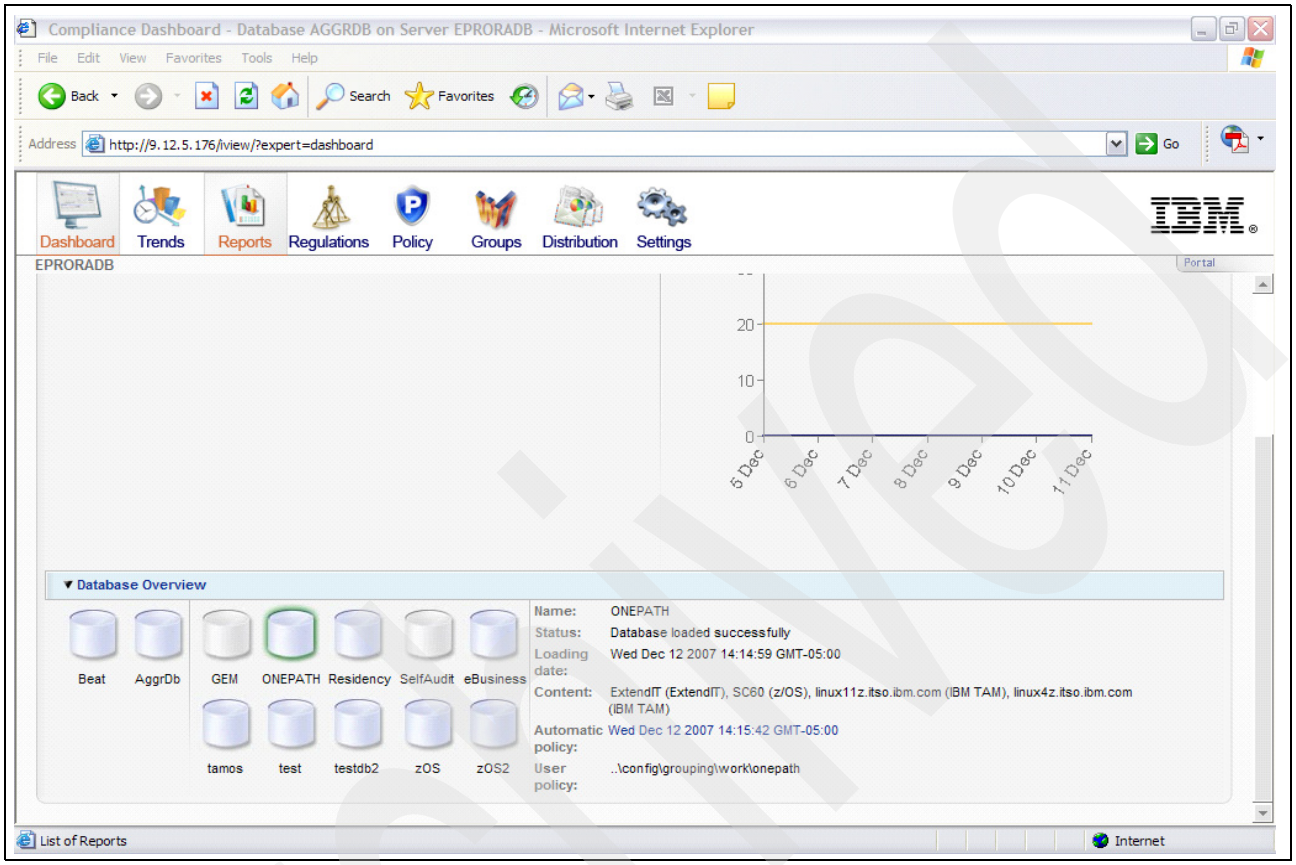


Figure 9-7 TCIM dashboard

2. From the Onepath summary page, we choose total events (Figure 9-8).

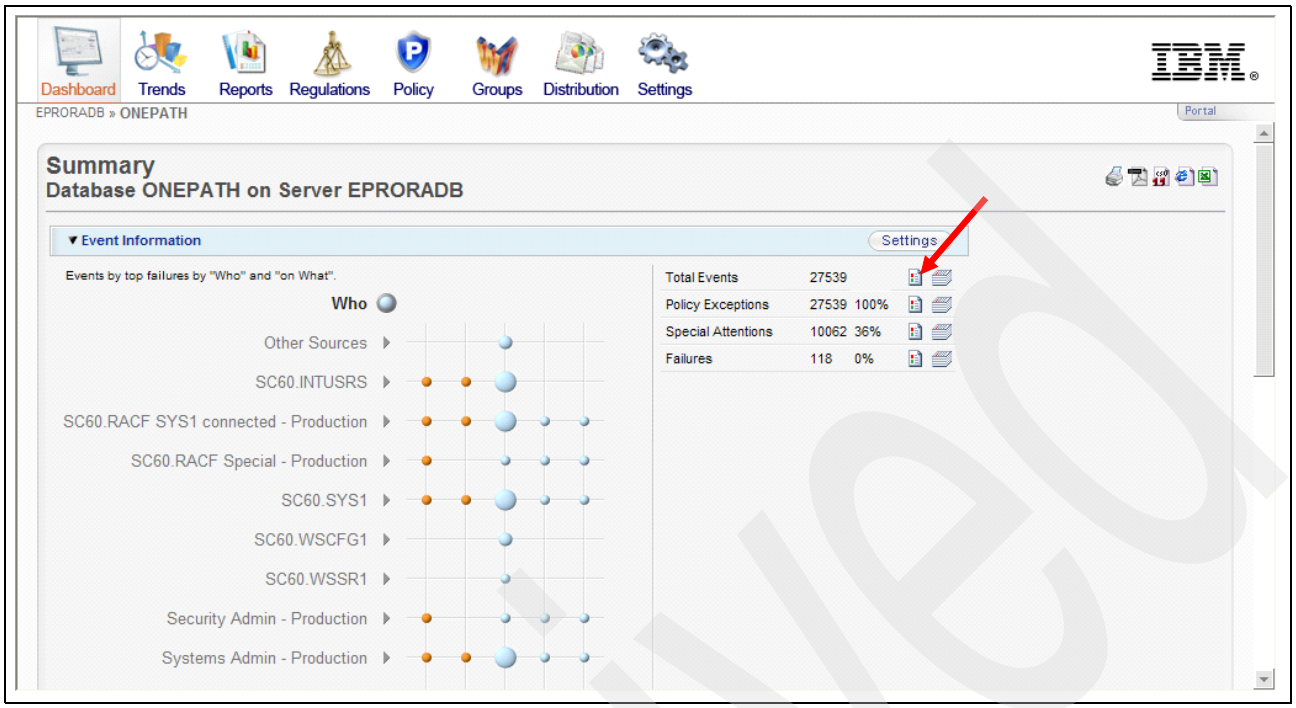


Figure 9-8 Onepath summary

3. On the resulting events page, we click the small gray square atop the Where column to filter on the z/OS system (SC60) and the small gray square atop the Who column to filter on the name of our administrator (Figure 9-9).

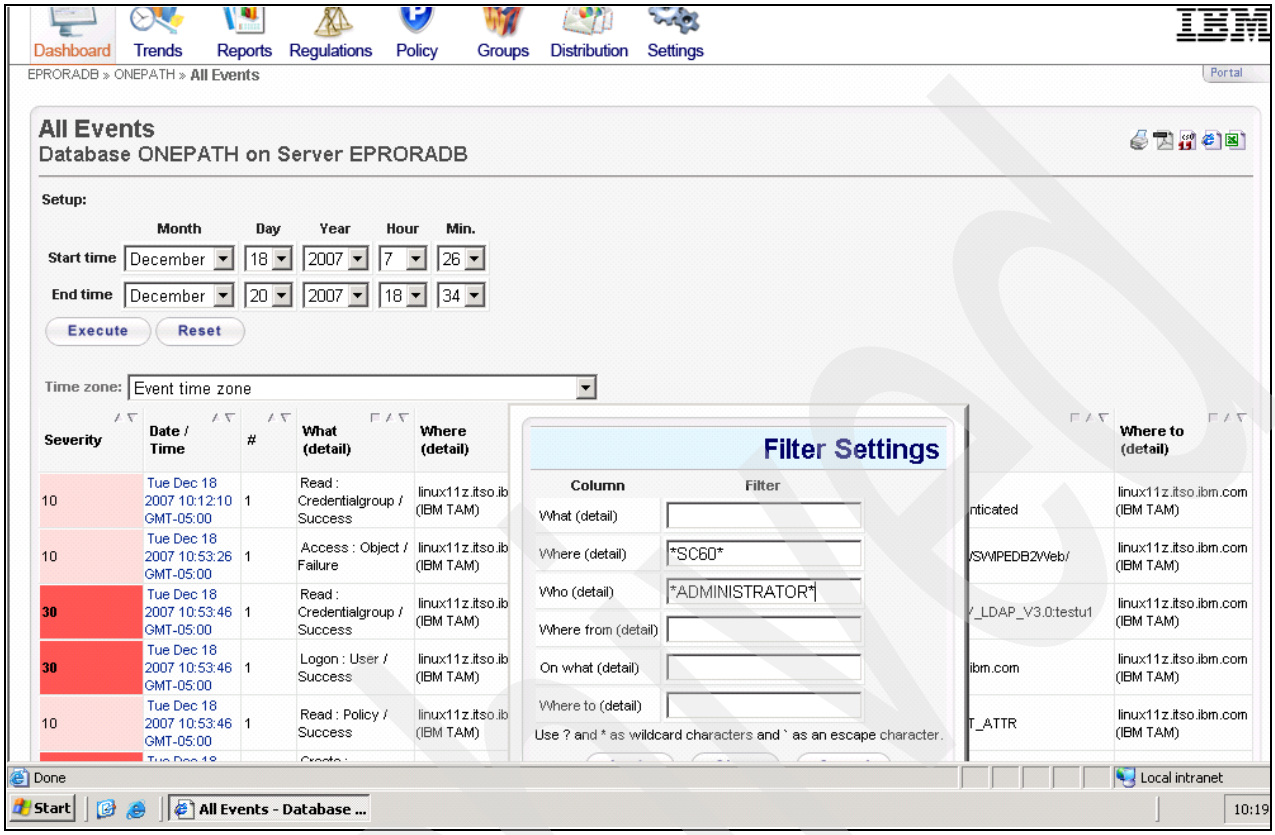
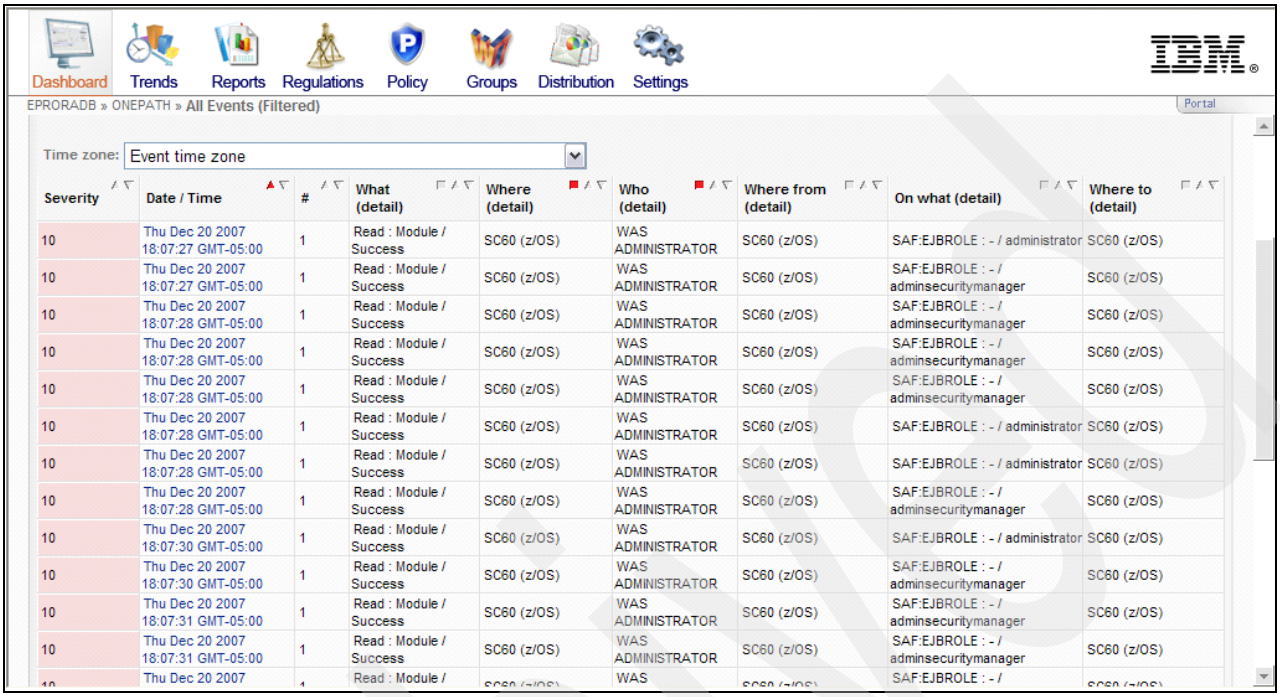


Figure 9-9 Filtering on WebSphere Application Server administrator

We see (Figure 9-10) that the administrator was logged in on December 20 around 18:07 and performed activities that required the administrator role. These records do not show us, however, what changes the administrator was making.



Time zone: Event time zone

Severity	Date / Time	#	What (detail)	Where (detail)	Who (detail)	Where from (detail)	On what (detail)	Where to (detail)
10	Thu Dec 20 2007 18:07:27 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / administrator	SC60 (z/OS)
10	Thu Dec 20 2007 18:07:27 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / adminsecuritymanager	SC60 (z/OS)
10	Thu Dec 20 2007 18:07:28 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / adminsecuritymanager	SC60 (z/OS)
10	Thu Dec 20 2007 18:07:28 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / adminsecuritymanager	SC60 (z/OS)
10	Thu Dec 20 2007 18:07:28 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / adminsecuritymanager	SC60 (z/OS)
10	Thu Dec 20 2007 18:07:28 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / administrator	SC60 (z/OS)
10	Thu Dec 20 2007 18:07:28 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / administrator	SC60 (z/OS)
10	Thu Dec 20 2007 18:07:28 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / adminsecuritymanager	SC60 (z/OS)
10	Thu Dec 20 2007 18:07:30 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / administrator	SC60 (z/OS)
10	Thu Dec 20 2007 18:07:30 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / adminsecuritymanager	SC60 (z/OS)
10	Thu Dec 20 2007 18:07:31 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / adminsecuritymanager	SC60 (z/OS)
10	Thu Dec 20 2007 18:07:31 GMT-05:00	1	Read : Module / Success	SC60 (z/OS)	WAS ADMINISTRATOR	SC60 (z/OS)	SAF:EJBROLE : - / adminsecuritymanager	SC60 (z/OS)
	Thu Dec 20 2007	1	Read : Module /		WAS		SAF:EJBROLE : - /	

Figure 9-10 WebSphere Application Server administrator activity

For further information, we can turn to the WebSphere Application Server messages, which can be found in either the z/OS system log or the SYSPRINT from the WebSphere Application Server servant address space. Example 9-1shows the messages that we find in the system log, which show that the security.xml document was modified.

Example 9-1 WebSphere Application Server SYSPRINT messages

```
18.09.42 STC21105  BBOO0222I: ADMR0016I: User  837
   837           wtsc60.itso.ibm.com/server:cl6603_nd6603_ws6603 modified document
   837           cells/cl6603/security.xml.
 18.10.23 STC21105  BBOO0222I: ADMR0016I: User  839
   839           wtsc60.itso.ibm.com/server:cl6603_nd6603_ws6603 modified document
   839           cells/cl6603/security.xml.
```

Note that the user called out in these records is not the WebSphere Application Server administrator per se, but the WebSphere Application Server server itself. This is because the administrator role gives the individual assigned to it the privilege necessary to access the administrative application. The administrative application, acting under the user ID of the servant address space, is actually making the changes. This explains why filtering on the administrator's name does not provide us with the detail about which files were accessed.

9.1.3 Linux test case

Another powerful capability of a WebSphere Application Server administrator is changing the mapping of references to resources. We thought it would be good to see what auditing information would be available when the mapping of JDBC data source references to data sources is changed.

Performing the test case

The test case consisted of the following steps:

1. Log on to the WebSphere Application Server administrative console with an administrator user ID (wasroot).
2. Expand the **Applications** tree and click **Enterprise applications**.
3. Click our test application, **SWIPEDB2** (Figure 9-11).

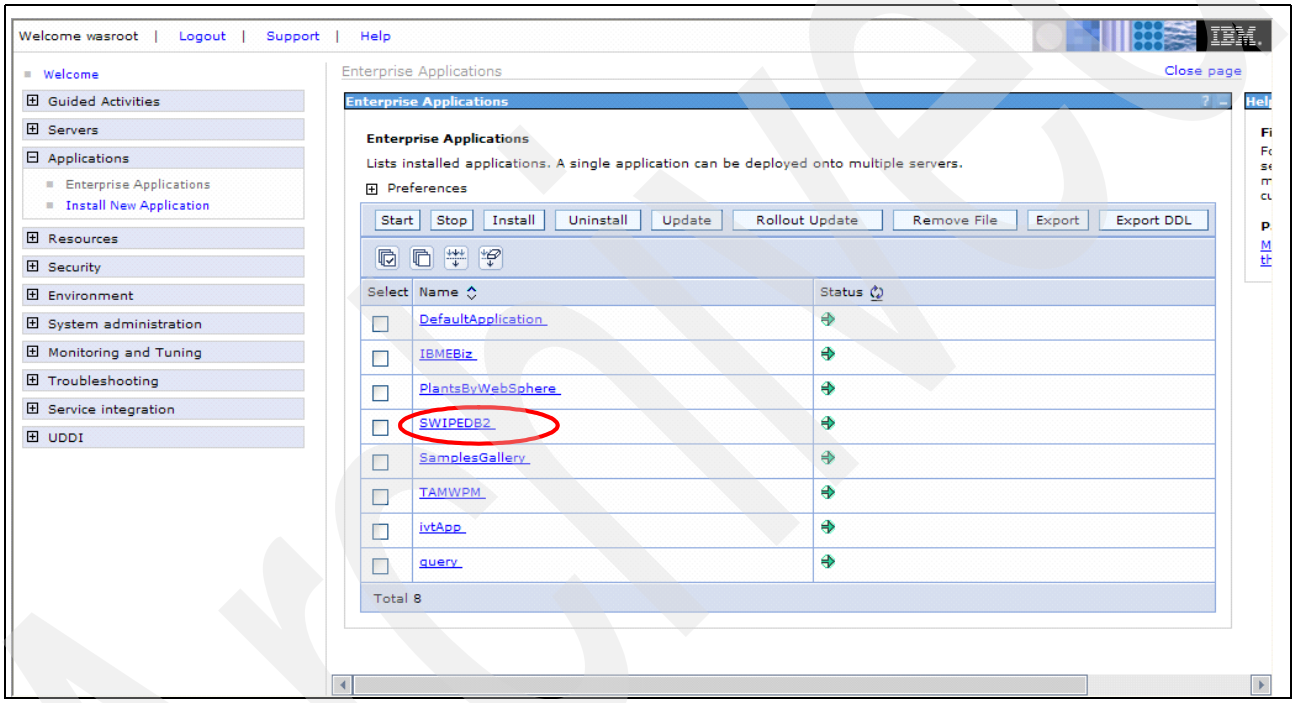


Figure 9-11 Enterprise applications

4. Click **Map resource references to resources**.

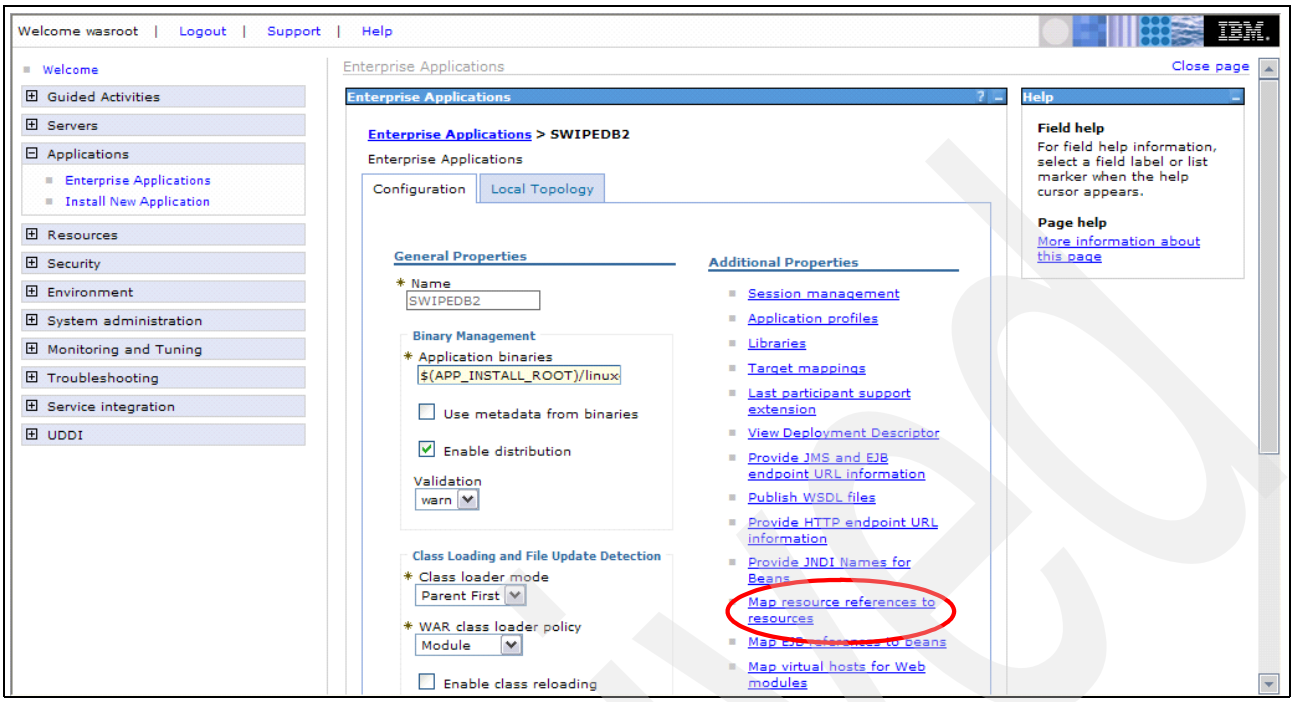


Figure 9-12 Application detail

5. Place a check mark in the box next to the resource with the reference binding jdbc/db2contJCA2.
6. At the top of the page, choose **jdbc/PlantsByWebSphereDataSource** from the pull-down and click **Apply** (Figure 9-13).

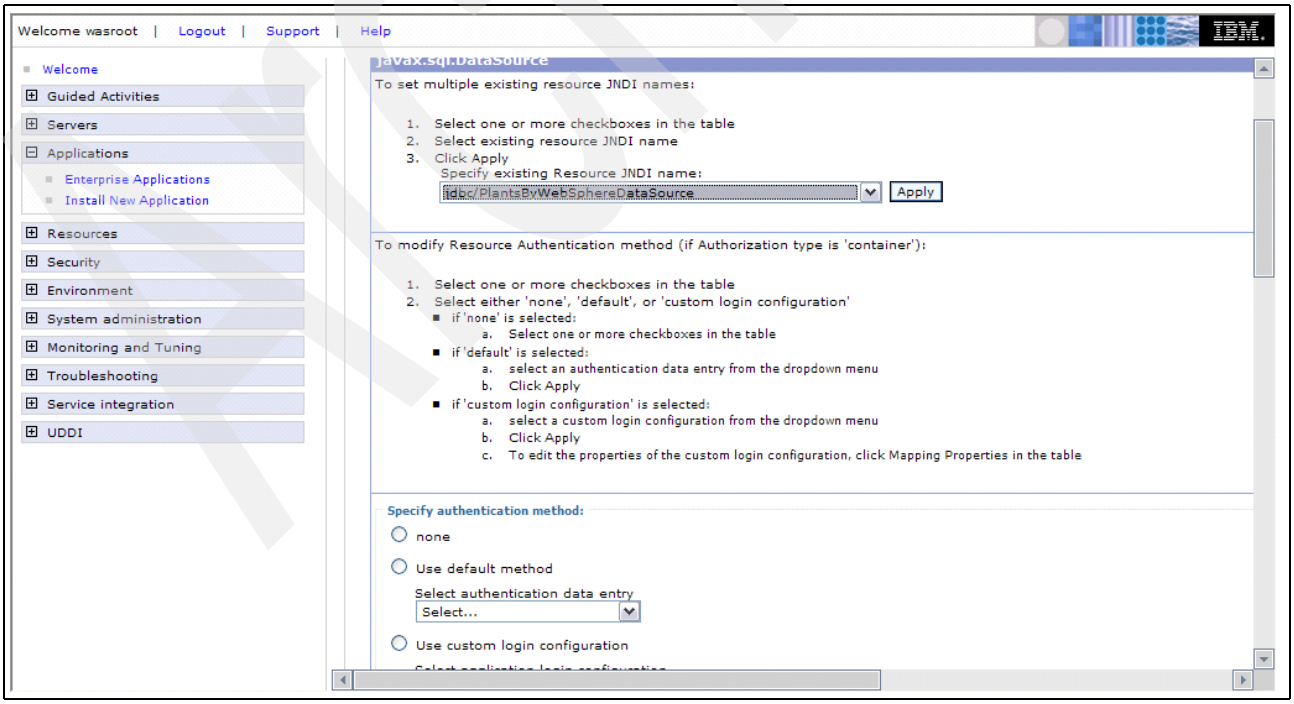


Figure 9-13 Choosing a resource

7. At the bottom of the page, click **OK**.
8. Click **Save** to apply the changes to the master configuration.
9. Log out from the administrative console.

Checking the audit trail

We begin by using the same techniques employed in “Checking the audit trail” on page 205, but this time filtering on our Linux administrator, wasroot (Figure 9-14).

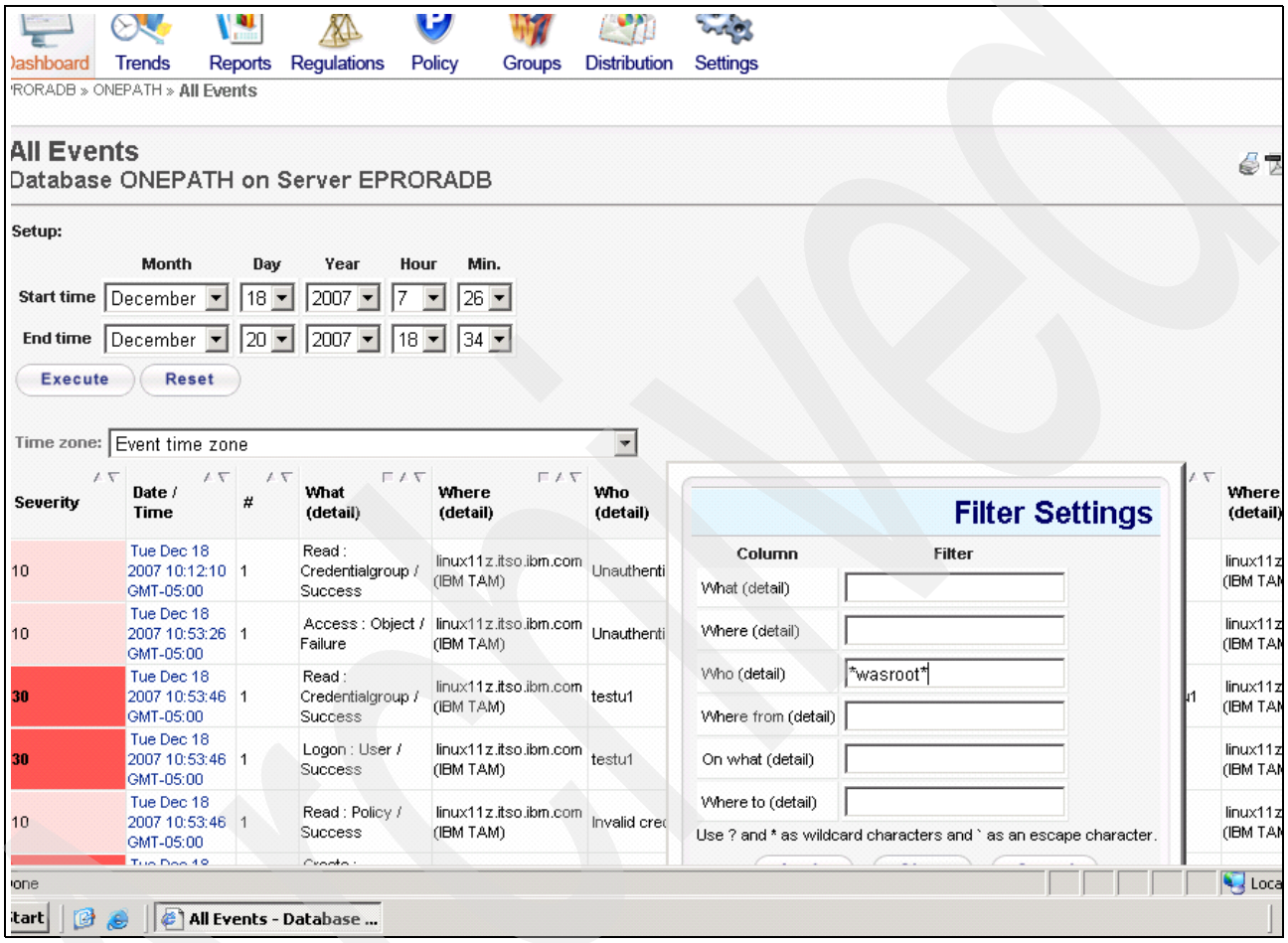
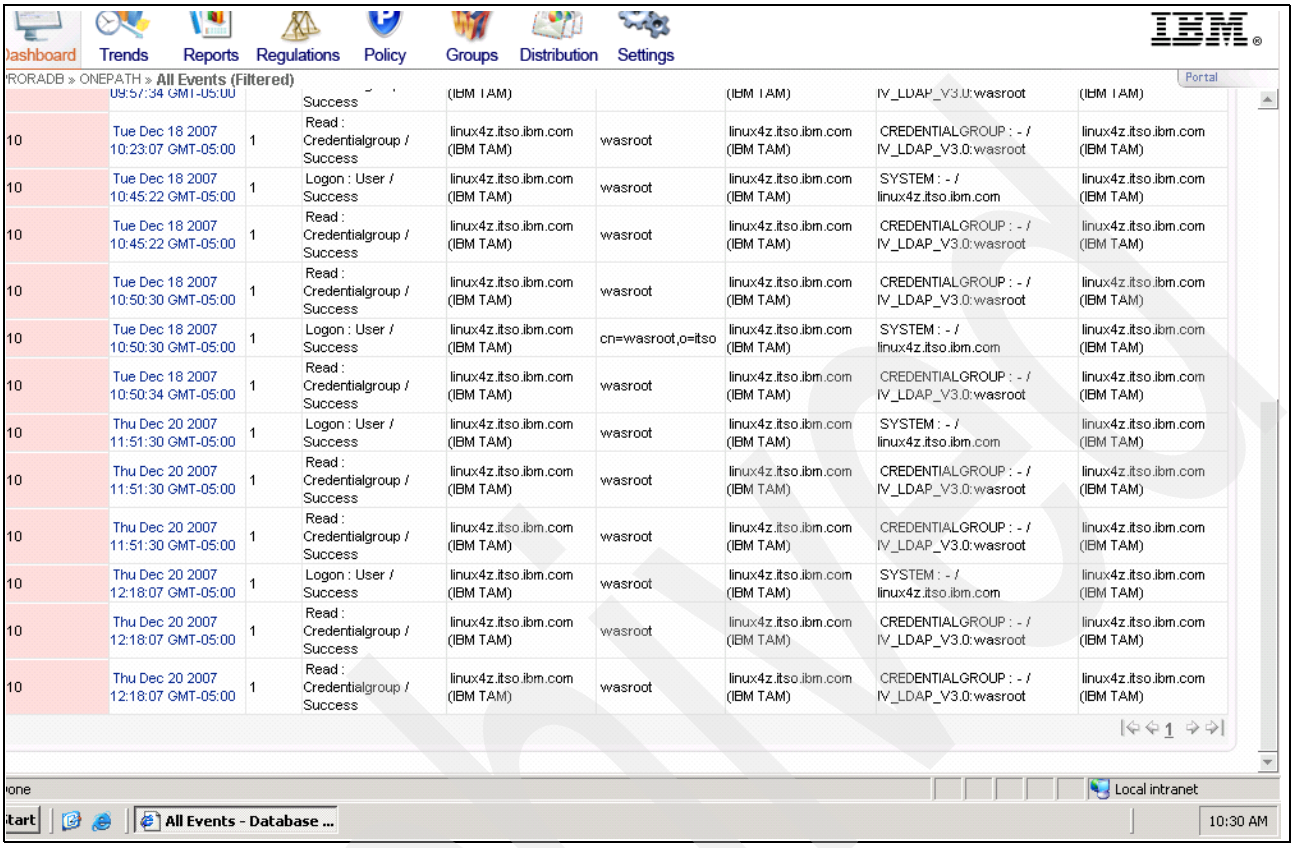


Figure 9-14 Filtering on wasroot

The results (Figure 9-15) show that the WebSphere Application Server administrator was logged in on several occasions, including 12:18 on December 20, but do not show what resources the administrator accessed.



	09:57:34 GMT-05:00		Success	(IBM TAM)		(IBM TAM)	IV_LDAP_V3.0:wasroot	(IBM TAM)
10	Tue Dec 18 2007 10:23:07 GMT-05:00	1	Read : Credentialgroup / Success	linux4z.itso.ibm.com (IBM TAM)	wasroot	linux4z.itso.ibm.com (IBM TAM)	CREDENTIALGROUP : - / IV_LDAP_V3.0:wasroot	linux4z.itso.ibm.com (IBM TAM)
10	Tue Dec 18 2007 10:45:22 GMT-05:00	1	Logon : User / Success	linux4z.itso.ibm.com (IBM TAM)	wasroot	linux4z.itso.ibm.com (IBM TAM)	SYSTEM : - / linux4z.itso.ibm.com	linux4z.itso.ibm.com (IBM TAM)
10	Tue Dec 18 2007 10:45:22 GMT-05:00	1	Read : Credentialgroup / Success	linux4z.itso.ibm.com (IBM TAM)	wasroot	linux4z.itso.ibm.com (IBM TAM)	CREDENTIALGROUP : - / IV_LDAP_V3.0:wasroot	linux4z.itso.ibm.com (IBM TAM)
10	Tue Dec 18 2007 10:50:30 GMT-05:00	1	Read : Credentialgroup / Success	linux4z.itso.ibm.com (IBM TAM)	wasroot	linux4z.itso.ibm.com (IBM TAM)	CREDENTIALGROUP : - / IV_LDAP_V3.0:wasroot	linux4z.itso.ibm.com (IBM TAM)
10	Tue Dec 18 2007 10:50:30 GMT-05:00	1	Logon : User / Success	linux4z.itso.ibm.com (IBM TAM)	cn=wasroot,o=itso	linux4z.itso.ibm.com (IBM TAM)	SYSTEM : - / linux4z.itso.ibm.com	linux4z.itso.ibm.com (IBM TAM)
10	Tue Dec 18 2007 10:50:34 GMT-05:00	1	Read : Credentialgroup / Success	linux4z.itso.ibm.com (IBM TAM)	wasroot	linux4z.itso.ibm.com (IBM TAM)	CREDENTIALGROUP : - / IV_LDAP_V3.0:wasroot	linux4z.itso.ibm.com (IBM TAM)
10	Thu Dec 20 2007 11:51:30 GMT-05:00	1	Logon : User / Success	linux4z.itso.ibm.com (IBM TAM)	wasroot	linux4z.itso.ibm.com (IBM TAM)	SYSTEM : - / linux4z.itso.ibm.com	linux4z.itso.ibm.com (IBM TAM)
10	Thu Dec 20 2007 11:51:30 GMT-05:00	1	Read : Credentialgroup / Success	linux4z.itso.ibm.com (IBM TAM)	wasroot	linux4z.itso.ibm.com (IBM TAM)	CREDENTIALGROUP : - / IV_LDAP_V3.0:wasroot	linux4z.itso.ibm.com (IBM TAM)
10	Thu Dec 20 2007 11:51:30 GMT-05:00	1	Read : Credentialgroup / Success	linux4z.itso.ibm.com (IBM TAM)	wasroot	linux4z.itso.ibm.com (IBM TAM)	CREDENTIALGROUP : - / IV_LDAP_V3.0:wasroot	linux4z.itso.ibm.com (IBM TAM)
10	Thu Dec 20 2007 12:18:07 GMT-05:00	1	Logon : User / Success	linux4z.itso.ibm.com (IBM TAM)	wasroot	linux4z.itso.ibm.com (IBM TAM)	SYSTEM : - / linux4z.itso.ibm.com	linux4z.itso.ibm.com (IBM TAM)
10	Thu Dec 20 2007 12:18:07 GMT-05:00	1	Read : Credentialgroup / Success	linux4z.itso.ibm.com (IBM TAM)	wasroot	linux4z.itso.ibm.com (IBM TAM)	CREDENTIALGROUP : - / IV_LDAP_V3.0:wasroot	linux4z.itso.ibm.com (IBM TAM)
10	Thu Dec 20 2007 12:18:07 GMT-05:00	1	Read : Credentialgroup / Success	linux4z.itso.ibm.com (IBM TAM)	wasroot	linux4z.itso.ibm.com (IBM TAM)	CREDENTIALGROUP : - / IV_LDAP_V3.0:wasroot	linux4z.itso.ibm.com (IBM TAM)

Figure 9-15 Events recorded for wasroot

Note also that in contrast to z/OS, we do not have information as to whether the administrator was actually being permitted to administrative roles, or was simply logging in to check e-mail. This is one reason why we recommend that WebSphere Application Server administrative user IDs act for reasons other than administration be considered a policy exception.

Again, we can turn to the WebSphere Application Server logs for further detail (Example 9-2). Note here that the user ID in the message is the administrator user ID. In contrast to z/OS, WebSphere Application Server on Linux performs the administrative functions under the user ID of the administrator rather than under that of the WebSphere Application Server process.

Example 9-2 SystemOut

```
[12/20/07 17:24:19:916 EST] 0000005a FileRepositor A   ADMR0016I: User
9.12.4.18:3391/wasroot modified document
cells/linux4zNode01Cell/applications/SWIPEDB2.ear/deployments/SWIPEDB2/META-
INF/application.xml.
```

Finally, note the discrepancy in time between the SystemOut messages and the iView event reports. This is due to the Linux time having been set to GMT, even though it appears to be

local time. In order to perform successful auditing, it is necessary to adopt a uniform policy of clock settings across the enterprise and to test to ensure that the time stamp reporting is consistent.

9.2 z/OS ITDS LDAP administrator DN

With the z/OS-based LDAPs, both the older *Integrate Security Services (ISS) LDAP* and the newer *Tivoli Directory Server (TDS) for z/OS*, there is a single high-powered administrative user. By default, with ISS LDAP this is called “cn=LDAP Administrator”. With the distributed ITDS, such as running on Linux for System z, there is both an administrative user and an administrative group.

The test cases in this section focus on the administrator on the z/OS ITDS server. We did not have a distributed ITDS installed, and there is no internal auditing mechanism for the z/OS ISS LDAP (see 2.2, “Tivoli Directory Server (TDS)” on page 29 discussion about auditing on LDAP).

9.2.1 Environment setup

Prior to running the test cases we checked that the RACF settings, as listed in Chapter 6, “Setting up the user ID and security for the LDAP server,” in *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191, are applied to our environment. Example 9-3 shows the commands from the manual.

Example 9-3 RACF commands to set up LDAP for generating audit records

```
RDEFINE FACILITY IRR.RAUDITX UACC(NONE)
PERMIT IRR.RAUDITX CLASS(FACILITY) ID(GLDSRV) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

We had already set up these definitions for other tasks in the project.

We also needed to ensure that the appropriate level of auditing was set for the LDAP server. It can be set in the configuration PDS member, but to confirm we ran the audit display command:

```
/F TDSSRV1,DISPLAY AUDIT
```

Example 9-4 shows the output from the command.

Example 9-4 TDS server audit setting report

```
SDSF JOB DATA SET DISPLAY - JOB TDSSRV1  (STC17201)     COMMAND ISSUED
COMMAND INPUT ===>                                            SCROLL ===> CSR
RESPONSE=SC60
 GLD1190I Audit status
 Option     Setting
 AUDIT      ON
 ERROR
 ALL        ADD BIND COMPARE CONNECT DELETE DISCONNECT EXOP MODIFY
  MODIFYDN SEARCH UNBIND
 NONE
```

We turned on all auditing for this test case, shown above as the levels add, bind, compare, connect, delete, disconnect, exop, modify, ModifyDN, search, and unbind. Depending on your audit requirements, you might choose to enable lower levels.

9.2.2 LDAP schema and user changes test case

The aim of this test case is to show the level of auditing associated with the z/OS ITDS administrative use. We performed some simple tests as the administrative user, cn=admin, which included modifying the schema, adding a user, and modifying a user to add new attributes. These functions are indicative of the types of activity performed in LDAP.

Performing the test case

The following steps were performed as the ITDS LDAP administrative user, in our case “cn=admin”. This is the user defined as the adminDN in the LDAP configuration.

The first task performed was to modify the LDAP schema to add a new attributetype and objectclass. The execution of this is shown in Example 9-5.

Example 9-5 LDAP administrator test case step 1

```
linux4z:~/scripts # cat ldap_case1.ldif

dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.4.1.4203.666.1.112 NAME
'com-ibm-example-jaas-ExamplePluggable2-Map'
    DESC 'For use with com.ibm.example.jaas.mapper '
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024})
-

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: ( 1.3.6.1.4.1.4203.666.1.113 NAME
'com-ibm-example-jaas-ExamplePluggable2'
    DESC 'For use with com.ibm.example.jaas.mapper '
    AUXILIARY
    MUST ( com-ibm-example-jaas-ExamplePluggable-Map ) )
-

linux4z:~/scripts # ./lmod_tds
+ LDAPHOST=9.12.4.18
+ LDAPPORT=3890
+ LDAPUSER=cn=admin
+ LDAPUPWD=secret
+ date
Fri Nov 16 07:46:39 EST 2007
+ idsldapmodify -h 9.12.4.18 -p 3890 -D cn=admin -w secret -f ldap_case1.ldif
modifying entry cn=schema

modifying entry cn=schema

+ date
```

```
Fri Nov 16 07:46:39 EST 2007
```

The first part of the example shows the schema changes to be made. The second part is the execution of the script (lmod_tds) to apply the changes.

The next task performed was to add a new user. This is shown in Example 9-6.

Example 9-6 LDAP administrator test case step 2

```
linux4z:~/scripts # ./lmod_add_tds
+ LDAPHOST=9.12.4.18
+ LDAPPORT=3890
+ LDAPUSER=cn=admin
+ LDAPUPWD=secret
+ date
Fri Nov 16 07:47:13 EST 2007
+ idsldapmodify -a -h 9.12.4.18 -p 3890 -D cn=admin -w secret -v -i
ldap_case2.ldif
ldap_init(9.12.4.18, 3890)
add objectclass:
        BINARY (3 bytes) top
        BINARY (6 bytes) person
        BINARY (20 bytes) organizationalPerson
add cn:
        BINARY (10 bytes) Jane Smith
add sn:
        BINARY (5 bytes) Smith
adding new entry cn=Jane Smith, ou=SampleData, o=ITSO

+ date
Fri Nov 16 07:47:13 EST 2007
```

The user was added successfully. The idsldapmodify output shows the changes being made—three objectclasses being added, and one cn and one sn being added.

Finally, we updated this user to add an additional attribute.

Example 9-7 LDAP administrator test case step 5

```
linux4z:~/scripts # ./lmod_tds
+ LDAPHOST=9.12.4.18
+ LDAPPORT=3890
+ LDAPUSER=cn=admin
+ LDAPUPWD=secret
+ date
Fri Nov 16 07:47:51 EST 2007
+ idsldapmodify -h 9.12.4.18 -p 3890 -D cn=admin -w secret -f ldap_case3.ldif
ldapmodify: expected "com-ibm-example-jaas-ExamplePluggable2-Map" but found
"com-ibm-example-jaas-ExamplePluggable-Map" (line 7 of entry: cn=Jane
Smith,ou=SampleData,o=ITSO)
+ date
Fri Nov 16 07:47:51 EST 2007
```

This last step failed, but would have still written an audit record.

There will have been a number of SMF Type 83 Sub-type 3 audit records written for this activity, including the schema modification ones, the user add, and the failed user modify.

Checking audit log capture

To extract the SMF records from the SMF data sets we ran a job on z/OS to dump all SMF Type 83 records to an XML-formatted file using the SMF dump program IFASMFDP. The job is shown in Example 9-8.

Example 9-8 Job to extract SMF Type 83 records

```
//SMF83  JOB (999,POK),'SYS  LABS',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=0M
//*
//STEP1   EXEC PGM=IEFBR14,REGION=0M
//XMLFORM DD  DISP=(SHR,DELETE),DSN=DAVIDED.SMF83.XML
//*
//STEP2   EXEC PGM=IFASMFDP,REGION=4M
//DUMPIN   DD  DSN=SYS1.SC60.MAN1,DISP=SHR
//ADUPRINT DD  SYSOUT=*
//XMLFORM  DD  DISP=(,CATLG),DSN=DAVIDED.SMF83.XML,
//             SPACE=(CYL,(15,10),RLSE),
//             LRECL=8192,RECFM=VB,BLKSIZE=24576
//DUMPOUT  DD  DUMMY
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 INDD(DUMPIN,OPTIONS(DUMP))
 OUTDD(DUMPOUT,TYPE(83))
 USER2(IRRADU00)
 USER3(IRRADU86)
```

This test case, with an additional **idsldapsearch** as the end, produced 18 SMF records in the XML file.

Example 9-9 shows some of the xml data for the first schema modification.

Example 9-9 XML SMF Type 83 ST 3 record for schema modification

```
<event>
    <eventType>*MODIFY</eventType>
    <eventQual>SUCCESS</eventQual>
    <timeWritten>07:46:39.27</timeWritten>
    <dateWritten>2007-11-16</dateWritten>
    <systemSmfid>SC60</systemSmfid>
    <prodFmid>HRSL380</prodFmid>
    <prodName>LDAP</prodName>
    <details xmlns:d="http://www.ibm.com/xmlns/zOS/LDAPSchema">
      <violation>Y</violation>
...
      <prodId>LDAP</prodId>
      <logRauditx>Y</logRauditx>
      <d:serverUrl>ldap://:3890</d:serverUrl>
      <d:connId>72</d:connId>
      <d:messageId>2</d:messageId>
      <d:bindDn>cn=admin</d:bindDn>
      <d:srcIpAddr>9.12.4.114</d:srcIpAddr>
      <d:auditVersion>V1</d:auditVersion>
```

```
      <d:eventCode>8</d:eventCode>
      <d:protocolVer>V3</d:protocolVer>
      <d:returnCode>0000000000</d:returnCode>
      <d:entryNm>cn=schema</d:entryNm>
      <d:relocate205>Fri Nov 16 07:46:39 2007</d:relocate205>
      <d:entrySuffix>CN=SCHEMA</d:entrySuffix>
      <d:cntrlsPresnt>1.3.6.1.4.1.42.2.27.8.5.1,FALSE</d:cntrlsPresnt>
      <d:modAttrAdd>attributetypes</d:modAttrAdd>
    </details>
  </event>
```

The top of the event entry shows the type of activity, the success or failure, and the date and time of execution. Much of the entry is taken up with standard SMF fields.

The information we are interested in is towards the bottom of the entry in the detail (d:*) fields. You can see the bind DN or user who performed the action, along with where they ran the command. The changes made are shown in the d:entrySuffix and the d:modAttrAdd fields.

Example 9-10 shows the audit record for the user add.

Example 9-10 XML SMF Type 83 ST 3 record for schema modification

```
<event>
    <eventType>*ADD</eventType>
    <eventQual>SUCCESS</eventQual>
    <timeWritten>07:47:13.82</timeWritten>
    <dateWritten>2007-11-16</dateWritten>
    <systemSmfid>SC60</systemSmfid>
...
      <d:serverUrl>ldap://:3890</d:serverUrl>
      <d:connId>73</d:connId>
      <d:messageId>2</d:messageId>
      <d:bindDn>cn=admin</d:bindDn>
      <d:srcIpAddr>9.12.4.114</d:srcIpAddr>
      <d:auditVersion>V1</d:auditVersion>
      <d:eventCode>1</d:eventCode>
      <d:protocolVer>V3</d:protocolVer>
      <d:returnCode>0000000000</d:returnCode>
      <d:entryNm>cn=Jane Smith, ou=SampleData, o=ITSO</d:entryNm>
      <d:relocate205>Fri Nov 16 07:47:13 2007</d:relocate205>
      <d:entrySuffix>O=ITSO</d:entrySuffix>
      <d:cntrlsPresnt>1.3.6.1.4.1.42.2.27.8.5.1,FALSE</d:cntrlsPresnt>
      <d:addAttr>objectclass</d:addAttr>
      <d:addAttr>cn</d:addAttr>
      <d:addAttr>sn</d:addAttr>
    </details>
  </event>
```

You can see that the entry added was cn=Jane Smith, ou=SampleData, o=ITSO (the d:entryNm field); the suffix affected was o=ITSO; and the attributes added were objectclass, cn, and sn (d:addAttr fields).

Note that the records only show that changes were made, not what data was changed.

We have not shown all records relating to the test cases.

Audit reporting

Unlike many of the other examples in this book, there is no TCIM collection of the SMF Type 83 records. The SMF Type 83 Subtype 3 records are well formed, so it would not be too hard to develop some integration to periodically dump SMF, pipe the XML files to the TCIM depot, and build a load process to parse the XML data into W7 format. TDI could certainly do this.

9.3 TAM sysadmin

The TAM products, such as TAMeb and TAMOS, share administrative users. There is an installation-wide system administrator, often called sec_master. There is an administrative group called iv-admin. If the installation is partitioned into multiple domains, there may also be domain administrators with privileged access to the objects in that domain.

These users are internal to the products and have no relationship to operating system users, so the only operating system auditing that will show use may be when the **pdadmin** command-line interface is called with the user listed, such as:

```
pdadmin -a sec_master -p secret
```

This section demonstrates some use cases related to the use of the privileged users in TAM and how their activity can be tracked. The last part of the section shows how the activity can be reported on in TCIM.

9.3.1 Environment setup

Before running the tests we need to confirm that auditing is enabled for administrative use. This is set in the TAM configuration files, such as ivmgrd.conf (for the TAM policy server) and the ivacld.conf (for the TAM authorization server).

Example 9-11 shows the audit settings for the TAM policy server.

Example 9-11 TAM ivmgrd.conf auditing settings

```
[aznapi-configuration]

# Audit logging configuration.
#
#       logcfg          Configure logging.  If uncommented, the following
#                       logcfg statements will enable authentication and/or
#                       authorization auditing for the application.
#
#       logcfg attributes
#          audit.azn:     Category indicates auditing of azn component
#          file           Destination (or agent) type file
#          path           For file agent, specifies the log filename
#          flush_interval Frequency for flushing log file buffers
#          log_id         Direct events from additional categories to same agent
#
logcfg = audit.azn:file
path=/var/PolicyDirector/audit/pdmgrd.log,flush_interval=20,log_id=PDMgrAudit,mode
=text
```

```
logcfg = audit.authn:file log_id=PDMgrAudit
logcfg = audit.authz:file log_id=PDMgrAudit
logcfg = audit.http:file log_id=PDMgrAudit
logcfg = audit.mgmt:file log_id=PDMgrAudit
```

In this example all possible auditing is enabled, which is not necessary and may produce too much data. For these test cases we are concerned with the audit.mgmt level. All auditing is being sent to the same destination, the /var/PolicyDirector/audit/pdmgrd.log file. You could specify different destinations for the different types of audit information.

Example 9-12 shows the audit settings for the TAM authorization server.

Example 9-12 TAM ivacld.conf auditing settings

```
[aznapi-configuration]

# Audit logging configuration.
#
#       logcfg          Configure logging.  If uncommented, the following
#                       logcfg statements will enable authentication and/or
#                       authorization auditing for the application.
#
#       logcfg attributes
#          audit.azn:     Category indicates auditing of azn component
#          file           Destination (or agent) type file
#          path           For file agent, specifies the log filename
#          flush_interval Frequency for flushing log file buffers
#          log_id         Direct events from additional categories to same agent
#
# Audit log settings
logcfg = audit.azn:file
path=/var/PolicyDirector/audit/ivacld.log,flush_interval=20,log_id=PDAclAudit
logcfg = audit.authn:file log_id=PDAclAudit
logcfg = audit.mgmt:file log_id=PDAclAudit
```

The important one for the use cases below is the audit.mgmt setting. All audit records will go to /var/PolicyDirector/audit/ivacld.log.

By default the audit sub-directory and the audit logs are owned (in UNIX/Linux) by the user ivmgr and group ivmgr. The user has read+write permission to the files and the group has read permission to the files, as shown in Example 9-13.

Example 9-13 Permission settings on TAM audit log files

```
linux4z:/var/PolicyDirector # ls -l
total 40
drwxrwxr-x  10 ivmgr ivmgr 4096 2007-10-24 17:30 .
drwxr-xr-x  20 root  root  4096 2007-10-23 15:46 ..
drwxrwxr-x   2 ivmgr ivmgr 4096 2007-11-07 17:58 audit
drwxrwxr-x   2 ivmgr ivmgr 4096 2005-10-29 16:07 cache
drwxrwxr-x   2 ivmgr ivmgr 4096 2007-11-12 16:01 db
drwxrwxr-x   2 ivmgr ivmgr 4096 2007-10-24 17:32 keytab
drwxrwxrwx   2 ivmgr ivmgr 4096 2007-11-12 07:17 log
drwxr-x---   2 ivmgr ivmgr 4096 2005-10-29 16:07 pdacld
drwxrwxr-x   2 ivmgr ivmgr 4096 2007-11-05 17:10 pdbackup
drwxr-x---   2 ivmgr ivmgr 4096 2005-10-29 16:07 pdmgrd
```

```
linux4z:/var/PolicyDirector # ls -l audit
total 15516
drwxrwxr-x   2 ivmgr ivmgr     4096 2007-11-07 17:58 .
drwxrwxr-x  10 ivmgr ivmgr     4096 2007-10-24 17:30 ..
-rw-r-----   1 ivmgr ivmgr  1189786 2007-11-13 07:48 ivacld.log
-rw-r-----   1 ivmgr ivmgr 14661270 2007-11-13 07:48 pdmgrd.log
```

The ivmgr user cannot be logged on to, but anyone with root (UID=0) privilege can access these files. We recommend applying operating system level auditing to monitor who has accessed and edited the files. If your audit control requirements dictate a higher level of control, you should implement a tool like TAMOS that can stop root from viewing and editing specific files.

9.3.2 Listing the users with iv-admin test case

There is one installation-wide administrative user for the TAM products, normally called sec_master. Other users can be made administrators by adding them to the iv-admin group.

There are two ways to identify the members of the administrative group iv-admin, through the TAM tools or by viewing the entries in the user repository.

This test case does not demonstrate tracking and reporting activity as the other test cases do. It is concerned with identifying who are the privileged users in a TAM system.

Using TAM tools to check iv-admin

TAM has two administrative interfaces that it provides: the Web Portal Manager (WPM) Java application accessed through a browser and the **pdadmin** command-line tool.

To view the group members in the Web Portal Manager select **Group** → **Search Groups** and enter the group name. When the group detail is presented select the **Members** tab. An example of this is shown in Figure 9-16.

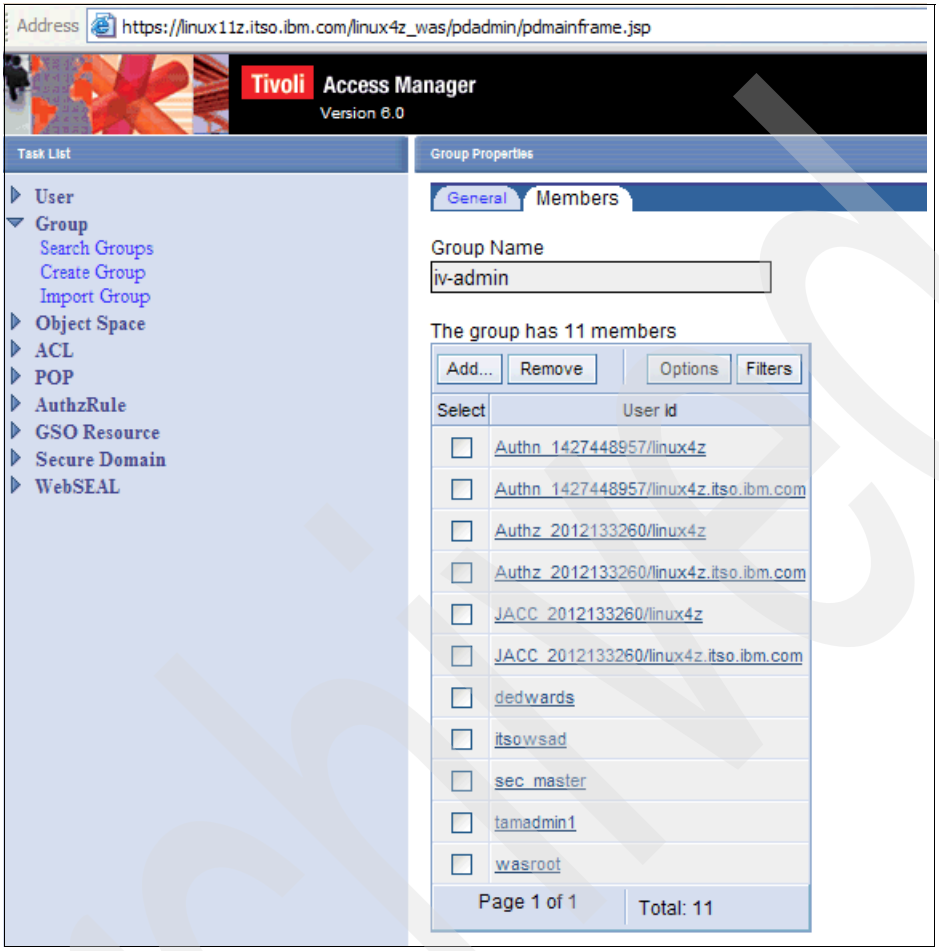


Figure 9-16 TAM Web Portal Manager iv-admin group members

The other way to do this is to use the **pdadmin** command, as shown in Example 9-14.

Example 9-14 pdadmin iv-admin group members

```
linux4z:/opt/PolicyDirector/etc # pdadmin -a sec_master
Enter Password:
pdadmin sec_master> group show-members iv-admin
sec_master
itsowsad
wasroot
Authn_1427448957/linux4z
JACC_2012133260/linux4z
Authz_2012133260/linux4z
dedwards
tamadmin1
Authn_1427448957/linux4z.itso.ibm.com
JACC_2012133260/linux4z.itso.ibm.com
Authz_2012133260/linux4z.itso.ibm.com
```

The members shown are the same as in the Web Portal Manager, just ordered differently.

Some of the users shown are internal users, such as the Authn_* and JACC_* users belonging to the TAM JACC Provider. The other users, such as dedwards and wasroot, may need to be monitored in addition to sec_master.

Using LDAP tools to check iv-admin membership

TAM holds its users and groups in a user repository. There are many supported repositories, but they are normally LDAP-compliant. This means that the ldap client tools, such as ldapsearch, can be used to view the data.

The TAM security (administrative) groups are held in cn=SecurityGroups,secAuthority=Default. Example 9-15 shows an **idsldapsearch** command to see the group members.

Example 9-15 idsldapsearch to view the iv-admin members

```
+ idsldapsearch -b cn=iv-admin,cn=SecurityGroups,secAuthority=Default -s base -h
9.12.4.18 -p 3391 -D 'cn=LDAP Administrator' -w secret 'objectclass=*'
cn=iv-admin,cn=SecurityGroups,secAuthority=Default
objectclass=top
objectclass=accessGroup
cn=iv-admin
member=secAuthority=Default
member=cn=SecurityMaster,secAuthority=Default
member=cn=itsowsad,o=itso
member=cn=wasroot,o=itso
member=cn=Authn_1427448957/linux4z,cn=SecurityDaemons,secAuthority=Default
member=cn=JACC_2012133260/linux4z,cn=SecurityDaemons,secAuthority=Default
member=cn=Authz_2012133260/linux4z,cn=SecurityDaemons,secAuthority=Default
member=cn=David Edwards, ou=Employees, o=ITSO
member=cn=TAM Admin1, ou=Employees, o=ITSO
member=cn=Authn_1427448957/linux4z.itso.ibm.com,cn=SecurityDaemons,secAuthority=De
fault
member=cn=JACC_2012133260/linux4z.itso.ibm.com,cn=SecurityDaemons,secAuthority=Def
ault
member=cn=Authz_2012133260/linux4z.itso.ibm.com,cn=SecurityDaemons,secAuthority=De
fault
description=DO NOT MODIFY: Controls Access Manager Administrators
```

In this case we are using the **idsldapsearch** command on a Linux for System z system and remotely accessing a z/OS ISS LDAP. The same command could have been used to view the entries on a distributed ITDS.

9.3.3 Creating another iv-admin user test case

This test case involves the sec_master user creating another administrative user. This may be a perfectly valid activity but should be audited so that the activity can be validated.

In this section we show the activity, the logging generated by the activity, and the reporting that can be performed on the activity.

Performing the test case

A script was created to run the **pdadmin** commands. Three commands were run—one to create the user, one to show the newly created user, and one to list the iv-admin group members. Example 9-16 shows the script execution, with the **pdamdin** commands highlighted.

Example 9-16 sec_master test case 1 execution

```
linux4z:~/scripts # ./pdadmin_audit pdadmin_case1.pd
+ date
Tue Nov 13 08:49:52 EST 2007
+ pdadmin -a sec_master -p secret pdadmin_case1.pd
cmd> user create newadmin cn=newadmin,o=itso newadmin admin secret00 iv-admin
cmd> user show newadmin
Login ID: newadmin
LDAP DN: cn=newadmin,o=itso
LDAP CN: newadmin
LDAP SN: admin
Description:
Is SecUser: Yes
Is GSO user: No
Account valid: No
Password valid: Yes
cmd> group show-members iv-admin
sec_master
itsowsad
wasroot
Authn_1427448957/linux4z
JACC_2012133260/linux4z
Authz_2012133260/linux4z
dedwards
tamadmin1
Authn_1427448957/linux4z.itso.ibm.com
JACC_2012133260/linux4z.itso.ibm.com
Authz_2012133260/linux4z.itso.ibm.com
newadmin
+ date
Tue Nov 13 08:49:53 EST 2007
```

The time stamps were added to make searching the audit trail easier.

Checking audit log capture

Audit records for the above activity are written to the TAM policy server audit log, pdmgrd.log. The first records show the authentication of sec_master and the authorization to run the command, as shown in Example 9-17.

Example 9-17 pdmgrd.log audit records for sec_master authentication and authorization

```
<event rev="1.2">
<date>2007-11-13-08:49:52.593-05:00I-----</date>
<outcome status="0">0</outcome>
<originator blade="pdmgrd"><component rev="1.3">authn</component>
<event_id>101</event_id>
<action>0</action>
<location>linux4z.itso.ibm.com</location>
</originator>
<accessor name="">
```

```
<principal auth="password" domain="Default">sec_master</principal>
</accessor><target resource="7"><object></object></target>
<data>
</data>
</event>
<event rev="1.2">
<date>2007-11-13-08:49:52.610-05:00I-----</date>
<outcome status="0">0</outcome>
<originator blade="pdmgrd"><component rev="1.2">azn</component>
<event_id>110</event_id>
<action>0</action>
<location>linux4z.itso.ibm.com</location>
</originator>
<accessor name="">
<principal auth="IV_LDAP_V3.0" domain="Default">sec_master</principal>
<name_in_rgy>cn=SecurityMaster,secAuthority=Default</name_in_rgy><user_location>9.
12.4.114</user_location><user_location_type>IPV4</user_location_type></accessor><t
arget resource="3"><object>IV_LDAP_V3.0:sec_master</object></target>
<data>
azn_id_get_creds</data>
</event>
```

Note that the authorization record does not show what authorization was permitted.

Note: The XML formatted TAM audit records can be hard to read. For the remaining TAM audit record examples in this chapter we omit the irrelevant XML entries for simplicity and mark the removed lines by ellipses (...).

Example 9-18 shows the audit record written for the **user create** command.

Example 9-18 pdmgrd.log audit records for user create

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">sec_master</principal>
...
<mgmtinfo><command>USER CREATE</command>
<objname>newadmin</objname>
<objname_rgy>cn=newadmin,o=itso</objname_rgy>
<objtype>user</objtype>
<parm><name>groupid</name><value>iv-admin</value></parm>
<parm><name>nopwdpolicy</name><value>false</value></parm>
<parm><name>gsouser</name><value>false</value></parm>
<parm><name>loginid</name><value>newadmin</value></parm>
<parm><name>dn</name><value>cn=newadmin,o=itso</value></parm>
<parm><name>cn</name><value>newadmin</value></parm>
<parm><name>sn</name><value>admin</value></parm>
</mgmtinfo>
...
```

There is no audit record written for the **user show** and **group show-members** commands.

9.3.4 Changing TAM policy test case

The TAM administrators are responsible for managing the security policy implemented in TAM. This includes the users and groups, the resources, the access control lists (ACLs), and the protected object policies (POPs). These are mapped together in the TAM protected object space, often just called the TAM *object space*.

A malicious or accidental change to policy could have significant impact on the operation of the business application. This test case shows changing ACLs applied to resources in TAM and how it is tracked and reported.

Performing the test case

For this test case we have a simple HTML application for displaying help pages. Access to the application is controlled by WebSEAL with a resource controlling the Web folder containing the help files and an ACL attached to it. Figure 9-17 shows the resource (intranet_help) under the /WebSEAL/linux11z.itso.ibm.com-default branch in the TAM objectspace.

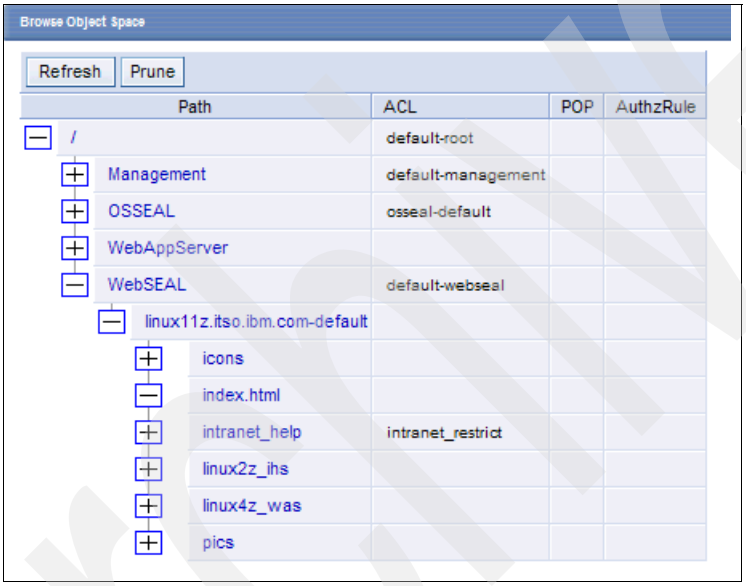


Figure 9-17 WPM Browse Object Space display

The ACL attached to the resource is called intranet_restrict. Figure 9-18 shows the access allowed by this ACL:

- The sec_master user has full control.
- The intranet-users group has read access (that is, they can access the files in the directory).
- The any-other and unauthenticated special types have no access.

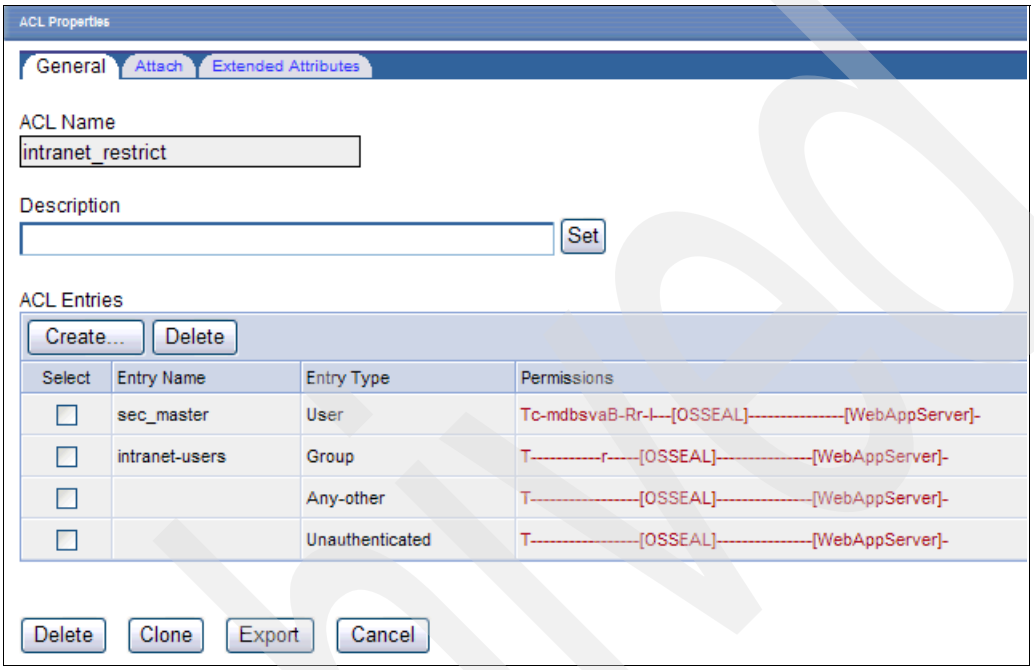


Figure 9-18 WPM ACL Properties display

In this test case the sec_master user makes three changes to this policy:

- Add the user jsmith to the ACL so that he can read resources (that is, view the html pages).
- Add the group iv-admin to the ACL so that the members have full control to the resources.
- Detach and delete the ACL completely so that it inherits the general access from the branch above.

Example 9-19 shows the relevant commands being executed through pdadmin.

Example 9-19 sec_master test case 2 execution

```
linux4z:~/scripts # ./pdadmin_audit pdadmin_case2.pd
+ date
Tue Nov 13 11:01:20 EST 2007
+ pdadmin -a sec_master -p secret pdadmin_case2.pd
cmd> acl modify intranet_restrict set user jsmith Tr
cmd> acl modify intranet_restrict set group iv-admin TcmdbsvaBRlr
cmd> acl detach /WebSEAL/linux11z.itso.ibm.com-default/intranet_help
cmd> acl delete intranet_restrict
+ date
Tue Nov 13 11:01:21 EST 2007
```

Note that if the commands were successful, there are no additional messages displayed.

Checking audit log capture

As in test case 1, the records are written to the TAM policy server audit log, which in our environment is pdmgrd.log. As before, there are authentication and authorization records written for the sec_master user running the **pdadmin** command.

The first mgmt record cut is the one for adding the user jsmith to the ACL, as shown in Example 9-20.

Example 9-20 pdmgrd.log audit records for acl modify #1

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">sec_master</principal>
...
<mgmtinfo><command>ACL SET</command>
<objname>intranet_restrict</objname>
<objtype>acl</objtype>
<parm><name>aclentrytype</name><value>3</value></parm>
<parm><name>aclentrytype</name><value>2</value></parm>
<parm><name>aclentrytype</name><value>1</value></parm>
<parm><name>aclentrytype</name><value>0</value></parm>
<parm><name>aclentrytype</name><value>3</value></parm>
<parm><name>aclentryactions</name><value>TcmdbsvaBRrl</value></parm>
<parm><name>aclentryactions</name><value>Tr</value></parm>
<parm><name>aclentryactions</name><value>T</value></parm>
<parm><name>aclentryactions</name><value>T</value></parm>
<parm><name>aclentryactions</name><value>Tr</value></parm>
<parm><name>aclid</name><value>intranet_restrict</value></parm>
<parm><name>acldesc</name><value></value></parm>
<parm><name>aclentrycount</name><value>5</value></parm>
<parm><name>aclentryid</name><value>sec_master</value></parm>
<parm><name>aclentryid</name><value>intranet-users</value></parm>
<parm><name>aclentryid</name><value></value></parm>
<parm><name>aclentryid</name><value></value></parm>
<parm><name>aclentryid</name><value>jsmith</value></parm>
<parm><name>aclid</name><value>intranet_restrict</value></parm>
</mgmtinfo>
...
```

We can see that the command being run is an ACL SET on the intranet_restrict ACL. The audit record does not show the changes to the ACL. It shows the ACL after the change. You cannot see from the record what has been changed.

Notice also the structure of the ACL entries in the audit record. The entries are an array within the ACL object. The new entry (for user jsmith) is the fifth entry in the ACL, but to see the values for that entry, you have to look at the fifth aclentrytype, aclentryactions, and aclentryid parms.

This approach to writing audit records makes identifying individual changes almost impossible.

Example 9-21 shows the audit record cut for the modify that added the iv-admin group to the ACL.

Example 9-21 pdmgrd.log audit records for acl modify #2

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">sec_master</principal>
...
<mgmtinfo><command>ACL SET</command>
<objname>intranet_restrict</objname>
<objtype>acl</objtype>
<parm><name>aclentrytype</name><value>3</value></parm>
<parm><name>aclentrytype</name><value>2</value></parm>
<parm><name>aclentrytype</name><value>1</value></parm>
<parm><name>aclentrytype</name><value>0</value></parm>
<parm><name>aclentrytype</name><value>3</value></parm>
<parm><name>aclentrytype</name><value>2</value></parm>
<parm><name>aclentryactions</name><value>TcmdbsvaBRrl</value></parm>
<parm><name>aclentryactions</name><value>Tr</value></parm>
<parm><name>aclentryactions</name><value>T</value></parm>
<parm><name>aclentryactions</name><value>T</value></parm>
<parm><name>aclentryactions</name><value>Tr</value></parm>
<parm><name>aclentryactions</name><value>TcmdbsvaBRlr</value></parm>
<parm><name>aclid</name><value>intranet_restrict</value></parm>
<parm><name>acldesc</name><value></value></parm>
<parm><name>aclentrycount</name><value>6</value></parm>
<parm><name>aclentryid</name><value>sec_master</value></parm>
<parm><name>aclentryid</name><value>intranet-users</value></parm>
<parm><name>aclentryid</name><value></value></parm>
<parm><name>aclentryid</name><value></value></parm>
<parm><name>aclentryid</name><value>jsmith</value></parm>
<parm><name>aclentryid</name><value>iv-admin</value></parm>
<parm><name>aclid</name><value>intranet_restrict</value></parm>
</mgmtinfo>
...
```

This record again shows the ACL after the change with the new entry being the sixth item in the entry array (fields shown in bold).

Example 9-22 shows the audit record for the **ACL detach** command.

Example 9-22 pdmgrd.log audit records for acl detach

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">sec_master</principal>
...
<mgmtinfo><command>ACL DETACH</command>
<objname>/WebSEAL/linux11z.itso.ibm.com-default/intranet_help</objname>
<objtype>protectedResource</objtype>
```

```
<parm><name>objid</name><value>/WebSEAL/linux11z.itso.ibm.com-default/intranet_hel
p</value></parm>
</mgmtinfo>
...
```

Example 9-23 shows the audit record for the **ACL delete** command.

Example 9-23 pdmgrd.log audit records for acl delete

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">sec_master</principal>
...
<mgmtinfo><command>ACL DELETE</command>
<objname>intranet_restrict</objname>
<objtype>acl</objtype>
<parm><name>aclid</name><value>intranet_restrict</value></parm>
</mgmtinfo>
...
```

In summary, all four activities performed by the user sec_master have been logged in the audit log. In all cases you can see the activity being performed, the date and time, the outcome (success or failure), the principal, and the object being changed. For the modify command you only see the *after* state, not what was changed, so you cannot easily identify the discrete changes to the system.

9.3.5 Changing TAM auditing test case

For most of the TAM products, the audit levels are set within the configuration files, such as the ivmgrd.conf file. Some products also allow overriding the global audit settings by applying resource-specific audit settings that could be used by administrators to hide activity. This test case shows the creation of a POP to hide http access auditing.

Performing the test case

This test case uses the same sample application as in the previous test case. The intranet_help resource in the TAM objectspace has the ACL applied, as shown in Figure 9-17 on page 225.

For this test case we create a POP that disables http auditing and apply it to the resource, as shown in Example 9-24.

Example 9-24 sec_master test case 3 execution

```
linux4z:~/scripts # ./pdadmin_audit pdadmin_case3.pd
+ date
Tue Nov 13 11:59:30 EST 2007
+ pdadmin -a sec_master -p secret pdadmin_case3.pd
cmd> pop create nohttpaudit
cmd> pop modify nohttpaudit set attribute audithttp no
cmd> pop attach /WebSEAL/linux11z.itso.ibm.com-default/intranet_help nohttpaudit
+ date
Tue Nov 13 11:59:31 EST 2007
```

As there were no violations, no additional messages were written out.

Checking audit log capture

As in test case 1, the records are written to the TAM policy server audit log, which in our environment is pdmgrd.log. As before, there are authentication and authorization records written for the sec_master user running the **pdadmin** command.

Example 9-25 shows the audit record for the **POP create** command.

Example 9-25 pdmgrd.log audit records for pop create

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">sec_master</principal>
...
<mgmtinfo><command>POP CREATE</command>
<objname>nohttpaudit</objname>
<objtype>pop</objtype>
<parm><name>popid</name><value>nohttpaudit</value></parm>
</mgmtinfo>
...
```

Example 9-26 shows the audit record for the POP modify (that is, setting the nohttpauth attribute).

Example 9-26 pdmgrd.log audit records for pop modify

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">sec_master</principal>
...
<mgmtinfo><command>POP MOD SET ATTR</command>
<objname>nohttpaudit</objname>
<objtype>pop</objtype>
<parm><name>popid</name><value>nohttpaudit</value></parm>
<parm><name>attributevalue</name><value>no</value></parm>
<parm><name>attributename</name><value>auditthttp</value></parm>
</mgmtinfo>
...
```

Example 9-27 shows the audit record for the **POP attach** command.

Example 9-27 pdmgrd.log audit records for pop attach

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">sec_master</principal>
...
<mgmtinfo><command>POP ATTACH</command>
<objname>/WebSEAL/linux11z.itso.ibm.com-default/intranet_help</objname>
<objtype>protectedResource</objtype>
<target>nohttpaudit</target>
<parm><name>popid</name><value>nohttpaudit</value></parm>
```

```
<parm><name>objectid</name><value>/WebSEAL/linux11z.itso.ibm.com-default/intranet_
help</value></parm>
</mgmtinfo>
...
```

Whilst there is no impact to the managed resource by creating and modifying the POP, there is no way (other than the naming of the POP) to see what settings are being applied when the POP is attached to the resource. If auditing changes for resources, you should consider auditing all POP (and ACL) actions and tying them together.

9.3.6 Changing TAMOS auditing test case

TAMOS uses a combination of configuration files, administration commands, and TAM policy objects to define its auditing. Global policy can be set in the osseal.conf file under the [audit] stanza, and through the **pdosctl** control command. User-specific and resource-specific auditing can be set within the TAM objectspace.

Global auditing requires operating system and TAMOS access rights to the configuration files and the configuration utilities. Setting the audit policy in the TAM objectspace requires the appropriate access to TAM. It is possible to hide activity by setting a user-level or resource-level policy with auditing set to none. This test case shows how this type of activity can be audited.

Performing the test case

For this test case we have global auditing enabled in TAMOS and set to:

admin Audit all administrative actions.

trace_exec_root Trace all activity for the root user.

trace_exec_l Trace all activity when a user's effective UID is not the same as her login ID (for example, they have performed a su).

trace_exec_file Trace all activity against all file resources defined to TAMOS.

This is shown in Example 9-28.

Example 9-28 TAMOS global policy settings

```
linux11z:/opt/pdweb/www-default/docs/intranet_help # pdosctl -a
pdosd has the following audit levels on: admin, trace_exec_root, trace_exec_l,
trace_file
pdoswdd has the following audit levels on: admin, trace_exec_root, trace_exec_l,
trace_file
pdoslpmd has the following audit levels on: admin, trace_exec_root, trace_exec_l,
trace_file
pdoslrd is not running
pdosauditd has the following audit levels on: admin, trace_exec_root,
trace_exec_l, trace_file
```

To disable authorization and trace audit logging for a user we define an AuditAuth object and a AuditTrace object in the TAM objectspace. The commands we ran are shown in Example 9-29.

Example 9-29 sec_master test case 4 execution

```
linux4z:~/scripts # ./pdadmin_audit pdadmin_case4.pd
+ date
Tue Nov 13 14:02:34 EST 2007
+ pdadmin -a sec_master -p secret pdadmin_case4.pd
cmd> object create /OSSEAL/zlinux/AuditAuth/User/jsmith/none "Disable auth audit"
11 ispolicyattachable no
cmd> object create /OSSEAL/zlinux/AuditTrace/User/jsmith/none "Disable trace
audit" 11 ispolicyattachable no
+ date
Tue Nov 13 14:02:35 EST 2007
```

Note that the sec_master user was able to do this without changes to the default policy. He has not been made a member of the osseal-admin group, which is used to control access to the TAMOS administrative commands.

Checking audit log capture

As in test case 1, the records are written to the TAM policy server audit log, which in our environment is pdmgrd.log. As before, there are authentication and authorization records written for the sec_master user running the **pdadmin** command.

Example 9-30 shows the audit record for the first **Object Create** command.

Example 9-30 pdmgrd.log audit records for audit object create #1

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">sec_master</principal>
...
<mgmtinfo><command>OBJ CREATE</command>
<objname>/OSSEAL/zlinux/AuditAuth/User/jsmith/none</objname>
<objtype>protectedResource</objtype>
<parm><name>objid</name><value>/OSSEAL/zlinux/AuditAuth/User/jsmith/none</value></
parm>
<parm><name>objdesc</name><value>Disable auth audit</value></parm>
<parm><name>objtype</name><value>11</value></parm>
<parm><name>objpolicyattach</name><value>false</value></parm>
</mgmtinfo>
...
```

This audit record shows the exact audit setting that has been applied as it is the resource name. The audit record for the second command is basically the same, as shown in Example 9-31.

Example 9-31 pdmgrd.log audit records for audit object create #2

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">sec_master</principal>
```

```
...
<mgmtinfo><command>OBJ CREATE</command>
<objname>/OSSEAL/zlinux/AuditTrace/User/jsmith/none</objname>
<objtype>protectedResource</objtype>
<parm><name>objid</name><value>/OSSEAL/zlinux/AuditTrace/User/jsmith/none</value><
/parm>
<parm><name>objdesc</name><value>Disable trace audit</value></parm>
<parm><name>objtype</name><value>11</value></parm>
<parm><name>objpolicyattach</name><value>false</value></parm>
</mgmtinfo>
...
```

We recommend capturing these types of audit events, where the user-level audit overrides can be set.

9.3.7 Administration by another iv-admin user test case

This test case is a repeat of 9.3.3, “Creating another iv-admin user test case” on page 222, where the TAM administrator modified some ACL entries and then removed the ACL from a resource. In this case, the activity was performed by another user in the iv-admin group, not the default administrator sec_master.

Performing the test case

The steps run in this test case are shown in Example 9-32.

Example 9-32 TAM test case 6 execution

```
linux4z:~/scripts # ./pdadmin_audit_newadmin pdadmin_case5.pd
+ date
Thu Nov 15 09:09:18 EST 2007
+ pdadmin -a newadmin -p secret00 pdadmin_case5.pd
cmd> acl modify intranet_restrict set user jsmith Tr
cmd> acl modify intranet_restrict set group iv-admin TcmdbsvaBRlr
cmd> acl detach /WebSEAL/linux11z.itso.ibm.com-default/intranet_help
cmd> acl delete intranet_restrict
+ date
Thu Nov 15 09:09:21 EST 2007
```

As there were no error messages, the commands were authorized and run successfully.

Checking audit log capture

The audit records for this test are written to the same audit log, the TAM policy server audit log.

Example 9-33 shows the mgmt audit record written for the *first acl modify* request.

Example 9-33 pdmgrd audit record for first ACL modify

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">newadmin</principal>
...
<mgmtinfo><command>ACL SET</command>
<objname>intranet_restrict</objname>
```

```
<objtype>acl</objtype>
<parm><name>aclentrytype</name><value>3</value></parm>
<parm><name>aclentrytype</name><value>2</value></parm>
<parm><name>aclentrytype</name><value>1</value></parm>
<parm><name>aclentrytype</name><value>0</value></parm>
<parm><name>aclentrytype</name><value>2</value></parm>
<parm><name>aclentrytype</name><value>3</value></parm>
<parm><name>aclentryactions</name><value>TcmdbsvaBRrl</value></parm>
<parm><name>aclentryactions</name><value>Tr</value></parm>
<parm><name>aclentryactions</name><value>T</value></parm>
<parm><name>aclentryactions</name><value>T</value></parm>
<parm><name>aclentryactions</name><value>TcmdbsvaBRrl</value></parm>
<parm><name>aclentryactions</name><value>Tr</value></parm>
<parm><name>aclid</name><value>intranet_restrict</value></parm>
<parm><name>acldesc</name><value></value></parm>
<parm><name>aclentrycount</name><value>6</value></parm>
<parm><name>aclentryid</name><value>sec_master</value></parm>
<parm><name>aclentryid</name><value>intranet-users</value></parm>
<parm><name>aclentryid</name><value></value></parm>
<parm><name>aclentryid</name><value></value></parm>
<parm><name>aclentryid</name><value>iv-admin</value></parm>
<parm><name>aclentryid</name><value>jsmith</value></parm>
<parm><name>aclid</name><value>intranet_restrict</value></parm>
</mgmtinfo>
...
```

This is basically the same record as written in Example 9-20 on page 227, but with the principal being newadmin rather than sec_master.

Example 9-34 shows the audit record for the *acl detach* request.

Example 9-34 pdmgrd audit record for first ACL detach

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
<principal auth="IV_LDAP_V3.0" domain="Default">newadmin</principal>
...
<mgmtinfo><command>ACL DETACH</command>
<objname>/WebSEAL/linux11z.itso.ibm.com-default/intranet_help</objname>
<objtype>protectedResource</objtype>
<parm><name>objid</name><value>/WebSEAL/linux11z.itso.ibm.com-default/intranet_hel
p</value></parm>
</mgmtinfo>
...
```

Again, this is the same record as for the sec_master activity, just with a different principal.

Example 9-35 shows the last audit record for this test case where the *acl delete* request is run as newadmin.

Example 9-35 pdmgrd audit record for first ACL delete

```
...
<originator blade="pdmgrd"><component rev="1.1">mgmt</component>
...
```

```
<principal auth="IV_LDAP_V3.0" domain="Default">newadmin</principal>
...
<mgmtinfo><command>ACL DELETE</command>
<objname>intranet_restrict</objname>
<objtype>acl</objtype>
<parm><name>aclid</name><value>intranet_restrict</value></parm>
</mgmtinfo>
...
```

This completes the execution of the test cases for the TAM administration. The next section shows how these records can be reported in TCIM.

9.3.8 Audit reporting for TAM test cases

TAM does not provide its own reporting mechanism. You could develop something to process the XML-formatted audit records, but to report on them as they are you need another tool such as the Tivoli Common Audit and Reporting Service (CARS) or TCIM. TCIM is the preferred direction for audit reporting with the Tivoli security products. We did not test forwarding these TAM events to TCIM, but we did test TAM WebSEAL messages in TCIM in the e-business scenarios in Chapter 8, “e-business scenarios” on page 159.

9.4 Linux privileged (uid=0)

This section is aimed at providing some test cases (examples) of how privileged user actions can be audited using the Linux Audit subsystem. As discussed in 5.1, “Linux auditing subsystem” on page 88, the Linux Audit subsystem relies on trusted programs to perform auditing on actions happening in user-space (like authentication).

A privileged user in this section’s context is defined by a user that can change the system state to behave in a different way, that is, by changing the system configuration, adding new users, or changing user configuration data. In UNIX terms, this is usually done through the root user (also known as superuser)—an administrative-only account that is associated with the first user ID (*uid=0*). In practice, every user with uid=0 can be considered a superuser, no matter what username it uses.

9.4.1 Environment setup

The baseline environment for all test cases listed here is based on Red Hat Enterprise Linux Version 5 for System z (s390x) using the *targeted* SELinux policy (although architecture probably will not matter in this case, and specific distribution and SELinux policy differences can be adjusted).

The system has the Audit Daemon (auditd) packages installed with standard kernel and user-space configuration for a static-IP networked node (unless otherwise noted).

Specific environment settings are covered in each test case for clarity.

9.4.2 root login auditing test case

In a standard UNIX environment, it is a well-known security practice to prevent direct superuser logins from remote locations. This is also especially useful for auditing purposes if

the root account needs to be shared among a group of system administrators (in case more than a single person needs access to administrative commands). This way the system enforces administrative users to perform login using their own (individual) accounts and then become root as needed (using utilities such as su or sudo), leaving an auditable trail of who performed operations under the superuser credentials.

The standard way of determining what users can log into the system and from where is made through the pam_access.so PAM configuration (Linux Pluggable Authentication Modules; see pam_access(8) manual page). The line in Example 9-36 shows a rule added to file /etc/security/access.conf determining a policy that the root user can only log in in the specified terminals (cron and crond allow the cron service to run).

Example 9-36 example line in /etc/security/access.conf file

```
- : root : ALL EXCEPT cron crond :0 console tty1 tty2 tty3 tty4 tty5 tty6
```

Make sure that the pam_access.so module is set as required for account service in the /etc/pam.d/sshd file, as shown in Example 9-37.

Example 9-37 /etc/pam.d/sshd

```
#%PAM-1.0
auth       include      system-auth
account    required     pam_nologin.so
account    required     pam_access.so
account    include      system-auth
password   include      system-auth
session    optional     pam_keyinit.so force revoke
session    include      system-auth
session    required     pam_loginuid.so
```

By doing so, the system will not allow root logins through ssh (or any other PAM-capable authentication service linking to pam_access.so) coming from the network. Even the localhost is restricted, so users cannot hide their identity by opening another login session while already inside the system.

Examining Audit trail

Every time an authentication is performed an USER_AUTH record is cut. If a user is trying to authenticate, even if it is not permitted to log in in that terminal, the USER_AUTH record will audit whether the authentication was successful or not. Example 9-38 shows how users aureport and ausearch inspect authentication failures.

Example 9-38 Authentication report

```
[root@linux2z ~]# aureport --auth --interpret

Authentication Report
============================================
# date time acct host term exe success event
============================================
1. 09/07/2007 16:41:15 acct=k ? pts/3 /bin/su yes 2242
2. 11/14/2007 17:53:11 acct=r ? console /bin/login yes 563
3. 11/14/2007 18:16:48 acct=r 9.57.139.29 ssh /usr/sbin/sshd yes 618
4. 11/14/2007 18:16:48 acct=r 9.57.139.29 ssh /usr/sbin/sshd no 619
5. 11/14/2007 18:19:08 acct=r 9.57.139.29 ssh /usr/sbin/sshd no 623
6. 11/14/2007 18:19:14 acct=r 9.57.139.29 ssh /usr/sbin/sshd no 625
```

```
7. 11/14/2007 18:19:18 acct=r 9.57.139.29 ssh /usr/sbin/sshd no 627
8. 11/14/2007 18:33:05 acct=r 9.57.139.29 ssh /usr/sbin/sshd no 635
[root@linux2z ~]# ausearch --interpret -a 635
----
type=USER_AUTH msg=audit(11/14/2007 18:33:05.486:635) : user pid=10064 uid=root
auid=klausk subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='PAM:
authentication acct=root : exe=/usr/sbin/sshd (hostname=9.57.139.29,
addr=9.57.139.29, terminal=ssh res=failed)'
[root@linux2z ~]#
```

The report shown by aureport in Example 9-39 shows both successful and failed authentication attempts. The **ausearch** command is also used to inspect events with serial 635 showing that the failed authentication occurred through ssh, coming from network address 9.57.135.29, against the root account and it failed.

Example 9-39 shows an **ausearch** command listing all failed authentications for the root account for the day.

Example 9-39 ausearch command listing

```
[root@linux2z ~]# ausearch --interpret -m USER_AUTH --uid root --success no
--start today
----
type=USER_AUTH msg=audit(11/14/2007 08:46:26.116:427) : user pid=8474 uid=root
auid=unset subj=system_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='PAM:
authentication acct=klausk : exe=/usr/sbin/sshd (hostname=9.57.139.29,
addr=9.57.139.29, terminal=ssh res=failed)'
----
type=USER_AUTH msg=audit(11/14/2007 18:19:08.376:623) : user pid=10011 uid=root
auid=klausk subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='PAM:
authentication acct=root : exe=/usr/sbin/sshd (hostname=9.57.139.29,
addr=9.57.139.29, terminal=ssh res=failed)'
----
type=USER_AUTH msg=audit(11/14/2007 18:19:14.766:625) : user pid=10011 uid=root
auid=klausk subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='PAM:
authentication acct=root : exe=/usr/sbin/sshd (hostname=9.57.139.29,
addr=9.57.139.29, terminal=ssh res=failed)'
----
type=USER_AUTH msg=audit(11/14/2007 18:19:18.636:627) : user pid=10011 uid=root
auid=klausk subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='PAM:
authentication acct=root : exe=/usr/sbin/sshd (hostname=9.57.139.29,
addr=9.57.139.29, terminal=ssh res=failed)'
----
type=USER_AUTH msg=audit(11/14/2007 18:33:05.486:635) : user pid=10064 uid=root
auid=klausk subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='PAM:
authentication acct=root : exe=/usr/sbin/sshd (hostname=9.57.139.29,
addr=9.57.139.29, terminal=ssh res=failed)'
```

Besides the authentication events, the Linux Audit subsystem also has a USER_LOGIN record cut every time a user attempts to perform login itself. This may or cannot involve authentication, depending on the system configuration. Example 9-40 shows an aureport for failed logins for the day.

Example 9-40 aureport sample

```
[root@linux2z ~]# aureport --login --start today --failed

Login Report
============================================
# date time auid host term exe success event
============================================
1. 11/14/2007 18:33:02 acct=r 9.57.139.29 sshd /usr/sbin/sshd no 634
2. 11/14/2007 18:33:05 acct=r 9.57.139.29 sshd /usr/sbin/sshd no 636
3. 11/14/2007 18:46:48 acct=r 9.57.139.29 sshd /usr/sbin/sshd no 637
4. 11/14/2007 18:46:48 acct=r 9.57.139.29 sshd /usr/sbin/sshd no 638
5. 11/14/2007 18:46:51 acct=r 9.57.139.29 sshd /usr/sbin/sshd no 641
[root@linux2z ~]#
```

In Example 9-41, further investigation around the time of event 641 revealed that the a user coming from ssh connection, attempting to login as root, has successfully authenticated, but failed to log in due to the pam_access.so policy in place.

Example 9-41 log example

```
[root@linux2z ~]# ausearch --interpret --start 11/14/2007 18:46:50 --end
11/14/2007 18:46:55
----
type=USER_AUTH msg=audit(11/14/2007 18:46:51.326:639) : user pid=10099 uid=root
auid=klausk subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='PAM:
authentication acct=root : exe=/usr/sbin/sshd (hostname=9.57.139.29,
addr=9.57.139.29, terminal=ssh res=success)'
----
type=USER_ACCT msg=audit(11/14/2007 18:46:51.326:640) : user pid=10099 uid=root
auid=klausk subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='PAM: accounting
acct=root : exe=/usr/sbin/sshd (hostname=9.57.139.29, addr=9.57.139.29,
terminal=ssh res=failed)'
----
type=USER_LOGIN msg=audit(11/14/2007 18:46:51.326:641) : user pid=10099 uid=root
auid=klausk subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='acct=root:
exe=/usr/sbin/sshd (hostname=?, addr=9.57.139.29, terminal=sshd res=failed)'
[root@linux2z ~]#
```

Example 9-42 shows how to list every successful root login that happened today.

Example 9-42 Successful root login

```
[root@linux2z ~]# ausearch --interpret -m USER_LOGIN --uid root --success yes
--start today
----
type=USER_LOGIN msg=audit(11/14/2007 08:46:35.826:436) : user pid=8478 uid=root
auid=root subj=system_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=9.57.139.29, addr=9.57.139.29, terminal=/dev/pts/3
res=success)'
----
```

```
type=USER_LOGIN msg=audit(11/14/2007 09:03:03.626:463) : user pid=8562 uid=root
auid=root subj=system_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=9.57.139.29, addr=9.57.139.29, terminal=/dev/pts/3
res=success)'
----
type=USER_LOGIN msg=audit(11/14/2007 17:36:23.516:518) : user pid=9516 uid=root
auid=root subj=system_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=9.12.5.171, addr=9.12.5.171, terminal=/dev/pts/4
res=success)'
----
type=USER_LOGIN msg=audit(11/14/2007 17:36:51.786:527) : user pid=9564 uid=root
auid=root subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=9.12.5.171, addr=9.12.5.171, terminal=/dev/pts/4
res=success)'
----
type=USER_LOGIN msg=audit(11/14/2007 17:37:58.506:536) : user pid=9599 uid=root
auid=root subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=9.12.5.171, addr=9.12.5.171, terminal=/dev/pts/4
res=success)'
----
type=USER_LOGIN msg=audit(11/14/2007 17:49:48.896:557) : user pid=9757 uid=root
auid=root subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=localhost.localdomain, addr=127.0.0.1,
terminal=/dev/pts/4 res=success)'
----
type=USER_LOGIN msg=audit(11/14/2007 17:53:12.016:569) : user pid=1451 uid=root
auid=root subj=system_u:system_r:local_login_t:s0-s0:c0.c1023 msg='uid=root
exe=/bin/login (hostname=?, addr=?, terminal=console res=success)'
----
type=USER_LOGIN msg=audit(11/14/2007 17:55:35.226:578) : user pid=9838 uid=root
auid=root subj=system_u:system_r:local_login_t:s0-s0:c0.c1023 msg='uid=root
exe=/bin/login (hostname=?, addr=?, terminal=console res=success)'
----
type=USER_LOGIN msg=audit(11/14/2007 17:57:38.486:591) : user pid=9880 uid=root
auid=root subj=system_u:system_r:local_login_t:s0-s0:c0.c1023 msg='uid=root
exe=/bin/login (hostname=?, addr=?, terminal=console res=success)'
[root@linux2z ~]#
```

Example 9-43 shows how to filter and flag remote root logins. These events happened before the policy enforcement defined in the parm_access.so configuration.

Example 9-43 Filtered root logins

```
[root@linux2z ~]# ausearch --interpret -m USER_LOGIN --uid root --success yes
--start today --executable /usr/sbin/sshd
----
type=USER_LOGIN msg=audit(11/14/2007 08:46:35.826:436) : user pid=8478 uid=root
auid=root subj=system_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=9.57.139.29, addr=9.57.139.29, terminal=/dev/pts/3
res=success)'
----
type=USER_LOGIN msg=audit(11/14/2007 09:03:03.626:463) : user pid=8562 uid=root
auid=root subj=system_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=9.57.139.29, addr=9.57.139.29, terminal=/dev/pts/3
res=success)'
----
```

```
type=USER_LOGIN msg=audit(11/14/2007 17:36:23.516:518) : user pid=9516 uid=root
auid=root subj=system_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=9.12.5.171, addr=9.12.5.171, terminal=/dev/pts/4
res=success)'
----
type=USER_LOGIN msg=audit(11/14/2007 17:36:51.786:527) : user pid=9564 uid=root
auid=root subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=9.12.5.171, addr=9.12.5.171, terminal=/dev/pts/4
res=success)'
----
type=USER_LOGIN msg=audit(11/14/2007 17:37:58.506:536) : user pid=9599 uid=root
auid=root subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=9.12.5.171, addr=9.12.5.171, terminal=/dev/pts/4
res=success)'
----
type=USER_LOGIN msg=audit(11/14/2007 17:49:48.896:557) : user pid=9757 uid=root
auid=root subj=user_u:system_r:unconfined_t:s0-s0:c0.c1023 msg='uid=root
exe=/usr/sbin/sshd (hostname=localhost.localdomain, addr=127.0.0.1,
terminal=/dev/pts/4 res=success)'
[root@linux2z ~]#
```

9.4.3 Tracking changes to the user database test case

Keeping track of changes in the user database is an important task when auditing a system. In Linux, user and group information can be acquired locally (local user database) or remotely (LDAP, NIS), and changes to this information can occur in a variety of methods.

This subsection presents techniques for auditing changes to a locally defined user database, but it can be adapted for remote databases since the concepts are the same. In order to track changes efficiently, two types of access must be considered:

- Direct (raw) access to databases - In the case of a locally defined database, this involves auditing low-level accesses to the files where user information can be found. The Linux Audit subsystem provides a very straightforward way to do this: file system *watches*. The following files should be watched for changes in content or attribute:
  - /etc/group - user group file
  - /etc/passwd - password file
  - /etc/gshadow - shadowed group file
  - /etc/shadow - encrypted password file
  - /etc/security/opasswd - old passwords file
- Database changes through specialized applications - User management systems often provide specialized applications to deal with user, group, and authentication management. In the specified Linux environment, these tools are bundled in the shadow-utils package and are considered trusted programs, meaning that they will cut audit records when performing changes to the databases by default, without need of additional audit rules. These applications are:
  - /usr/bin/chage - Change user password expiry information.
  - /usr/bin/gpasswd - Administer the /etc/group file.
  - /usr/bin/newgrp - Log in to a new group.
  - /usr/bin/sg - Execute a command as a different group ID.
  - /usr/sbin/adduser - Create a new user or update default new user information.
  - /usr/sbin/groupadd - Create a new group.
  - /usr/sbin/groupdel - Delete a group.

- /usr/sbin/groupmod - Modify a group.
- /usr/sbin/useradd - Create a new user or update default new user information.
- /usr/sbin/userdel - Delete a user account and related files.
- /usr/sbin/usermod - Modify a user account.

Auditing both low-level changes and also specialized programs can help the auditor to distinguish whether the change was performed in a legitimate way (using trusted programs) or not (only raw accesses). Besides this, the auditor can usually extract more detailed information about the change made using the specialized records from trusted programs.

Example 9-44 shows the set of rules used to audit changes to the local user information database. (Remember that trusted programs will audit every change by default.) The rules shown must be entered in the /etc/audit/audit.rules file so that they can be loaded at system startup.

Example 9-44 Rules for auditing changes to the local user information database

```
-w /etc/group -p wa -k user_CFG
-w /etc/passwd -p wa -k user_CFG
-w /etc/gshadow -p wa -k user_CFG
-w /etc/shadow -p wa -k user_CFG
-w /etc/security/opasswd -p wa -k user_CFG
```

Also note that the user_CFG key is used to identify this set of rules, making it easier to search for events generated by this rules later.

Examining audit trail

Example 9-45 shows how user aureport lists account modifications (when performed by trusted programs), followed by inspections to the event details by using ausearch.

Example 9-45 aureport sample

```
[root@linux2z audit]# aureport --interpret --mods

Account Modifications Report
===========================================
# date time auid addr term exe success event
===========================================
1. 11/15/2007 12:09:00 klausk ? pts/1 /usr/sbin/useradd yes 121
2. 11/15/2007 12:09:00 klausk ? pts/1 /usr/sbin/useradd yes 130
3. 11/15/2007 12:09:10 klausk ? pts/1 /usr/bin/passwd yes 133
4. 11/15/2007 12:09:10 klausk ? pts/1 /usr/bin/passwd yes 134
[root@linux2z audit]# ausearch --interpret -a 121
----
type=USER_CHAUTHTOK msg=audit(11/15/2007 12:09:00.420:121) : user pid=2937
uid=root auid=klausk subj=user_u:system_r:unconfined_t:s0 msg='op=adding user
acct=testuser exe=/usr/sbin/useradd (hostname=?, addr=?, terminal=pts/1
res=success)'
[root@linux2z audit]# ausearch --interpret -a 130
----
type=USER_CHAUTHTOK msg=audit(11/15/2007 12:09:00.640:130) : user pid=2937
uid=root auid=klausk subj=user_u:system_r:unconfined_t:s0 msg='op=adding home
directory acct=testuser exe=/usr/sbin/useradd (hostname=?, addr=?, terminal=pts/1
res=success)'
[root@linux2z audit]# ausearch --interpret -a 133
----
```

```
type=USER_CHAUTHTOK msg=audit(11/15/2007 12:09:10.000:133) : user pid=2942
uid=root auid=klausk subj=user_u:system_r:unconfined_t:s0 msg='PAM: chauthtok
acct=testuser : exe=/usr/bin/passwd (hostname=?, addr=?, terminal=pts/1
res=success)'
[root@linux2z audit]# ausearch --interpret -a 134
----
type=USER_CHAUTHTOK msg=audit(11/15/2007 12:09:10.000:134) : user pid=2942
uid=root auid=klausk subj=user_u:system_r:unconfined_t:s0 msg='op=change password
id=testuser exe=/usr/bin/passwd (hostname=?, addr=?, terminal=pts/1 res=success)'
[root@linux2z audit]#
```

An event with serial 121 shows that the user klausk (auid=klausk), using root credentials (uid=root), used the /usr/bin/useradd trusted program to add the user account testuser, and that the operation performed successfully. The similar event 130 follows with the creation of the user's home directory. Events 133 and 134 show that the password for user testuser was then changed (as reported by useradd itself as well as by PAM).

Example 9-46 shows the same operations as they where audited by low-level access. From the total of 9 events, some were suppressed for better clarity. Tracking low-level operations can generate a fairly good amount of data. System administrators must balance the necessity of tracking such operations with the resource usage.

Example 9-46 ausearch sample

```
[root@linux2z audit]# ausearch --interpret -k user_CFG
----
type=PATH msg=audit(11/15/2007 12:09:00.420:117) : item=0 name=/etc/passwd
inode=358397 dev=5e:05 mode=file,644 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0
type=CWD msg=audit(11/15/2007 12:09:00.420:117) :  cwd=/root
type=SYSCALL msg=audit(11/15/2007 12:09:00.420:117) : arch=s390x syscall=open
success=yes exit=5 a0=800130f8 a1=2 a2=1b6 a3=1b6 items=1 ppid=2803 pid=2937
auid=klausk uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root
fsgid=root tty=pts1 comm=useradd exe=/usr/sbin/useradd
subj=user_u:system_r:unconfined_t:s0 key="user_CFG"
----
type=PATH msg=audit(11/15/2007 12:09:00.420:118) : item=0 name=/etc/shadow
inode=358398 dev=5e:05 mode=file,400 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0
type=CWD msg=audit(11/15/2007 12:09:00.420:118) :  cwd=/root
type=SYSCALL msg=audit(11/15/2007 12:09:00.420:118) : arch=s390x syscall=open
success=yes exit=6 a0=800139d8 a1=2 a2=1b6 a3=1b6 items=1 ppid=2803 pid=2937
auid=klausk uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root
fsgid=root tty=pts1 comm=useradd exe=/usr/sbin/useradd
subj=user_u:system_r:unconfined_t:s0 key="user_CFG"
----
...
----
type=PATH msg=audit(11/15/2007 12:09:10.000:132) : item=4 name=/etc/shadow
inode=357455 dev=5e:05 mode=file,400 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0
type=PATH msg=audit(11/15/2007 12:09:10.000:132) : item=3 name=/etc/shadow
inode=358397 dev=5e:05 mode=file,400 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0
```

```
type=PATH msg=audit(11/15/2007 12:09:10.000:132) : item=2 name=/etc/nshadow
inode=357455 dev=5e:05 mode=file,400 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0
type=PATH msg=audit(11/15/2007 12:09:10.000:132) : item=1 name=/etc/ inode=357281
dev=5e:05 mode=dir,755 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0
type=PATH msg=audit(11/15/2007 12:09:10.000:132) : item=0 name=/etc/ inode=357281
dev=5e:05 mode=dir,755 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0
type=CWD msg=audit(11/15/2007 12:09:10.000:132) :  cwd=/root
type=SYSCALL msg=audit(11/15/2007 12:09:10.000:132) : arch=s390x syscall=rename
success=yes exit=0 a0=2000003753c a1=20000037530 a2=20000031fe4 a3=0 items=5
ppid=2803 pid=2942 auid=klausk uid=root gid=root euid=root suid=root fsuid=root
egid=root sgid=root fsgid=root tty=pts1 comm=passwd exe=/usr/bin/passwd
subj=user_u:system_r:unconfined_t:s0 key="user_CFG"
[root@linux2z audit]#
```

In the previous example, the first two events show how /usr/sbin/useradd performed the open syscall in the files /etc/passwd and /etc/shadow. The last event shows the low-level operation behind the passwd trusted program. The application actually writes changes to a copy of the /etc/shadow file, named /etc/nshadow, replacing the original file afterwards (using the rename syscall).

9.4.4 Tracking root commands test case

There may be some situations where is desirable to track every superuser command. In this scenario, one possible solution might include restricting interactive use of the root account completely, so that administrative users must use the **sudo** command to execute commands with the superuser credentials. The **sudo** command executions can then be tracked by a watch audit rule. Executing commands using **sudo** will cut an EXECVE record that contains, along with other information, the actual command line executed with superuser credentials.

To achieve this, the system administrator must make sure that the root account is locked and a proper sudo configuration is in place. Example 9-47 shows a sample configuration for sudo that disables interactive sessions (shells) execution, while allowing all other commands for the users in the wheel group.

Example 9-47 Sample configuration

```
Cmnd_Alias     SHELLS = /bin/bash, /bin/sh, /bin/tcshc, /bin/csh, \
                        /bin/ksh, /bin/rsh
Cmnd_Alias     SU = /usr/bin/su

%wheel  ALL=(ALL)       NOEXEC: ALL, !SHELLS, !SU
```

The NOEXEC prevents that executed commands to execute other commands, like spawning a new interactive session from inside an editor, for example.

The rule in Example 9-48 shows how to enable auditing in every sudo execution.

Example 9-48 Enable auditing

```
-w /usr/bin/sudo -p x -k sudo_EXEC
```

After the system administrator is sure that he can execute commands with the root credentials, he can lock himself out of interactive root sessions by issuing **usermod -L root**.

Note that this configuration will not prevent the privileged user from executing arbitrary commands. Instead, it is aimed at tracking the sudo usage, which would allow auditors to identify policy violations. The privileged user (in group wheel) can always edit the sudo configuration and allow himself to execute interactive sessions, but if by doing so it is considered a policy violation, the auditor can successfully track it.

Examining audit trail

Example 9-49 shows an **ausearch** command, matching events with the sudo_EXEC key.

Example 9-49 ausearcg command sample

```
[root@linux2z audit]# ausearch --interpret -k sudo_EXEC
----
type=PATH msg=audit(11/15/2007 17:38:10.530:393) : item=1 name=(null) inode=649604
dev=5e:05 mode=file,755 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:ld_so_t:s0
type=PATH msg=audit(11/15/2007 17:38:10.530:393) : item=0 name=/usr/bin/sudo
inode=1082199 dev=5e:05 mode=file,suid,111 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:bin_t:s0
type=CWD msg=audit(11/15/2007 17:38:10.530:393) :  cwd=/home/klausk
type=EXECVE msg=audit(11/15/2007 17:38:10.530:393) : a0="sudo" a1="cp"
a2="/etc/shadow" a3="/home/klausk/passwords"
type=SYSCALL msg=audit(11/15/2007 17:38:10.530:393) : arch=s390x syscall=execve
success=yes exit=11 a0=800f7aa0 a1=800eb010 a2=800f7ef0 a3=4e747112c8 items=2
ppid=4562 pid=4601 auid=klausk uid=klausk gid=klausk euid=root suid=root
fsuid=root egid=klausk sgid=klausk fsgid=klausk tty=pts1 comm=sudo
exe=/usr/bin/sudo subj=user_u:system_r:unconfined_t:s0 key="sudo_EXEC"
----
type=PATH msg=audit(11/15/2007 17:38:34.400:397) : item=1 name=(null) inode=649604
dev=5e:05 mode=file,755 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:ld_so_t:s0
type=PATH msg=audit(11/15/2007 17:38:34.400:397) : item=0 name=/usr/bin/sudo
inode=1082199 dev=5e:05 mode=file,suid,111 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:bin_t:s0
type=CWD msg=audit(11/15/2007 17:38:34.400:397) :  cwd=/home/klausk
type=EXECVE msg=audit(11/15/2007 17:38:34.400:397) : a0="sudo" a1="/sbin/ifconfig"
a2="eth0" a3="promisc"
type=SYSCALL msg=audit(11/15/2007 17:38:34.400:397) : arch=s390x syscall=execve
success=yes exit=11 a0=800f7aa0 a1=800f79c0 a2=800f7ef0 a3=4e747112c8 items=2
ppid=4562 pid=4603 auid=klausk uid=klausk gid=klausk euid=root suid=root
fsuid=root egid=klausk sgid=klausk fsgid=klausk tty=pts1 comm=sudo
exe=/usr/bin/sudo subj=user_u:system_r:unconfined_t:s0 key="sudo_EXEC"
[root@linux2z audit]#
```

Note that there are two sudo executions by the same user (auid=klausk). As seen in the execve syscall arguments (a0, a1, a2, a3) in the EXECVE record, the user executed a copy from file /etc/shadow to /home/klausk/passwords. In the second command, the user enabled promiscuous mode for network device eth0 (ifconfig eth0 promisc).

9.5 UNIX/Linux privileged user with TAMOS (uid=0)

One of the strengths of the TAMOS product is its auditing ability. Not only can it audit the authorization decisions that it makes, it can also audit all activity for different types of users, such as the root (UID=0) user or users who have performed some form of change user (such as su).

This section contains a number of test cases run on a system with TAMOS auditing enabled to show how well TAMOS can audit privileged user use on a UNIX/Linux system.

9.5.1 Environment setup

These test cases were run on a Linux for System z system running TAMOS 6.0. The audit settings for the environment (prior to any test cases) are as detailed in 9.3.6, “Changing TAMOS auditing test case” on page 231.

On the Linux for System z system there are two specific users that we are interested in: root (UID=0) and a new user itsoaud. The root user is a member of the osseal-admin group, which means that it is entitled to perform TAMOS administrative functions but not audit functions. The itsoaud user is a member of the osseal-auditors group, which means that it is entitled to access the TAMOS audit data and commands but not administrative functions. Example 9-50 shows the group membership as defined in TAM.

Example 9-50 TAMOS administrative and auditing groups in pdadmin

```
pdadmin sec_master> group list oss* 0
osseal-admin
osseal-auditors
ossaudit
osseal
pdadmin sec_master> group show-members osseal-admin
root
osseal
insight
pdadmin sec_master> group show-members osseal-auditors
osseal
insight
itsoaud
pdadmin sec_master>
```

The following sections use these users and the audit settings.

9.5.2 Create another privileged user test case

The power of the root user on UNIX/Linux is of concern to auditors. The root user can perform any function on the system irrespective of the file permissions set and can also wipe any operating system audit trail.

While there is focus on root, there also needs to be focus on other users with UID=0. This test case looks at the root user creating another user and setting it to UID=0.

Performing the test case

We used a **useradd** command to add the new UID=0 user, as shown in Example 9-51.

Example 9-51 root test case 1 execution

```
linux11z:~/scripts # ./tamos_case1.sh
+ date
Tue Nov 13 15:30:57 EST 2007
+ useradd -g root -p secret00 -u 0 -o -r -s /bin/bash testroot
+ date
Tue Nov 13 15:30:57 EST 2007
linux11z:~/scripts # tail -3 /etc/passwd
insight1:x:1101:100::/home/insight1:/bin/sh
itsoaud:x:1105:100::/home/itsoaud:/bin/bash
testroot:x:0:0::/home/testroot:/bin/bash
linux11z:~/scripts #
```

This user is now defined on the system and can act as another root user.

Checking audit log capture

The audit records for this are captured by the TAMOS auditing mechanism and written into the binary audit.log file. The **pdosaudview** command is used to extract and format the records.

As root is not a member of the osseal-auditors group it cannot run the **pdosaudview** command to access the audit records, as shown in Example 9-52.

Example 9-52 root attempts to view the TAMOS audit records

```
linux11z:~ # whoami
root
linux11z:~ # pdoswhoami
root
linux11z:~ # pdosaudview
-bash: /usr/bin/pdosaudview: Permission denied
```

We log in as the itsoaud user and use it to extract the audit records into a readable format, as shown in Example 9-53.

Example 9-53 itsoaud user extracts the TAMOS audit records

```
itsoaud@linux11z:/> whoami
itsoaud
itsoaud@linux11z:/> pdoswhoami
itsoaud
itsoaud@linux11z:/> pdosaudview -s now-20
Processed output is in /var/pdos/audit/text.log.
/var/pdos/audit/audit.log.2007-11-13-12-27-20 opened for processing.
/var/pdos/audit/audit.log opened for processing.
```

The **pdosaudview** command has been run without a formatting option so it will write into a compressed text format. We specified to extract the records starting from twenty minutes ago (-s now-20). The readable audit records were written to /var/pdos/audit/text.log.

Example 9-54 shows the audit records produced by executing our script (tamos_case1.sh) and the **date** command in the script.

Example 9-54 TAMOS audit records for user add #1

```
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/root/scripts/tamos_cas
e1.sh (./tamos_case1.sh),APID=16337,RPSN=/bin/bash,UQ=0
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/sh
(./tamos_case1.sh),APID=16337,RPSN=/root/scripts/tamos_case1.sh,UQ=1
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/root/scripts/tamos_cas
e1.sh (/bin/sh -x
./tamos_case1.sh),APID=16337,RPSN=/root/scripts/tamos_case1.sh,UQ=2
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/sh (/bin/sh -x
./tamos_case1.sh),APID=16337,RPSN=/root/scripts/tamos_case1.sh,UQ=3
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/date
(date),APID=16338,RPSN=/root/scripts/tamos_case1.sh,UQ=4
```

Example 9-55 shows the second set of audit records related to the test script.

Example 9-55 TAMOS audit records for user add #2

```
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/usr/sbin/useradd
(useradd -g root -p secret00 -u 0 -o -r -s /bin/bash
testroot),APID=16339,RPSN=/root/scripts/tamos_case1.sh,UQ=5
TS=Tue 13 Nov 2007 03:30:57 PM EST,E=7,V=P,R=5,RT=File,AN=root,AEN=root,A=Check
Access,P=wr,Q=34,PBN=zlinux,PON=File/dev/kazndrv,SRN=/dev/kazndrv,APID=16339,RPSN=
/usr/sbin/useradd,O=S,UQ=6
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=28,V=P,R=1,RT=TraceFile,AN=root,AEN=root,A=Trace,P=wr,PRS=/dev/kazndrv,ARS=/
dev/kazndrv,APID=16339,RPSN=/usr/sbin/useradd,UQ=7
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/usr/sbin/useradd.local
(/usr/sbin/useradd.local testroot 0 0
/home/testroot),APID=16340,RPSN=/usr/sbin/useradd,UQ=8
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/bash
(/usr/sbin/useradd.local testroot 0 0
/home/testroot),APID=16340,RPSN=/usr/sbin/useradd.local,UQ=9
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/usr/sbin/useradd.local
(/bin/bash /usr/sbin/useradd.local testroot 0 0
/home/testroot),APID=16340,RPSN=/usr/sbin/useradd.local,UQ=10
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/bash (/bin/bash
/usr/sbin/useradd.local testroot 0 0
/home/testroot),APID=16340,RPSN=/usr/sbin/useradd.local,UQ=11
TS=Tue 13 Nov 2007 03:30:57 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/date
(date),APID=16341,RPSN=/root/scripts/tamos_case1.sh,UQ=12
```

The first record (highlighted) shows the **useradd** command that we used, including all of the command-line options. The following record shows the TAMOS authorization check to execute the **useradd** command.

The focus on these test cases has been to ensure that activity is audited and can be reported on. As an aside, its worthwhile looking at one of these records displayed in a more readable format. The record in Example 9-56 was produced using the -F verbose option with pdosaudview. This is the record highlighted above.

Example 9-56 TAMOS audit record in verbose mode

```
*** START OF NEW RECORD ***

Timestamp                                                Tue 13 Nov 2007
03:30:57 PM EST
Audit Event                                              TRACE Exec program
Audit View                                               Trace
Audit Reason                                             Global Audit
Audit Resource Type                                      TraceExec
Accessor Name                                            root
Accessor Effective Name                                  root
Audit Action                                             Trace
Accessed Resource Specification                          /usr/sbin/useradd
(useradd -g root -p secret00 -u 0 -o -r -s /bin/bash testroot)
Accessor Process ID                                      16339
Running Program System Resource Name
/root/scripts/tamos_case1.sh
Audit Uniqifier                                          5
```

In this case there is extensive auditing to show that root has performed a useradd. You would need to drill through the audit records to identify that a UID=0 user has been created instead of a normal user.

9.5.3 Log in as a new user and perform suspect activity test case

This test case continues on from the previous one where a new UID=0 user has been created. In this test case the new user will log in and modify the /etc/inittab as though he is introducing some malicious code to run at system startup.

Performing the test case

The first step is to log in as the new user. This is shown in Example 9-57.

Example 9-57 root test case 2 execution part 1

```
login as: testroot
Password:
Last login: Tue Nov 13 16:09:59 2007 from 9.12.5.148
Could not chdir to home directory /home/testroot: No such file or directory
/usr/X11R6/bin/xauth:  error in locking authority file /home/testroot/.Xauthority
linux11z:/ # date
Tue Nov 13 16:31:35 EST 2007
linux11z:/ # whoami
testroot
```

```
linux11z:/ # pdoswhoami
testroot
linux11z:/ #
```

Both Linux and TAMOS consider this user to be testroot.

Example 9-58 shows the user modifying the inittab file.

Example 9-58 root test case 2 execution part 2

```
linux11z:/etc # vi inittab
... edit file and save ...
linux11z:/etc # ls -ltr
...
-rw-r--r--   1 testroot root      1298 2007-11-13 15:30 passwd
-rw-r-----   1 testroot shadow     695 2007-11-13 16:06 shadow
-rw-r--r--   1 testroot root      1971 2007-11-13 16:32 inittab
drwxr-xr-x  59 testroot root      4096 2007-11-13 16:32 .
linux11z:/etc # date
Tue Nov 13 16:33:31 EST 2007
```

Notice that now every file in the directory has testroot as the owner after modifying the inittab file.

Checking audit log capture

Using the same approach as for the previous section, we can view the audit records associated with this test case. Example 9-59 shows the audit records for the SSH login, starting with the network connection (the NetIncoming record).

Example 9-59 testroot user logging onto system using SSH session

```
TS=Tue 13 Nov 2007 04:31:25 PM
EST,E=7,V=P,R=5,RT=NetIncoming,AN=root,AEN=root,A=Check
Access,P=C,Q=34,PBN=zlinux,PON=NetIncoming,NRH=9.12.5.148,NP=tcp,NS=22,APID=1258,R
PPN=/usr/sbin/sshd,RPSN=/usr/sbin/sshd,O=S,UQ=0
TS=Tue 13 Nov 2007 04:31:29 PM EST,E=7,V=P,R=5,RT=File,AN=root,AEN=root,A=Check
Access,P=wr,Q=34,PBN=zlinux,PON=File/dev/kazndrv,SRN=/dev/kazndrv,APID=16927,RPSN=
/usr/sbin/sshd,O=S,UQ=0
TS=Tue 13 Nov 2007 04:31:29 PM EST,E=7,V=P,R=5,RT=File,AN=root,AEN=root,A=Check
Access,P=wr,Q=34,PBN=zlinux,PON=File/var/pdos/lpm,SRN=/var/pdos/lpm/auth_rw.lock,A
PID=16927,RPPN=/usr/sbin/sshd,RPSN=/usr/sbin/sshd,O=S,UQ=1
TS=Tue 13 Nov 2007 04:31:29 PM EST,E=7,V=P,R=5,RT=File,AN=root,AEN=root,A=Check
Access,P=l,Q=34,PBN=zlinux,PON=File/var/pdos/login,SRN=/var/pdos/login,APID=16927,
RPPN=/usr/sbin/sshd,RPSN=/usr/sbin/sshd,O=S,UQ=2
TS=Tue 13 Nov 2007 04:31:32 PM EST,E=7,V=P,R=5,RT=File,AN=root,AEN=root,A=Check
Access,P=l,Q=34,PBN=zlinux,PON=File/var/pdos/login,SRN=/var/pdos/login,APID=16929,
RPPN=/usr/sbin/sshd,RPSN=/usr/sbin/sshd,O=S,UQ=0
...
TS=Tue 13 Nov 2007 04:31:32 PM EST,E=7,V=P,R=5,RT=File,AN=root,AEN=root,A=Check
Access,P=wr,Q=34,PBN=zlinux,PON=File/dev/kazndrv,SRN=/dev/kazndrv,APID=16927,RPSN=
/usr/sbin/sshd,O=S,UQ=5
TS=Tue 13 Nov 2007 04:31:32 PM EST,E=7,V=P,R=5,RT=File,AN=root,AEN=root,A=Check
Access,P=wr,Q=34,PBN=zlinux,PON=File/dev/kazndrv,SRN=/dev/kazndrv,APID=16927,RPSN=
/usr/sbin/sshd,O=S,UQ=6
```

```
TS=Tue 13 Nov 2007 04:31:32 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/sh (sh -c
/usr/X11R6/bin/xauth -q -),APID=16932,RPSN=/usr/sbin/sshd,UQ=7
...
```

There are a number of records related to TAMOS checking to see whether sshd is considered a valid login mechanism (that is, defined to TAMOS as a login program) and whether the user is allowed to use this login process.

Notice that all records are logged against the user root. There are no records here that indicate that the user logged in as testroot. You could not tie any activity here to the testroot login.

Note also that there is a lot of *noise* generated by having the level of logging turned on. Most of the records shown above have been generated by the TraceFile and TraceExec audit settings.

Example 9-60 shows the audit records relating to the testroot user performing the **date**, **whoami**, **pdoswhoami**, and **ls** commands.

Example 9-60 testroot user performing commands

```
TS=Tue 13 Nov 2007 04:31:35 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/date
(date),APID=16951,RPSN=/bin/bash,UQ=0
TS=Tue 13 Nov 2007 04:31:42 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/usr/bin/whoami
(whoami),APID=16953,RPSN=/bin/bash,UQ=0
TS=Tue 13 Nov 2007 04:31:45 PM
EST,E=27,V=P,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,PRS=/opt/pdos/bin/pdoswhoam
i,ARS=/usr/bin/pdoswhoami [SG] (pdoswhoami),APID=16954,RPSN=/bin/bash,UQ=0
TS=Tue 13 Nov 2007 04:31:45 PM
EST,E=28,V=P,R=1,RT=TraceFile,AN=root,AEN=root,A=Trace,P=r,PRS=/opt/pdos/lib,ARS=/
opt/pdos/lib/libosseal.so,APID=16954,RPSN=/usr/bin/pdoswhoami,UQ=1
... (7 similar messages relating to /usr/bin/pdoswhoami removed) ...
TS=Tue 13 Nov 2007 04:32:27 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/ls (/bin/ls -a -N
--color=tty -T 0 -ltr),APID=16959,RPSN=/bin/bash,UQ=0
```

Again, there is no indication that the user is testroot. Finally, Example 9-61 shows the user editing the inittab file and running another **ls** command and another **date** command.

Example 9-61 testroot user modifying /etc/inittab

```
TS=Tue 13 Nov 2007 04:32:33 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/usr/bin/vi (vi
inittab),APID=16961,RPSN=/bin/bash,UQ=0
TS=Tue 13 Nov 2007 04:32:59 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=testroot,AEN=testroot,A=Trace,ARS=/bin/ls
(/bin/ls -a -N --color=tty -T 0 -ltr),APID=16964,RPSN=/bin/bash,UQ=0
TS=Tue 13 Nov 2007 04:33:31 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=testroot,AEN=testroot,A=Trace,ARS=/bin/date
(date),APID=16969,RPSN=/bin/bash,UQ=0
```

It is important to note that the audit records for the **vi** command only indicate the file name entered on the command line, which in this case is just inittab. If the user has not entered the full path name on the command line you will not see it in the audit trace.

Also, the auditing has no way of knowing what has been changed in the file. It does not write before and after records.

Notice that after the file has been edited the audit records now show the user as testroot. This matches with the change of ownership of all the files shown in Example 9-58 on page 249. At this point the operating system has switched to associating the user *testroot* with UID 0 rather than root. All of the files are still owned by UID 0 but the operating system now associates activity with testroot.

In summary, the TAMOS audit facility has been able to trace the activity of a user, but has not been able to identify activity for the testroot user as distinct to activity for the root user. So in this test case TAMOS auditing is not very effective.

9.5.4 Root attempts to hide audit trail test case

A key concern of the UNIX security model is that the root user can access any file bypassing the file permission settings. This means that if the operating system produces an audit trail of activity, root can come along and delete the file or modify the file to remove the relevant records.

TAMOS implements its own auditing mechanism and can stop root from modifying the files. The exposure is that if TAMOS is not running, the root user can access the files. This test case shows the root user attempting to access the files while TAMOS is up and then stopping TAMOS to access the files.

Performing the test case

First we attempted to access the TAMOS audit files and copy them to another location, as shown in Example 9-62.

Example 9-62 root test case 3 execution part 1

```
linux11z:~/scripts # date
Tue Nov 13 17:08:22 EST 2007
linux11z:~/scripts # cd /var/pdos/audit
linux11z:/var/pdos/audit # ls -l
/bin/ls: .: Permission denied
linux11z:/var/pdos/audit # cp audit.log /tmp
cp: cannot open `audit.log' for reading: Permission denied
linux11z:/var/pdos/audit # date
Tue Nov 13 17:08:44 EST 2007
linux11z:/var/pdos/audit #
```

In this case TAMOS has stopped root from listing the contents of the directory and copying the file to another location.

Next we shut down TAMOS and repeated the steps, as shown in Example 9-63.

Example 9-63 root test case 3 execution part 2

```
linux11z:/var/pdos/audit # date
Tue Nov 13 17:21:20 EST 2007
linux11z:/var/pdos/audit # pdosctl -k
```

```
pdosd shutdown
pdoswdd shutdown
pdoslpmd shutdown
pdosauditd shutdown
linux11z:/var/pdos/audit # ls -l
total 29092
drwxrwx---   2 root    ossaudit    4096 2007-11-13 16:36 .
dr-xr-xr-x  25 osseal  osseal      4096 2006-03-29 12:05 ..
-rw-r-----   1 root    ossaudit  547814 2007-11-13 17:21 audit.log
-rw-r-----   1 root    ossaudit 1000127 2007-10-31 00:05
audit.log.2007-10-31-00-05-49
linux11z:/var/pdos/audit # cp audit.log /tmp
linux11z:/var/pdos/audit # date
Tue Nov 13 17:22:00 EST 2007
linux11z:/var/pdos/audit # rm audit.log
linux11z:/var/pdos/audit # date
Tue Nov 13 17:23:28 EST 2007
```

As root is a TAMOS administrator, it is allowed to shut down the TAMOS deamons. Once TAMOS is down and no longer providing access control over the files, root can view, copy, and delete the audit log files.

Checking audit log capture

Because the root user has deleted the live audit log and TAMOS has created a new one on startup, there is a large hole in the audit record flow between 12:27:19 and 17:24:15, as shown in Example 9-64.

Example 9-64 Audit records missing after delete

```
TS=Tue 13 Nov 2007 12:27:19 PM
EST,E=7,V=P,R=5,RT=Surrogate,AN=root,AEN=root,A=Check
Access,P=G,Q=34,PBN=zlinux,PON=Surrogate/Group,SN=postfix,APID=1375,RPSN=/usr/lib/
postfix/master,O=S,UQ=1
TS=Tue 13 Nov 2007 12:27:19 PM
EST,E=7,V=P,R=5,RT=Surrogate,AN=root,AEN=root,A=Check
Access,P=G,Q=34,PBN=zlinux,PON=Surrogate/User,SN=postfix,APID=1375,RPSN=/usr/lib/p
ostfix/master,O=S,UQ=2
TS=Tue 13 Nov 2007 05:24:15 PM
EST,E=35,V=H,R=6,RT=Health,AN=root,AEN=root,A=CertLife,SRN=/var/pdos/certs/pdosd.k
db:APPL_LDAP_CERT,NRH=wtsc60.itso.ibm.com,NP=tcp,NS=3392,APID=17394,AP=Expiration
Date=2017-10-23 Remaining Days=3631,O=S,UQ=0
TS=Tue 13 Nov 2007 05:24:17 PM EST,E=7,V=P,R=5,RT=File,AN=root,AEN=root,A=Check
Access,P=x,Q=34,PBN=zlinux,PON=File/opt/pdos/bin/pdosshowmsg,SRN=/opt/pdos/bin/pdo
sshowmsg,APID=17524,RPSN=/opt/pdos/bin/rc.osseal,O=S,UQ=0
TS=Tue 13 Nov 2007 05:24:17 PM
EST,E=27,V=P,R=5,RT=TraceExec,AN=root,AEN=root,A=Trace,PRS=/opt/pdos/bin/pdosshowm
sg,ARS=/opt/pdos/bin/pdosshowmsg (/opt/pdos/bin/pdosshowmsg pdosr.cat
35a61082),APID=17524,RPSN=/opt/pdos/bin/rc.osseal,UQ=1
```

The size of this gap will be dictated by the frequency of audit logs rolling over. But if the root user can delete the current audit log file, she can also delete the old copies.

Out of interest we shut down TAMOS, restored the file that was deleted, and restarted TAMOS. Example 9-65 shows the audit records for the date and **pdosctl** commands corresponding to the activity in Example 9-63 on page 251.

Example 9-65 Audit records for TAMOS shutdown

```
TS=Tue 13 Nov 2007 05:21:20 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,ARS=/bin/date
(date),APID=17360,RPSN=/bin/bash,UQ=0
TS=Tue 13 Nov 2007 05:21:24 PM
EST,E=27,V=P,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,PRS=/opt/pdos/bin/pdosctl,A
RS=/usr/bin/pdosctl [SUG] (pdosctl -k),APID=17361,RPSN=/bin/bash,UQ=0
TS=Tue 13 Nov 2007 05:21:24 PM EST,E=7,V=P,R=5,RT=File,AN=root,AEN=root,A=Check
Access,P=x,Q=34,PBN=zlinux,PON=File/opt/pdos,SRN=/usr/bin/pdosctl,APID=17361,RPSN=
/bin/bash,O=S,UQ=1
TS=Tue 13 Nov 2007 05:21:24 PM
EST,E=27,V=P,R=1,RT=TraceExec,AN=root,AEN=root,A=Trace,PRS=/opt/pdos,ARS=/usr/bin/
pdosctl [SUG] (pdosctl -k),APID=17361,RPSN=/bin/bash,UQ=2
TS=Tue 13 Nov 2007 05:21:25 PM
EST,E=24,V=A,R=1,RT=Process,AN=root,AEN=root,A=Stop,APID=7328,RPPN=/opt/pdos/bin/p
dosd,RPSN=/opt/pdos/bin/pdosd,O=S,UQ=0
TS=Tue 13 Nov 2007 05:21:27 PM
EST,E=24,V=A,R=1,RT=Process,AN=root,AEN=root,A=Stop,APID=7344,RPPN=/opt/pdos/bin/p
doswdd,RPSN=/opt/pdos/bin/pdoswdd,O=S,UQ=0
TS=Tue 13 Nov 2007 05:21:29 PM
EST,E=24,V=A,R=1,RT=Process,AN=root,AEN=root,A=Stop,APID=7336,RPPN=/opt/pdos/bin/p
doslpmd,RPSN=/opt/pdos/bin/pdoslpmd,O=S,UQ=0
TS=Tue 13 Nov 2007 05:21:31 PM
EST,E=24,V=A,R=1,RT=Process,AN=root,AEN=root,A=Stop,APID=7332,RPPN=/opt/pdos/bin/p
dosauditd,RPSN=/opt/pdos/bin/pdosauditd,O=S,UQ=0
```

Here we can see the **date** command followed by three records relating to the **pdosctl -k** command. The last records, with A=Stop, are administrative audit records showing the TAMOS processes being stopped.

How do you solve this dilemma of root being able to shut down TAMOS and delete the files? There are a number of options:

- Implement some form of live backup mechanism that is continuously taking a backup of the audit files that a root user cannot access.
- Implement system monitoring that alerts an operator and logs a trouble ticket when the TAMOS deamons are shut down. This will not stop the root user from deleting files, but will record the fact that they have been shut down. If only a limited number of users are defined as osseal-administrators there will be focus on who could have stopped the deamons. This could be supplemented by a monitoring watch on the active audit log. If it disappears then an alarm is raised.
- Remove root from the osseal-admin group so that the user is no longer considered an administrator and cannot shut down TAMOS. However, depending on procedures, this may cause issues with system maintenance.
- Treat the root user as a special account that can only be logged into from the master console (this can be set as policy in TAM) and the root password is stored securely (such as in a safe) and only accessed by exception (with a trouble ticket raised). When root is used, it is under exceptional circumstances and its use is closely monitored. If the audit file disappeared during this time, there would be a record of who accessed the root password and logged in as root.

Thus, this exposure can be managed proceduraly.

9.5.5 Tracking su use test case

Many customers struggle with the issue of users getting the root password and su'ing to root to perform tasks. They may have a legitimate need and be very disciplined about its use or it may be that the root passwords are rarely changed and have become common knowledge over time.

This use case shows an ordinary user su'ing to root and performing some tasks before exiting out.

Performing the test case

This test case involves an ordinary user su'ing to root, as shown in Example 9-66.

Example 9-66 root test case 4 execution part 1

```
login as: jsmith
Password:
Last login: Tue Nov 13 18:18:35 2007 from 9.12.5.148
jsmith@linux11z:~> whoami
jsmith
jsmith@linux11z:~> pdoswhoami
jsmith
jsmith@linux11z:~> pdoswhoami -a
1106 jsmith
jsmith@linux11z:~> su root
Password:
linux11z:/home/jsmith # whoami
root
linux11z:/home/jsmith # pdoswhoami -a
1106 jsmith
```

Before the su is executed, both the Linux **whoami** command and the TAMOS **pdoswhoami** command show the same user, jsmith. After the user executes su, the operating system considers him root. But TAMOS is still tracking his login identity, jsmith (with UID 1106).

Example 9-67 shows the user performing a series of commands as root.

Example 9-67 root test case 4 execution part 2

```
linux11z:/home/jsmith # date
Tue Nov 13 18:52:48 EST 2007
linux11z:/home/jsmith # cd ~/scripts
linux11z:~/scripts # ls
.   add_nativeid.ldif  findjava         lmod   lsrch_ssl
..  amoscfg            junct_create.pd  lsrch  tamos_case1.sh
linux11z:~/scripts # ./lsrch > /dev/nul 2>&1
linux11z:~/scripts # ls -l /etc/passwd
-rw-r--r--  1 root root 1340 2007-11-13 18:15 /etc/passwd
linux11z:~/scripts # touch /etc/passwd
linux11z:~/scripts # ls -l /etc/passwd
-rw-r--r--  1 root root 1340 2007-11-13 18:53 /etc/passwd
linux11z:~/scripts # pdosctl -a
bash: /usr/bin/pdosctl: Permission denied
linux11z:~/scripts # pdosctl -k
```

```
bash: /usr/bin/pdosctl: Permission denied
linux11z:~/scripts # date
Tue Nov 13 18:54:05 EST 2007
linux11z:~/scripts # exit
exit
jsmith@linux11z:~> touch /etc/passwd
touch: cannot touch `/etc/passwd': Permission denied
jsmith@linux11z:~> pdosctl -a
-bash: /usr/bin/pdosctl: Permission denied
jsmith@linux11z:~> whoami
jsmith
```

The user may have been able to execute some of the commands as himself, but modifying the /etc/passwd inode entry and running **pdosctl** require root authority (in our environment). Notice that the user has been allowed to touch the passwd file but not run the **pdosctl** command. Access to touch the passwd file as jsmith was not allowed.

TAMOS has allowed some access according to the assumed authority (root) but not allowed running of the TAMOS control command, as the user is not a member of the osseal-admin group.

Checking audit log capture

As we have a trace level of trace_exec_l enabled. Trace records are cut when a user is running under another identity. Example 9-68 shows the audit records for jsmith running the **date**, **ls**, and **lsrch** commands.

Example 9-68 Audit records for su'd user jsmith

```
TS=Tue 13 Nov 2007 06:52:48 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/bin/date
(date),APID=18552,RPSN=/bin/bash,UQ=0
TS=Tue 13 Nov 2007 06:52:58 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/bin/ls (/bin/ls -a
-N --color=tty -T 0),APID=18554,RPSN=/bin/bash,UQ=0
TS=Tue 13 Nov 2007 06:53:11 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/root/scripts/lsrch
(./lsrch),APID=18557,RPSN=/bin/bash,UQ=0
TS=Tue 13 Nov 2007 06:53:11 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/bin/sh
(./lsrch),APID=18557,RPSN=/root/scripts/lsrch,UQ=1
TS=Tue 13 Nov 2007 06:53:11 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/root/scripts/lsrch
(/bin/sh ./lsrch),APID=18557,RPSN=/root/scripts/lsrch,UQ=2
TS=Tue 13 Nov 2007 06:53:11 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/bin/sh (/bin/sh
./lsrch),APID=18557,RPSN=/root/scripts/lsrch,UQ=3
TS=Tue 13 Nov 2007 06:53:11 PM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/usr/bin/idsldapsearc
h (idsldapsearch -b o=itso -s sub -h 9.12.4.18 -p 3391 -D cn=LDAP Administrator -w
secret ibm-nativeId=*),APID=18558,RPSN=/root/scripts/lsrch,UQ=4
```

There are a few points to note about these audit records:

- All show the login user name (AN=) and effective user name (AEN=), so even though the user is running as root, we can still tie activity to his login identity. This is a very powerful feature.
- When the user executes a script, such as the lsrch script, you see the full path of the executable, not just the local name. This is different from the records cut in Example 9-61 on page 250 where the file being edited was an argument on a command line.
- The records show the chain of commands in the lsrch script, such as the **idsldapsearch** command, not just the executed script.

The remaining audit records for this test case have not been shown. They are similar to the above.

There was no audit record showing the su to root. This was because the audit settings (trace_exec_root and trace_exec_l) only audit activity by a user who logged in as root or whose effective ID is different from their login ID. At the time of running the **su** command the user jsmith has an effective ID of jsmith.

9.5.6 Tracking sudo use test case

The *sudo* (super user do) program is common on UNIX/Linux systems to give users the ability to run higher-authority commands without giving them access to the user ID or password to run the commands. It uses a file to define who can run higher-authority commands, what systems they can run these on, and other settings such as whether their password is required, the target user password, or no password at all.

This test case involves an ordinary user running sudo-allowed commands to administer users, such as adding users, setting their password, modifying the attributes, and deleting users.

> **Note:** TAMOS provides its own sudo function that is tied to the central TAMOS policy model. Users run the **pdossudo** command and access to the command is checked against TAMOS SUDO resources in TAM.

Performing the test case

For this test case we turned all auditing on to see what records we got in relation to sudo. The audit levels enabled were permit, deny, admin, loginpermit, logindeny, trace_exec_root, trace_exec_l, and trace_file. For this test case TAMOS was running.

In addition to the TAMOS configuration we needed to define the sudo configuration file, called /etc/sudoers. This is shown in Example 9-69.

Example 9-69 sudoers file for root test case 5

```
linux11z:/usr/sbin # cat /etc/sudoers
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the sudoers man page for the details on how to write a sudoers file.
#
# Host alias specification
# User alias specification
# Cmnd alias specification
```

```
Cmnd_Alias      USRADM = /usr/sbin/useradd, /usr/sbin/usermod, \
                         /usr/sbin/userdel, /usr/bin/passwd

# Defaults specification
# Defaults targetpw    # ask for the password of the target user i.e. root
# %users ALL=(ALL) ALL # WARNING! Only use this together with 'Defaults targetpw'!

# User privilege specification
# You should not use sudo as root in an SELinux environment
# If you use SELinux, remove the following line
root    ALL=(ALL) ALL

# Uncomment to allow people in group wheel to run all commands
# %wheel        ALL=(ALL)       ALL

# Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL

# Samples
# %users  ALL=/sbin/mount /cdrom,/sbin/umount /cdrom
# %users  localhost=/sbin/shutdown -h now
jsmith          ALL=(ALL)       USRADM
```

Most of the file shown above is the default file. The lines that we configured are highlighted:

- The Cmnd_Alias statement associates the **useradd**, **usermod**, **userdel**, and **passwd** commands with the USRADM label.
- The last line associates this command label with the user jsmith (our test user). The ALL = USRADM means that jsmith can run the user administration commands on all systems where this file is used.

The test case involves jsmith running a series of user administration commands through sudo, as shown in Example 9-70.

Example 9-70 root test case 5 execution

```
login as: jsmith
Password:
Last login: Wed Nov 14 08:57:15 2007 from 9.12.5.148
jsmith@linux11z:~> date
Wed Nov 14 09:11:14 EST 2007
jsmith@linux11z:~> sudo -l
Password:
Sorry, try again.
Password:
User jsmith may run the following commands on this host:
    (ALL) /usr/sbin/useradd, /usr/sbin/usermod, /usr/sbin/userdel, /usr/bin/passwd
jsmith@linux11z:~> sudo /usr/sbin/useradd testuser
jsmith@linux11z:~> sudo /usr/bin/passwd testuser
Changing password for testuser.
New password:
Re-enter new password:
Password changed
jsmith@linux11z:~> sudo /usr/sbin/usermod -c "Test User" -p secret00 testuser
jsmith@linux11z:~> grep testuser /etc/passwd
testuser:x:1107:100:Test User:/home/testuser:/bin/bash
```

```
jsmith@linux11z:~> sudo /usr/sbin/userdel -r testuser
no crontab for testuser
jsmith@linux11z:~> date
Wed Nov 14 09:13:01 EST 2007
```

There are four user administration commands run in the following sequence; **useradd**, **passwd**, **usermod**, and **userdel**. These are a good example of commands that you would give out to ordinary users through sudo instead of giving them root access.

Checking audit log capture

The default audit log for sudo is the /usr/log/messages file written through syslog. Example 9-71 shows the entries in the file that relate to the test case activity.

Example 9-71 /var/log/messages entries for test case

```
Nov 14 09:11:06 linux11z sshd[25214]: Accepted keyboard-interactive/pam for jsmith
from 9.12.5.148 port 2015 ssh2
Nov 14 09:11:25 linux11z sudo:   jsmith : TTY=pts/2 ; PWD=/home/jsmith ; USER=root
; COMMAND=list
Nov 14 09:11:49 linux11z sudo:   jsmith : TTY=pts/2 ; PWD=/home/jsmith ; USER=root
; COMMAND=/usr/sbin/useradd testuser
Nov 14 09:11:49 linux11z useradd: new user: name=testuser, uid=1107, gid=100,
home=/home/testuser, shell=/bin/bash
Nov 14 09:12:02 linux11z sudo:   jsmith : TTY=pts/2 ; PWD=/home/jsmith ; USER=root
; COMMAND=/usr/bin/passwd testuser
Nov 14 09:12:33 linux11z sudo:   jsmith : TTY=pts/2 ; PWD=/home/jsmith ; USER=root
; COMMAND=/usr/sbin/usermod -c Test User -p secret00 testuser
Nov 14 09:12:56 linux11z sudo:   jsmith : TTY=pts/2 ; PWD=/home/jsmith ; USER=root
; COMMAND=/usr/sbin/userdel -r testuser
Nov 14 09:12:56 linux11z crontab[25258]: (root) DELETE (testuser)
```

The audit log is quite effective. It identifies the login user (jsmith), the directory that she is running the command from, the user that the command is being run as (root), and the command that was run. It would be reasonably simple to translate these messages into TCIMs W7 format using a tool like TDI.

The TAMOS audit log also shows this activity. Some of the records associated with the **useradd** command are shown in Example 9-72.

Example 9-72 TAMOS audit log for sudo useradd command

```
TS=Wed 14 Nov 2007 09:11:49 AM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/usr/sbin/useradd
(/usr/sbin/useradd testuser),APID=25245,RPSN=/usr/bin/sudo,UQ=10
TS=Wed 14 Nov 2007 09:11:49 AM EST,E=7,V=P,R=1,RT=File,AN=jsmith,AEN=root,A=Check
Access,P=wr,Q=34,PBN=zlinux,PON=File/dev/kazndrv,SRN=/dev/kazndrv,APID=25245,RPSN=
/usr/sbin/useradd,O=S,UQ=11
TS=Wed 14 Nov 2007 09:11:49 AM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/usr/sbin/useradd.loc
al (/usr/sbin/useradd.local testuser 1107 100
/home/testuser),APID=25246,RPSN=/usr/sbin/useradd,UQ=13
TS=Wed 14 Nov 2007 09:11:49 AM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/bin/bash
(/usr/sbin/useradd.local testuser 1107 100
/home/testuser),APID=25246,RPSN=/usr/sbin/useradd.local,UQ=14
```

```
TS=Wed 14 Nov 2007 09:11:49 AM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/usr/sbin/useradd.loc
al (/bin/bash /usr/sbin/useradd.local testuser 1107 100
/home/testuser),APID=25246,RPSN=/usr/sbin/useradd.local,UQ=15
TS=Wed 14 Nov 2007 09:11:49 AM
EST,E=27,V=T,R=1,RT=TraceExec,AN=jsmith,AEN=root,A=Trace,ARS=/bin/bash (/bin/bash
/usr/sbin/useradd.local testuser 1107 100
/home/testuser),APID=25246,RPSN=/usr/sbin/useradd.local,UQ=16
```

Note that TAMOS considers this access as surrogate, the same as though the user had performed a su to root. The audit records show the login user (AN=) and the effective user (AEN=). The audit trace also shows the command arguments used by the lower-level commands (such as the **useradd.local** command).

The audit trace detail for the **passwd**, **usermod**, and **userdel** commands is similar.

9.5.7 Audit reporting for UID=0 TAMOS test cases

There are a number of ways that the audit reporting can be performed for these test cases. On the most basic level you could use the **pdosaudview** command to produce a comma-separated format output and feed it into Microsoft Excel, Microsoft Access, or some other reporting tool.

In this book we focus on reporting in TCIM. It provides actuators to access the raw TAMOS audit data and the mapping of TAMOS records into the TCIM W7 reporting format.

Note: We noticed that TCIM was no longer collecting TAMOS audit records after a certain time. This time seemed to coincide with the test case of deleting the TAMOS audit log. This action may have corrupted a link to the file. Re-installing the actuator resolved the issue.

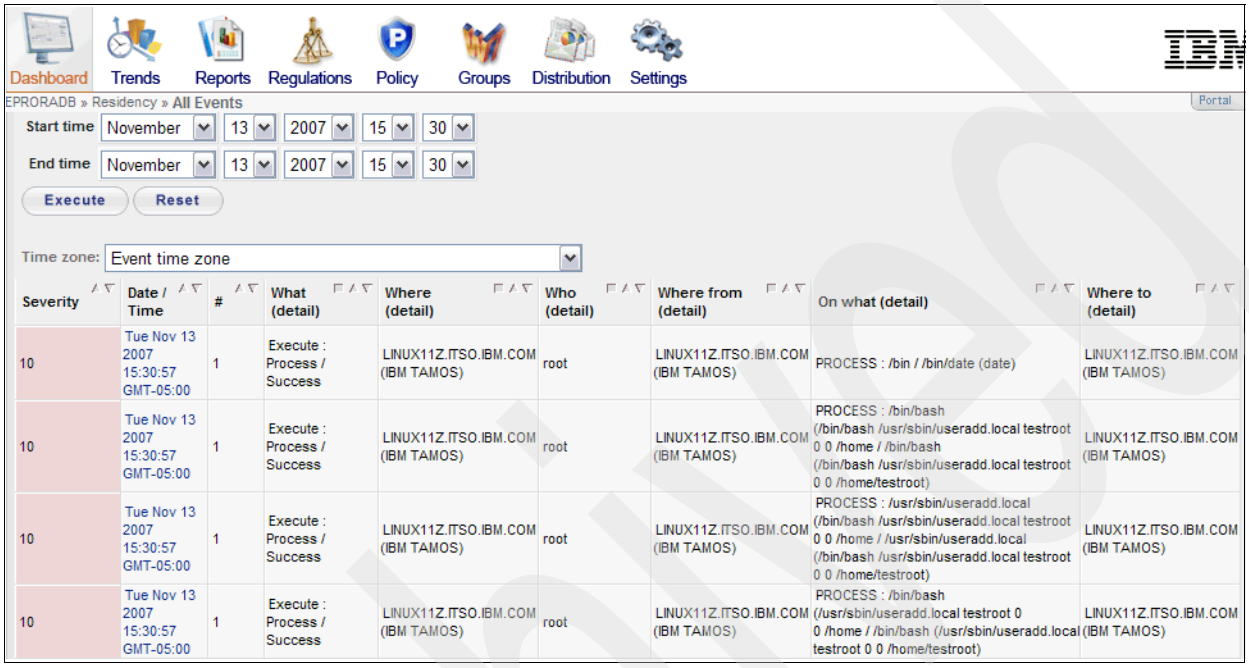
The following sections provide examples of standard and custom reporting against this test case data.

Test case data in TCIM All Events view

This section shows how the TCIM All Events view can show the TAMOS audit data for the test cases.

TCIM Events for the creation of another privilege user test case

The first TAMOS test case (9.5.2, “Create another privileged user test case” on page 245) involved root creating another privileged user (that is, with UID=0). The TAMOS audit records for this were shown in Example 9-54 on page 247 and Example 9-55 on page 247. The TCIM mapping pulls all TAMOS audit records from TAMOS. Figure 9-19 shows some of these records, beginning with the first **`date`** command.



Severity	Date / Time	#	What (detail)	Where (detail)	Who (detail)	Where from (detail)	On what (detail)	Where to (detail)
10	Tue Nov 13 2007 15:30:57 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /bin / /bin/date (date)	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)
10	Tue Nov 13 2007 15:30:57 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /bin/bash (/bin/bash /usr/sbin/useradd.local testroot 0 0 /home / /bin/bash (/bin/bash /usr/sbin/useradd.local testroot 0 0 /home/testroot)	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)
10	Tue Nov 13 2007 15:30:57 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /usr/sbin/useradd.local (/bin/bash /usr/sbin/useradd.local testroot 0 0 /home / /usr/sbin/useradd.local (/bin/bash /usr/sbin/useradd.local testroot 0 0 /home/testroot)	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)
10	Tue Nov 13 2007 15:30:57 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /bin/bash (/usr/sbin/useradd.local testroot 0 0 /home / /bin/bash (/usr/sbin/useradd.local testroot 0 0 /home/testroot)	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)

Figure 9-19 TCIM report for TAMOS test case

The mapping process has categorized the TAMOS data in the W7 mapping process. The TraceExec audit record for the date command has been mapped as follows:

- The *when* (date/time) value has been derived from the timestamp in the audit record.
- The *what* value has been set to Execute : Process / Success as the TraceExec record showed a success.
- The *where* value is a combination of the server ID (host name) and the actuator (IBM TAMOS).
- The *who* value is the user issuing the date command.
- The *WhereFrom* and *WhereTo* values are the same as the *where* (these would be the same except for NetOutgoing and NetIncoming TAMOS records).
- The *OnWhat* value is a combination of PROCESS and the process name.

You will notice that a severity has been assigned in the TCIM load process based on some default policy associated with the TAMOS audit records.

If you look at the Date/Time field you will notice that the time is only recorded to the second, not hundreds of seconds that TAMOS records them as. This can result in records appearing out of time order in TCIM. The records shown above for the **`useradd.local`** command appear before the **`useradd`** command record (see Figure 9-20 on page 261). However, in the TAMOS audit log (Example 9-55 on page 247) the **`useradd`** command comes before the **`useradd.local`** commands. You should be careful associating a sequence of activities based on the sequence of records displayed in TCIM.

Figure 9-20 shows the **useradd** command from test case 1.

Tue Nov 13 2007 15:30:57 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /usr/sbin/useradd (useradd -g root -p secret00 -u 0 -o -r -s /bin / /usr/sbin/useradd (useradd -g root -p secret00 -u 0 -o -r -s /bin/bash testroot)

Figure 9-20 TCIM record for TAMOS user add

Like the TAMOS audit record, the entire command with arguments is shown. In this case it includes the password being set for the new user. If there will be this type of information available through TCIM, you need to consider who has access to view the data.

TCIM Events for the new user login test case

With test case 2 (9.5.3, “Log in as a new user and perform suspect activity test case” on page 248) we demonstrated that the TAMOS audit log did not identify that the new UID 0 user (testroot) logged into the system. This would not be different under TCIM.

TCIM Events for attempt to hide audit trail test case

With test case 3 (9.5.4, “Root attempts to hide audit trail test case” on page 251) the root user shut down TAMOS and deleted the TAMOS audit log. We can see the shutdown commands in TCIM, as shown in Figure 9-21.

Tue Nov 13 2007 17:21:20 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /bin / /bin/date (date)
Tue Nov 13 2007 17:21:24 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /usr/bin / /usr/bin/pdosctl [SUG] (pdosctl -k)
Tue Nov 13 2007 17:21:24 GMT-05:00	1	Authorize : User / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	FILE : File/opt/pdos / /usr/bin/pdosctl
Tue Nov 13 2007 17:21:24 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /usr/bin / /usr/bin/pdosctl [SUG] (pdosctl -k)
Tue Nov 13 2007 17:21:25 GMT-05:00	1	Stop : System / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	SYSTEM : - / LINUX11Z.ITSO.IBM.COM
Tue Nov 13 2007 17:21:27 GMT-05:00	1	Stop : System / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	SYSTEM : - / LINUX11Z.ITSO.IBM.COM
Tue Nov 13 2007 17:21:29 GMT-05:00	1	Stop : System / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	SYSTEM : - / LINUX11Z.ITSO.IBM.COM
Tue Nov 13 2007 17:21:31 GMT-05:00	1	Stop : System / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	root	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	SYSTEM : - / LINUX11Z.ITSO.IBM.COM

Figure 9-21 TCIM records for TAMOS shutdown command

There are three records for the shutdown command **pdosctl -k**. These are from the TAMOS audit records shown in Example 9-65 on page 253.

The last four records show a successful system stop. These are misleading. The system has not stopped. The four TAMOS deamons have been stopped. Compare these records with the TAMOS audit records shown in Example 9-65 on page 253. They show the four TAMOS processes, pdosd, pdoswdd, pdoslpmd, and pdosauditd being shut down. This is one case where relevant data has not been effectively mapped from the TAMOS audit records to TCIM.

Note that there are no records relating to the TAMOS audit files being copied and deleted. This is because TAMOS itself was down and so could not cut records. Covering this auditing hole would require operating system level monitoring of the audit files. TCIM could then show the activities chronologically.

TCIM Events for su tracking test case

With test case 4 (9.5.5, “Tracking su use test case” on page 254) the user jsmith logged in, su’d to root, and performed some activities. Some of these activities are shown in Figure 9-22.

Tue Nov 13 2007 18:52:48 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	jsmith	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /bin / /bin/date (date)
Tue Nov 13 2007 18:52:58 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	jsmith	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /bin/ls (/bin / /bin/ls (/bin/ls -a -N --color=tt
Tue Nov 13 2007 18:53:11 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	jsmith	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /opt/ibm/ldap/V6.0/bin/32/ldapsearch (/opt/ibm/ldap/V6.0/bin/32 / /opt/ibm/ldap/V6.0/bin/32/ld (/opt/ibm/ldap/V6.0/bin/32/ldapsearch -b o=itso -s sub 9.12.4.18 -p 33

Figure 9-22 TCIM records for jsmith su’d to root

While these records show what activity jsmith was performing on the system, there is no indication that the user has su’d to root and was running under the higher authority. While the TAMOS audit records show that the login user ID is different from the effective user, this information has not been mapped across to the TCIM records. The W7 format does not seem to allow for two levels of who (login who and effective who) in process records.

As mentioned previously, there is no record in the TAMOS audit log of the user performing the su to root due to the level of TAMOS tracing turned on. This means that there is no record of this in TCIM. Had the original su record been audited by TAMOS and carried to TCIM, you would see the user su’ing to root and then the activity shown above and could assume that they were still running with the higher authority.

TCIM Events for the sudo use test case

With test case 5 (9.5.6, “Tracking sudo use test case” on page 256) we used the **sudo** command to allow jsmith to perform some user administration tasks. Some of the TCIM records are shown in Figure 9-23.

Wed Nov 14 2007 09:11:49 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	jsmith	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	USER : - / root	LIN (IBM
Wed Nov 14 2007 09:11:49 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	jsmith	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	USER : - / ivmgr	LIN (IBM
Wed Nov 14 2007 09:11:49 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	jsmith	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	USER : - / tivoli	LIN (IBM
Wed Nov 14 2007 09:11:49 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	jsmith	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /usr/sbin/useradd (/usr/sbin / /usr/sbin/useradd (/usr/sbin/useradd testuser)	LIN (IBM
Wed Nov 14 2007 09:11:49 GMT-05:00	1	Authorize : User / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	jsmith	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	FILE : File/dev/kazndrv / /dev/kazndrv	LIN (IBM
Wed Nov 14 2007 09:11:49 GMT-05:00	1	Access : File / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	jsmith	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	FILE : /dev / /dev/kazndrv	LIN (IBM
Wed Nov 14 2007 09:11:49 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	jsmith	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /usr/sbin/useradd.local (/usr/sbin/useradd.local testuser 1107 100 /home / /usr/sbin/useradd.local (/usr/sbin/useradd.local testuser 1107 100 /home/testuser)	LIN (IBM
Wed Nov 14 2007 09:11:49 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	jsmith	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	PROCESS : /bin/bash (/usr/sbin/useradd.local testuser 1107 100 /home / /bin/bash (/usr/sbin/useradd.local testuser 1107 100 /home/testuser)	LIN (IBM

Figure 9-23 TCIM records for jsmith using sudo

Unlike with the su test case, the sudo commands show records in TCIM. These are the *Authorize : Su / Success* records shown in the figure. However, there is no indication in the execute records that the commands were run through sudo.

As with the TAMOS audit records, you can see the command-line arguments, such as for the **useradd.local** command, so care needs to be taken with access to this data.

It is worth highlighting the search filtering capabilities of TCIM. Figure 9-24 on page 264 shows the filtering options to show the records for the last test case.

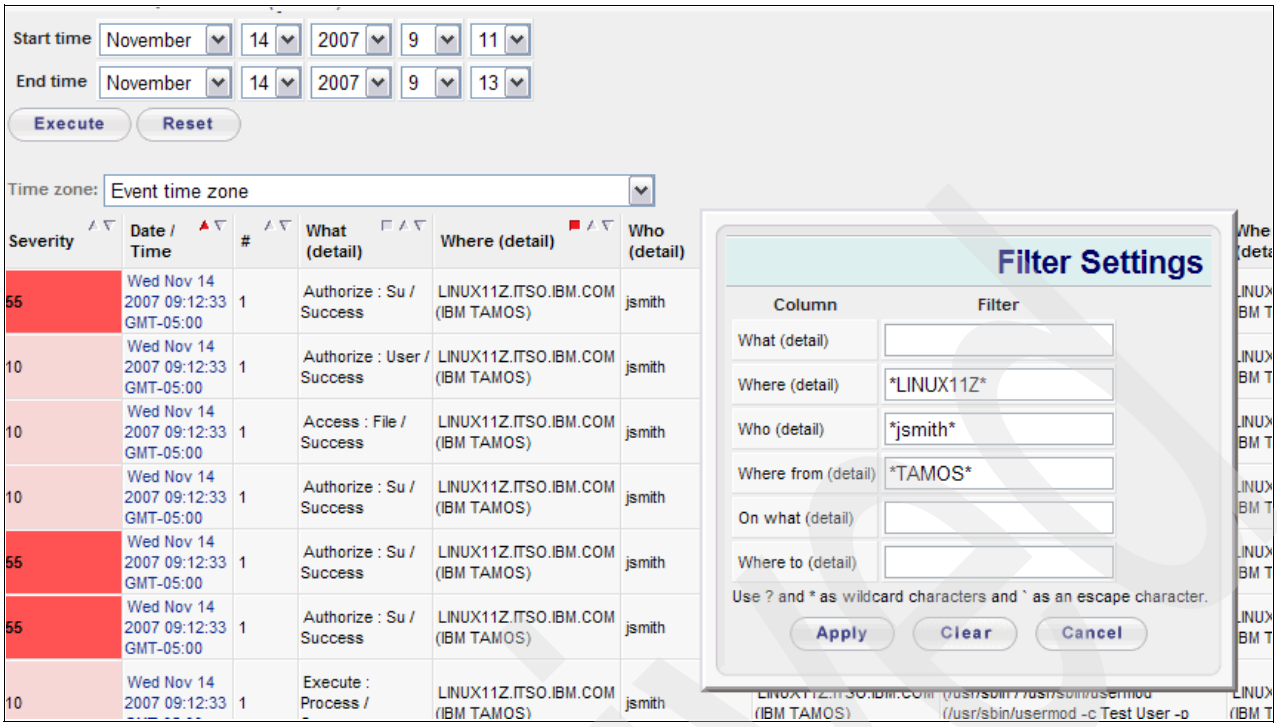


Figure 9-24 TCIM filtering options

On the top left of the figure are the start and end time filters. We also used the filter settings accessible through the small box icons in the column headers to further reduce the data shown. (Note that if there is a filter applied to one of the columns, this box is shown as red.)

Custom TCIM policy created for test case data

It would be common in most deployments to build custom policy to enable better searching of data and to enable TCIM automated actions. This section details some custom policy created to highlight some of the events created in the test cases.

Custom policy - highlight su to root

The first condition that we wanted to highlight was when a user su's to root. We need to be careful, as there are many audit records cut when system processes run as root, so we need to make an exception for system accounts.

TCIM policy is built from a set of groupings that can be applied to policy rules and attention rules. The groupings are defined using the TCIM Console and involve record filters based on boolean logic. Getting the logic correct in the UI can be challenging until you understand how the conditions work.

The simplest approach is to define what conditions you want in your grouping and look at the data that you have available. For this example we want:

- To capture all records where a su to root is performed
- To discard any records where the originator is a system account

Figure 9-25 shows one of the su records in TCIM that we can use to help build the filters.

Event Detail

	Field	Group
Severity	10	This is a policy exception.
When	Wed Nov 14 2007 14:28:19 GMT-05:00	Other Periods (10)
What	Authorize : Su / Success	SU (10)
Where	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	Other Platforms (10)
Who	root	ROOT unix (10)
From Where	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	Other Platforms (10)
On What	USER : - / root	UsysAct (10)
Where To	LINUX11Z.ITSO.IBM.COM (IBM TAMOS)	Other Platforms (10)

Figure 9-25 Event Detail for su record

The two fields we are interested in are the *What* field, showing the su activity, and the *On What* field, showing the user performing the su.

We need to set up filters that will show records where the What field contains “Authorize : Su / Success” and the OnWhat field contains “USER : - / *name*”, where *name* is not one of the system accounts.

We look at the What condition first. You will see that each Field and Group in the Event Detail is a link. If you hover the mouse over the data in the What Field cell, a pop-up will show the structure of the field:

```
mainclass:eventclass/successclass - Show event list for Event type “Authorize : Su
/ Success”
```

In this What field:

- mainclass = Authorize
- eventclass = Su
- successclass = Success

Using the same approach for the On What field we can see that:

- type = USER
- path = -
- name = root

These arguments can be used in the TCIM console to define the grouping filters. The filters we want to apply for this are:

- What; eventclass = Su
- OnWhat; type = USER and name not in (list of system accounts)

The first is simple to implement. The second requires that an OnWhat group be created containing all of the system accounts. Then a condition can be created where the type is USER and the name is not in this group. The implementation of these is shown in Figure 9-26.

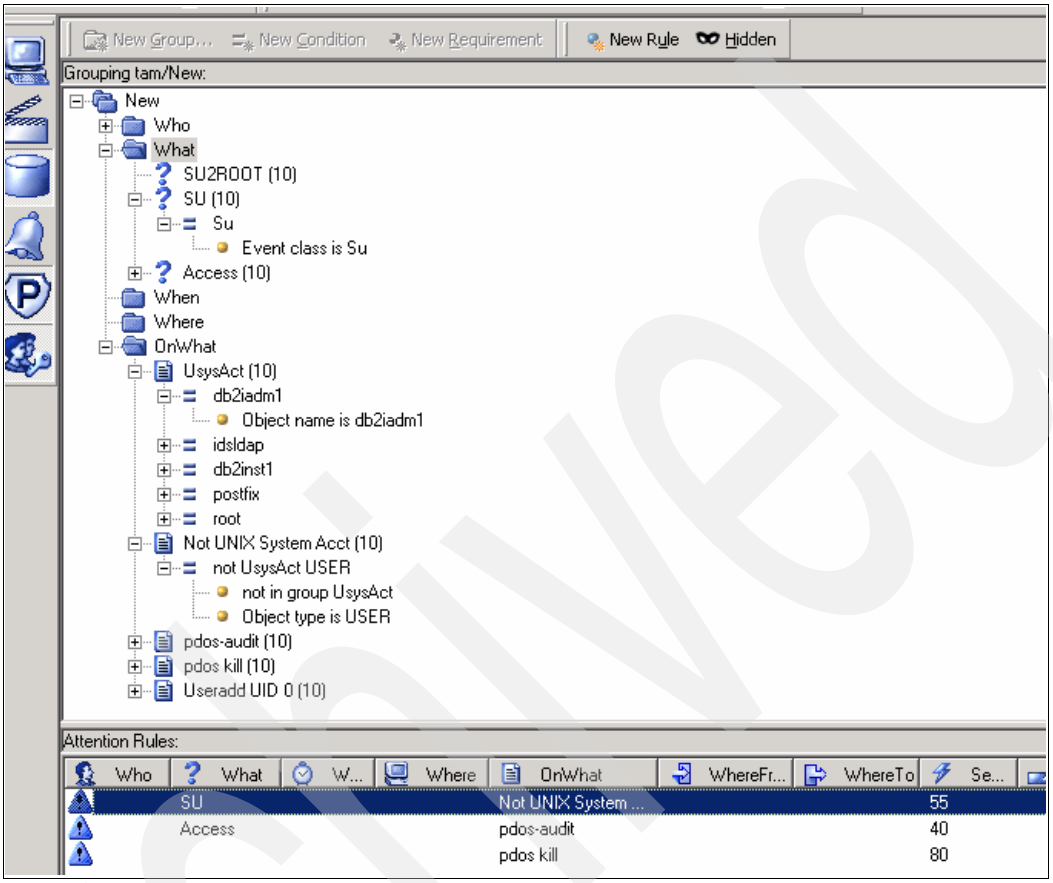


Figure 9-26 Grouping conditions for su to root policy

Towards the top of the figure there is the What condition where Eventclass is Su.

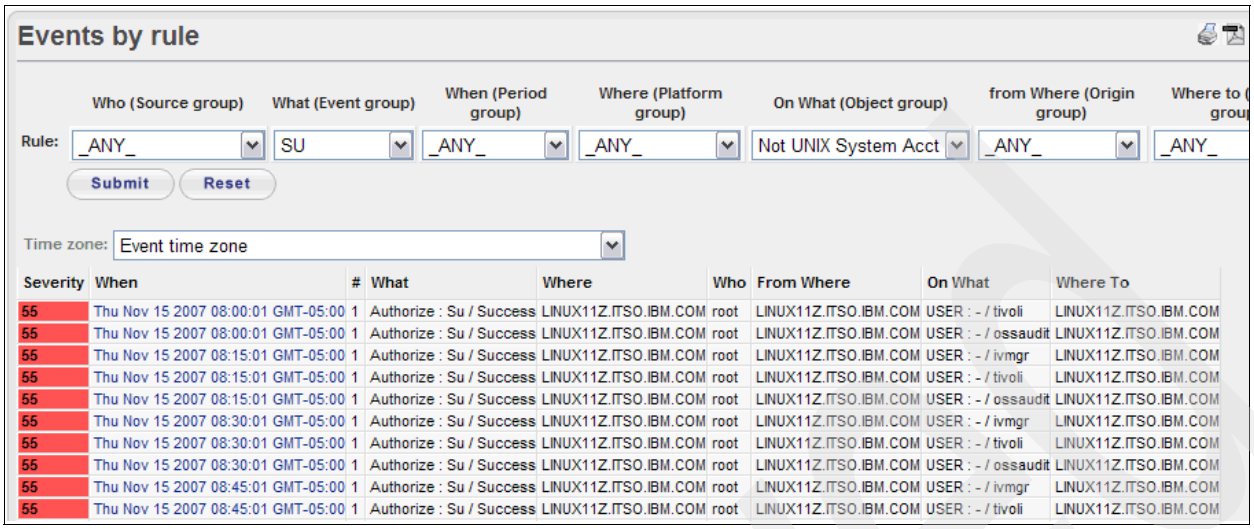
Under the OnWhat branch there is a *group* called UsysAct that contains the accounts that we have defined as UNIX system accounts. Each one of the *conditions* under this grouping is joined with an OR. Each of these conditions contains a single requirement of object name is *name*. So the group UsysAct is Object name is db2iadm1 OR Object name is idsldap OR Object name is db2inst1, and so on.

Under this is a group called Not UNIX System Acct. This contains a single condition of not UsysAct USER. This condition has two requirements:

- Not in group UsysAct
- Object type is USER

Requirements are joined with an AND. So the grouping statement is Object type is USER and user not in group UsysAct.

These policy rules can now be used to filter the display in the TCIM iView. Figure 9-27 shows some records with the SU What rule and the Not UNIX System Accts On what rule.



Severity	When	#	What	Where	Who	From Where	On What	Where To
55	Thu Nov 15 2007 08:00:01 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	USER : - / tivoli	LINUX11Z.ITSO.IBM.COM
55	Thu Nov 15 2007 08:00:01 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	USER : - / ossaudit	LINUX11Z.ITSO.IBM.COM
55	Thu Nov 15 2007 08:15:01 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	USER : - / ivmgr	LINUX11Z.ITSO.IBM.COM
55	Thu Nov 15 2007 08:15:01 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	USER : - / tivoli	LINUX11Z.ITSO.IBM.COM
55	Thu Nov 15 2007 08:15:01 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	USER : - / ossaudit	LINUX11Z.ITSO.IBM.COM
55	Thu Nov 15 2007 08:30:01 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	USER : - / ivmgr	LINUX11Z.ITSO.IBM.COM
55	Thu Nov 15 2007 08:30:01 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	USER : - / tivoli	LINUX11Z.ITSO.IBM.COM
55	Thu Nov 15 2007 08:30:01 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	USER : - / ossaudit	LINUX11Z.ITSO.IBM.COM
55	Thu Nov 15 2007 08:45:01 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	USER : - / ivmgr	LINUX11Z.ITSO.IBM.COM
55	Thu Nov 15 2007 08:45:01 GMT-05:00	1	Authorize : Su / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	USER : - / tivoli	LINUX11Z.ITSO.IBM.COM

Figure 9-27 TCIM Events by Rule report with custom policy applied

Note that the report still contains system accounts, such as tivoli and ivmgr. You would need to further tune the report to add these to the list of system accounts.

You can see that the severity for these events is 55. This has been set by creating an attention rule, as shown in Figure 9-26 on page 266.

Custom policy - highlight access to the TAMOS audit files

The next condition that we wanted to highlight was when someone accesses the TAMOS audit files. TAMOS tightly controls the files, but you might want to monitor access against them.

The policy rule that we want to apply shows all access to files in the /var/pdos/audit directory.

Using the same approach as in the previous section, we define two conditions:

- What; Event main class contains Access.
- OnWhat; Object name starts with /vaer/pdos/audit.

The implementation of these into TCIM groups, conditions, and requirements is shown in Figure 9-28.

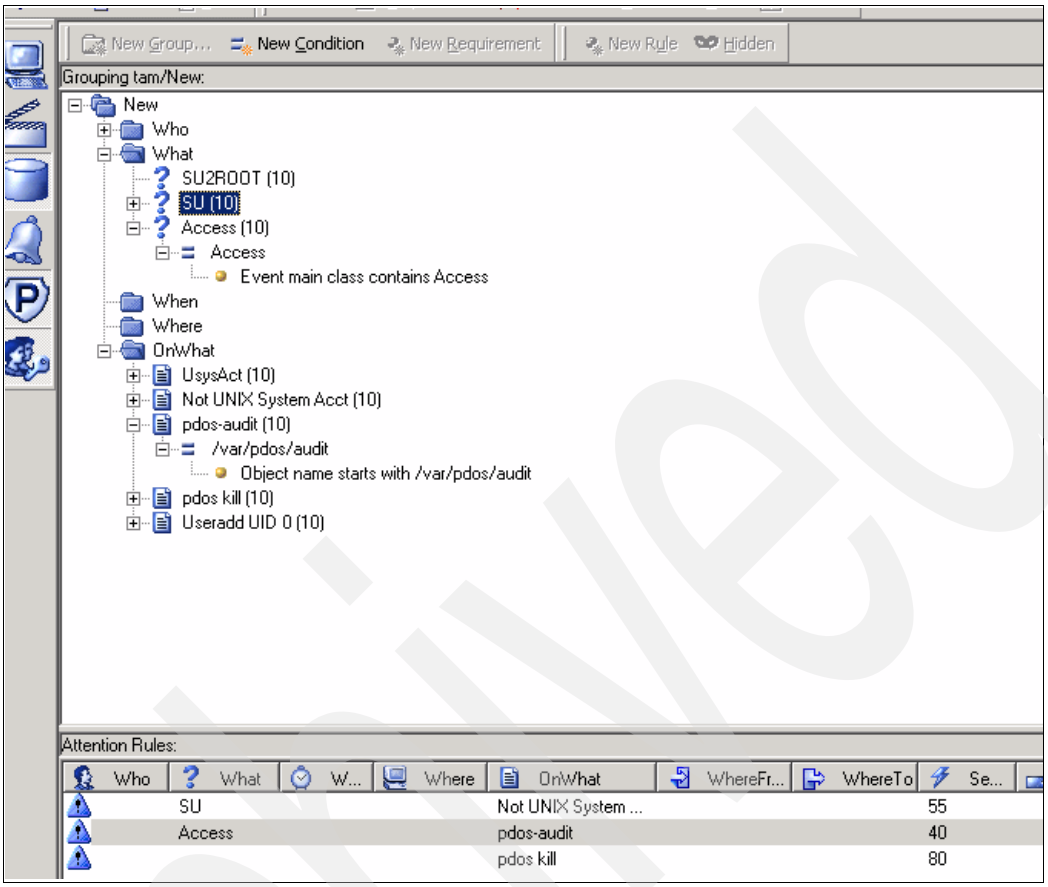
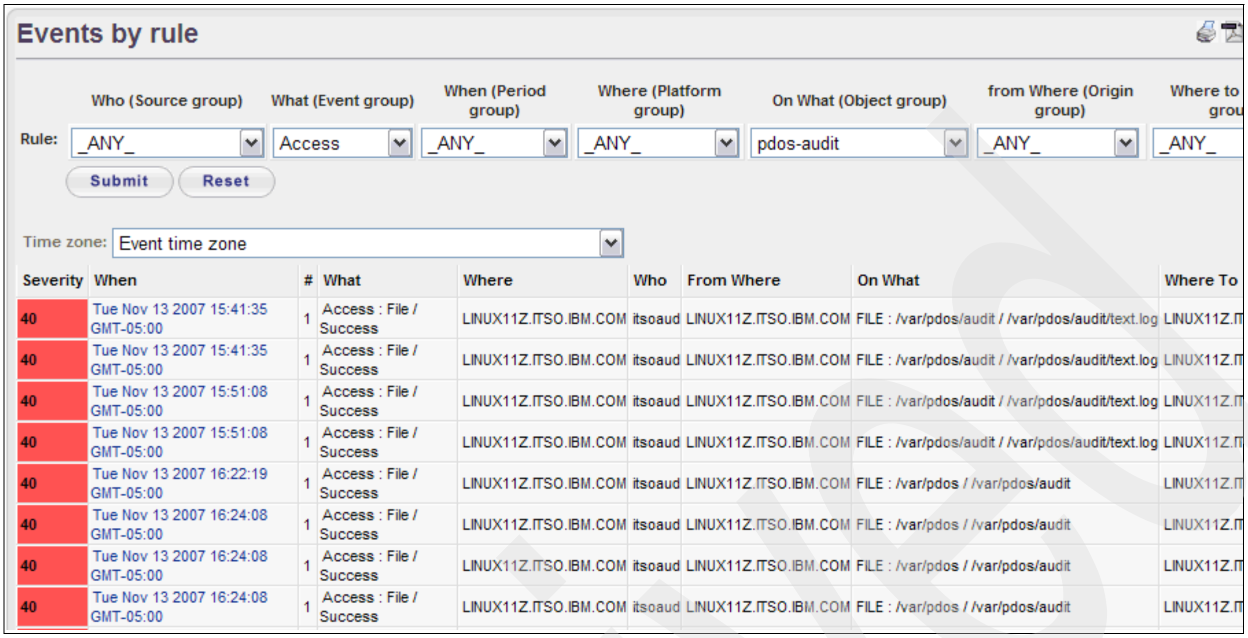


Figure 9-28 Grouping conditions for access to TAMOS audit files policy

There is only a single requirement for each condition and a single condition for each group. Figure 9-29 shows these filter rules applied in the TCIM iView Events by Type report.



Events by rule

	Who (Source group)	What (Event group)	When (Period group)	Where (Platform group)	On What (Object group)	from Where (Origin group)	Where to grou
Rule:	_ANY_	Access	_ANY_	_ANY_	pdos-audit	_ANY_	_ANY_

Time zone: Event time zone

Severity	When	#	What	Where	Who	From Where	On What	Where To
40	Tue Nov 13 2007 15:41:35 GMT-05:00	1	Access : File / Success	LINUX11Z.ITSO.IBM.COM	itsoaud	LINUX11Z.ITSO.IBM.COM	FILE : /var/pdos/audit / /var/pdos/audit/text.log	LINUX11Z.I
40	Tue Nov 13 2007 15:41:35 GMT-05:00	1	Access : File / Success	LINUX11Z.ITSO.IBM.COM	itsoaud	LINUX11Z.ITSO.IBM.COM	FILE : /var/pdos/audit / /var/pdos/audit/text.log	LINUX11Z.I
40	Tue Nov 13 2007 15:51:08 GMT-05:00	1	Access : File / Success	LINUX11Z.ITSO.IBM.COM	itsoaud	LINUX11Z.ITSO.IBM.COM	FILE : /var/pdos/audit / /var/pdos/audit/text.log	LINUX11Z.I
40	Tue Nov 13 2007 15:51:08 GMT-05:00	1	Access : File / Success	LINUX11Z.ITSO.IBM.COM	itsoaud	LINUX11Z.ITSO.IBM.COM	FILE : /var/pdos/audit / /var/pdos/audit/text.log	LINUX11Z.I
40	Tue Nov 13 2007 16:22:19 GMT-05:00	1	Access : File / Success	LINUX11Z.ITSO.IBM.COM	itsoaud	LINUX11Z.ITSO.IBM.COM	FILE : /var/pdos / /var/pdos/audit	LINUX11Z.I
40	Tue Nov 13 2007 16:24:08 GMT-05:00	1	Access : File / Success	LINUX11Z.ITSO.IBM.COM	itsoaud	LINUX11Z.ITSO.IBM.COM	FILE : /var/pdos / /var/pdos/audit	LINUX11Z.I
40	Tue Nov 13 2007 16:24:08 GMT-05:00	1	Access : File / Success	LINUX11Z.ITSO.IBM.COM	itsoaud	LINUX11Z.ITSO.IBM.COM	FILE : /var/pdos / /var/pdos/audit	LINUX11Z.I
40	Tue Nov 13 2007 16:24:08 GMT-05:00	1	Access : File / Success	LINUX11Z.ITSO.IBM.COM	itsoaud	LINUX11Z.ITSO.IBM.COM	FILE : /var/pdos / /var/pdos/audit	LINUX11Z.I

Figure 9-29 TCIM Events by Rule report with custom policy applied

An Attention Rule was built with the combination of the Access What group and the pdos-audit On What group with a severity of 40.

Custom policy - highlight use of the TAMOS process kill command

The next condition that we wanted to highlight was when someone uses the TAMOS kill command (`pdosctl -k`). For normal operation there should be no need to kill the processes, so this situation is worth highlighting.

The policy rule that we want to apply is, show all execution of **`pdosctl -k`**. This only required a single OnWhat condition where the Object name contains pdosctl -k.

The implementation of these into TCIM groups, conditions, and requirements is shown in Figure 9-30.

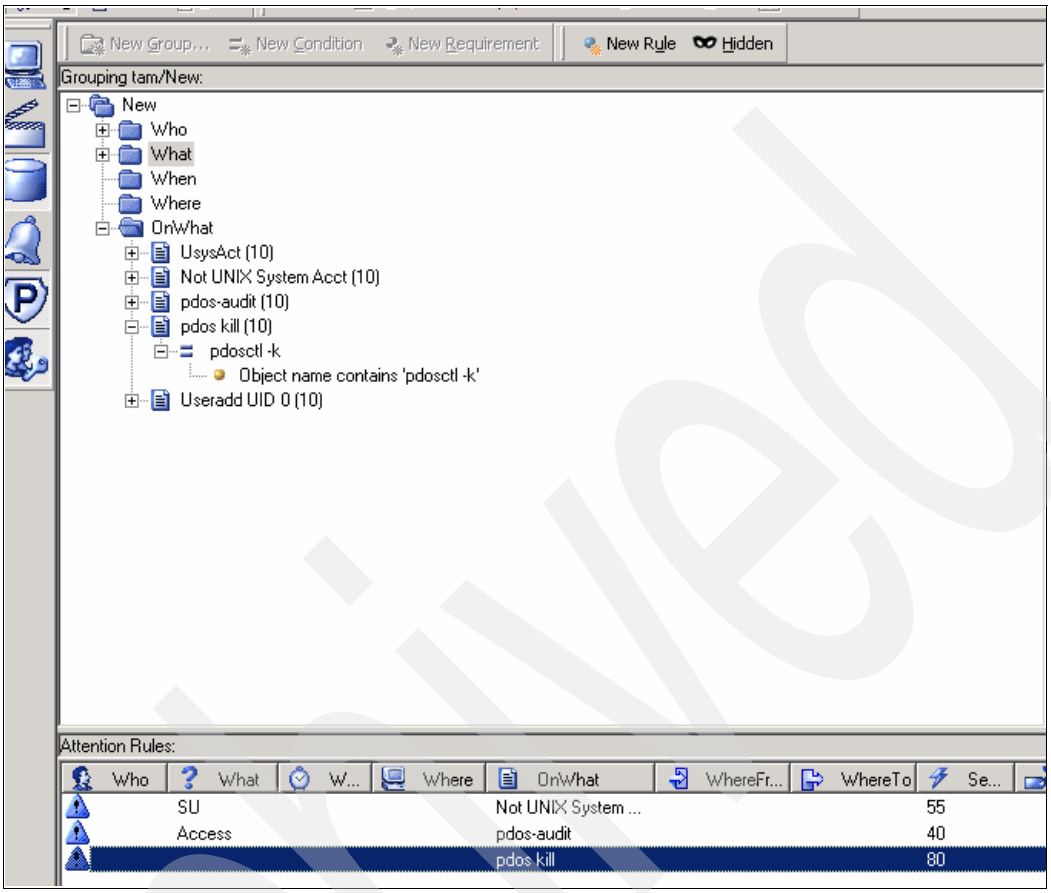
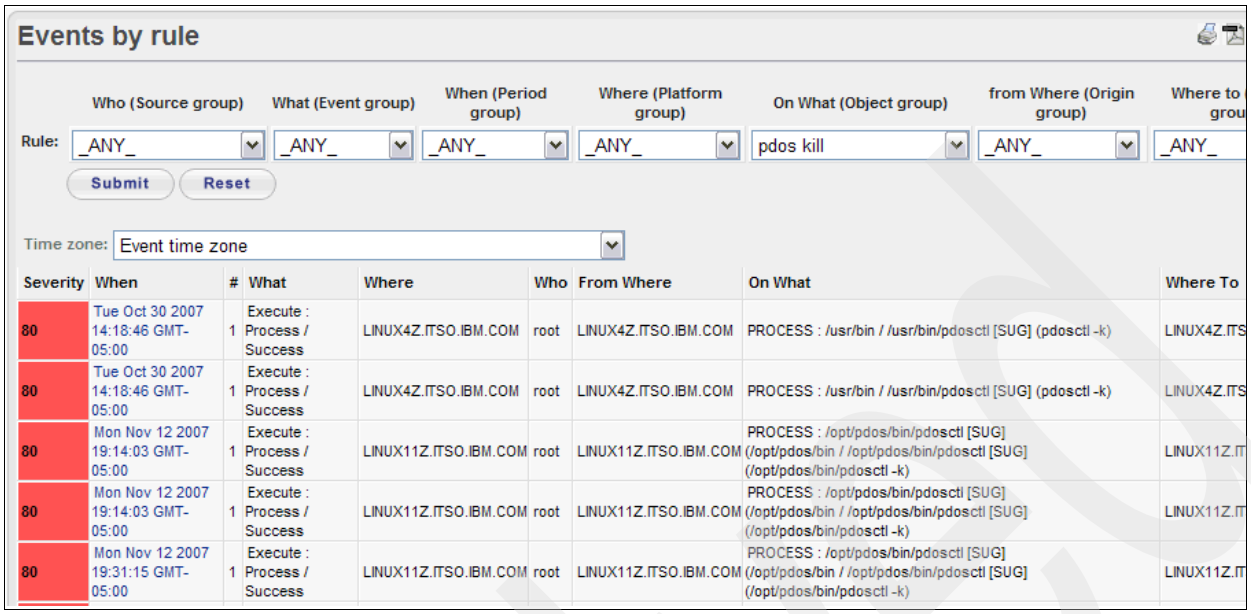


Figure 9-30 Grouping conditions for running the TAMOS kill command policy

The single requirement of Object name contains “pdosctl -k” under the pdos kill group. Note that this requires the command to have a single space between the “pdosctl” and “-k” strings. You could also code two separate requirements under the same condition with one for Object name contains pdosctl and the other for Object name contains -k.

The TCIM Events by Rule report with this grouping applied is shown in Figure 9-31.



Events by rule

	Who (Source group)	What (Event group)	When (Period group)	Where (Platform group)	On What (Object group)	from Where (Origin group)	Where to grou
Rule:	_ANY_	_ANY_	_ANY_	_ANY_	pdos kill	_ANY_	_ANY_

Submit Reset

Time zone: Event time zone

Severity	When	#	What	Where	Who	From Where	On What	Where To
80	Tue Oct 30 2007 14:18:46 GMT-05:00	1	Execute : Process / Success	LINUX4Z.ITSO.IBM.COM	root	LINUX4Z.ITSO.IBM.COM	PROCESS : /usr/bin / /usr/bin/pdosctl [SUG] (pdosctl -k)	LINUX4Z.ITS
80	Tue Oct 30 2007 14:18:46 GMT-05:00	1	Execute : Process / Success	LINUX4Z.ITSO.IBM.COM	root	LINUX4Z.ITSO.IBM.COM	PROCESS : /usr/bin / /usr/bin/pdosctl [SUG] (pdosctl -k)	LINUX4Z.ITS
80	Mon Nov 12 2007 19:14:03 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	PROCESS : /opt/pdos/bin/pdosctl [SUG] (/opt/pdos/bin / /opt/pdos/bin/pdosctl [SUG] (/opt/pdos/bin/pdosctl -k)	LINUX11Z.IT
80	Mon Nov 12 2007 19:14:03 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	PROCESS : /opt/pdos/bin/pdosctl [SUG] (/opt/pdos/bin / /opt/pdos/bin/pdosctl [SUG] (/opt/pdos/bin/pdosctl -k)	LINUX11Z.IT
80	Mon Nov 12 2007 19:31:15 GMT-05:00	1	Execute : Process / Success	LINUX11Z.ITSO.IBM.COM	root	LINUX11Z.ITSO.IBM.COM	PROCESS : /opt/pdos/bin/pdosctl [SUG] (/opt/pdos/bin / /opt/pdos/bin/pdosctl [SUG] (/opt/pdos/bin/pdosctl -k)	LINUX11Z.IT

Figure 9-31 TCIM Events by Rule report with custom policy applied

An Attention Rule was built with the combination of the Access What group and the pdos-audit On What group with a severity of 80.

Custom policy - highlight creation of UID=0 users

The next condition that we wanted to highlight was when someone creates a user using the **`useradd`** command and sets the UID to be zero. This is an exception condition that should be highlighted.

The policy rule that we want to apply is, show all execution of **`useradd`** where the argument “-u 0” is entered. So there are three conditions required:

- OnWhat; Object type is PROCESS
- OnWhat; Object name contains useradd
- OnWhat; Object name contains “-u 0”

The implementation of these conditions into a TCIM group is shown in Figure 9-32.

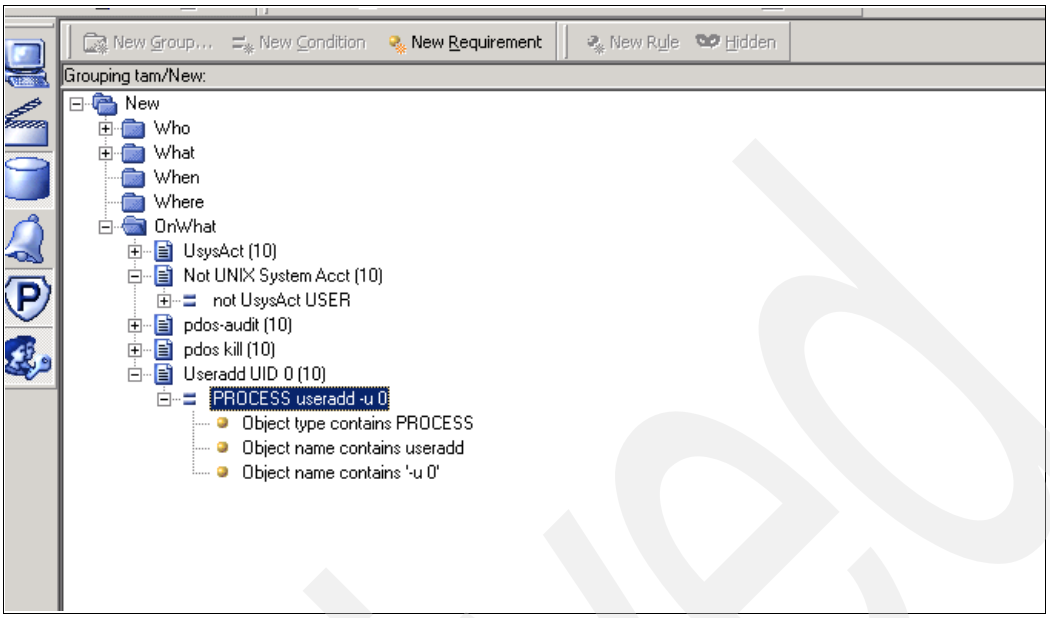


Figure 9-32 Grouping conditions for executing a useradd with UID=0 policy

There is only a single condition under the group. This condition contains three requirements, which are joined with an AND.

The TCIM Events by Rule report with this grouping applied is shown in Figure 9-33.

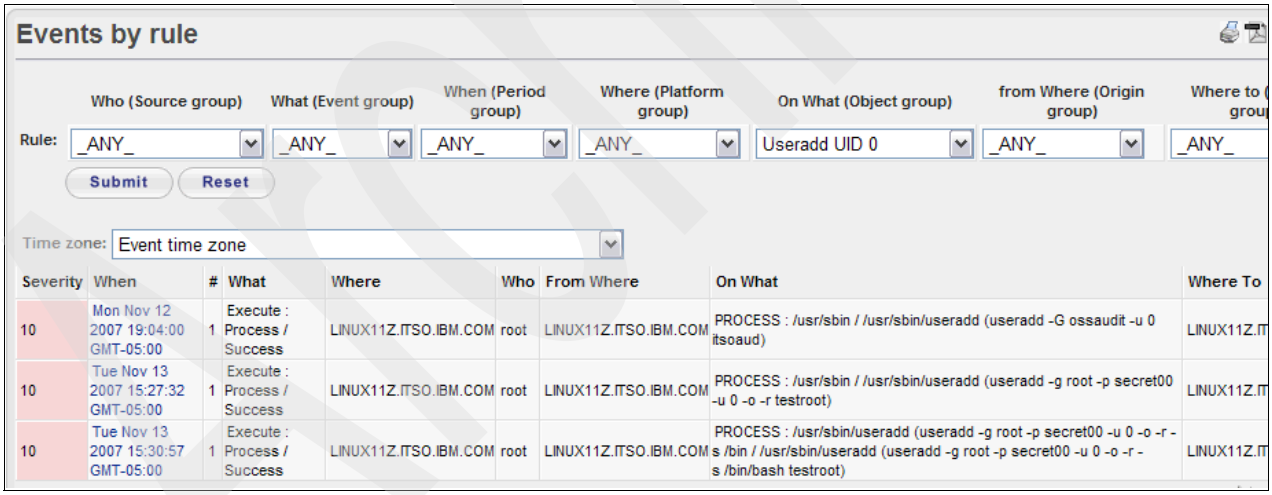


Figure 9-33 TCIM Events by Rule report with custom policy applied

In this example we did not set an attention rule.

These four examples have shown how to implement custom policy for the TAMOS audit events. The approach can be used for any type of event for any source. One of the key points that we wanted to highlight with these examples was the way to implement the OR and AND conditions within the groups, conditions, and requirements.

9.6 z/OS privilege users scenarios

The following scenarios show samples of how the z/OS privileged users can be audited.

9.6.1 Environment setup

The following tasks were performed to set up the z/OS environment and perform the following auditing scenarios:

- Make sure that all the needed SMF records are cut (SMFPRMnn). RACF uses records 80, 81, 82, and 83.
- Make sure to set the attributes in under SYSTEM options in RACF to: SAUDIT CMDVIOL OPERAUDIT

For more background on these use cases, see 7.2.3, “Privileged user scenarios” on page 151. It details the type of privileges that a RACF account can have and the security concerns each represents.

9.6.2 User with attribute system special test case

In this scenario, the RACF user PTST01, with system special privilege, is providing the user PTST02 with the operation privilege. PTST02 will access a special protected resource (HANNE.PROD) using the new acquired privilege.

Performing the test case

For the test case:

1. We created a group called Payroll under the onwhat category in TCIM and assigned the HANNE.PROD to it.

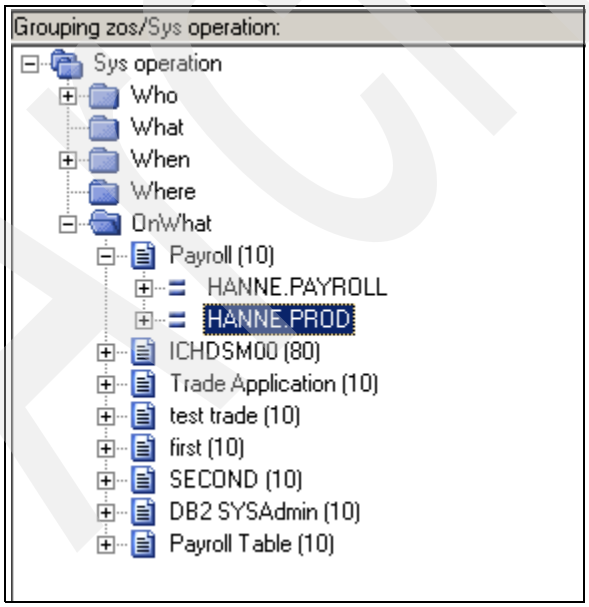


Figure 9-34 Defining a group in TCIM with the HANNE.PROD data set

2. We defined a RACF user with System Special, PTST01, and set audit ALL.

3. We log on using PTST01 (with RACF Special) and verify that all the activities performed by this use are audited and the audit all is set on.
4. We define the new user, PTST02. The resulting audit record from SMF has been sent and analyzed with TCIM, as shown in Figure 9-35.

Severity	1		
When	Tue Nov 13 2007 12:06:15 GMT-05:00	Other Periods (10)	
What	Create : User / Success	Other Events (10)	
Where	SC60 (z/OS)	z/OS platform (10)	
Who	SYSTEM SPECIAL (SC60.PTST01)	Test (10) SC60.RACF Special - Production (10)	More...
From Where	USIBMSC.SC60TC02 (LU)	Other Platforms (10)	
On What	SAF:USER : - / PTST02	Other Objects (10)	
Where To	SC60 (z/OS)	z/OS platform (10)	

▼ Incident Tracking

This event is not tracked yet. Its unique event-id is: WTSC60.ITSO.IBM.COM.125\JJNGRJ1.9042

1) Click here to open the ticketing service. (change)

2) Copy the event information into the ticket.

```
Event Id   : WTSC60.ITSO.IBM.COM.125\JJNGRJ1.9042
When       : Tue Nov 13 2007 12:06:15 GMT-05:00 - Other Periods
fromWhere  : USIBMSC.SC60TC02 (LU)              - Other Platforms
Who        : SC60.PTST01 (SYSTEM SPECIAL)       - Test
Where      : SC60 (z/OS)                        - z/OS platform
What       : Create: User / Success             - Other Events
WhereTo    : SC60 (z/OS)                        - z/OS platform
onWhat     : SAF:USER: - / PTST02               - Other Objects
```

Figure 9-35 User PTST01 with System SPECIAL privilege is defining user PTST02

5. We assigned operations privilege to the PTST02 user, as shown in Figure 9-36.

Incident Tracking

This event is not tracked yet. Its unique event-id is: WTSC60.ITSO.IBM.COM.125\JJNGRJ1.9053

1) Click here to open the ticketing service. (change)

2) Copy the event information into the ticket.

```
Event Id   : WTSC60.ITSO.IBM.COM.125\JJNGRJ1.9053
When       : Tue Nov 13 2007 12:06:28 GMT-05:00 - Other Periods
fromWhere  : USIBMSC.SC60TC02 (LU)            - Other Platforms
Who        : SC60.PTST01 (SYSTEM SPECIAL)     - Test
Where      : SC60 (z/OS)                      - z/OS platform
What       : Modify: User / Success           - Other Events
WhereTo    : SC60 (z/OS)                      - z/OS platform
onWhat     : SAF:USER: - / PTST02             - Other Objects
```

3) Enter the ticket number and press [submit].

Submit

Click here to see all incidents.

Additional information

Aspect	Value
Event :: logrecordtype	SMF,80-13
Event :: command	ALTUSER PTST02 OPERATIONS
Event :: recorddesc	RACF ALTUSER success for PTST01: ALTUSER PTST02
Who :: originator	SC60.PTST01

Figure 9-36 User PTST01 issued the ALTER command to provide PTST02 user with OPERATION privilege

6. Set AUDIT = YES on PTST02 user.

Incident Tracking

This event is not tracked yet. Its unique event-id is: WTSC60.ITSO.IBM.COM.125\JJNGRJ1.9811

1) Click here to open the ticketing service. (change)

2) Copy the event information into the ticket.

```
Event Id   : WTSC60.ITSO.IBM.COM.125\JJNGRJ1.9811
When       : Tue Nov 13 2007 12:10:16 GMT-05:00 - Other Periods
fromWhere  : USIBMSC.SC60TC02 (LU)            - Other Platforms
Who        : SC60.PTST01 (SYSTEM SPECIAL)     - Test
Where      : SC60 (z/OS)                      - z/OS platform
What       : Modify: User / Success           - Other Events
WhereTo    : SC60 (z/OS)                      - z/OS platform
onWhat     : SAF:USER: - / PTST02             - Other Objects
```

3) Enter the ticket number and press [submit].

Submit

Click here to see all incidents.

Additional information

Aspect	Value
Event :: logrecordtype	SMF,80-13
Event :: recorddesc	RACF ALTUSER success for PTST01: ALTUSER PTST02
Event :: command	ALTUSER PTST02 TSO(MSGCLASS(X)) TSO(ACCTNUM(MVS)) TSO(PROC(BPXPROC)) TSO(SIZE(2000000)) TSO(UNIT(SYSDA)) TSO(COMMAND(ISPPDF))
Who :: originator	SC60.PTST01

Figure 9-37 Turning AUDIT on PTST02

7. Set Audit all(READ) in RACF on HANNE.PROD resource.
8. Log on as the PTST02 user and edit the HANNE.PROD data set. As you can see in Figure 9-38, the event can be visible through TCIM reporting and can be used for auditing special users.

Severity	1		
When	Tue Nov 13 2007 12:06:15 GMT-05:00	Other Periods (10)	
What	Create : User / Success	Other Events (10)	
Where	SC60 (z/OS)	z/OS platform (10)	
Who	SYSTEM SPECIAL (SC60.PTST01)	Test (10) SC60.RACF Special - Production (10)	More...
From Where	USIBMSC.SC60TC02 (LU)	Other Platforms (10)	
On What	SAF:USER : - / PTST02	Other Objects (10)	
Where To	SC60 (z/OS)	z/OS platform (10)	

▼ Incident Tracking

This event is not tracked yet. Its unique event-id is: WTSC60.ITSO.IBM.COM.125\JJNGRJ1.9042

1) Click here to open the ticketing service. (change)

2) Copy the event information into the ticket.

```
Event Id   : WTSC60.ITSO.IBM.COM.125\JJNGRJ1.9042
When       : Tue Nov 13 2007 12:06:15 GMT-05:00 - Other Periods
fromWhere  : USIBMSC.SC60TC02 (LU)            - Other Platforms
Who        : SC60.PTST01 (SYSTEM SPECIAL)     - Test
Where      : SC60 (z/OS)                      - z/OS platform
What       : Create: User / Success           - Other Events
WhereTo    : SC60 (z/OS)                      - z/OS platform
onWhat     : SAF:USER: - / PTST02             - Other Objects
```

Figure 9-38 PTST02 is accessing Hanne.prod

9.6.3 Access to resource provided by special privilege user test case

In this scenario, the RACF user PTST01, with system special privilege, is providing a standard user, ZUE, with the access to view a protected resource, HANNE.PAYROLL data set.

Performing the test case

For this test case:

1. Define ‘HANNE.PAYROLL’ in RACF with UACC NONE and AUDIT(READ).
2. Set audit(read) both failures and success in RACF on HANNE.PAYROLL.
3. Create a group called Payroll under the onwhat category in TCIM and assign the HANNE.PAYROLL to it.

4. Create a group called `non-business hours` under the when category in TCIM and define under this group the time between 13:00 PM to 8:00 AM the next business day.

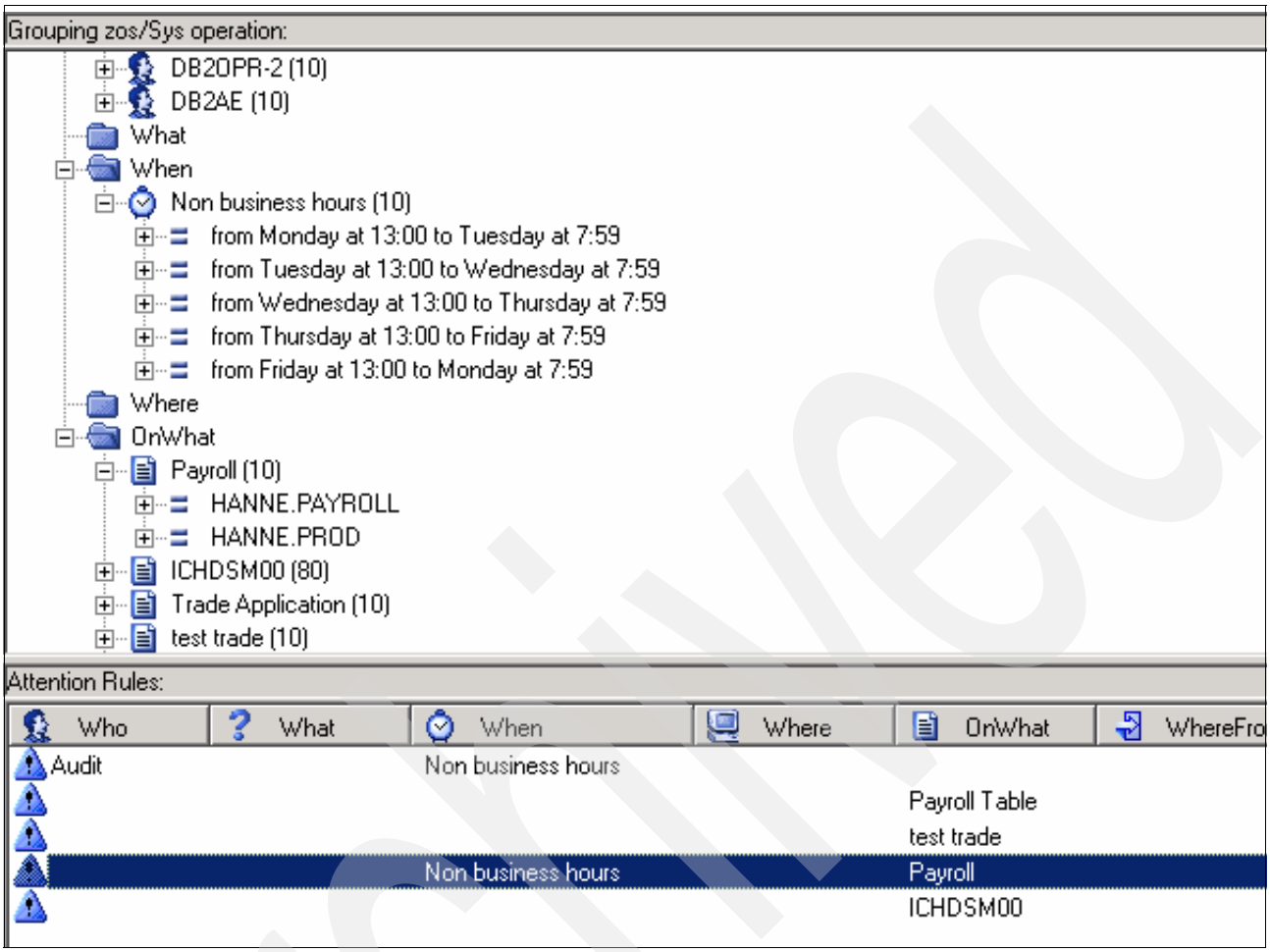


Figure 9-39 Group and role definition in TCIM

5. We defined an attention rule in TCIM that will notify you when someone is accessing HANNE.PAYROLL during non-business hours. As you can see in Figure 9-40, the access to the HANNE.PAYROLL data set during non-business hours has been flagged red in the TCIM report.

Severity	Date / Time	What (detail)	Who (detail)	On what (detail)
65	Tue Nov 13 2007 12:12:49 GMT-05:00	Grant : Profile / Failure	SYSTEM OPERATIONS	SAF:DATASET : - / HANNE.PAYROLL
1	Tue Nov 13 2007 12:13:34 GMT-05:00	Grant : Profile / Success	SYSTEM SPECIAL	SAF:DATASET : - / HANNE.PAYROLL
1	Tue Nov 13 2007 12:14:47 GMT-05:00	Grant : Profile / Success	SYSTEM SPECIAL	SAF:DATASET : - / HANNE.PAYROLL
10	Tue Nov 13 2007 12:17:11 GMT-05:00	Read : File / Success	MIN M ZHU	SAF:DATASET : - / HANNE.PAYROLL
10	Tue Nov 13 2007 12:17:14 GMT-05:00	Read : File / Success	MIN M ZHU	SAF:DATASET : - / HANNE.PAYROLL
70	Tue Nov 13 2007 14:28:54 GMT-05:00	Read : File / Success	MIN M ZHU	SAF:DATASET : - / HANNE.PAYROLL
70	Tue Nov 13 2007 14:28:56 GMT-05:00	Read : File / Success	MIN M ZHU	SAF:DATASET : - / HANNE.PAYROLL
70	Tue Nov 13 2007 14:28:58 GMT-05:00	Read : File / Success	MIN M ZHU	SAF:DATASET : - / HANNE.PAYROLL

Figure 9-40 The result of the attention rule that we defined in TCIM

6. Using ID PTST01, which has the audit flag sat on, we gave ZHU read access to *HANNE.PAYROLL'.

Event Detail

	Field	Group
Severity	1	
When	Tue Nov 13 2007 12:14:47 GMT-05:00	Other Periods (10)
What	Grant : Profile / Success	Other Events (10)
Where	SC60 (z/OS)	z/OS platform (10)
Who	SYSTEM SPECIAL (SC60.PTST01)	Test (10) SC60.RACF Special - Production (10) More...
From Where	USIBMSC.SC60TC02 (LU)	Other Platforms (10)
On What	SAF:DATASET : - / HANNE.PAYROLL	Payroll (10)
Where To	SC60 (z/OS)	z/OS platform (10)

▶ Incident Tracking

▼ Additional information

Aspect	Value
Event :: logrecordtype	SMF,80-19
Event :: recorddesc	RACF PERMIT success for PTST01: PERMIT DATASET HANNE.PAYROLL
Event :: command	PERMIT 'HANNE.PAYROLL' ACCESS(READ) CLASS(DATASET) GENERIC ID(ZHU)
Who :: originator	SC60.PTST01

Figure 9-41 User PTST01 gave read access to user ZHU

7. Using ID ZHU, the data set HANNE.PAYROLL was browsed after 13.00. As you can see in Figure 9-42, the exception is still valid since the user ZHU is trying to browse the resource during non-business hours.

	Field	Group	
Severity	70	This is a policy exception. This is a special attention event.	
When	Tue Nov 13 2007 14:28:54 GMT-05:00	Non business hours (10)	
What	Read : File / Success	Other Events (10)	
Where	SC60 (z/OS)	z/OS platform (10)	
Who	MIN M ZHU (SC60.ZHU)	ZHU (10) SC60.RACF Special - Production (10)	More...
From Where	USIBMSC.SC60TC06 (LU)	Other Platforms (10)	
On What	SAF:DATASET : - / HANNE.PAYROLL	Payroll (10)	
Where To	SC60 (z/OS)	z/OS platform (10)	

▸ Incident Tracking

▾ Additional information

Aspect	Value
Event :: logrecordtype	SMF,80-2
Event :: miscellaneous	JOBNAME:ZHU
Event :: recorddesc	RACF ACCESS success for ZHU: (READ,READ) on DATASET HANNE.PAYROLL
Who :: originator	SC60.ZHU
On What :: sysplex	PLEX60

Figure 9-42 User ZHU accessed the HAMME.PAYROLL data set

9.6.4 Operation privilege user test case

In this scenario, the RACF user PTST02 with operation privilege will edit a data set, HANNE.PROD, without having access through the access list.

Performing the test case

For this test case:

1. Define 'HANNE.PROD' in RACF with UACC NONE and AUDIT(READ).
2. Set audit(read) both failures and success in RACF on HANNE.PROD.
3. We created a group called Payroll under the onwhat category in TCIM and assigned the HANNE.PROD to it.
4. User PTST02 is defined with the OPERATIONS attribute and AUDIT is sat on the user in RACF.
5. We created a group called SYSTEM OPERATION under the who category in TCIM and assigned the PTST02 to it.
6. PTST02 logs onto the system and goes into edit mode on HANNE.PROD and edits the data.

As you can see in Figure 9-43, TCIM is reporting the fact that PTST02 with the SYSTEM OPERATION privilege has browsed and edited the HANNE.PROD data set.

70	Tue Nov 13 2007 16:25:17 GMT-05:00	Read : File / UserPriv	SC60.PTST02	SYSTEM OPERATIONS	SAF:DATASET HANNE.PROD
70	Tue Nov 13 2007 16:25:32 GMT-05:00	Write : File / UserPriv	SC60.PTST02	SYSTEM OPERATIONS	SAF:DATASET HANNE.PROD

Figure 9-43 TCIM reporting for PTST02 user

Important: To avoid this scenario, you can define the RACF rule for your sensitive data, with access=none to users with the Operation attribute.

9.6.5 Audit privilege test case

In this scenario, the RACF user PTST03, with the audit privilege, will execute the RACF audit program ICHDSM00 (the DSMON program), which is protected through the RACF Program class.

Performing the test case

For this test case:

1. Using PTST01, define a RACF user, PTST03, with AUDIT privilege.
2. Run the DSMON audit report.
3. We created a group called ICHDSM00 under the onwhat category in TCIM and assigned the ICHDSM00 program to it.

4. We created a group called AUDIT under the who category in TCIM and assigned the PTST03 to it.

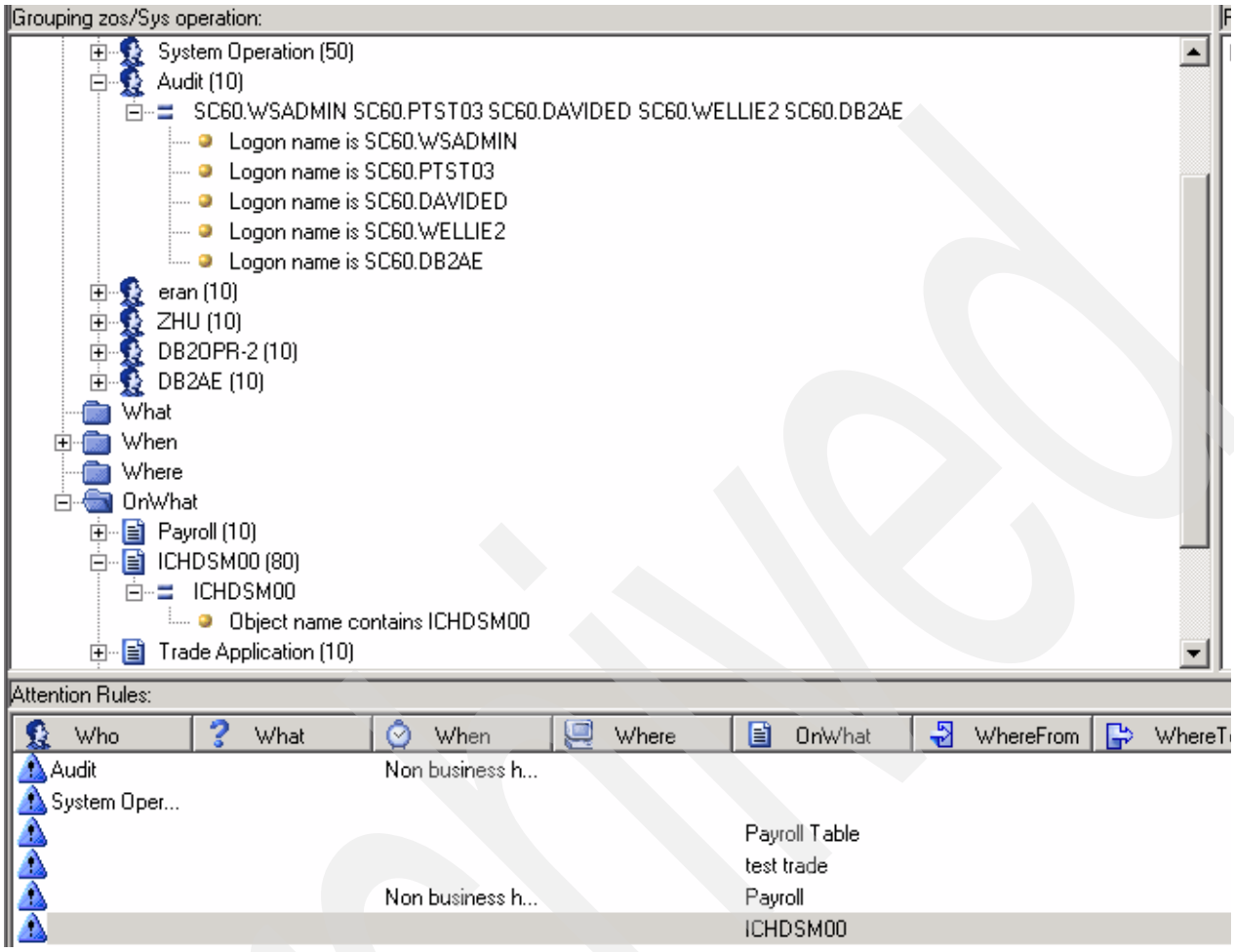


Figure 9-44 Group and role definition in TCIM

5. We defined a profile in the program class for program ICHDSM00 (the DSMON program), with no access list, using the listed command:

   ```
   RDEF Program ICHDSM00 addmem('SYS1.LINKLIB'//NOPADCHK) UACC(NONE) OWNER(SYS1)
   AUDIT(FAILURES(READ)
   ```

6. We submitted a job executing the ICHDSM00 program. As you can see in Figure 9-45, TCIM is showing the violation for the PTST03 user.

30	Tue Nov 13 2007 12:17:41 GMT-05:00	Read : Module / Failure	SC60.PTST03	SYSTEM AUDITOR	SAF:PROGRAM : - / ICHDSM00

Figure 9-45 TCIM reporting that PTST03 has tried to execute the ICHDSM00 program

7. We then added PTST03 to the access list to allow the user to execute the program.
8. We submitted a job executing the ICHDSM00 program. As you can see in Figure 9-46, the user PTST03 was now able to execute the program with success.

30	Tue Nov 13 2007 17:02:59 GMT-05:00	Read : Module / Success	SC60.PTST03	SYSTEM AUDITOR	SAF:PROGRAM ICHDSM00

Figure 9-46 TCIM reporting that PTST03 has tried to execute the ICHDSM00 program and succeeded

9.7 DB2 database administrator scenarios

In this section we analyze different scenario depending on where authorization is performed for a DB2 subsystem.

9.7.1 RACF DB2 External Security Auditing using TCIM and SMF Type 80

In this section we discuss RACF DB2 External Security Auditing using TCIM and SMF Type 80.

Environment setup

The following tasks have ben performed on our system to prepare the environment:

- Create users in RACF called ZHU, DB2OPR, and DB2AE.
- Set AUDIT to YES on ZHU,DB2OPR, DB2AE.

  Make sure that all DB2 IDs with privileges (SYSAdmin, sysctlr, sysoperations, and so on) are audited.

  We will use an ID with DB2 SYSADMIN for our use case.
- Connect all users with different privileges to the different groups defined for RACF Control of DB2.
- Create all DB2 RACF profiles (in all RACF DB2 classes) and set to WARNING.

  For all connects to the DB2 subsystem from distributed sources we activated the DSNR class in RACF. In our test scenarios, we connect to the DB8L subsystem.

For DB2 DSNR:

- Activate the DSNR class.
- Define the profile *DB8L*.DIST, where *DB8L* is the name of your subsystem.

  The active RACF Class DSNR will protect DB2 resources on the mainframe from all connections from distributed sources. We put a warning on profile DB8L.DIST for db2 connection auditing.

RACF WARNING on unauthorized DB2 object access test case

An unprivileged user (DB2OPR) tries to access a table with INSUFFICIENT AUTHORITY. RACF generates the warning and records it in SMF. TCIM shows a consolidated picture for a quick and clear understanding of what happened in the system.

Performing the test case

To perform the test case:

1. Define user DB2OPR in RACF as unprivileged DB2 User.

   Add “DB8L.AETEST.TEST01” and “DB8l.AETEST.TEST02“ in the Payroll table in the On What item of the TCIM policy.

2. Define attention rules with payroll table as severity 75.

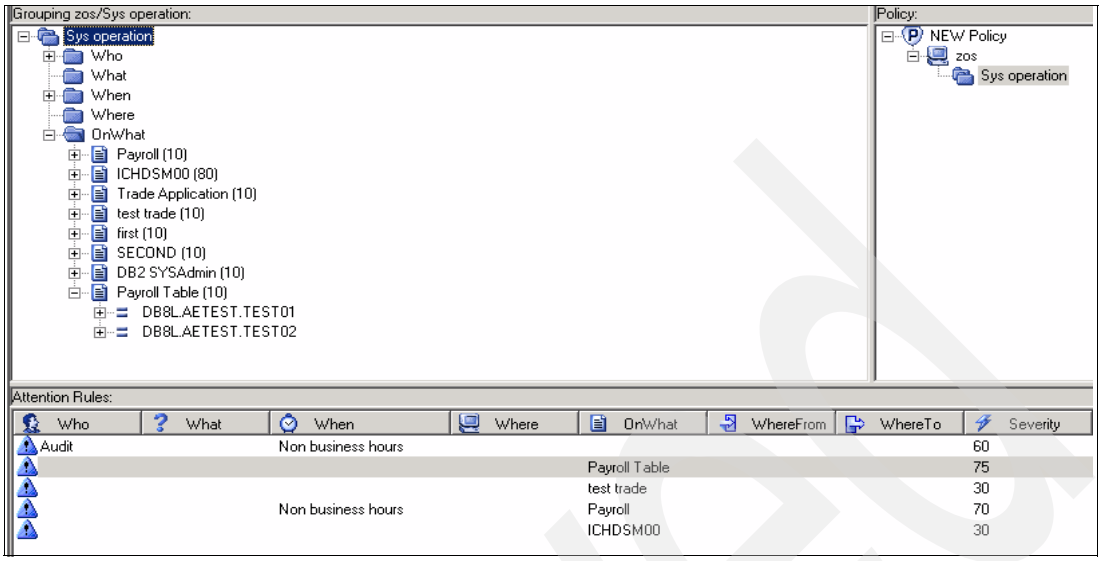


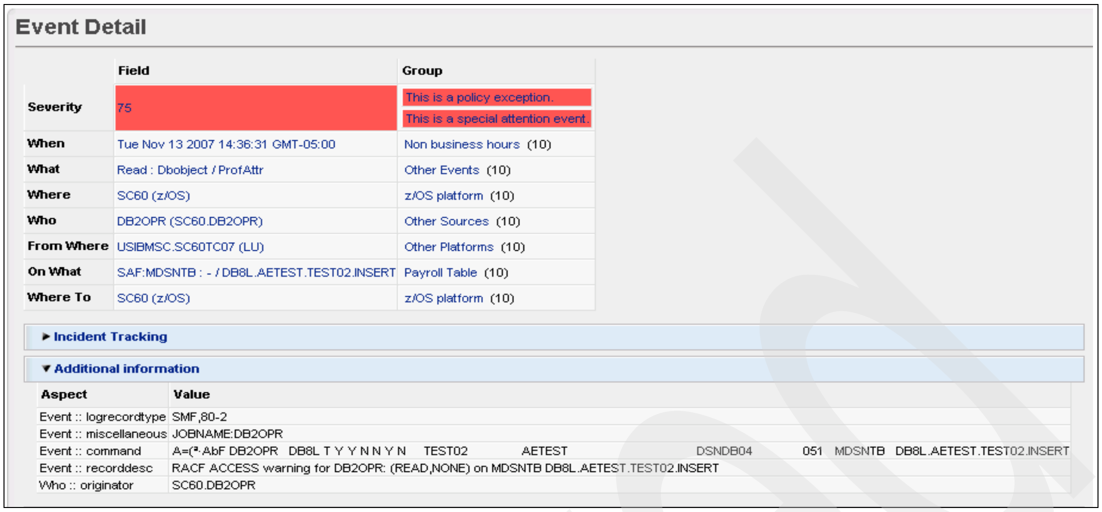
Figure 9-47 Definition of attention rules

3. We access the table AETEST.TEST02 using the DB2OPR user and INSERT few records.
4. The operation is not allowed and RACF generates a warning message for unauthorized access that can be seen on the z/OS syslog.
5. We check the TCIM portal by attention events.

Severity	Date / Time	#	What (detail)	Where (detail)	Who (detail)	Where from (detail)	On what (detail)	Where to (detail)
30	Tue Nov 13 2007 12:17:41 GMT-05:00	1	Read : Module / Failure	SC60 (z/OS)	SYSTEM AUDITOR	SC60 (z/OS)	SAF:PROGRAM : - / ICHDSM00	SC60 (z/OS)
70	Tue Nov 13 2007 14:28:58 GMT-05:00	1	Read : File / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC06 (LU)	SAF:DATASET : - / HANNE.PAYROLL	SC60 (z/OS)
70	Tue Nov 13 2007 14:28:56 GMT-05:00	1	Read : File / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC06 (LU)	SAF:DATASET : - / HANNE.PAYROLL	SC60 (z/OS)
70	Tue Nov 13 2007 14:28:54 GMT-05:00	1	Read : File / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC06 (LU)	SAF:DATASET : - / HANNE.PAYROLL	SC60 (z/OS)
75	Tue Nov 13 2007 14:37:30 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60 (z/OS)	DB2OPR	USIBMSC.SC60TC07 (LU)	SAF:MDSNTB : - / DB8L.AETEST.TEST02.SELECT	SC60 (z/OS)
75	Tue Nov 13 2007 14:37:30 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60 (z/OS)	DB2OPR	USIBMSC.SC60TC07 (LU)	SAF:MDSNTB : - / DB8L.AETEST.TEST02.DELETE	SC60 (z/OS)
75	Tue Nov 13 2007 14:36:31 GMT-05:00	1	Read : Dbobject / ProfAttr	SC60 (z/OS)	DB2OPR	USIBMSC.SC60TC07 (LU)	SAF:MDSNTB : - / DB8L.AETEST.TEST02.INSERT	SC60 (z/OS)

Figure 9-48 TCIM attention events report

6. Click the TCIM reporting record and get detail information.



Event Detail

	Field	Group
Severity	75	This is a policy exception. This is a special attention event.
When	Tue Nov 13 2007 14:36:31 GMT-05:00	Non business hours (10)
What	Read : Dbobject / ProfAttr	Other Events (10)
Where	SC60 (z/OS)	z/OS platform (10)
Who	DB2OPR (SC60.DB2OPR)	Other Sources (10)
From Where	USIBMSC.SC60TC07 (LU)	Other Platforms (10)
On What	SAF:MDSNTB : - / DB8L.AETEST.TEST02.INSERT	Payroll Table (10)
Where To	SC60 (z/OS)	z/OS platform (10)

Aspect	Value
Event :: logrecordtype	SMF,80-2
Event :: miscellaneous	JOBNAME:DB2OPR
Event :: command	A=(ª·AbF DB2OPR DB8L T Y Y N N Y N TEST02 AETEST DSNDB04 051 MDSNTB DB8L.AETEST.TEST02.INSERT
Event :: recorddesc	RACF ACCESS warning for DB2OPR: (READ,NONE) on MDSNTB DB8L.AETEST.TEST02.INSERT
Who :: originator	SC60.DB2OPR

Figure 9-49 TCIM attention report details

This record comes from SMF type 80 RACF record. When using RACF DB2 external security, SMF cut more information about the DB2 subsystem from RACF records.

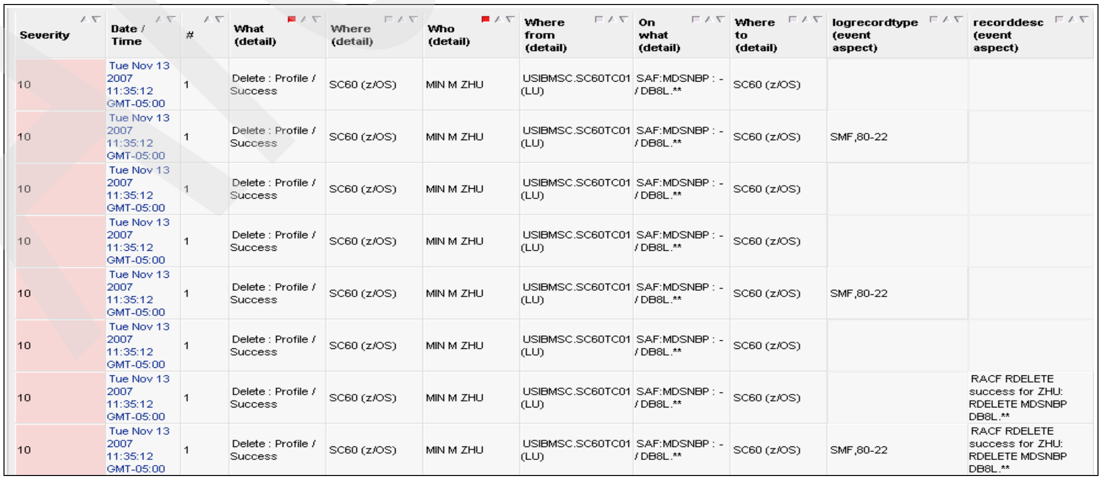
RACF WARNING on DB2 profile operation in RACF test case

This test case shows that the deletion of the privileged user, ZHU. RACF, generates a warning and will record into SMF type 80.

Performing the test case

To perform the test case:

1. Define the user ZHU as a privileged user with special authority in RACF.
2. Make sure that type 80 is specified for dump.
3. Define severity 10 in the TCIM policy.
4. Delete the generic profile “DB8L.**“ under Class MDSNBP in RACF with the command RDELETE MDSNBP (DB8L.**).



Severity	Date / Time	#	What (detail)	Where (detail)	Who (detail)	Where from (detail)	On what (detail)	Where to (detail)	logrecordtype (event aspect)	recorddesc (event aspect)
10	Tue Nov 13 2007 11:35:12 GMT-05:00	1	Delete : Profile / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC01 (LU)	SAF:MDSNBP : - / DB8L.**	SC60 (z/OS)		
10	Tue Nov 13 2007 11:35:12 GMT-05:00	1	Delete : Profile / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC01 (LU)	SAF:MDSNBP : - / DB8L.**	SC60 (z/OS)	SMF,80-22	
10	Tue Nov 13 2007 11:35:12 GMT-05:00	1	Delete : Profile / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC01 (LU)	SAF:MDSNBP : - / DB8L.**	SC60 (z/OS)		
10	Tue Nov 13 2007 11:35:12 GMT-05:00	1	Delete : Profile / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC01 (LU)	SAF:MDSNBP : - / DB8L.**	SC60 (z/OS)		
10	Tue Nov 13 2007 11:35:12 GMT-05:00	1	Delete : Profile / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC01 (LU)	SAF:MDSNBP : - / DB8L.**	SC60 (z/OS)	SMF,80-22	
10	Tue Nov 13 2007 11:35:12 GMT-05:00	1	Delete : Profile / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC01 (LU)	SAF:MDSNBP : - / DB8L.**	SC60 (z/OS)		
10	Tue Nov 13 2007 11:35:12 GMT-05:00	1	Delete : Profile / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC01 (LU)	SAF:MDSNBP : - / DB8L.**	SC60 (z/OS)		RACF RDELETE success for ZHU: RDELETE MDSNBP DB8L.**
10	Tue Nov 13 2007 11:35:12 GMT-05:00	1	Delete : Profile / Success	SC60 (z/OS)	MIN M ZHU	USIBMSC.SC60TC01 (LU)	SAF:MDSNBP : - / DB8L.**	SC60 (z/OS)	SMF,80-22	RACF RDELETE success for ZHU: RDELETE MDSNBP DB8L.**

Figure 9-50 TCIM report for ZHU user deletion

Drill down to the record from Nov 13th 2007, 11:35:12 to see more detailed information.

Event Detail

	Field	Group
Severity	10	This is a policy exception.
When	Tue Nov 13 2007 11:35:12 GMT-05:00	Other Periods (10)
What	Delete : Profile / Success	Other Events (10)
Where	SC60 (z/OS)	z/OS platform (10)
Who	MIN M ZHU (SC60.ZHU)	ZHU (10) SC60.RACF Special - Production (10) More...
From Where	USIBMSC.SC60TC01 (LU)	Other Platforms (10)
On What	SAF:MDSNBP : - / DB8L.**	Other Objects (10)
Where To	SC60 (z/OS)	z/OS platform (10)

▶ Incident Tracking

▼ Additional information

Aspect	Value
Event :: command	RDELETE MDSNBP (DB8L.**)
Event :: logrecordtype	SMF,80-22
Event :: recorddesc	RACF RDELETE success for ZHU: RDELETE MDSNBP DB8L.**
Who :: originator	SC60.ZHU

Figure 9-51 Details on ZHU user deletion

9.7.2 DB2 Auditing using TCIM and SMF Type 102

In this section we discuss DB2 Auditing using TCIM and SMF Type 102.

RACF WARNING for unprivileged USER access test case

All unauthorized activities by the unprivileged user (DNB2AE) will be recorded by RACF before checking in DB2. If related, DB2 Resource is defined in RACF profile and put in an UACCESS warning. The information can be cut into SMF.

Performing the test case

To perform the test case:

1. Define the user profile DB2AE in RACF.
2. Define general resource profiles for DB2 in RACF:

```
RDEF DSNADM DB8L.SYSADM UACC(NONE) OWNER(DB2) AUDIT(ALL(READ)) WARNING
RDEF DSNADM DB8L.SYSOPR UACC(READ) OWNER(DB2) AUDIT(ALL(READ)) WARNING
RDEF DSNADM DB8L.SYSCTRL UACC(NON) OWNER(DB2) AUDIT(ALL(READ)) WARNING
```

3. Connect to DB8L using DB2AE:

```
Set Current SQLID='DB2OPR';
```

4. A RACF warning massage shows up in the system console log:

```
ICH408I USER(DB2AE   ) GROUP(SYS1    ) NAME(DB2AE              ) 508
```

```
DB8L.SYSADM CL(DSNADM  )
WARNING: INSUFFICIENT AUTHORITY - TEMPORARY ACCESS ALLOWED
ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

Make sure to put *warning* and *audit (all (READ))* for the DB8L.SYSADM profile definition.

When using RACF for DB2 authorization management, RACF documents the profiles required to access DB2 resources, and the RACF/DB2 External Security Module will check the RACF profiles corresponding to that set of privileges or authorities. (A related record can be checked in the TCIM report as well.)

RACF-managed privileged user access test case

Privileged users who get RACF access authorization can normally access the DB2 resource. If they are put into the UAUDIT list, SMF cuts related records that can be audited by TCIM.

Performing the test case

To perform the test case:

1. Define three groups as:

   ```
   DB8LADM- DB2 DB8L SYSADM group
   DB8LOPR- DB2 DB8L SYSOPR group
   DB8LCTRL- DB2 DB8L SYSCTRL group
   ```

2. Put DB2AE into the UAUDIT list.
3. Add DB2AE into the group DB8LADM.
4. Connect group DB8LADM to profile DB8L.SYSADM.
5. Connect to DB8L using DB2AE and issue the command:

   ```
   Set current sqlid='db2opr'
   ```

6. There is no warning message from RACF in console log because DB2AE got DB2 DB8L SYSADM privilege when adding DB2AE into group DB8LADM (as DB8LADM is DB2 DB8L SYSADM group).
7. Check the TCIM report for the DB2AE auditing record. (The system has been configured to record SMF Types 100–102 data, and connected to TCIM Server as one source.)

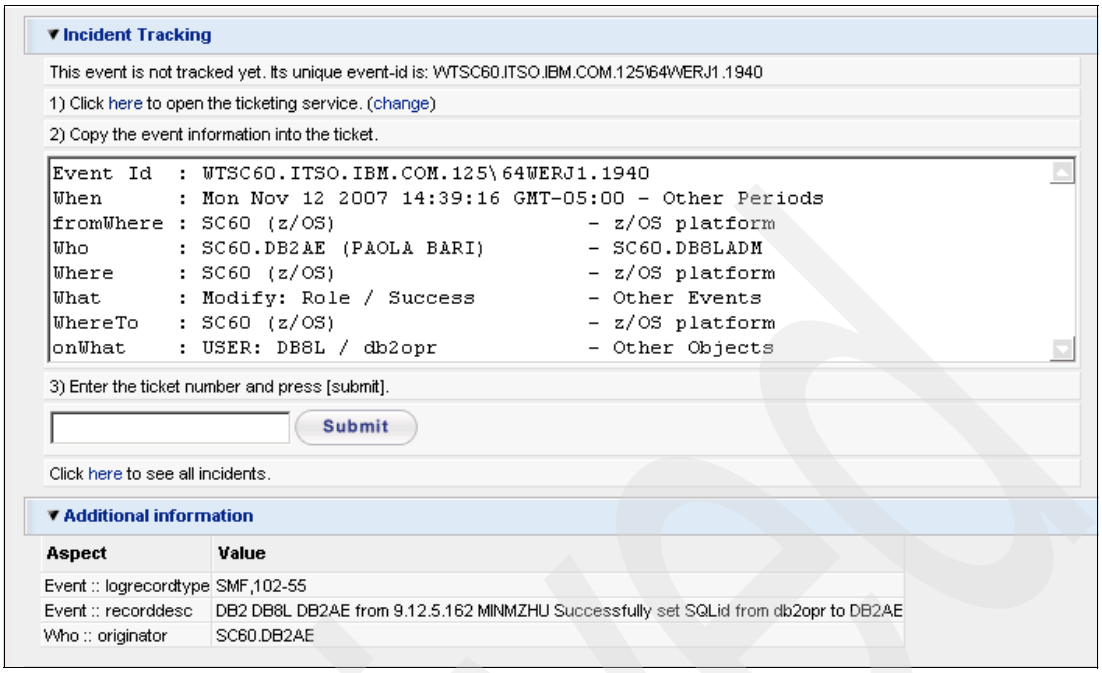


Figure 9-52 Incident Tracking

TCIM cut DB2 auditing related records from SMF and maps to “W7” format with detailed auditing report. As it is using RACF/DB2 External Security management, TCIM offers an entire auditing picture not only for mainframe but combined with other operation systems, application, or components.

Auditing failed DB2 authorization using SMF record 102 test case

SMF type 102 records DB2 auditing related information including DB2 authorization failures.

Performing the test case

To perform the test case:

1. Define DB2AE in the TCIM policy item “WHO” given severity 10.
2. Log on user DB2AE as an unprivileged user.
3. Issue a DB2 command (by DB2AE): Grant Trace to Alina.
4. Get failed code=552.
5. Check in TCIM.

First find severity 10 records or filter by log record type “SMF 102” & What “Grant”.

EPRORADB » zOS » Reports » Check (Filtered)

Check

Database zOS on Server EPRORADB - filtered by logrecordtype (event aspect) *102*, recorddesc (event aspect) *grant*

Setup:

	Month	Day	Year	Hour	Min.
Start time	November	12	2007	9	45
End time	November	13	2007	14	51

Execute Reset

Time zone: Event time zone

Severity	Date / Time	#	What (detail)	Where (detail)	Who (detail)	Where from (detail)	On what (detail)	Where to (detail)	logrecordtype (event aspect)	recorddesc (event aspect)
10	Mon Nov 12 2007 10:46:50 GMT-05:00	1	Grant : User / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	CREDENTIAL : - / -	SC60 (z/OS)	SMF,102-141	DB2 DB8L ZHU Grant userauth by id ZHU with SYSADM: GRANT SYSADM TO HANNE
10	Mon Nov 12 2007 10:49:59 GMT-05:00	1	Grant : User / Success	SC60 (z/OS)	MIN M ZHU	SC60 (z/OS)	CREDENTIAL : - / -	SC60 (z/OS)	SMF,102-141	DB2 DB8L ZHU Grant userauth by id ZHU with SYSADM: GRANT SYSADM TO HANNE
10	Mon Nov 12 2007 11:46:11 GMT-05:00	1	Grant : User / Success	SC60 (z/OS)	DB2AE	SC60 (z/OS)	CREDENTIAL : - / -	SC60 (z/OS)	SMF,102-141	DB2 DB8L DB2AE Grant userauth by id DB2AE failed code=552: GRANT TRACE TO ALINA

Figure 9-53 Grant verification in TCIM

Drill down to see detailed information.

Event Detail

	Field	Group	
Severity	10	This is a policy exception.	
When	Mon Nov 12 2007 11:46:11 GMT-05:00	Other Periods (10)	
What	Grant : User / Success	Other Events (10)	
Where	SC60 (z/OS)	z/OS platform (10)	
Who	DB2AE (SC60.DB2AE)	DB2AE (10) SC60.RACF Special - Production (10)	More...
From Where	SC60 (z/OS)	z/OS platform (10)	
On What	CREDENTIAL : - / -	Other Objects (10)	
Where To	SC60 (z/OS)	z/OS platform (10)	

▸ Incident Tracking

▾ Additional information

Aspect	Value
Event :: logrecordtype	SMF,102-141
Event :: recorddesc	DB2 DB8L DB2AE Grant userauth by id DB2AE failed code=552: GRANT TRACE TO ALINA
Who :: originator	SC60.DB2AE

Figure 9-54 TCIM detailed information about grant access

9.7.3 DB2 Auditing using audit Management Expert

In this section we discuss DB2 auditing using the Audit Management Expert.

Environment setup

For all the connections to the DB2 subsystem from distributed sources, we activated the DSNR class in RACF. In our test scenarios, we connect to the DB8L subsystem.

1. Activate the DSNR class.
2. Define the profile DB8L.DIST, where DB8L is the name of your subsystem.
3. Define a user in RACF, DB2AME, for your distributed task DB2 Audit Expert Manager that will connect to your subsystem.
4. Define a RACF GROUP: DB2DIST.
5. Connect user DB2AM to group DB2DIST.
6. Connect group DB2DIST to resource DB8L.DIST in the DSNR class.
7. Connect user DB2AME to GROUP xxx (to be defined after reading logs).
8. Connect.
9. Check that all distributed connects to DB2 are verified through the DSNR class and that audit records are cut for these user IDs connecting to DB2 to the mainframe.

Audit failed access attempt test case

When unauthorized users try to connect to DB2 on z/OS, access attempt denied by DB2 will be recorded into DB2 audit trace class 1, which is the default.

Performing the test case

For the test case:

1. Define the collection profile in the AME admininstration Interface and specify to audit “Failed Access Attempt”.
2. Active the collection.
3. Check auditing records from the AME Reporting Interface for type “a. failed access attempt”.

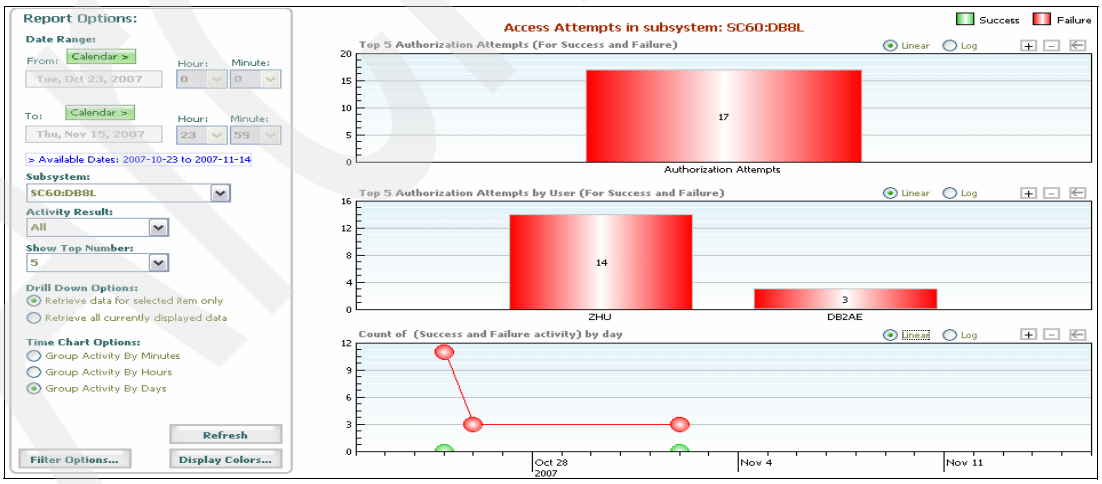


Figure 9-55 Auditing record in AME reporting

4. Export auditing detail report.

Record Count: 17

ROW	TIME	RESULT	RETURNED	IFICODE	CORREL...	CONTEXT_TYPE	TYPE	AUTH...	ORIGIN...	REQUESTOR...	DB2_...	CONNECTIO...	REQUESTOR_LOCATION	LUNAME
12	2007-10-26 18...	128	FAILURE	00083	487585264	AUTH ATTEMPT -...	USER AUTH	ZHU		MINMZHU	DB8L	DISCN-NC	9.12.5.162	G965
13	2007-10-26 18...	128	FAILURE	00083	487585264	AUTH ATTEMPT -...	USER AUTH	ZHU		MINMZHU	DB8L	DISCN-NC	9.12.5.162	G966
14	2007-10-26 18...	128	FAILURE	00083	487585264	AUTH ATTEMPT -...	USER AUTH	ZHU		MINMZHU	DB8L	DISCN-NC	9.12.5.162	G967
15	2007-11-02 13...	128	FAILURE	00083	489417104	AUTH ATTEMPT -...	USER AUTH	DB2AE		MINMZHU	DB8L	DISCN-NC	9.12.5.162	GC63
16	2007-11-02 13...	128	FAILURE	00083	489417104	AUTH ATTEMPT -...	USER AUTH	DB2AE		MINMZHU	DB8L	DISCN-NC	9.12.5.162	GC64
17	2007-11-02 13...	128	FAILURE	00083	489417104	AUTH ATTEMPT -...	USER AUTH	DB2AE		MINMZHU	DB8L	DISCN-NC	9.12.5.162	GC65
1	2007-10-25 12...	128	FAILURE	00083	523846832	AUTH ATTEMPT -...	USER AUTH	ZHU	SYSOPR	MINMZHU	DB8L	DB8L	9.12.5.162	G78A
2	2007-10-25 12...	128	FAILURE	00083	523846832	AUTH ATTEMPT -...	USER AUTH	ZHU	SYSOPR	MINMZHU	DB8L	DB8L	9.12.5.162	G78B
3	2007-10-25 12...	128	FAILURE	00083	523846832	AUTH ATTEMPT -...	USER AUTH	ZHU	SYSOPR	MINMZHU	DB8L	DB8L	9.12.5.162	G7A0
4	2007-10-25 12...	128	FAILURE	00083	523846832	AUTH ATTEMPT -...	USER AUTH	ZHU	SYSOPR	MINMZHU	DB8L	DB8L	9.12.5.162	G7A1
5	2007-10-25 12...	128	FAILURE	00083	523846832	AUTH ATTEMPT -...	USER AUTH	ZHU	SYSOPR	MINMZHU	DB8L	DB8L	9.12.5.162	G7A2
6	2007-10-25 12...	128	FAILURE	00083	523846832	AUTH ATTEMPT -...	USER AUTH	ZHU	SYSOPR	MINMZHU	DB8L	DB8L	9.12.5.162	G7A3
7	2007-10-25 14...	128	FAILURE	00083	523854856	AUTH ATTEMPT -...	USER AUTH	ZHU	SYSOPR	MINMZHU	DB8L	DB8L	9.12.5.162	G7F4
8	2007-10-25 14...	128	FAILURE	00083	523854856	AUTH ATTEMPT -...	USER AUTH	ZHU	SYSOPR	MINMZHU	DB8L	DB8L	9.12.5.162	G7F5
9	2007-10-25 14...	128	FAILURE	00083	523854856	AUTH ATTEMPT -...	USER AUTH	ZHU	SYSOPR	MINMZHU	DB8L	DB8L	9.12.5.162	G7F6
10	2007-10-25 14...	128	FAILURE	00083	523854856	AUTH ATTEMPT -...	USER AUTH	ZHU	SYSOPR	MINMZHU	DB8L	DB8L	9.12.5.162	G7F7
11	2007-10-25 14...	128	FAILURE	00083	523854856	AUTH ATTEMPT -...	USER AUTH	ZHU	SYSOPR	MINMZHU	DB8L	DB8L	9.12.5.162	G7F8

Figure 9-56 Exporting auditing trail

Audit DB2 table first read and first change test case

For the defined target auditing table, auditors can retrieve all insert/delete/update operation information organized by time range.

Performing the test case

For the test case:

1. Make sure that the target table attribute is AUDIT ALL. To check the table audit attribute:

   ```
   SELECT AUDITING FROM SYSIBM.SYSTABLES WHERE CREATOR='DB2AE'AND
   NAME=AETEST.TEST01';
   ```

   To change target table's attribute to be auditable:

   ```
   ALTER TABLE AETEST.TEST01 AUDIT ALL;
   ```

2. Define the collection operation in the AME administration Interface, add AETEST.TEST01 to the target table list, and specify the first read and first change event to be audited.

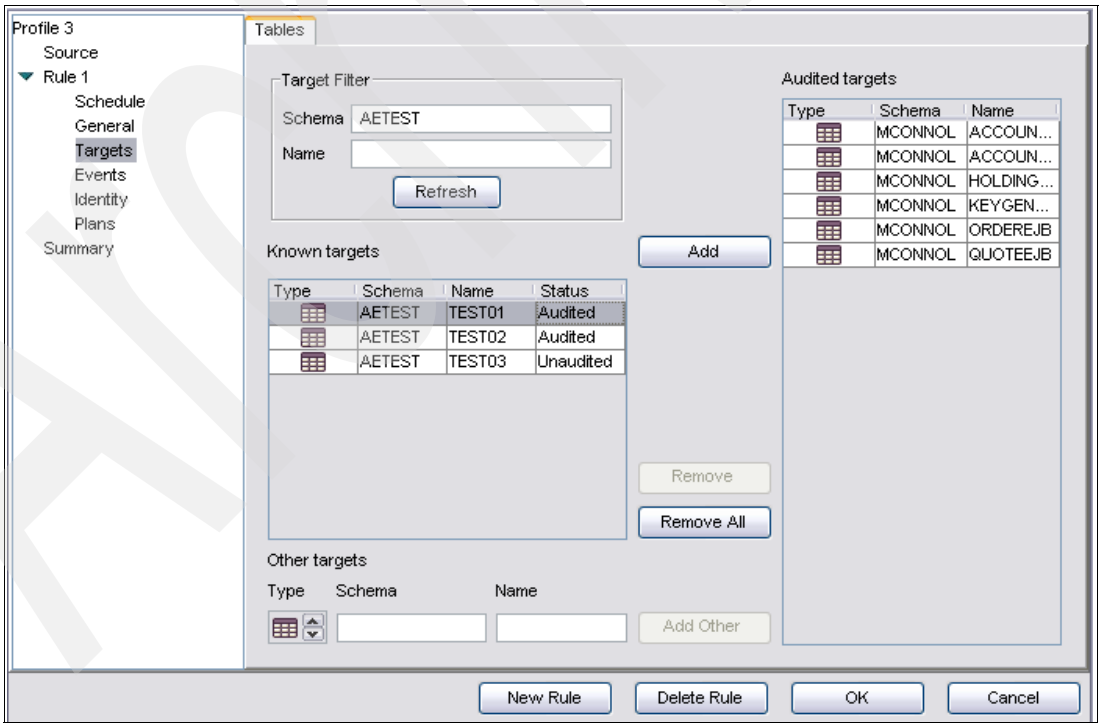


Figure 9-57 Defining collection

3. Activate the collection in the AME admininstration interface.

4. Open the AME Reporting Interface to check the first read or first change record and generate an auditing report if required.

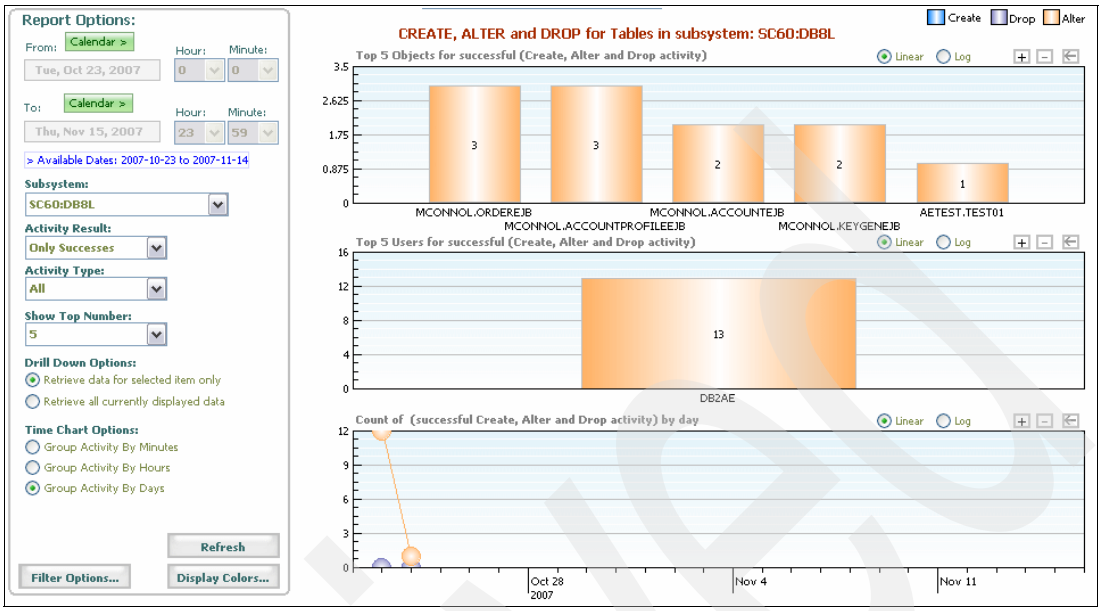


Figure 9-58 Auditing report

Audit DB2 table insert/delete/update using AME log analysis test case

If an auditor is interested in DB2 access during a certain time range and wants to include table record changes before and after an update operation, she can use the AME support log analysis function to filter the data needed for the auditing analysis.

Performing the test case

For the test case:

1. Make sure that the log analysis time range covers the possible auditing event in which auditors are interested.
2. Specify the DB2 subsystem that agents have been installed on.
3. Specify the auditing target tables by refreshing the schema or the given full names.

4. Specify the filters (time and date) and which reports are requested (as part of the statement type selection and report output options).

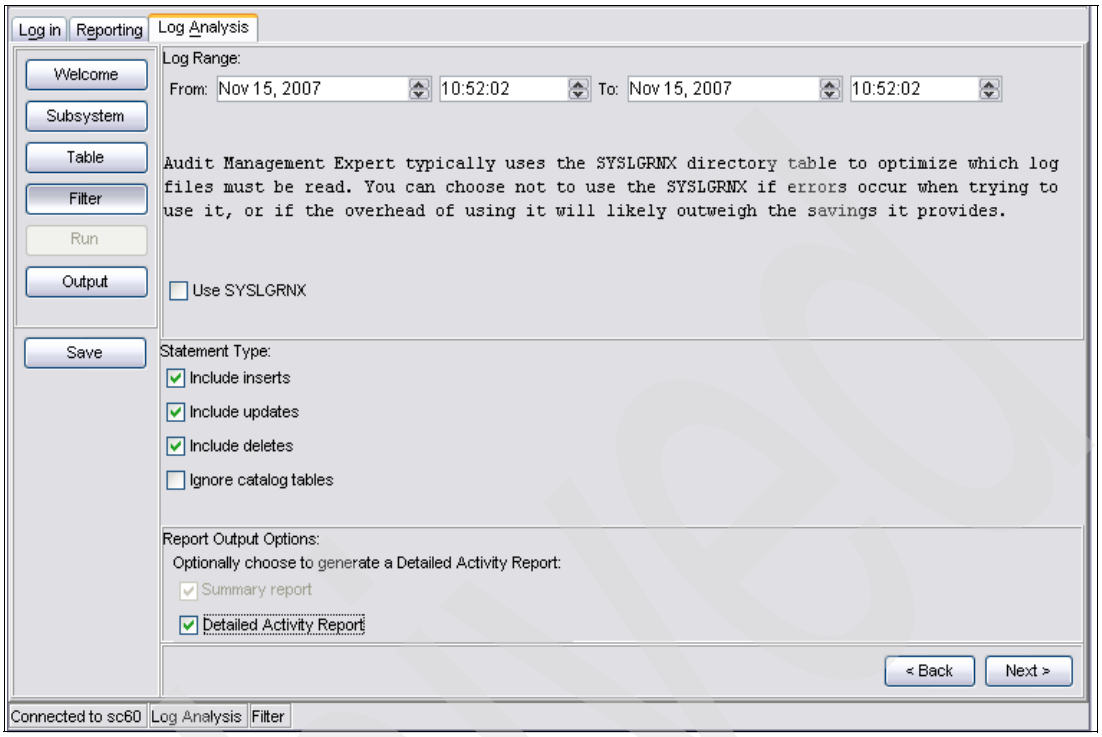


Figure 9-59 Defining the time intervals

5. Generate JCL and submit it with the TSO user.
6. In the output label, auditing log analysis reports are available as soon as the batch job completes on z/OS. The reports can be then saved and used for further analysis.

A

Linux Audit record description

This appendix describes the most-common Linux Audit records. The descriptions are based on Red Hat Enterprise Linux Version 5.

Reading the record descriptions

The several record types described here are made available through five tables:

- Table A-1 on page 295 shows records that can be emitted by trusted programs running in userland.
- Table A-2 on page 301 shows records that can be emitted by the audit daemon itself.
- Table A-3 on page 302 shows records that can be emitted by generic events in the Linux kernel.
- Table A-4 on page 305 shows records related to mandatory access control events as implemented by SELinux.
- Table A-5 on page 306 shows records related to anomalies or special conditions.

All tables follow the same base format, composed of one line for each event type, and three fixed columns, as described:

- The first column provides the textual name of the record type. All names are capitalized identifiers, and all records carry them under the type field. It is possible to search for specific record types using **`ausearch -m TYPE`**, where *TYPE* is the record name.
- The second column provides a brief description of the event and possible fields description. The descriptions also bring specific information about what kind of data can be found in each field, following the convention:
  - interpreted - This field can be interpreted as a textual meaning by auparse libraries (including commands **`ausearch`** and **`aureport`**).
  - decimal - This field is a number in base 10 (decimal).
  - octal - This field is a number base 8 (octal).
  - hex - This field is a number base 16 (hexadecimal).
  - string - This field is a character string (text).
- The third column lists what reports can possibly bring this type of record. Names used here are the same as those used in report parameters for **`aureport`** (**`aureport --name`**):
  - auth - report about authentication attempts
  - avc - report about avc messages
  - config - report about config changes
  - event - report about events
  - file - report about files
  - host - report about hosts
  - login - report about logins
  - mods - report about account modifications
  - mac - report about mandatory access control (MAC) events
  - pid - report about processes
  - syscall - report about syscalls
  - terminal - report about terminals
  - user - report about users
  - executable - report about executables

Format tables: user-space records

Table A-1 User-space record types

Record type	Description	Report
USER_AUTH	Authentication - Subject performed an authentication operation. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if subj cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► exe - calling task's executable name, if any. String. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String. ► res - authentication outcome. Success or failed.	► auth ► event ► user ► pid ► host ► terminal
USER_ACCT	Accounting update. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if subj cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► exe - calling task's executable name, if any. String. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String. ► res - operation outcome. Success or failed.	► event ► user ► pid ► host ► terminal

Record type	Description	Report
CRED_ACQ	Credential acquired - user acquired a new identity. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if subj cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► exe - calling task's executable name, if any. String. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String. ► res - operation outcome. Success or failed.	► event ► pid ► user ► host ► terminal
CRED_DISP	Credential disposed - user-disposed previously acquired identity. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if subj cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► exe - calling task's executable name, if any. String. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String. ► res - operation outcome. Success or failed.	► event ► pid ► user ► host ► terminal

Record type	Description	Report
USER_START	Session start - user started interacting with the system. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if subj cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► exe - calling task's executable name, if any. String. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String. ► res - operation outcome. Success or failed.	► event ► pid ► user ► host ► terminal
USER_END	Session end. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if subj cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► exe - calling task's executable name, if any. String. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String. ► res - operation outcome. Success or failed.	► event ► pid ► user ► host ► terminal

Record type	Description	Report
USER_AVC	Access Vector Cache check user-space message. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if subj cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► exe - calling task's executable name, if any. String. ► sauid - the original (audit) user ID of the entity related to the AVC message. Decimal or interpreted. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String.	► avc ► event ► pid ► user ► host ► terminal

Record type	Description	Report
USER_CHAUTHTOK	Account attribute changed - password changes, group changes, and so on. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if *subj* cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► exe - calling task's executable name, if any. String. ► sauid - the original (audit) user ID of the entity related to the AVC message. Decimal or interpreted. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String. ► op - operation description, if available. String. ► acct - account name where operation is being performed, if available. String. ► id - account ID where operation is being performed, if available, and if acct could not be determined. Decimal. ► res - operation outcome. Success or failed.	► mods ► event ► pid ► user ► host ► terminal
USER_ERR	Account state error. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if subj cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► exe - calling task's executable name, if any. String. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String. ► res - operation outcome. Success or failed.	► event ► pid ► user ► host ► terminal

Record type	Description	Report
CRED_REFR	Credential refreshed - a credential has been assigned. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if subj cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► exe - calling task's executable name, if any. String. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String. ► res - operation outcome. Success or failed.	► event ► pid ► user ► host ► terminal
USYS_CONFIG	System config changed. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if subj cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► exe - calling task's executable name, if any. String. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String. ► res - operation outcome. Success or failed.	► config ► event ► pid ► user ► host ► terminal

Record type	Description	Report
USER_LOGIN	User has logged in. Fields: ► pid - calling task's process ID. Decimal or interpreted. ► uid - calling task's user ID. Decimal or interpreted. ► auid - calling task's original (audit) user ID. Decimal or interpreted. ► subj - calling task's security context, if any. String. ► ssid - calling task's security ID, if any, and if subj cannot be determined. Decimal. ► msg - application-defined message about the operation audited. String. ► uid - user ID performing login, if available. Decimal. ► exe - calling task's executable name, if any. String. ► hostname - calling user's host name if known. String. ► addr - calling user's network address if known. String. ► terminal - calling user's terminal if known. String. ► res - operation outcome. Success or failed.	► login ► event ► pid ► user ► host ► terminal

Format tables: audit daemon records

Table A-2 Audit daemon record types

Record type	Description	Report
DAEMON_START	Audit daemon startup. Fields: ► ver - audit daemon version. String. ► format - audit log format. Currently fixed as raw. ► auid - audit daemon original (audit) user ID. Decimal or interpreted. ► pid - audit daemon process ID. Decimal. ► res - outcome of the start procedure. Success or failed.	► event ► pid
DAEMON_END	Audit daemon normal stop. Fields: ► auid - requester's original (audit) user ID. Decimal or interpreted. ► pid - requester's process ID. Decimal. ► subj - requester's security context. String. ► res - outcome of the shutdown procedure. Success or failed.	► event ► pid
DAEMON_ABORT	Audit daemon abnormal stop. Fields: ► auid - requester's original (audit) user ID. Decimal or interpreted. ► pid - requester's process ID. Decimal. ► res - always failed.	► event ► pid

Record type	Description	Report
DAEMON_CONFIG	Audit daemon configuration change. Fields: ► auid - requester's original (audit) user ID. Decimal or interpreted. ► pid - requester's process ID. Decimal. ► subj - requester's security context. String. ► res - outcome of the reconfigure procedure. Success or failed.	► config ► event ► pid

Format tables: generic kernel events records

Table A-3 Generic kernel events record types

Record type	Description	Report
SYSCALL	Generic syscall event. Fields: ► arch - architecture number. Hex or interpreted. ► syscall - syscall numberm. Decimal or interpreted. ► per - personality if not Linux. Hex. ► success - operation success, yes or no. ► exit - error code (errno) if operation was unsucessful. Decimal. ► a0, a1, a2, a3 - syscall's first four parameters, if any. Hex. ► items - number of related object names reported in the same event (such as PATH records). Decimal. ► ppid - parent process ID of the calling process. Decimal. ► pid - calling process's pid. Decimal. ► auid - calling user original (audit) user ID. Decimal or interpreted. ► uid - calling user current user ID. Decimal or interpreted. ► gid - calling user primary group ID. Decimal or interpreted. ► euid - calling user's effective user ID. Decimal orinterpreted. ► suid - calling user's saved user ID. Decimal or interpreted. ► fsuid - caller user's file system checks user ID. Decimal or interpreted. ► egid - calling user's effective group ID. Decima or interpreted. ► sgid - calling user's saved group ID. Decimal or interpreted. ► fsgid - calling user's file system checks group ID. Decima or interpreted. ► tty - terminal name if any. String. ► comm - calling task's command name. String. ► exe - calling task's executable name, if any. String. ► key - filter key if any. String.	► syscall ► Any other report when syscalls are involved

Record type	Description	Report
PATH	Brings filename path information for syscall dealing with files. Fields: ► item - item reference number, from zero to items minus one in the SYSCALL record. Decimal. ► name - name of file, object, or resource, if any. String. ► inode - inode number if applicable. Decimal. ► dev - device's major and minor numbers if applicable. Hex in nn:mm format. ► mode - object's permission modes. Octal. ► ouid - object owner's uid if applicable. Decimal or interpreted. ► rdev - raw device's major and minor numbers if applicable. Hex in nn:mm format. ► obj - object's security label, if any. String. ► osid - object's security ID, if any ,and if obj cannot be determined. Decimal.	Any syscall report
IPC	Brings IPC objects related with IPC syscalls. Fields: ► ouid - owner's user ID. Decimal or interpreted. ► ogid - owner's group. ID. Decimal or interpreted. ► mode - permission modes. Hex	Any syscall report
SOCKETCALL	Brings detailed parameters for socketcall syscall. Fields: ► nargs - number of arguments. Decimal. ► a0, a1, ..., aN - argument's values. Hex.	Any syscall report
CONFIG_CHANGE	Audit system configuration change. Fields: ► audit_rate_limit - new rate limit. Decimal. ► audit_backlog_limit - new backlog limit. Decimal. ► audit_enabled - new value for audit state. 0 for disabled, 1 for enabled. ► audit_failure - new value for audit failure action. Decimal. ► audit_pid - new value for audit daemon process ID. Decimal. ► old - previous value for configuration, if available. Decimal. ► auid - requeste's original (audit) user ID. Decimal or interpreted. ► subj - caller's security context, if available. String.	► config
SOCKADDR	Brings the socket address of bind, connect, and sendto syscalls. Fields: ► saddr - the socket address. Hex.	Any syscall report
CWD	Brings the current working directory for a SYSCALL record (if available). Fields: ► cwd - calling task's current working directory. String.	Any syscall report

Record type	Description	Report
EXECVE	Brings the execve arguments for a SYSCALL record (if available). Fields: ► a0, a1, ..., aN - argument's value. String.	Any syscall report
IPC_SET_PERM	Brings the new permissions for IPC operations with IPC_SET command. Fields: ► qbytes - max number of bytes in queue. Hex. ► ouid - owner's user ID. Decimal or interpreted. ► ogid - owner's group ID. Decimal or interpreted. ► mode - IPC object's permission mode. Octal.	Any syscall report
MQ_OPEN	Brings parameters for a mq_open syscall. Fields ► oflag - operation flags. Hex. ► mode - permission modes. Octal. ► mq_flags - message queue flags. Hex. ► mq_maxmsg - max number of messages. Decimal. ► mq_msgsize - max message size. Decimal. ► mq_curmsgs - number of messages currently in queue. Decimal.	Any syscall report
MQ_SENDRECV	Brings parameters for mq_timedsend and mq_timedreceive syscalls. Fields: ► mqdes - message queue descriptor. Decimal. ► msg_len - message size. Decimal. ► msg_prio - message priority. Decimal. ► abs_timeout_sec - timeout seconds. Decimal. ► abs_timeout_nsec - timeout nanoseconds. Decimal.	Any syscall report
MQ_NOTIFY	Brings parameters for mq_notify syscall. Fields: ► mqdes - message queue descriptor. Decimal. ► segev_signo - notification signal. Decimal.	Any syscall report
MQ_GETSETATTR	Brings parameters for mq_getsetattr syscall. Fields: ► mqdes - message queue descriptor. Decimal. ► mq_flags - message queue flags. Hex. ► mq_maxmsg - max number of messages. Decimal. ► mq_msgsize - max message size. Decimal. ► mq_curmsgs - number of messages currently in queue. Decimal.	Any syscall report

Format tables: SELinux events records

Table A-4 SELinux events record types

record type	description	report
AVC	SELinux Access Vector Cache check, meaning a failed MAC authorization. Fields: ► pid - caller task's process ID if available. Decimal. ► comm - caller task's command name if available. String. ► key - ipc-id if available. Decimal. ► capability - capability if avaiable. Decimal. ► name - resource name if available. String. ► path - resource path if available. String. ► dev - device name if available. String. ► ino - inode number if available. Decimal. ► laddr or saddr- local network address if available. String. ► lport or src- local network port if available. Decimal. ► faddr or daddr - foreign network address if available. String. ► fport or dest - foreign network port if available. Decimal. ► netif - name of the network interface if available. String. ► scontext - source security context if available. String. ► ssid - source security ID if available and if scontext cannot be determined. Decimal. ► tcontext - target security context if available. String. ► tsid - target security ID if available and if tcontext cannot be determined. Decimal. ► tclass - target security class if available. String.	► avc ► Any other report with AVC checks
AVC_PATH	Brings dentry, vsfmout pair from AVC records if aplicable. Fields: ► path - path to resource involved. String.	Any AVC reports
MAC_POLICY_LOAD	SELinux policy load. Fields: ► auid - caller's original (audit) user ID. Decimal or interpreted.	► mac ► config ► syscall

record type	description	report
MAC_STATUS	Changed SELinux enforcing status (enforcing, permissive, off). Fields: ► enforcing - new enforce value. 0 for permissive, 1 for enforcing. ► old_enforcing - old enforce value. 0 for permissive, 1 for enforcing. ► auid - caller's original (audit) user ID. Decimal or interpreted.	► mac ► config ► syscall
MAC_CONFIG_CHANGE	SELinux booleans change. Fields: ► bool - boolean name. String ► val - boolean new value. 0 for off, 1 for on. ► old_val - boolean old value. 0 for off, 1 for on. ► auid - caller's original (audit) user ID. Decimal or interpreted.	► mac ► config ► syscall

Format tables: anomalies records

Table A-5 Anomalies record types

Record type	Description	Report
ANOM_PROMISCUOUS	Device changed promiscuous mode. Fields: ► dev - device name. String. ► prom - new promiscuous mode. 0 for disabled, 1 for enabled. ► old_prom - previous promiscuous mode. 0 for disabled, 1 for enabled. ► auid - requester's original (audit) user ID. Decimal or interpreted.	► anomaly ► syscall

B

SSL/SSH configuration guides

The body of this book does not cover installation or configuration of the products that we have used. There are many books that cover this and are referred to throughout the book. However, there have been parts of the installation that we found were not adequately covered in available documentation.

This appendix contains a series of guides to the areas of secure communication configuration, such as configuring SSL for clients connecting to a z/OS LDAP server and configuring SSH for TCIM actuators.

Configuring SSL for LDAP clients on Linux for System z communicating with a z/OS LDAP server

In this section we show the steps for establishing SSL between an LDAP client (such as idsldapsearch) running on a Linux for System z image and the LDAP server running on z/OS. It involves using a self-signed certificate between the client and the server.

Note: In a production environment you probably would not use a self-signed certificate, as it is considered less secure than a publicly signed certificate.

We only used command-line tools, such as *gskkyman* on z/OS UNIX and *gsk7cmd* on Linux, for these examples.

The steps shown are:

1. Create a keystore, or key database, on z/OS.
2. Create a self-signed certificate.
3. Export the self-signed certificate.
4. Create a keystore, or key database, on Linux for System z.
5. Import the self-signed certificate.
6. Configure z/OS LDAP to use the newly created certificate.

Step 1 - Create key database on z/OS

The first step is to create the key database using the **gskkyman** command (shipped as part of the Cryptographic Services PKI Services in z/OS). Use **gskkyman** to start the database menu, as shown in Example B-1.

Example: B-1 gskkyman database menu

```
DAVIDED @ SC60:/SYSTEM/var>gskkyman

       Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database
   6 - Create key parameter file
   7 - Display certificate file (Binary or Base64 ASN.1 DER)

  11 - Create new token
  12 - Delete token
  13 - Manage token
  14 - Manage token from list of tokens

   0 - Exit program

Enter option number:
```

To create a keystore, or key database, select **Option 1 - Create new database**. You need to specify the key database name, a password for it, the expiration date, and the database record length.

For our system we accepted the default expiration and database record length values, as shown in Example B-2.

Example: B-2 gskkyman KeyDB creation

```
Enter option number: 1
Enter key database name (press ENTER to return to menu): zos_ldap.kdb
Enter database password (press ENTER to return to menu):
Re-enter database password:
Enter password expiration in days (press ENTER for no expiration):
Enter database record length (press ENTER to use 2500):

Key database /SYSTEM/var/zos_ldap.kdb created.

Press ENTER to continue.
```

This concludes the key creation. You can continue to work with certificates and keys within the newly created key database, manage other databases, or exit.

The key database has been created with a password that must be entered, or hardcoded into scripts, when accessing the database. To reduce this security exposure, you can store the password in a stash file, which is hashed and virtually unreadable. To create the stash file, exit the gskkyman utility and run it again with the -s command-line option, as shown in Example B-3.

Example: B-3 Using gskkyman to stash a key database password

```
DAVIDED @ SC60:/SYSTEM/var>gskkyman -s
Enter key database name (press ENTER to cancel): zos_ldap.kdb
Enter database password (press ENTER to cancel):

Database password stored in zos_ldap.sth.
```

As per the message from the utility, the key database password is now stored in the file zos_ldap.sth, which can be used instead of entering a password for client access to the key database.

Depending on who or what will be accessing the key database and stash file you may need to make the file group or world readable.

Step 2 - Create a self-signed certificate

Now that we have a key database defined we can create a self-signed certificate to use in our SSL communication. The key database also contains signing authority certificates that you would use in conjunction with certificates that you have had signed by these external authorities. But for a simple test environment, self-signed certificates are considered adequate.

To create the self-signed certificate run gskkyman. On the gskkyman Database Menu select option **2 - Open database**. It will prompt for the database name and password. In our case we open the key database that we created in the previous section, zos_ldap.kdb.

After opening the database, the key management menu is shown and the database listed, as shown in Example B-4.

Example: B-4 gskkyman opening a key database and the key management menu

```
Enter option number: 2
Enter key database name (press ENTER to return to menu): zos_ldap.kdb
Enter database password (press ENTER to return to menu):

       Key Management Menu

       Database: /SYSTEM/var/zos_ldap.kdb

   1 - Manage keys and certificates
   2 - Manage certificates
   3 - Manage certificate requests
   4 - Create new certificate request
   5 - Receive requested certificate or a renewal certificate
   6 - Create a self-signed certificate
   7 - Import a certificate
   8 - Import a certificate and a private key
   9 - Show the default key
  10 - Store database password
  11 - Show database record length

   0 - Exit program
```

We need to create a self-signed certificate in our key database, so select option **6 - Create a self-signed certificate**. This brings up the Certificate Type menu.

To work with the LDAP clients on System z images, you need to select one of the user or server options (options 5–8). CA certificates do not have the correct key usage to be used as a server end entity certificate.

For our environment we selected option **5 - User or server certificate with 1024-bit RSA key**, as shown in Example B-5.

Example: B-5 gskkyman certificate type menu for self-signed certificate creation

```
Enter option number (press ENTER to return to previous menu): 6

       Certificate Type

   1 - CA certificate with 1024-bit RSA key
   2 - CA certificate with 2048-bit RSA key
   3 - CA certificate with 4096-bit RSA key
   4 - CA certificate with 1024-bit DSA key
   5 - User or server certificate with 1024-bit RSA key
   6 - User or server certificate with 2048-bit RSA key
   7 - User or server certificate with 4096-bit RSA key
   8 - User or server certificate with 1024-bit DSA key

Select certificate type (press ENTER to return to menu): 5
```

On this panel you are prompted for the certificate attributes:

- Certificate label - a unique identifier for this certificate. It will be used both within the gskkyman tool and externally to refer to this specific certificate.
- Certificate subject name - an LDAP-formatted name consisting of a number of entries:
  - Common name (required) - a name (cn) for the certificate. It can be the same as the label, but does not have to be.
  - Organizational unit (optional) - an ou= for the certificate.
  - Organization (required) - an o= for the certificate.
  - City/locality (optional) - a l= for the certificate.
  - State/province (optional) - a s= for the certificate.
  - Country/region (required) - a two-character country code (for example, c=us or c=au).
- Validity period - the number of days that the certificate will be valid.

You can also specify alternate subject names. For our environment we set the values as shown in Example B-6.

Example: B-6 gskkyman self-signed certificate settings

```
Enter label (press ENTER to return to menu): zos_ldap_svr
Enter subject name for certificate
  Common name (required): zos_ldap_svr2
  Organizational unit (optional):
  Organization (required): itso
  City/Locality (optional):
  State/Province (optional):
  Country/Region (2 characters - required): us
Enter number of days certificate will be valid (default 365): 3650

Enter 1 to specify subject alternate names or 0 to continue: 0

Please wait .....

Certificate created.

Press ENTER to continue.
```

This results in a certificate that can be exported to the LDAP client system for use in SSL connections.

Before doing this, let us have a look at the contents of the certificate. Example B-7 shows the contents using the openssl toolkit (after exporting to a file, which we have not covered yet). You can see that the subject is C=US, O=itso, CN=zos_ldap_svr2, which was built from the common name, organization, and country/region specified in Example B-7. As this is a self-signed certificate, the issuer is the same as the subject.

Example: B-7 openssl view of new self-signed certificate

```
linux11z:/var/idsldap/V6.0 # openssl x509 -text -in zos_ldap_svr.b64
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            47:22:3f:c0:00:0a:48:54
```

```
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=itso, CN=zos_ldap_svr2
        Validity
            Not Before: Oct 26 19:28:00 2007 GMT
            Not After : Oct 23 19:28:00 2017 GMT
        Subject: C=US, O=itso, CN=zos_ldap_svr2
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b8:37:81:6a:3b:b1:43:7c:72:40:2a:12:a3:03:
                    dc:eb:70:ab:df:52:de:4c:27:f1:8b:d3:1b:e8:46:
                    6f:5a:a6:c9:cb:d2:a2:32:cc:f1:c7:c6:31:36:82:
                    bf:0a:b1:7b:93:6b:6c:5a:83:ef:70:20:2e:23:1e:
                    d3:96:89:9c:4e:d6:3c:6c:d6:50:66:99:68:68:92:
                    af:de:8d:55:0d:d8:04:98:04:f5:80:36:10:c5:2f:
                    f7:e4:7a:66:a9:25:44:81:eb:f5:1d:da:59:50:04:
                    aa:24:88:8f:a4:a9:6f:0d:93:b6:b8:43:23:55:bf:
                    04:13:06:da:ae:57:a0:6b:77
                Exponent: 65537 (0x10001)
...
```

Step 3 - Export the self-signed certificate

Now that the certificate has been created inside the database it needs to be exported to a file that can be sent to the client systems.

From the gskkyman key management menu (see Example B-4 on page 310) select option **1 - Manage keys and certificates**. You will be prompted for the key/certificate that you want to work with on the key and certificate list menu. In our example we selected the **zos_ldap_svr** certificate that we created above (as shown in Example B-8).

Example: B-8 gskkyman key and certificate list menu

```
Enter option number (press ENTER to return to previous menu): 1

       Key and Certificate List

       Database: /SYSTEM/var/zos_ldap.kdb

   1 - zos_ldap_svr

   0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list): 1
```

You are then presented with the key and certificate menu for that certificate, as shown in Example B-11 on page 314.

Example: B-9 gskkyman key and certificate menu

```
       Key and Certificate Menu

       Label: zos_ldap_svr
```

```
  1 - Show certificate information
  2 - Show key information
  3 - Set key as default
  4 - Set certificate trust status
  5 - Copy certificate and key to another database
  6 - Export certificate to a file
  7 - Export certificate and key to a file
  8 - Delete certificate and key
  9 - Change label
 10 - Create a signed certificate and key
 11 - Create a certificate renewal request

  0 - Exit program

Enter option number (press ENTER to return to previous menu):
```

Before exporting the certificate, you can use option **1 - Show certificate information** to view the contents of the certificate. Our examples is shown in Example B-10.

Example: B-10 gskkyman certificate information display

```
Enter option number (press ENTER to return to previous menu): 1

                        Certificate Information

                 Label: zos_ldap_svr
             Record ID: 15
      Issuer Record ID: 15
               Trusted: Yes
               Version: 3
         Serial number: 47223fc0000a4854
           Issuer name: zos_ldap_svr2
                        itso
                        US
          Subject name: zos_ldap_svr2
                        itso
                        US
        Effective date: 2007/10/26
       Expiration date: 2017/10/23
   Signature algorithm: sha1WithRsaEncryption
      Issuer unique ID: None
    Subject unique ID: None
  Public key algorithm: rsaEncryption
       Public key size: 1024
            Public key: 30 81 89 02 81 81 00 B8 37 81 6A 3B B1 43 7C 72
                        40 2A 12 A3 03 DC EB 70 AB DF 52 DE 4C 27 F1 8B
                        D3 1B E8 46 6F 5A A6 C9 CB D2 A2 32 CC F1 C7 C6
                        31 36 82 BF 0A B1 7B 93 6B 6C 5A 83 EF 70 20 2E
                        23 1E D3 96 89 9C 4E D6 3C 6C D6 50 66 99 68 68
                        92 AF DE 8D 55 0D D8 04 98 04 F5 80 36 10 C5 2F
                        F7 E4 7A 66 A9 25 44 81 EB F5 1D DA 59 50 04 AA
                        24 88 8F A4 A9 6F 0D 93 B6 B8 43 23 55 BF 04 13
```

```
                        06 DA AE 57 A0 6B 77 02 03 01 00 01
  Number of extensions: 3

Enter 1 to display extensions, 0 to return to menu:
```

Compare this with the contents shown by **openssl** in Example B-7 on page 311.

To export the certificate to a file, go back up to the key and certificate menu and select option **6 - Export certificate to a file**, as shown in Example B-11.

Example: B-11 gskkyman key and certificate menu

```
       Key and Certificate Menu

       Label: zos_ldap_svr

   1 - Show certificate information
   2 - Show key information
   3 - Set key as default
   4 - Set certificate trust status
   5 - Copy certificate and key to another database
   6 - Export certificate to a file
   7 - Export certificate and key to a file
   8 - Delete certificate and key
   9 - Change label
  10 - Create a signed certificate and key
  11 - Create a certificate renewal request

   0 - Exit program

Enter option number (press ENTER to return to previous menu): 6
```

On the export file format menu you can select one of four export formats and the file to which to export the certificate.

Note: It is common practice to use a .der extension for binary ASN.1 DER format files and a .pem extension for Base64 encoded ASN1. DER format files.

For the LDAP client SSL between the Linux for System z client and the z/OS LDAP server we chose option **1 - Binary ASN.1 DER** and named the export file zos_ldap_svr.der (see Example B-12).

Example: B-12 gskkyman export file format window

```
       Export File Format

   1 - Binary ASN.1 DER
   2 - Base64 ASN.1 DER
   3 - Binary PKCS #7
   4 - Base64 PKCS #7

Select export format (press ENTER to return to menu): 1
Enter export file name (press ENTER to return to menu): zos_ldap_svr.der
```

```
Certificate exported.

Press ENTER to continue.
```

This file can now be transferred to the LDAP client machines. As an example, we used ftp to pull files from one of the Linux on System z images. Note that a binary transfer was used in Example B-13.

Example: B-13 ftp certificate file from z/OS UNIX

```
linux11z:/var/idsldap/V6.0 # ftp 9.12.4.19
Connected to wtsc60oe.itso.ibm.com.
220-FTPD1 IBM FTP CS V1R9 at WTSC60.ITSO.IBM.COM, 13:13:01 on 2007-10-26.
220-Welcome to the SC60 system
220 Connection will not timeout.
Name (9.12.4.19:root): davided
331 Send password please.
Password:
230 DAVIDED is logged on.  Working directory is "DAVIDED.".
Remote system type is MVS.
ftp> binary
200 Representation type is Image
ftp> cd /var
250 HFS directory /var is the current working directory.
ftp> get zos_ldap_svr.der
local: zos_ldap_svr.der remote: zos_ldap_svr.der
229 Entering Extended Passive Mode (|||1384|)
125 Sending data set /var/zos_ldap_svr.der
250 Transfer completed successfully.
586 bytes received in 00:00 (14.69 KB/s)
```

The file is now on the client system and is ready to be imported into a local key database.

Step 4 - Create key database on Linux for System z

Before importing the new self-signed certificate into a key store on the Linux for System z image, you need to have a key database to load it into. If you already have one defined, you can skip to the next section.

To create the database, you can use the Global Security Kit (GSKit) IKEYCMD command-line interface (gsk7cmd utility) on the distributed systems. You can also use the GSKit JavaUI gsk7ikm utility or the GSKit command-line GSKCapiCmd program to perform the same functions.

Note: Before running gsk7cmd ensure that you have set up Java and that it includes the security.provider for CMS (com.ibm.spi.IBMCMSProvider). Otherwise, you will not see CMS as one of the key database types (or the command will fail if you try to specify CMS as the type). You should also have the Java specified in the PATH and JAVA_HOME environment variables.

Unlike the gskkyman utility that provided a text-based menu system to guide you through the steps, the gsk7cmd utility is purely command-line based. To perform a function, specify all of

the arguments. The full list of command-line arguments can be found in the *IBM Global Security Kit Secure Sockets Layer Introduction and iKeyman User's Guide*, SC32-1700-00.

To create a key database use the -keydb and -create arguments. The values that we used are shown in Example B-14.

Example: B-14 gsk7cmd Create KeyDB

```
linux11z:/var/idsldap/V6.0 # gsk7cmd -keydb -create -db zos_ldap.kdb -pw secret
-type cms -expire 3650 -stash
linux11z:/var/idsldap/V6.0 # ls -l zos*
-rw-r--r--  1 root root     80 Oct 26 09:33 zos_ldap.crl
-rw-r--r--  1 root root 110080 Oct 26 09:33 zos_ldap.kdb
-rw-r--r--  1 root root     80 Oct 26 09:33 zos_ldap.rdb
-rw-r--r--  1 root root    129 Oct 26 09:33 zos_ldap.sth
-rw-r--r--  1 root root    586 Oct 26 09:14 zos_ldap_svr.der
```

The key database is now ready to receive the self-signed certificate.

Step 5 - Import self-signed certificate

To add the new certificate to the database use the -cert and -add arguments. You must use the -add option. The -receive option expects to find a matching certificate creation request in the database (for requests sent to a certificate authority for signing).

The values we used are shown in Example B-15.

Example: B-15 gsk7cmd add certificate

```
linux11z:/var/idsldap/V6.0 # gsk7cmd -cert -add -db zos_ldap.kdb -pw secret -label
zos_ldap_svr -format binary -trust enable -file zos_ldap_svr.der
```

You can use the -cert and -details arguments to view the contents of the certificate, as shown in Example B-16.

Example: B-16 gsk7cmd list certificate

```
linux11z:/var/idsldap/V6.0 # gsk7cmd -cert -details -db zos_ldap.kdb
-pw secret -label zos_ldap_svr

Label: zos_ldap_svr
Key Size: 1024
Version: X509 V3
Serial Number: 47 21 DF 7B 00 07 0C 18
Issued By: zos_ldap_svr2
itso
US
Subject: zos_ldap_svr2
itso
US
Valid From: Friday, October 26, 2007 8:37:15 AM EDT To: Monday, October 23, 2017
8:37:15 AM EDT
Fingerprint: 67:AD:0E:DC:49:EB:52:F8:C8:7F:FF:28:33:57:34:3E:B5:57:C1:7F
Signature Algorithm: 1.2.840.113549.1.1.5
Trust Status: enabled
```

Note that this is similar to the output from gskkycmd in Example B-10 on page 313.

Step 6 - Configure z/OS LDAP for SSL and test

Now that the certificate is defined in both key databases (on z/OS and on the Linux for System z) the next step is to configure the LDAP server (on z/OS) to use it for its SSL connections.

We are concerned with a SSL/TLS-only secure connection. Also, we are only using the key database from above, not the RACF-maintained key ring.

The listening port definitions in our LDAP configuration file (we modified the SLAPDCNF member in the LDAP PDS) were:

- listen ldap://:3391
- listen ldaps://:3392

LDAP is listening for SSL connections on port 3392 and any host/IP address the z/OS system is using.

The SSL definitions that we used were:

- sslAuth serverAuth
- sslCertificate zos_ldap_svr
- sslCipherspecs ANY
- sslKeyRingFile /var/zos_ldap.kdb
- sslKeyRingPWStashFile /var/zos_ldap.sth

The ANY Cipherspecs was chosen for simplicity. We should have specified only those that were acceptable. As the stash file is being used for the key db password, the sslKeyRingFilePW parameter is commented out.

> **Note:** As the changes have been made to the z/OS PDS member, rather than the z/OS UNIX HFS member (such as /u/ldapsrv/ldap.profile), running the ldapcnf utility will wipe the changes. Make sure that your procedures allow for this scenario so that PDS member changes are not lost.

The z/OS LDAP server must be restarted for these changes to take effect.

Further details on the z/OS LDAP configuration can be found in the *Integrated Security Services LDAP Server Administration and Use manual*, SC24-5923-07.

To test the connection, use the idsldapsearch utility with the -Z (ssl) and -K (key db) arguments, as shown in Example B-17.

Example: B-17 idsldapsearch with SSL arguments

```
idsldapsearch -b o=itso -s sub -h 9.12.4.18 -p 3392 -D 'cn=LDAP Administrator' -w
secret -Z -K /var/idsldap/V6.0/zos_ldap.kdb 'objectclass=*'
O=ITSO
objectclass=organization
objectclass=top
o=ITSO

ou=Customers, o=ITSO
ou=Customers
objectclass=top
```

```
objectclass=organizationalUnit
seealso=cn=David Edwards, ou=Employees, o=ITSO
...
```

This example uses idsldapsearch to search the o=itso branch in the z/OS LDAP server from the Linux for System z image. It uses the zos_ldap.kdb key database that we created earlier to confirm the authenticity of the certificate that the z/OS LDAP server will present when establishing the SSL connection.

Configuring SSL for TAMOS on Linux for System z communicating with a z/OS LDAP server

In this section we show the steps for establishing SSL between TAMOS running on a Linux for System z image and the LDAP server running on z/OS. It involves using a self-signed certificate between the client and the server.

As part of the TAMOS configuration you need to specify a certificate for SSL communications to the user registry. In our environment the user registry is the z/OS LDAP server. The certificate needs to be in Base64 encoded ASN.1 DER format. This section expands on the previous one to show how this Base64 encoded certificate is extracted from the z/OS key database and used in the TAMOS configuration steps.

As we used the same keystore as in the previous section, the steps required were:

1. Export the self-signed certificate.
2. Configure TAMOS to use the certificate.

Step 1 - Export the self-signed certificate

The process to export the self-signed certificate using gskkyman is basically the same as in the previous section (see “Step 3 - Export the self-signed certificate” on page 312). You need to start gskkyman, open the key database, select manage keys and certificates, then select the self-signed certificate, as before.

On the key and certificate menu select option **6 - Export certificate to a file** (see Example B-18).

Example: B-18 gskkyman key and certificate menu

```
     Key and Certificate Menu

     Label: zos_ldap_svr

  1 - Show certificate information
  2 - Show key information
  3 - Set key as default
  4 - Set certificate trust status
  5 - Copy certificate and key to another database
  6 - Export certificate to a file
  7 - Export certificate and key to a file
  8 - Delete certificate and key
  9 - Change label
 10 - Create a signed certificate and key
```

```
  11 - Create a certificate renewal request

   0 - Exit program

Enter option number (press ENTER to return to previous menu): 6
```

When presented with the export file format list, select option **2 - Base64 ASN.1 DER** (see Example B-19).

Example: B-19 gskkyman export Base64 certificate

```
       Export File Format

   1 - Binary ASN.1 DER
   2 - Base64 ASN.1 DER
   3 - Binary PKCS #7
   4 - Base64 PKCS #7

Select export format (press ENTER to return to menu): 2
Enter export file name (press ENTER to return to menu): zos_ldap_svr.b64

Certificate exported.
```

The base64-encoded file, called zos_ldap_svr.b64, has been created in the local directory. As it is text, you can view the file (see Example B-20).

Example: B-20 Base64-encoded certificate file


```
-----BEGIN CERTIFICATE-----
MIICODCCAaGgAwIBAgIIRyI/wAAKSFQwDQYJKoZIhvcNAQEFBQAwNDELMAkGA1UE
BhMCVVMxDTALBgNVBAoTBGl0c28xFjAUBgNVBAMMDXpvc19sZGFwX3N2cjIwHhcN
MDcxMDI2MTkyODAwWhcNMTcxMDIzMTkyODAwWjAOMQswCQYDVQQGEwJVUzENMAsG
A1UEChMEaXRzbzEWMBQGA1UEAwwNem9zX2xkYXBfc3ZyMjCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEAuDeBajuxQ3xyQCoSowPc63Cr31LeTCfxi9Mb6EZvWqbJ
y9KiMszxx8YxNoK/CrF7k2tsWoPvcCAuIx7TlomcTtY8bNZQZploaJKv3o1VDdgE
mAT1gDYQxS/35HpmqSVEgev1HdpZUASqJIiPpKlvDZO2uEMjVb8EEwbarlega3cC
AwEAAaNTMFEwHQYDVR0OBBYEFJm/znPL/aLOZ4zOAIQBcpvTzkzFMB8GA1UdIwQY
MBaAFJm/znPL/aLOZ4zOAIQBcpvTzkzFMA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZI
hvcNAQEFBQADgYEALl7Qk99oV1oUOyZ3Nlmk+c9I/zIojAuZQFWywJ7QXDIXwhQU
nInNvi64FZ2aOXtrNJ7G8nIPWfdwjWXtOBJIZlFJjCGfrGO/UtOAHwhM/576rXQv
Nl+bpZPYsYtO9BrBliCbeLRc2s+FWY6p7k4IOKrYYJaxmS8klw+bROExmC8=
-----END CERTIFICATE-----
```


The file can now be copied to the machines where TAMOS is to be configured.

Step 2 - Configure TAMOS to use the certificate

Unlike Tivoli Access Manager for e-business (TAMeb), TAMOS must use SSL when communicating with the user registry. The certificate file that matches the server certificate must be available at configuration time. If you are using the InstallShield installation methods, you need to have the certificate file available prior to starting InstallShield. If you are using native methods, as we did, you can install the product without the certificate file, but you must have it prior to running the configuration utility (pdoscfg).

Attention: If you are installing TAMOS onto a system that already has other access manager components running and they do no use SSL to talk to the z/OS LDAP, make sure that you check the ssl_port setting in the /opt/PolicyDirector/etc/ldap.conf file.

When copying the file from the z/OS system to Linux for System z you need to make sure that any codepage differences are catered for with the Base64-encoded file. We used ftp in ASCII mode.

Once the file is on the TAMOS system, you might want to check that is has been transferred successfully, as shown in Example B-21.

Example: B-21 Checking the Base64-encoded certificate file

```
linux4z:/var/idsldap/V6.0 # file zos_ldap_svr.b64
zos_ldap_svr.b64: ASCII text
linux4z:/var/idsldap/V6.0 # cat zos_ldap_svr.b64
-----BEGIN CERTIFICATE-----
MIICODCCAaGgAwIBAgIIRyI/wAAKSFQwDQYJKoZIhvcNAQEFBQAwNDELMAkGA1UE
BhMCVVMxDTALBgNVBAoTBGlOc28xFjAUBgNVBAMMDXpvc19sZGFwX3N2cjIwHhcN
MDcxMDI2MTkyODAwWhcNMTcxMDIzMTkyODAwWjAOMQswCQYDVQQGEwJVUzENMAsG
A1UEChMEaXRzbzEWMBQGA1UEAwwNem9zX2xkYXBfc3ZyMjCBnzANBgkqhkiG9wOB
AQEFAAOBjQAwgYkCgYEAuDeBajuxQ3xyQCoSowPc63Cr31LeTCfxi9Mb6EZvWqbJ
y9KiMszxx8YxNoK/CrF7k2tsWoPvcCAuIx7TlomcTtY8bNZQZploaJKv3o1VDdgE
mAT1gDYQxS/35HpmqSVEgev1HdpZUASqJIiPpKlvDZO2uEMjVb8EEwbarlega3cC
AwEAAaNTMFEwHQYDVROOBBYEFJm/znPL/aLOZ4zOAIQBcpvTzkzFMB8GA1UdIwQY
MBaAFJm/znPL/aLOZ4zOAIQBcpvTzkzFMA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZI
hvcNAQEFBQADgYEALl7Qk99oV1oUOyZ3Nlmk+c9I/zIojAuZQFWywJ7QXDIXwhQU
nInNvi64FZ2aOXtrNJ7G8nIPWfdwjWXtOBJIZlFJjCGfrGO/UtOAHwhM/576rXQv
Nl+bpZPYsYtO9BrBliCbeLRc2s+FWY6p7k4IOKrYYJaxmS8klw+bROExmC8=
-----END CERTIFICATE-----
```

Now run the pdoscfg utility specifying the certificate file (as shown in Example B-22).

Example: B-22 pdoscfg using the self-signed certificate file

```
linux4z:/var/idsldap/V6.0 # pdoscfg -admin_name sec_master -admin_pwd secret
-registry_ssl_cacert /var/idsldap/V6.0/zos_ldap_svr.b64 -branch zlinux -suffix
o=itso
Determining user registry type.
Gathering information.
Processing the current configuration files.
Processing the command line.
Validating the information.
Configuring IBM Tivoli Access Manager for Operating Systems.
Configuring the PDOSD daemon.
Initializing the Tivoli Access Manager Policy Server context.
Registering with Tivoli Access Manager Policy Server. This may take a few minutes.
Registering the machine-specific policy information.
Initializing the PDOSD daemon's Trusted Computing Base.
Configuring the PDOSAUDITD daemon.
Configuring the PDOSWDD daemon.
Configuring the kernel extensions.
Configuring the PDOSLPMD daemon.
```

```
Tivoli Common Directory support has been enabled.  The directory is
/var/ibm/tivoli/common/AOS.
The configuration process completed successfully.
```

TAMOS will now connect to the user registry (z/OS LDAP) over SSL.

Configuring SSH between TCIM and the TAM servers

To collect TAMeb and TAMOS audit logs from their Linux for System z systems, you need to use a Tivoli Compliance Insight Manager (TCIM) SSH actuator running on the TCIM server (the TAM and TAMOS actuators are not supported on Linux for System z).

TCIM V8.0 uses PuTTY as the SSH client to connect from the point of presence (TCIM server) to the audited system (TAM or TAMOS running on Linux for System z in our case). It relies on a public-private key pair for the SSH connection to Linux for System z.

In the following steps we used OpenSSH to generate the key pair and manage the public key on the Linux for System z system.

The steps are:

1. Load the PuTTY utilities on the TCIM server.
2. Generate a key pair on the Linux for System z system.
3. Copy a private key to the TCIM server.
4. Convert the private key to PuTTY file format.
5. Set up a PuTTY link for TCIM use.
6. Configure TCIM for SSH.

Step 1 - Load the PuTTY utilities on the TCIM server

The TCIM server needs the following PuTTY executables:

- plink.exe - a command-line interface to the PuTTY utility so that you can run remote commands over SSH
- pscp.exe - a secure-copy client, used to copy the private key securely to the TCIM server
- putty.exe - the Telnet and SSH client
- puttygen.exe - a RSA/DSA key generation utility

These are available from a number of sites on the Web, but we used the *PuTTY Download Page* at:

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

These files can be placed anywhere on the TCIM server. We put them in c:\putty.

Next copy the plink.exe to the TCIM \Server\bin directory. In our case this was c:\IBM\TCIM\Server\bin.

This step only needs to be performed once.

Step 2 - Generate key pair on Linux for System z system

Before creating the key pair, you need to define the TCIM user (the user the SSH actuator on the TCIM server will use to connect to the Linux for System z system). See the *IBM Tivoli Compliance Insight Manager: Installation Guide*, GI11-8176-00, for details of creating this user.

Log on to the Linux for System z system as this user. The key generation procedure will depend on the SSH software installed. In our case, the OpenSSL SSH (OpenSSH) was installed, so we used the **ssh-keygen** command to generate a public/private rsa key pair. We created a user insight and su'd to the user from root before running the **ssh-keygen** command. An example is shown in Example B-23.

Example: B-23 ssh-keygen to generate RSA public/private key pair

```
linux11z:/var/pdos/audit # su insight
sh-2.05b$ cd ~
sh-2.05b$ pwd
/home/insight
sh-2.05b$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/insight/.ssh/id_rsa):
Created directory '/home/insight/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/insight/.ssh/id_rsa.
Your public key has been saved in /home/insight/.ssh/id_rsa.pub.
The key fingerprint is:
1c:e8:43:05:7e:27:01:d0:55:b7:b9:a3:89:0b:96:2f insight@linux11z
```

For the TCIM SSH connection we do not need a passphrase.

Next we need to copy the public key into the OpenSSH authorized_keys file, as shown in Example B-24.

Example: B-24 Add new key pair to SSH authorized keys file

```
sh-2.05b$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Note: Check that the ~/.ssh directory and its files are not world readable or world writeable. OpenSSH will not work with any files or directories that are world readable or writeable.

You could use this public key on every Linux for System z system that TCIM needs to communicate with. To do so, copy the id_rsa.pub file to the other Linux for System z systems and copy into the authorized_keys files there.

If you want a unique public/private key pair for each server, you need to repeat these steps on each Linux for System z system.

Step 3 - Copy private key to TCIM server

On the TCIM server use the **pscp.exe** command to copy the private key from the Linux for System z system to the TCIM server (see Example B-25).

Example: B-25 pscp.exe to copy private key to TCIM server

```
C:\putty>pscp.exe insight@linux11z.itso.ibm.com:.ssh/id_rsa .
Using keyboard-interactive authentication.
Password:
id_rsa                    | 0 kB |   0.9 kB/s | ETA: 00:00:00 | 100%

C:\putty>dir
 Volume in drive C has no label.
 Volume Serial Number is A03B-AA88

 Directory of C:\putty

10/30/2007  06:16 PM    <DIR>          .
10/30/2007  06:16 PM    <DIR>          ..
10/30/2007  06:16 PM               887 id_rsa
```

Step 4 - Convert private key to PuTTY file format

Next use the puttygen.exe GUI to convert the id_rsa private key file into PuTTY format (*.ppk file). Run the puttygen.exe to bring up the GUI, as shown in Figure B-1.

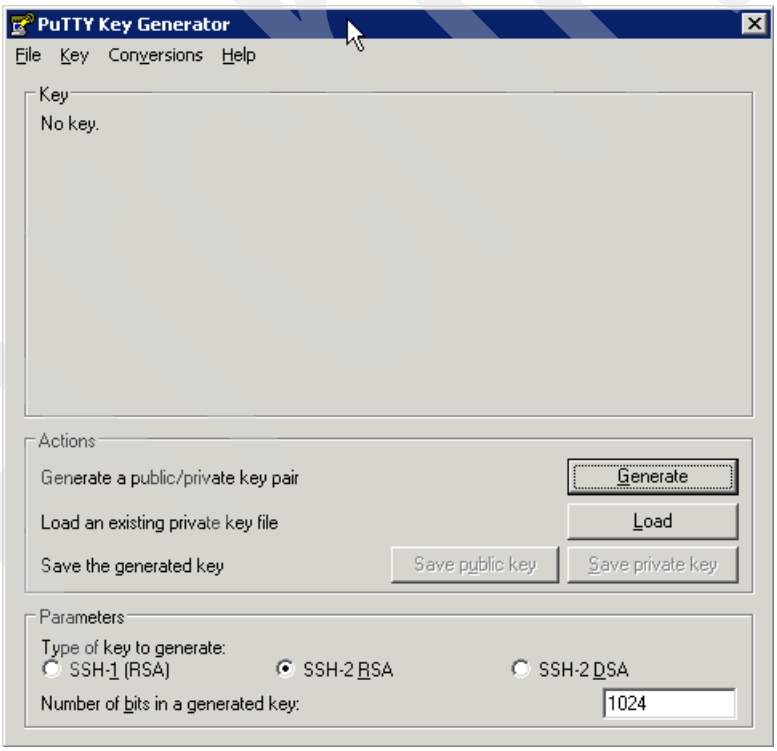


Figure B-1 PuTTY Key Generator GUI

First load the id_rsa file by clicking the **Load** button. In the previous step we saved the file in the c:\putty directory, so it is loaded from there, as shown in Figure B-2.

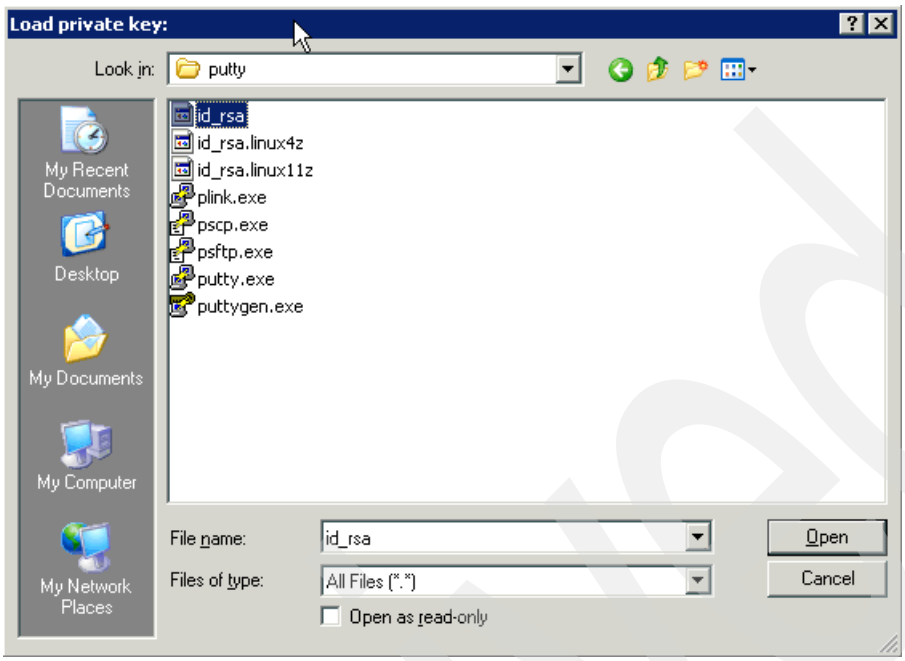


Figure B-2 puttygen loading id_rsa file

You will see a confirmation notice, as shown in Figure B-3.

Figure B-3 puttygen load successful message

The puttygen GUI now shows the details of the loaded private key, as in Figure B-4.

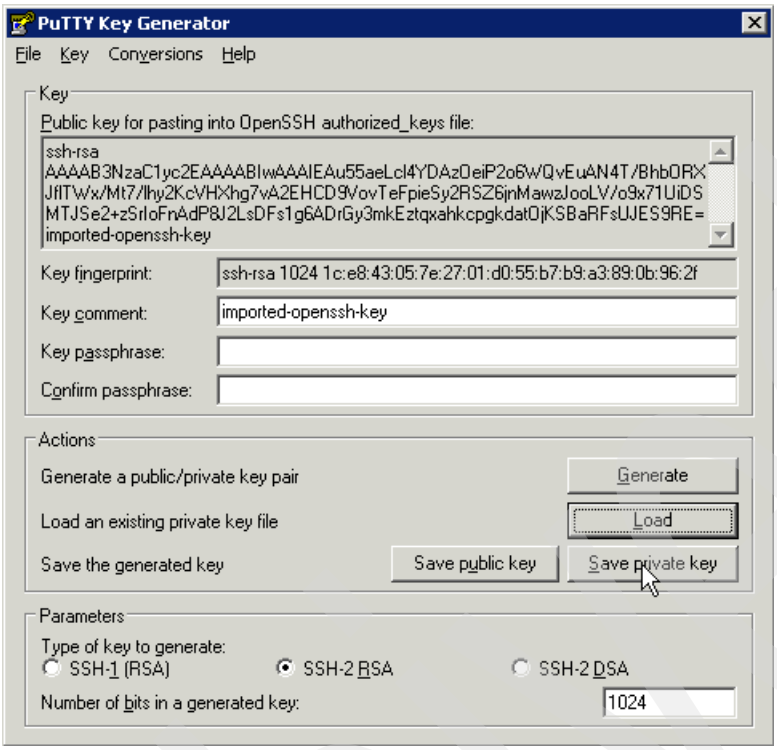


Figure B-4 puttygen loaded private key

This key must now be saved as a private key in the format that PuTTY can access. Click the **Save private key** button. As we saved the private key without a passphrase, puttygen will prompt for confirmation, as shown in Figure B-5.

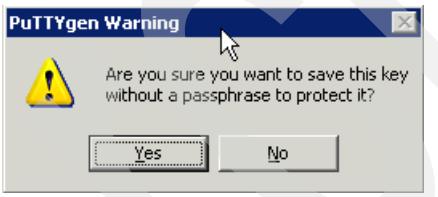


Figure B-5 puttygen no passphrase confirmation dialog

The private key needs to be saved in the PuTTY Private Key file format (*.ppk) so that the PuTTY utilities can use it. It must be saved in the TCIM Server\run\SSHKeys directory so that the TCIM server can access it. We saved it to c:\IBM\TCIM\Server\run\SSHKeys, as shown in Figure B-6.

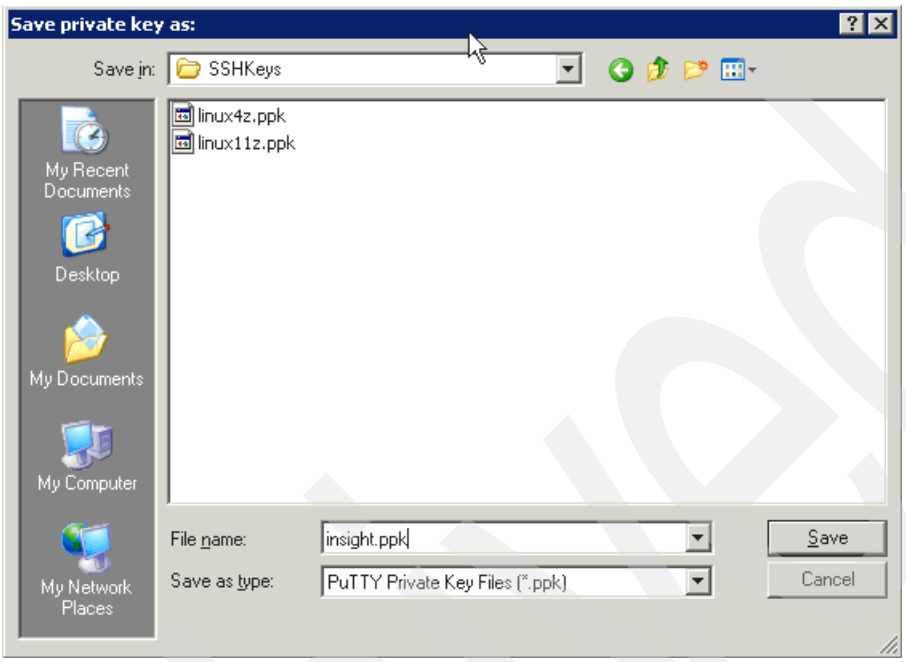


Figure B-6 puttygen file save dialog

This file is now ready to be used by PuTTY.

Example: B-26 TCIM SSHKeys directory with new private key file

```
C:\IBM\TCIM\Server\run\SSHKeys>dir
 Volume in drive C has no label.
 Volume Serial Number is A03B-AA88

 Directory of C:\IBM\TCIM\Server\run\SSHKeys

10/30/2007  06:32 PM    <DIR>          .
10/30/2007  06:32 PM    <DIR>          ..
10/30/2007  06:32 PM               847 insight.ppk
10/30/2007  04:13 PM               847 linux11z.ppk
10/30/2007  03:44 PM               851 linux4z.ppk
               3 File(s)          2,545 bytes
               2 Dir(s)  149,444,063,232 bytes free
```

Step 5 - Set up PuTTY link for TCIM use

Prior to configuring TCIM to use the private key file for the SSH actuators, we need to set up the puttylink mechanism for unattended operation. This involves running the **plink** command as the TCIM server account and letting the system cache the server's host key.

Log in to the TCIM server as the TCIM server administrator account (such as cearoot). Open a command window and change to the TCIM \Server\bin directory. Run the **plink.exe** command with the new .ppk file for the user, as shown in Example B-27 on page 327. You will be asked whether you want to add the key to PuTTY's cache. You must do this for the TCIM SSH actuators to work. Rerun the command to confirm that you are not prompted again.

Example: B-27 PuTTY plink to set up host key cache

```
C:\Documents and Settings\cearoot>cd c:\ibm\tcim\server\bin

C:\IBM\TCIM\Server\bin>plink.exe -i ..\run\SSHKeys\linux11z.ppk insight@linux11z.
itso.ibm.com ls
The server's host key is not cached in the registry. You
have no guarantee that the server is the computer you
think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 d8:c9:47:25:6c:69:20:57:2f:5f:01:fd:05:5f:18:0f
If you trust this host, enter "y" to add the key to
PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without
adding the key to the cache, enter "n".
If you do not trust this host, press Return to abandon the
connection.
Store key in cache? (y/n) y
test.sh
test2.sh

C:\IBM\TCIM\Server\bin>plink.exe -i ..\run\SSHKeys\linux11z.ppk insight@linux11z.
itso.ibm.com ls
test.sh
test2.sh

C:\IBM\TCIM\Server\bin>
```

Now everything is ready to configure TCIM for SSH.

Step 6 - Configure TCIM for SSH

Rather than showing all the steps to create the event sources, we only show the SSH configuration. The details of this are covered in Chapter 9, "Enabling collect using SSH event sources," in the *IBM Tivoli Compliance Insight Manager: Installation Guide,* GI11-9176. For more details see the IBM Redbooks publication *Compliance Management Design Guide with IBM Tivoli Compliance Insight Manager*, SG24-7530.

Figure B-7 shows the event source properties for a TAMeb through SSH event source. There are three SSH-related settings:

- SSH keyfile: This is the .ppk file that we created earlier and tested with plink.exe.
- SSH port: This is normally port 22 (by convention).
- SSH user: This is the user on the audited machine that we used to create the public/private key pair (such as the user insight).

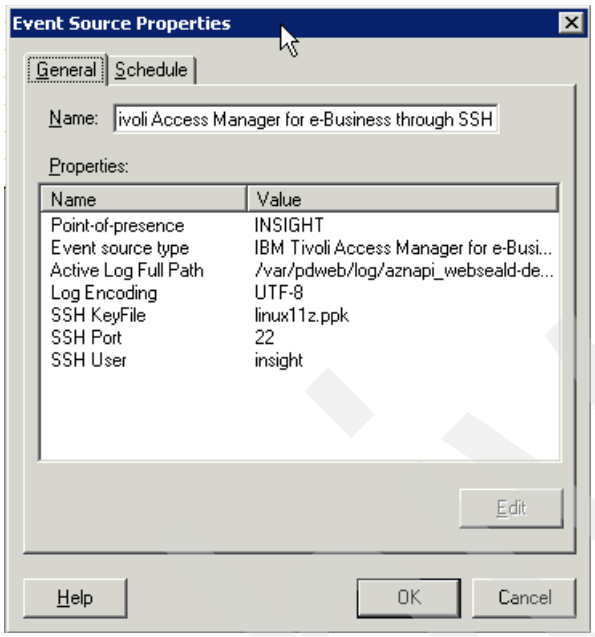


Figure B-7 TAMeb through SSH Event Source Properties

Figure B-8 shows the event source properties for a TAMOS through SSH event source. Notice that the SSH settings are the same.

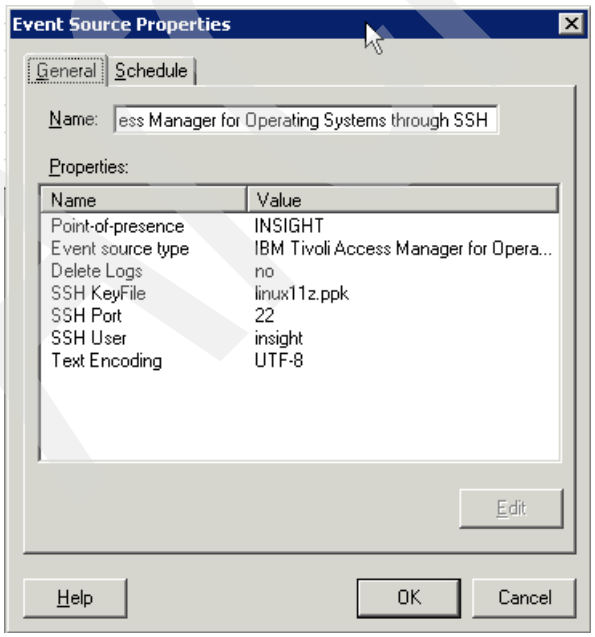


Figure B-8 TAMOS through SSH Event Source Properties

Important: Make sure that the SSH KeyFile and SSH user are correct for the system that you will be collecting the logs from.

This concludes the configuration steps. Once the event source is configured, you should be able to collect the relevant log files. You can check for errors or other messages on the Linux for System z system. In the user's home directory there will be directories created for each collection of the form nn.x.yyyy.previous (as shown in Example B-28).

Example: B-28 TCIM collection log directories

```
linux4z:/home/insight # ls -ltr | grep previous
drwxr-xr-x   5 insight users 4096 Oct 30 14:54 18.1.116.previous
drwxr-xr-x   5 insight users 4096 Oct 30 14:54 18.1.117.previous
drwxr-xr-x   5 insight users 4096 Oct 30 15:47 18.1.120S.previous
drwxr-xr-x   5 insight users 4096 Oct 30 15:47 18.1.121S.previous
```

Within each directory are bin, log, and run directories. If there are entries/updates that match the time of the latest collection in TCIM, then the SSH connection is working (any errors encountered with the collection will not be due to the SSH connection).

C

Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

ftp://www.redbooks.ibm.com/redbooks/SG247472

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247472.

Using the Web material

The additional Web material that accompanies this book includes the following files:

File name	*Description*
sg247472.zip	Zipped code samples and instructions

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 334. Note that some of the documents referenced here may be available in softcopy only.

- *Understanding LDAP - Design and Implementation*, SG24-4986
- *WebSphere Application Server for z/OS V5 and J2EE 1.3 Security Handbook*, SG24-6086
- *Linux on IBM zSeries and S/390: Securing Linux for zSeries with a Central z/OS LDAP Server (RACF)*, REDP-0221
- *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-6014
- *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-6014

Other publications

These publications are also relevant as further information sources:

- *IBM Tivoli Directory Server Administration Guide*, SC32-1674
- *z/OS V1R9.0 Security Server RACF Security Administrator's Guide*, SA22-7683-11
- *z/OS V1R9.0 Security Server RACF Auditor's Guide*, SA22-7684-09
- *z/OS V1R9.0 MVS System Management Facilities (SMF)*, SA22-7630-15

Online resources

These Web sites are also relevant as further information sources:

- Method of authentication done by RSA

  http://www.rsa.com/rsalabs/staff/bios/ajuels/publications/fourth-factor/ccs084-juels.pdf

- Tivoli Identity Manager

  http://www.ibm.com/software/tivoli/products/identity-mgr

- IT Governance Institute:

  http://www.itgi.org/
  http://www.isaca.org

- IBM Data Governance Council

  http://www.ibm.com/software/tivoli/governance/servicemanagement/data-governance.html

- ITDI product manuals

  http://www.ibm.com/software/tivoli/products/directory-integrator/

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

R

S

T

U

V

W

Z

IBM

Redbooks

Enterprise Multiplatform Auditing

Enterprise Multiplatform Auditing

Auditing tools available for the enterprise

Auditing in a WebSphere Application Server environment

How to audit the privilege users

In today's global marketplace clients must demonstrate compliance with laws like Sarbanes-Oxley, HIPPA, the EU 8th directive, and privacy laws in an enterprise (horizontal) environment. The environments become more and more complex with the rapid growth of e-business, and they often span several geographies. Most IT organizations are still very much vertical, often with different organizations within each country, with little cooperation between them. Lack of skills on cross-platform skills are also a major issue.

Integration and automation at the infrastructure layer is key to enabling e-business. With the growing number of security databases, proving compliance across the enterprise is very complex. This is also true for auditors, as they need a very deep knowledge of IT and a variety of solutions and IT Infrastructures to be able to do a reliable audit.

This IBM Redbooks publication documents the results of our efforts and tests to show how you can perform audit trail and report in an enterprise. Our configuration, even though a not very complex one, shows results of using internal (intranet) and external (internet) configurations. We have been testing with standard users and privilege users, using tools that are available in today's market.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7472-00 ISBN 0738485047