

Oracle Database 12c: Security

Student Guide - Volume I

D81599GC10

Edition 1.0

June 2014

D84089

ORACLE®

Author

Dominique Jeunot

Technical Contributors and Reviewers

Jean-François Verrier

Donna Keesling

James Spiller

Gerlinde Frenzen

Pat Huey

veerabhadra Rao Putrevu

Joel Goodman

Editors

Malavika Jinka

Anwesha Ray

Daniel Milne

Graphic Designer

Rajiv Chandrabhanu

Publishers

Jobi Varghese

Pavithran Adka

Sumesh Koshy

Srividya Rameshkumar

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

1 Database Security: Introduction

Course Objectives 1-2

Appendices 1-5

Prerequisites 1-6

Practice: Overview 1-7

2 Understanding Security Requirements

Objectives 2-3

Fundamental Data Security Requirements 2-4

Data Security Concerns 2-6

Compliance Mandates 2-7

Security Risks 2-9

Security Standards 2-11

Developing Your Security Policy 2-12

Defining a Security Policy 2-13

Implementing a Security Policy 2-15

Quiz 2-16

Enforcing Security at Different Levels 2-17

Principle of Least Privilege 2-18

Defense in Depth 2-19

Common Exploits 2-20

Preventing Exploits 2-22

Summary 2-24

Practice: Overview 2-25

3 Choosing Security Solutions

Objectives 3-3

Database Security Solutions 3-4

Encrypting Network Traffic 3-5

Authenticating Users to Database 3-7

Encrypting Application Data with DBMS_CRYPTO 3-9

Encrypting Application Data with TDE 3-10

Redacting Sensitive Data Displayed 3-12

Masking Data for Non-Production Use 3-13

Protecting Multiple Sensitive Columns 3-14

Controlling Data Access Using Authorization	3-15
Controlling Data Access Based on Views	3-16
Controlling Data Access by Privileged Users	3-17
Controlling Data Access Based on Function	3-19
Controlling Backup Access Based on Privilege	3-20
Controlling Data Access Based on Label	3-21
Monitoring Database Activity	3-23
Oracle Audit Vault and Database Firewall	3-24
Auditing and Alerting in Real-Time	3-26
Unified Auditing	3-27
Fine-Grained Auditing	3-28
Discovering Use of Privileges and Roles	3-29
Discovering Sensitive Data	3-30
Managing Configuration, Compliance, and Change Management	3-31
Enforcing Security at Different Levels	3-33
Quiz	3-34
Summary	3-35
Practice: Overview	3-36

4 Implementing Basic Database Security

Objectives	4-3
Database Security: Checklist	4-4
Reducing Administration Effort	4-5
Installing Only What Is Required	4-6
Applying Security Patches	4-7
Secure Password Support	4-8
Automatic Secure Configuration	4-9
Locking, Expiring, or Changing Passwords of Default Accounts	4-10
Privileged Accounts	4-12
Administrative Privileges	4-13
Limiting Users with Administrative Privileges	4-14
Separation of Responsibilities	4-16
Protecting the Data Dictionary	4-18
Limiting Privileges	4-19
Limiting External Procedures Privileges	4-21
Managing Scheduler Security	4-22
External Jobs	4-23
Summary	4-24
Practice: Overview	4-25

5 Securing Network Services

- Objectives 5-3
- Network Security: Checklist 5-4
- Restricting Network IP Addresses: Valid Node Checking 5-5
- Restricting Network IP Addresses: Guidelines 5-6
- Configuring IP Restrictions with Net Manager 5-7
- Quiz 5-8
- Listener Security: Checklist 5-9
- Restricting Nodes Registration: Valid Node Checking Registration (VNCR) 5-10
- Moving the Listener to a Nondefault Port 5-11
- Administering the Listener 5-12
- Administering the Listener Securely 5-13
- Preventing Online Administration of the Listener 5-14
- INBOUND_CONNECT_TIMEOUT 5-15
- Setting Listener-Logging Parameters 5-17
- Analyzing Listener Log Files 5-19
- Listener Log Connect: Examples 5-21
- Listener Log Command: Examples 5-23
- Managing Fine-Grained Access to External Network Services 5-25
- Summary 5-26
- Practice: Overview 5-27

6 Using Basic and Strong User Authentication

- Objectives 6-3
- User Authentication 6-4
- Basic User Authentication by Password 6-6
- Protecting Passwords 6-8
- Secure External Password Store 6-9
- Basic User Authentication by Operating System 6-10
- Quiz 6-11
- Strong User Authentication 6-12
- Public Key Infrastructure (PKI) Tools 6-15
- Certificates 6-16
- How to Use Certificates for Authentication 6-17
- Quiz 6-19
- How to Use Kerberos for Authentication 6-20
- How to Use KDC with Windows 2000 for Authentication 6-22
- RADIUS Authentication: Overview 6-24
- Fixed User Database Links 6-25
- Database Links Without Credentials 6-26
- Database Links and Changing Passwords 6-28

Auditing with Database Links 6-29
Restricting a Database Link with Views 6-30
Summary 6-32
Practice: Overview 6-33

7 Using Global User Authentication

Objectives 7-3
Enterprise User Security 7-4
Oracle Identity Management Software 7-5
Directory Structure: Overview 7-7
Oracle Database: Enterprise User Security Architecture 7-8
Authenticating Enterprise Users 7-9
Enterprise Users 7-11
Setting Up Enterprise User Security 7-12
Identifying the Enterprise User 7-13
Enabling Current User Database Links 7-14
Using Enterprise Roles 7-15
User Migration Utility 7-16
Enterprise-User Auditing 7-18
Quiz 7-19
Summary 7-20
Practice or Demo: Overview 7-21

8 Using Proxy Authentication

Objectives 8-3
User Authentication 8-4
Security Challenges of Three-Tier Computing 8-5
Common Implementations of Authentication 8-6
Quiz 8-8
Using Proxy Authentication for Database Users 8-9
Using Proxy Authentication for Enterprise Users 8-11
Revoking Proxy Authentication 8-12
Application-User Model 8-13
Using Proxy Authentication with Roles 8-15
Data Dictionary Views for Proxy Authentication 8-16
Data Dictionary Views: DBA_PROXIES and USER_PROXIES 8-17
Data Dictionary Views: V\$SESSION_CONNECT_INFO 8-18
Auditing Actions Taken on Behalf of the Real User 8-19
Summary 8-20
Practice: Overview 8-21

9 Using Privileges and Roles

- Objectives 9-3
- Authorization 9-4
- Administrative Privileges 9-5
- New Administrative Privileges 9-6
 - New Administrative Privilege: SYSBACKUP 9-7
 - New Administrative Privilege: SYSDG 9-8
 - New Administrative Privilege: SYSKM 9-9
- OS Authentication and OS Groups 9-10
- Password Authentication for SYSBACKUP 9-12
- Password Authentication for SYSDG 9-14
- Roles 9-15
 - Benefits of Roles 9-16
 - Creating Common and Local Roles 9-17
 - Granting Common and Local Privileges 9-18
 - Granting Common or Local Privileges to Roles, Common or Local Roles to Roles 9-19
- Granting Common and Local Roles to Users 9-20
- Predefined Roles 9-21
- Quiz 9-22
 - Secure Application Role 9-23
 - Implementing a Secure Application Role 9-24
 - Securing Objects with Procedures 9-25
 - Using Code-Based Access Control 9-26
 - Privilege Checking During PL/SQL Calls 9-27
 - INHERIT (ANY) PRIVILEGES Privileges 9-28
 - Privilege Checking with New BEQUEATH Views 9-29
 - Quiz 9-30
 - Creating and Using Virtual Private Catalogs 9-31
 - Summary 9-33
 - Practice: Overview 9-34

10 Using Privilege Analysis

- Objectives 10-3
- Privilege Analysis 10-4
- Privilege Analysis Flow 10-5
- Creating Policies: Database and Role Analysis 10-6
- Creating Policies: Context Analysis 10-7
- Creating Policies: Combined Analysis Types 10-8
- Analyzing and Reporting 10-9

SYSTEM and OBJECT Used Privileges 10-10
Compare Used and Unused Privileges 10-12
Views 10-13
Dropping an Analysis Policy 10-14
Quiz 10-15
Summary 10-16
Practice: Overview 10-17

11 Using Application Contexts

Objectives 11-3
Application Context: Description 11-4
Using the Application Context 11-5
Setting and Retrieving Application Context Values 11-6
Application Context Data Sources 11-7
Quiz 11-9
Application Context Accessed Globally 11-10
Application Context Accessed Globally in Action 11-12
Implementing the Application Context Accessed Globally 11-14
Viewing Application Context Information 11-15
Application Context Usage Guidelines 11-16
Summary 11-18
Practice: Overview 11-19

12 Implementing Virtual Private Database

Objectives 12-3
Fine-Grained Access Control: Overview 12-4
Understanding FGAC Policy Execution 12-6
Benefits of Using Fine-Grained Access Control 12-8
Virtual Private Database 12-9
Examples of VPD 12-10
Using DBMS_RLS to Manage Policies 12-11
Column-Level VPD 12-12
Policy Types: Overview 12-13
Designing and Implementing a VPD Solution 12-14
Implementing a VPD Policy 12-15
Writing a Function That Returns Different Predicates 12-16
Exceptions to VPD Policies 12-17
Quiz 12-18
Guidelines for Policies and Context 12-19
Policy Performance 12-21
Export and Import 12-23

Policy Views 12-24
Summary 12-25
Practice: Overview 12-26

13 Implementing Oracle Label Security

Objectives 13-3
Access Control: Overview 13-4
OLS 13-5
Enabling and Managing OLS 13-6
Quiz 13-7
OLS: Features 13-8
OLS and VPD Comparison 13-10
Analyzing Application Requirements 13-11
Implementing an OLS Solution 13-12
Create Policies With Enforcement Options 13-14
Define Labels 13-16
Assign User Authorization Labels 13-18
Apply the Policy to a Table 13-20
Adding Labels to Data 13-21
Access Mediation 13-22
Quiz 13-23
OLS Special User Privileges 13-24
Example: READ Privilege 13-25
Example: FULL Privilege 13-26
Example: COMPACCESS Privilege 13-27
Using the PROFILE_ACCESS Privilege 13-28
Trusted Stored Package Units 13-29
Exporting and Importing with OLS 13-30
Performance Tips 13-31
Summary 13-32
Practice: Overview 13-33

14 Oracle Data Redaction

Objectives 14-3
Oracle Data Redaction: Overview 14-4
Oracle Data Redaction and Operational Activities 14-6
Available Redaction Methods 14-7
Oracle Data Redaction: Examples 14-8
What Is a Redaction Policy? 14-9
Managing Redaction Policies 14-10
Defining a Redaction Policy 14-11

Adding a Redaction Policy to a Table or View	14-12
Full Redaction: Examples	14-13
Partial Redaction: Examples	14-14
Regular Expression	14-16
Modifying the Redaction Policy	14-17
Exempting Users from Redaction Policies	14-18
Defining Data Redaction Policies by Using Cloud Control 12c	14-19
Creating a Data Redaction Policy	14-20
Using Oracle Data Redaction with Other Oracle Database Security Solutions	14-22
Oracle Database Security Features	14-23
Best Practices: Preventing Unauthorized Policy Modifications and Exemptions	14-25
Best Practices: Considerations	14-26
Summary	14-27
Practice: Overview	14-28

15 Application Data Model and Oracle Data Masking

Objectives	15-3
Application Data Model: Overview	15-4
ADM and Data Masking Process	15-5
Creating an ADM	15-6
Viewing ADM Content	15-7
Discovering Sensitive Columns	15-8
Data Masking: Overview	15-9
Using the Data Masking Pack	15-10
Data Masking Pack: Features	15-11
Implementing Data Masking	15-12
Identifying Sensitive Data for Masking	15-14
Creating or Using Masking Formats	15-15
Using Oracle-Supplied Mask Formats and Built-in Masking Routines	15-16
Example: Data Masking of the EMPLOYEES Table	15-18
Creating a Masking Format Using a User-Defined Function	15-19
Creating Data Masking Definitions	15-20
Importing Formats and Modifying Properties	15-21
Using Condition-Based Masking	15-22
Using Compound Masking	15-23
Using a User-Defined Masking Function and Post-Processing Masking Function	15-24
Generating the Data Masking Script	15-25
Understanding the Data Masking Process	15-26

Creating an Application Masking Template 15-27
Controlling Data Masking Operations 15-28
Summary 15-29
Practice: Overview 15-30

16 Implementing Transparent Sensitive Data Protection

Objectives 16-3
Benefits of TSDP 16-4
TSDP: Overview 16-5
Using a TSDP Policy with VPD 16-7
Using a TSDP Policy with Data Redaction 16-9
Using the Predefined REDACT_AUDIT TSDP Policy 16-11
Disabling the REDACT_AUDIT TSDP Policy 16-13
TSDP Policies in a CDB and PDBs 16-14
Exempting Users from TSDP Policies 16-15
Maintaining TSDP Types and Sensitive Columns 16-16
Maintaining TSDP Policies 16-17
Data Dictionary Views for TDSP 16-18
Summary 16-19
Practice: Overview 16-20

17 Encryption Concepts

Objectives 17-3
Understanding Encryption 17-4
Encryption Is Not Access Control 17-5
Access by Privileged Users 17-6
What Problems Does Encryption Solve? 17-8
Cost of Encryption 17-9
What to Encrypt 17-10
Quiz 17-11
Data Encryption: Challenges 17-12
Encryption Key Management: Key Generation 17-13
Encryption Key Management: Key Modification and Transmission 17-14
Encryption Key Management: Storage 17-15
Storing the Key in the Database 17-16
Storing the Key in the File System 17-18
Letting the User Manage the Key 17-19
Solutions 17-20
Summary 17-21

18 Using Application-Based Encryption

- Objectives 18-3
- Overview 18-4
- DBMS_CRYPTO Package 18-5
- Generating Keys Using RANDOMBYTES 18-7
- Quiz 18-10
- Using ENCRYPT and DECRYPT 18-11
- Enhanced Security Using Cipher Block Modes 18-14
- Hash and Message Authentication Code 18-15
- Summary 18-18
- Practice: Overview 18-19

19 Applying Transparent Data Encryption

- Objectives 19-3
- Transparent Data Encryption 19-4
- Components of TDE 19-5
- Using TDE 19-6
- Using Hardware Security Modules 19-7
- Defining the Keystore Location 19-8
- Creating and Opening the Keystore 19-10
- Generating the Master Key 19-11
- Backing Up the Keystore 19-12
- Managing the Keystore in CDB and PDBs 19-13
- Creating and Opening the Keystore 19-14
- Setting Master Encryption Keys 19-15
- Quiz 19-16
- Re-Keying Table Keys 19-17
- Creating an Encrypted Column 19-18
- Creating an Index on an Encrypted Column 19-19
- TDE Column Encryption Support 19-20
- TDE Column-Level Storage Requirements 19-22
- TDE Column Encryption: Restrictions 19-23
- Tablespace Encryption: Advantages 19-24
- Creating an Encrypted Tablespace 19-25
- SECUREFILE LOB Encryption 19-26
- Summary 19-27
- Practice: Overview 19-28

20 Database Storage Encryption

- Objectives 20-3
- RMAN-Encrypted Backups 20-4

Oracle Secure Backup Encryption	20-5
Encrypted Backups to Tape	20-7
Creating RMAN-Encrypted Backups	20-8
Using Transparent-Mode Encryption	20-9
Using Password-Mode Encryption	20-11
Using Dual-Mode Encryption	20-12
Quiz	20-13
Restoring Encrypted Backups	20-14
RMAN-Encrypted Backups: Considerations	20-15
Data Pump Encryption	20-16
ENCRYPTION Parameter	20-17
ENCRYPTION_PASSWORD Parameter	20-18
ENCRYPTION_MODE Parameter	20-19
Encrypting Dump Files	20-20
Summary	20-21
Practice: Overview	20-22

21 Using Unified Auditing

Objectives	21-3
Types of Auditing	21-4
Oracle Database 12c Auditing	21-5
Security and Performance: Audit Architecture	21-6
Tolerance Level for Loss of Audit Records	21-7
Consolidation: Unique Audit Trail	21-8
Basic Audit Versus Extended Audit Information	21-9
Extended Audit Information	21-10
Data Pump Audit Policy	21-11
Oracle RMAN Audit Information	21-12
Unified Audit Implementation	21-13
Quiz	21-15
Security: Roles	21-17
Security: SYS Auditing	21-18
Simplicity: Audit Policy	21-19
Step 1: Creating the Audit Policy	21-20
Creating the Audit Policy: Object-Specific Actions	21-21
Creating the Audit Policy: Condition	21-22
Step 2: Enabling / Disabling the Audit Policy	21-23
Auditing Actions in a CDB and PDBs	21-24
Viewing the Audit Policy	21-25
Viewing the Audit Records CDB_UNIFIED_AUDIT_TRAIL	21-26
Using Predefined Audit Policies	21-27

Including Application Context Data 21-28
Dropping the Audit Policy 21-29
Audit Cleanup 21-30
Quiz 21-31
Summary 21-32
Practice: Overview 21-33

22 Using Fine-Grained Audit

Objectives 22-3
Fine-Grained Auditing (FGA) 22-4
FGA Policy 22-5
Triggering Audit Events 22-7
Data Dictionary Views 22-8
DBA_FGA_AUDIT_TRAIL / UNIFIED_AUDIT_TRAIL 22-9
Quiz 22-10
DBMS_FGA Package 22-11
Enabling / Disabling Dropping an FGA Policy 22-12
FGA Policy Guidelines 22-13
FGA Policy Errors 22-14
Maintaining the Audit Trail 22-15
Summary 22-16
Practice 22 Overview: Using Fine-grained Audit 22-17

A Implementing Virtual Private Database Policy Groups

Objectives A-2
Implementing Policy Groups A-3
Grouping Policies A-5
Default Policy Group A-6
Creating a Driving Context A-8
Making the Context a Driving Context A-10
Creating a Policy Group A-12
Adding a Policy to a Group A-13
Summary A-15

B Encrypting Network Traffic

Encrypting Network Traffic B-2
End-to-End Encryption B-4
Configuring Network Encryption B-5
Checksumming B-6
Configuring Checksumming B-7

C General Security Reports

- Using Database Vault Reports C-2
- General Security Reports C-3
- Database Account Password Reports C-4
- Using Powerful Database Accounts and Roles Reports C-5
- Privileged Database Accounts and Roles Reports C-6
- Initialization Parameter and Operating System Directory Permission Reports C-7
- Review Privilege Reports C-8
- General Database Privilege and Resource Profile Reports C-9
- Database Audit and Privilege Reports C-10
- Object Privilege Reports C-11
- Sensitive Objects Reports C-12
- Unified Audit Trail C-14
- Other Security Vulnerability Reports C-15

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

1

Database Security: Introduction



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Course Objectives

After completing this course, you should be able to do the following:

- Describe security risks and Oracle solutions
- Implement basic database security
- Ensure security over the network
- Choose a user-authentication model for database users and enterprise users
- Control data access
- Ensure data confidentiality
- Implement auditing

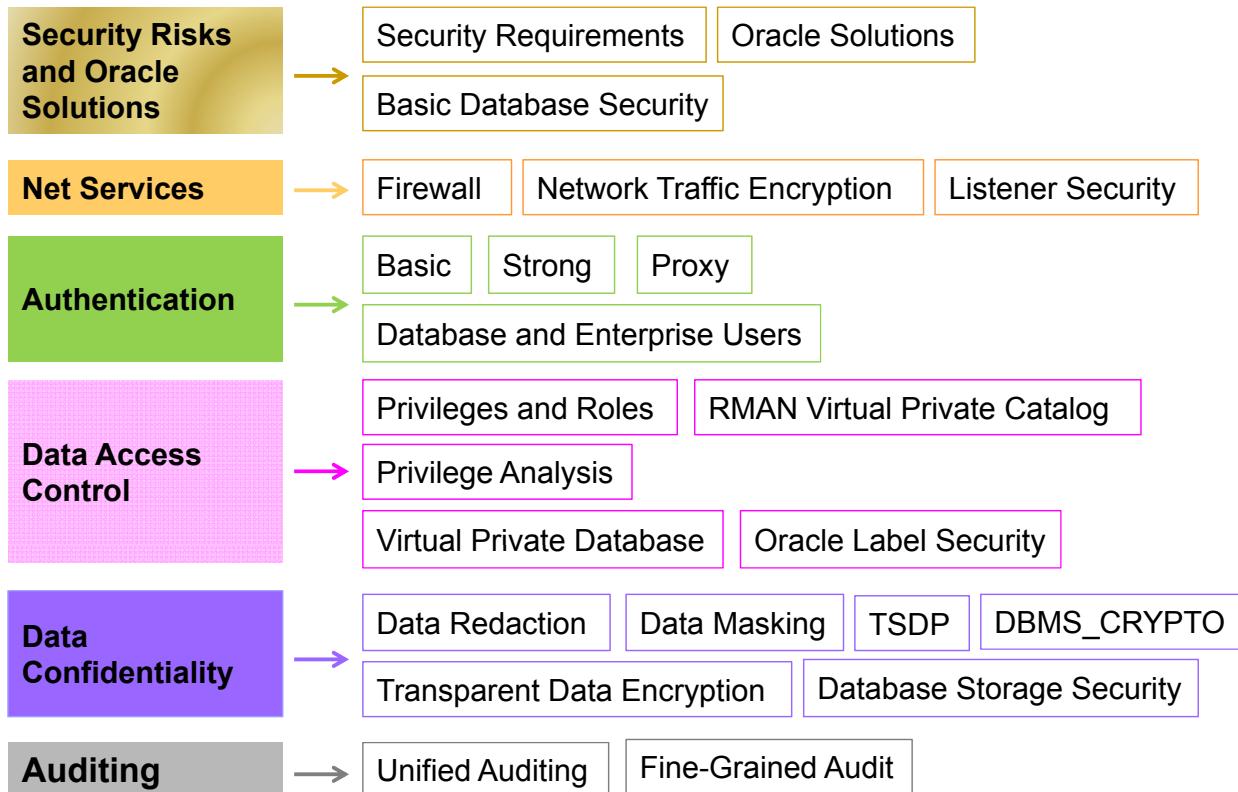


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete list of the new database security features for Oracle Database 12c Release 1, refer to the *Oracle Database New Features Guide 12c Release 1 (12.1)* guide in Oracle documentation.

Course Structure



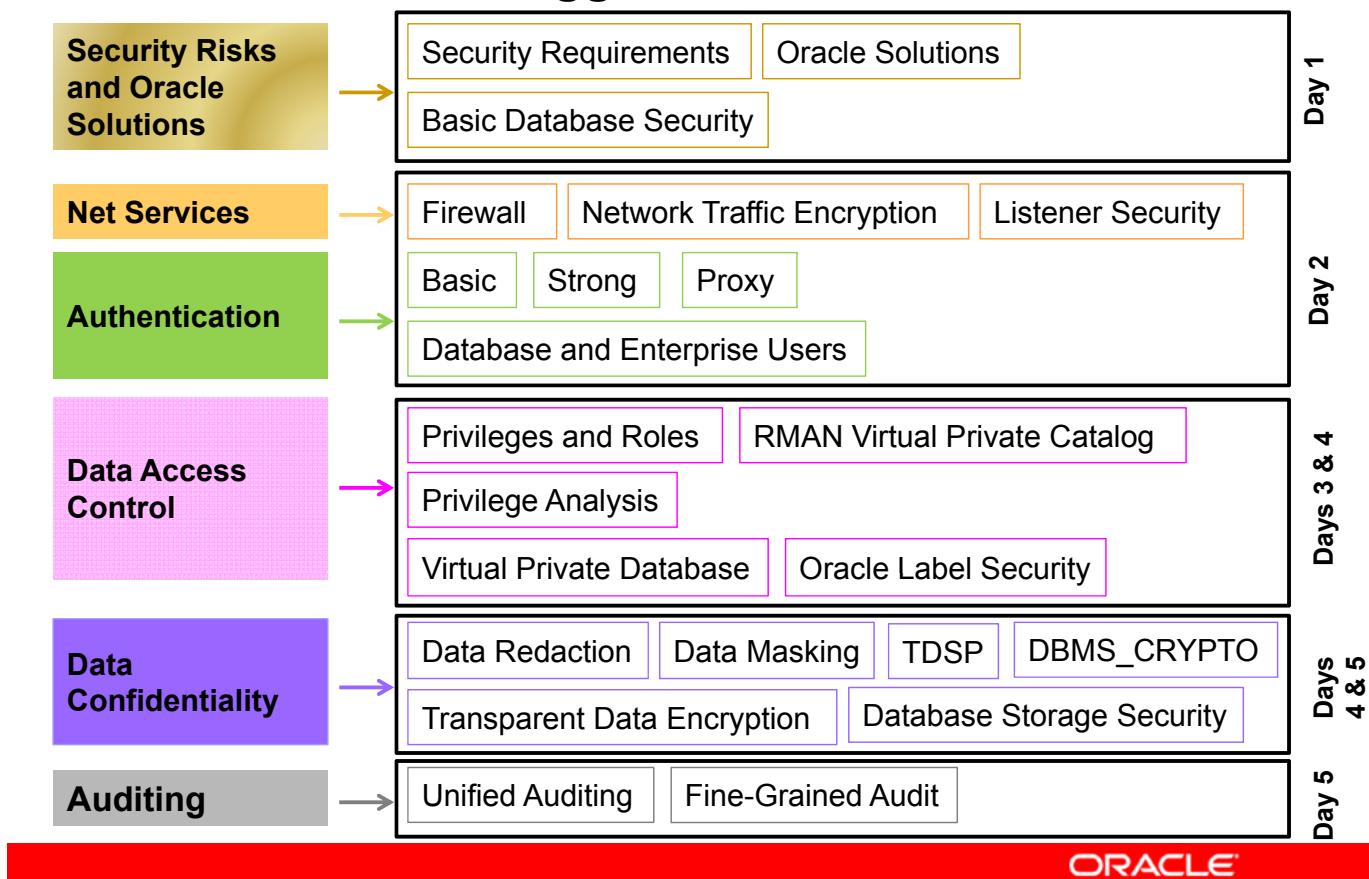
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The following lessons explain how to:

- Analyze the security risks of your organization
- Find appropriate Oracle solutions to meet the security, privacy, and compliance requirements of your organization
- Find solutions to secure database access through the network
- Configure appropriate authentication for the database or enterprise users in the organization
- Control data access and integrity in your organization by using an appropriate feature or option or product such as privileges, Oracle Label Security, or Audit Vault and Database Firewall
- Ensure data confidentiality by using an encryption solution such as Transparent Data Encryption, Data Redaction, or Oracle Data Masking
- Audit user actions by using any of the auditing features such as unified auditing or a product such as Audit Vault and Database Firewall

Suggested Schedule



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Appendices

Appendix	Topic
A	Oracle Virtual Private Database Policy Groups
B	Encrypting Network Traffic
C	General Security Reports
D	Source Code
E	USERENV and SYS_SESSION_ROLES Contexts



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Prerequisites

For this course, it is assumed that you have attended the Oracle Database 12c: Administration Workshop (or equivalent experience).



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

Practice 1-1 covers environment familiarization.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

Module

Security Risks and Oracle Solutions

A solid red horizontal bar spanning most of the slide width.

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

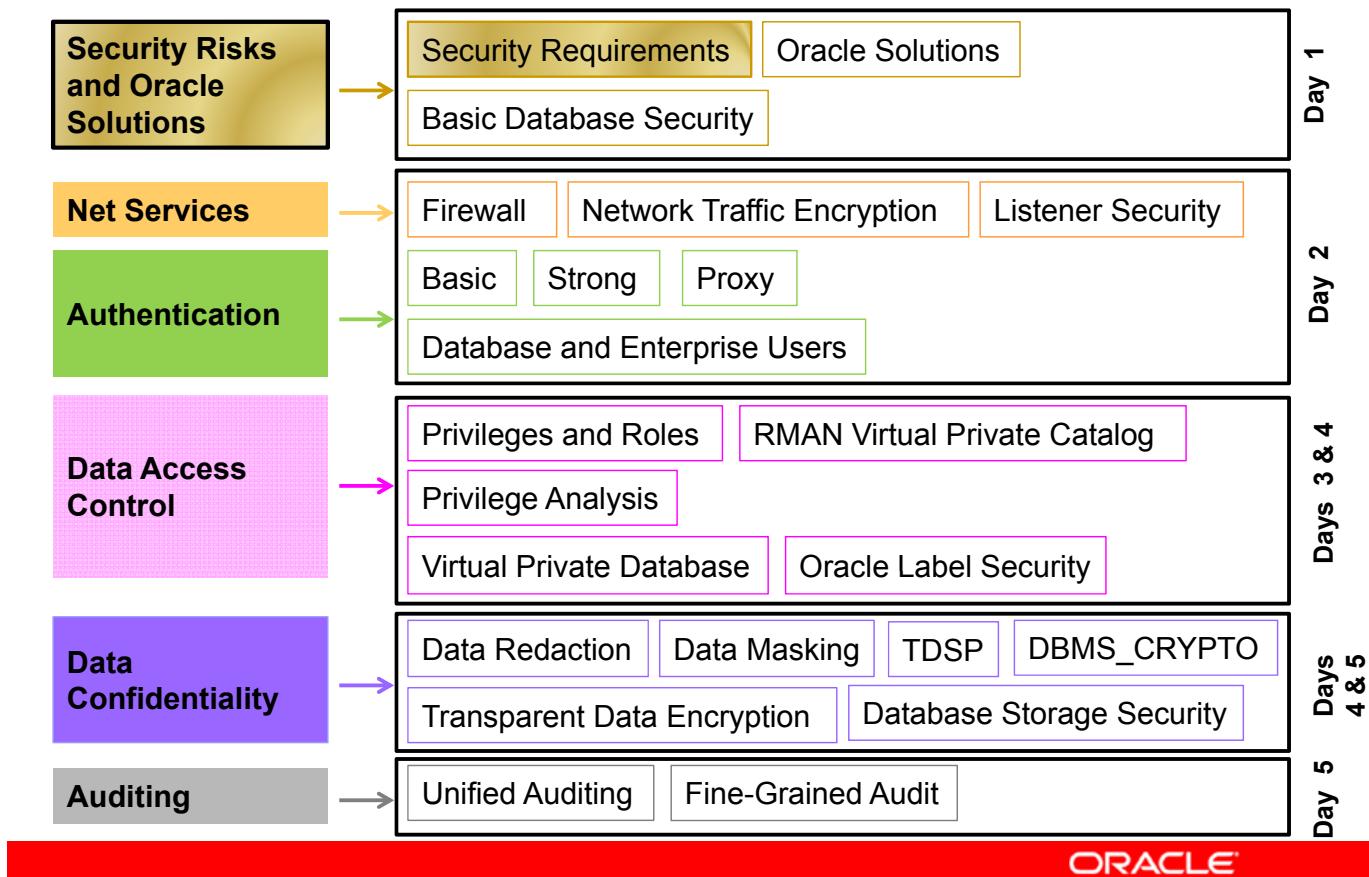
Understanding Security Requirements



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Schedule



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe business security requirements
- Define the following terms:
 - Least privilege
 - Authorization
 - Authentication
- Describe security policies
- Describe the concept of in-depth security
- Apply these concepts to prevent SQL injection



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Fundamental Data Security Requirements

Security standards require security policies to meet fundamental data security requirements:

- **Confidentiality:** Ensure that individuals see only the data they are supposed to see
- **Integrity:** Ensure that data remains valid
- **Availability:** Ensure that denial-of-service attacks do not impact access to and use of systems



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Security standards include both legal and business standards. These standards require that your security policy meet the following fundamental requirements:

- **Confidentiality:** Confidentiality ensures that individuals see only the data that they are supposed to see. This includes:
 - Privacy of communication
 - Secure storage of data, including backup media (may require encryption)
 - Secure transmission of sensitive data (may require network encryption)
 - Restricted access to services
 - Authenticated users (only authenticated users may see data)
 - Granular access control (only authorized users may view allowed data)
- **Integrity:** A secure system ensures that the data remains valid. Data integrity means that data is protected from deletion and corruption, both while it resides within the database and while it is being transmitted over the network. Integrity has several aspects:
 - System and object privileges control access to application tables and system commands, so that only authorized users can change data.

- Constraints contribute to integrity by enforcing rules on data to keep it valid. For example, referential integrity is a constraint that maintains valid relationships between entities in the database, according to the rules that have been defined.
- A database must be protected against viruses that are designed to corrupt the data.
- Network traffic must be protected from deletion, corruption, and eavesdropping.
- **Availability:** Availability is usually considered as a backup and recovery issue, and not as a security issue. However, denial-of-service (DoS) attacks attempt to block authorized users' ability to access and use the system when needed. Preventing DoS attacks is a security issue. These attacks usually gain unauthorized access to computers and then use these computers to generate requests that flood the targeted system.
Availability can sometimes be hindered by your own security measures. Very restrictive firewalls can protect your resources, but can also block access for authorized users.

Data Security Concerns

Security threats



Industrial espionage

Identity theft

Insider threats

- Data consolidation
- Globalization
- Right sourcing



Compliance mandates



SOX

HIPAA

PCI

Basel II

EU directives

FDA

GLBA

SB1386

ORACLE

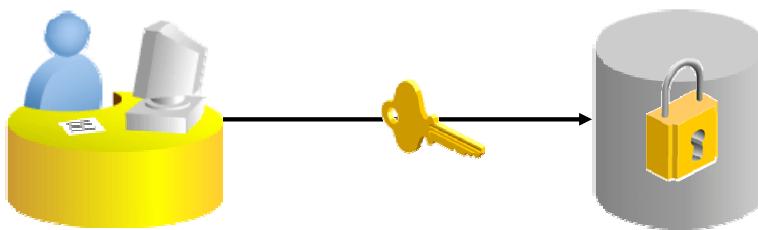
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Every business should recognize and develop policies that protect it. Accounting policies provide investors with the assurance that the company is sound and well managed. Engineering and safety policies assure customers and employees that proper precautions are being taken to protect the health and safety of both. These policies also protect the business from liability. Computer security policies serve a similar function. Privacy policies are required by the Payment Card Industry (PCI) for businesses that process credit cards. Customers may be attracted or repelled by certain privacy policies. Data security can prevent proprietary data from being stolen or abused.

Worldwide, businesses are adopting new computer security policies. For some, these policies are driven by law; for others, by the threat of theft, fraud, or sabotage; and for yet others, by the need for approval by financial regulators, access to credit cards, or investors.

Compliance Mandates

- Sarbanes-Oxley Act (SOX), J-SOX
- European Union Data Protection Directive
- Other U.S. laws
- Payment Card Industry Data Security Standard (PCI DSS)
- Reasonable care
- Security audits



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Security requirements have been a matter of individual concern until recently. Unless you were handling government or military data, there were few legal requirements. This is rapidly changing. A variety of laws have been passed to enforce privacy and accuracy of data in North America and Europe. Along with these laws comes a requirement to audit the security measures that are in place. These laws are becoming a pattern for laws in other countries, such as India and Japan.

Legal: Each of the laws listed here has specific requirements. This list is representative of many other laws that are being passed worldwide. These laws and industry standards are being held as a measure of reasonable care.

- **Sarbanes-Oxley Act (SOX)** requires that publicly traded companies in the United States strengthen and document internal controls to prevent individuals from committing fraudulent acts that may compromise an organization's financial position or the accuracy of its financial statements. The chief executive officer and the chief financial officer must attest to the adequacy of the internal controls and accuracy of financial reports. These officers are subject to fines and imprisonment for fraudulent reports. The requirements of SOX include providing information that is used to generate reports and providing documentation about the internal controls used to assure the integrity of the financial information. Other countries, such as Japan, are implementing similar laws.

- **European Union Data Protection Act** is intended to protect individual privacy by restricting access to individually identifiable data. It has eight points, one of which requires that data be kept secure.
- **Other U.S. laws:**
 - **Health Information Portability and Accountability Act (HIPAA)** is intended to protect personally identifiable health information (PII) from release or misuse. Information holders must provide audit trails of all those who access this data.
 - **Family Educational Rights and Privacy Act (FERPA)** covers health and personally identifiable information held by schools.
 - **California Security Breach Notification Law** requires that an organization holding a variety of personally identifiable information (PII)—for example, credit card, driver's license, and government identity numbers—must protect this information. If the information has been compromised, the organization must notify any resident of California whose unencrypted personal information is reasonably believed to have been acquired by an unauthorized person. There are two laws that apply to organizations holding PII.
 - **Federal Information Security Management Act (FISMA)** creates security guidance and standards through Federal Information Processing Standard (FIPS) documents that are managed by the National Institute of Standards and Technology (NIST). These standards are applied to organizations that process information for the U.S. government.

Payment Card Industry Data Security Standard (PCI DSS): For a business that captures credit card information, strict rules are imposed by contract and possible fines.

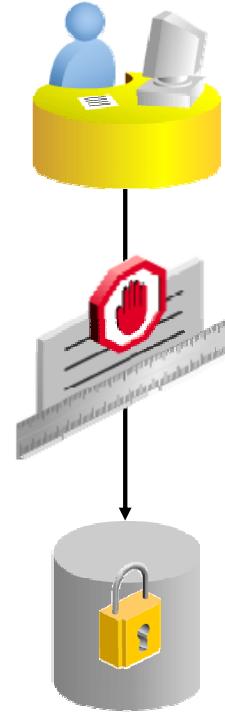
Security Audits: Many of these laws include provisions that require that the security plans (internal controls) be audited periodically. SOX requirements are vague and subject to interpretation by the officers of the organization. The implementation details can vary widely, depending on the level of detail that the officers require. Although SOX is vague, its penalties are severe. It is thus important to protect your company. The cost of security measures must be balanced against the risk. No one can certify that you are 100% secure.

Reasonable Care: A good solution to determining the proper level of security measures to implement is industry consensus. If you meet the agreed-upon minimum security practices and have accomplished due diligence, you may be safe from the worst penalties of the law. The current legal environment also requires reasonable care or due care. Due care is defined as the conduct that a reasonable man or woman will exercise in a particular situation, in looking out for the safety of others. If one uses due care, an injured party cannot prove negligence. If the security audit is part of due diligence, applying patches and hardening the system could be considered part of due care.

You, as DBA, security officer, or auditor, may be held responsible if you know of good practices and do not follow them, or if you neglect to be informed of common security solutions. Because of due diligence and due care, you must be able to advise management on the technical security measures that are available in the database. This course helps you to be informed of common security issues and solutions.

Security Risks

- External threats:
 - Unauthorized users
 - Denial of service
 - Unauthorized data access
 - Exploits: SQL injection and others
- Internal threats:
 - Abuse: Data theft
 - Sabotage: Data or service corruption
 - Complexity
 - Recovery
 - Omission
 - External threats listed above
- Partners



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Security risks come with accessibility. Accessibility makes your data valuable. If all your data were in filing cabinets or vaults, you could be sure that it was secure, but the time to retrieve it would make it worthless. External exploits by criminals breaking into systems get big headlines, but industry specialists estimate that 80% to 90% of the damage to information systems is done by insiders.

No system can be guaranteed to be 100% secure. The costs of securing a system are weighed against the possible costs of a security breach, whether it is internal or external. A thorough review can be expensive. However, it is important to be aware of the issues, set the priorities, and determine the funding. Some of the security standards, such as ISO/IEC 17799:2005, require a risk assessment and provide guidance.

External Threats

- **Unauthorized users:** These are outsiders who gain access to your system. They may use software exploits, bypass login information, crack passwords, or use social engineering. They may be helped by poor passwords, unattended terminal sessions, and unsecured servers or modem lines. Even your trash may contain information that allows them to get into your system.
- **Denial of service:** It can be an attack from a malicious source that requests limited resources such as port allocations to disrupt authorized users. It can be accidentally caused by authorized users who make inappropriate requests.
- **Unauthorized data and service access:** When an outsider obtains authentication, the system sees him or her as a valid user. This user then has the privileges granted to insiders.

- **Exploits:** This is a method of gaining access. **SQL injection** is a type of exploit that may accomplish one or more of the external threats listed.

Internal Threats

All the threats that are listed as outsider threats may also be performed by insiders. For example, an insider can gain unauthorized access by watching an authorized user enter a password.

- **Abuse of privilege:** Internal users have a justified authorization to your system. However, using that authorization to gain unauthorized access to certain data—or using the services in unauthorized ways—is considered to be abuse.
- **Data or service theft:** After access to data and services is obtained, theft is easy. This can be as minor as playing games on the server or as severe as selling proprietary information to a competitor. Personally identifiable information (PII) has become the most valuable information on systems, both in terms of the value to unauthorized users (identity theft) and cost when this information is compromised.
- **Sabotage (data or service corruption):** Sabotage by authorized users is always a possibility. This can be subtle or well intentioned: a data clerk altering data to help friends or a programmer leaving debug code in a program. The corruption can be intentional—for example, altering financial reports to move the stock price or cover embezzlement.
- **Complexity:** Increased complexity of the system increases the likelihood of security holes or avenues of attack that have been overlooked. One of the most overlooked security holes is passwords. Users forget or write down passwords when passwords are required to meet stronger complexity checks. Users often use simple passwords or write passwords in easy-to-find locations when they have many accounts. Some customers report that as much as 70% of their help-desk time is consumed by password resets. Single sign-on systems help reduce the complexity, allowing the user to remember one strong password instead of trying to remember several. However, a poorly designed single sign-on system can allow a single breach to access multiple systems.
- **Recovery:** What measures are in place to recover your systems if a breach should occur? How long will it take to have your system operational? Is there another system that can be placed in service so that the breached system may be examined forensically?
- **Omission:** How do you verify that the policies that are determined to be necessary are in place, and consistently applied across your systems? If access control and authorization policies are not applied properly, OS security patches will not prevent unauthorized access. Omission in policies allows holes in the security that do not require sophisticated skills to exploit.

Partners

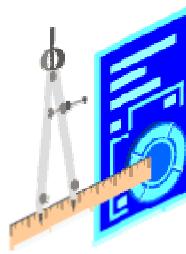
A relatively new area of concern and risk is partner access. The partner has authorized access to your system, but you do not have control of the partner's security policies.

- **External threat:** When external attackers breach the partner system, they have the same access that the partner is permitted.
- **Partner threat:** A growing number of breaches occur when the partner accesses sensitive data or proprietary information that is not appropriate for the partner to access. This occurs when the partner is allowed internal user type access.

Security Standards

Organizations publishing recognized security standards:

- SANS Institute
- Computer Emergency Response Team (CERT/CC)
- International Standards Organization (ISO 17799/27002)



Do your policies meet the standards?

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Several organizations produce guidelines for industry standard practices. Some of these are: SysAdmin, Audit, Network, Security (SANS) Institute, and the Computer Emergency Response Team (CERT/CC) operated by Carnegie Mellon University for the United States of America Department of Defense. More information about these organizations and the services that they provide can be found at www.sans.org and www.cert.org.

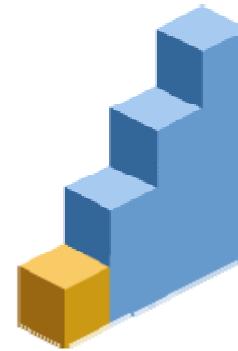
The International Standards Organization (ISO) produces standards for a broad range of industries. ISO-17799/27002 is an international standard of computer security practices. It includes best practices, certification, and risk assessment. It covers a broad range of issues and includes prewritten policies. For more information about ISO-17799/27002, see www.iso.org or www.computersecuritynow.com.

Security requirements are changing. It is important to be aware of the changes and their impact. The organizations listed in the slide (and others) provide newsletters and study opportunities.

Developing Your Security Policy

To develop your security policy, perform the following steps:

1. Assemble your security team.
2. Define your security requirements.
3. Develop procedures and systems to meet these requirements.
4. Implement security procedures.
5. Audit.



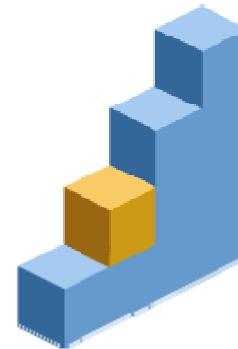
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

1. **Assembling Your Security Team:** First, determine which people in your organization will participate on the security team. To be effective, this team must include system architects, system administrators, database administrators (DBAs), network administrators, data owners, and representatives of end users. Although technical personnel monitor security, you must also include data owners and end users when defining your security requirements. Others to be included in the team are business experts who understand the organization, security, and disclosure requirements, and legal experts who know the legal requirements for handling the data of the organization.
2. **Defining Security Requirements:** To define your security requirements, examine who needs access to which data and services, and why they need it.
3. **Developing Security Procedures and Systems:** After you know the security requirements of your organization, you can develop procedures and systems to meet these requirements.
4. **Implementing Security Procedures:** Now that you have defined a policy, implement it to secure the data on a day-to-day basis.
5. **Auditing:** A security policy will have specific details that can be audited. An audit of your systems and procedures will confirm that the security policy has been implemented.

Defining a Security Policy

- What is a security policy?
 - A set of rules
 - Specific to an area and site
 - Required
 - Approved by management
- What is a standard?
 - Rules specific to a system or process
 - Required for everyone
- What are guidelines?
 - Suggestions and best practices
 - Specific to a system or a process
- Best practices



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A **security policy** is a set of documents that are specific to your site. Policy documents are approved by the management. They identify what is required by the company. The policy applies to a specific security area, such as acceptable use of modems, or formation of passwords. A policy is a set of mandatory rules. It must be only as long as required (a couple of pages) and easy to understand. It must be enforceable. To be successful, a policy must balance protection and productivity. Policies should include:

- **Configuration management:** Who is responsible for the security and to what level?
- **Incident reports:** How are security incidents handled?
- **Infractions:** What are the consequences of security violations?

A **standard** is a set of rules that apply to a specific system or process (for example, securing external procedures). Policies refer to standards. Standards, procedures, and guidelines are established at the department level and are much more detailed. They spell out what must be done to meet the policy—for example, your developers should have a secure coding standard that specifies certain coding practices that must be followed.

A **guideline** is a suggestion or a best practice for a specific system or process. Policies refer to guidelines. For novices, it is critical to get examples, especially for meeting the requirements of SOX or HIPPA. The SANS Institute and CERT/CC are very good resources.

Policies must be reviewed and updated regularly. By creating policies in a modular format, the pieces may be updated as needed without having to change a massive document. Policies should not go into details that change frequently, such as OS versions or hardware models.

Legal Implications

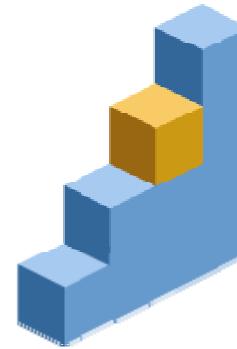
The legal department must be consulted and must approve the policy or procedure. How the policies are written and enforced can have a direct impact on whether you can prosecute violations and on whether your company is financially liable when breaches occur. The legal comment is critical. Security consultants have been sued for running password crackers on the network without written permission, even when they were being proactive. Legal advice is also critical when looking at warning banners and email monitoring. For your own protection, establish approved procedures for all forms of monitoring, sniffing, and cracking; such procedures should specify approved monitoring activities and should explicitly identify who performs these activities.

Best Practices

Security policies can often include references to best practice recommendations. Oracle provides a set of recommendations that can be accessed at <http://www.oracle.com/technetwork/topics/security/articles/index.html>.

Implementing a Security Policy

- Implement your standards and procedures.
- Implement the plan for developing new systems and applications.
- Monitor and enforce the policy.
- Keep systems and applications up-to-date with security patches.
- Educate users.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Now that you have a policy defined, implement it to secure the data on a day-to-day basis.

If you need to add systems or develop new applications to complete your security policy, immediately implement as much of the policy as you can without these systems. Then as you complete the new systems and applications, revise your security policy to include them.

Monitoring and Enforcing Security

Ensure that all employees understand the significance of keeping your organization secure. Employees must understand the security issues associated with their functions in the organization. They must also be aware of how they may be disciplined if they breach the security policy.

Applying Security Patches

System administrators and DBAs who are responsible for software installations must search for and apply security patches on a periodic basis. As part of the security policy, include a schedule to search for and apply new security patches.

Quiz

A successful security policy should balance protection and productivity.

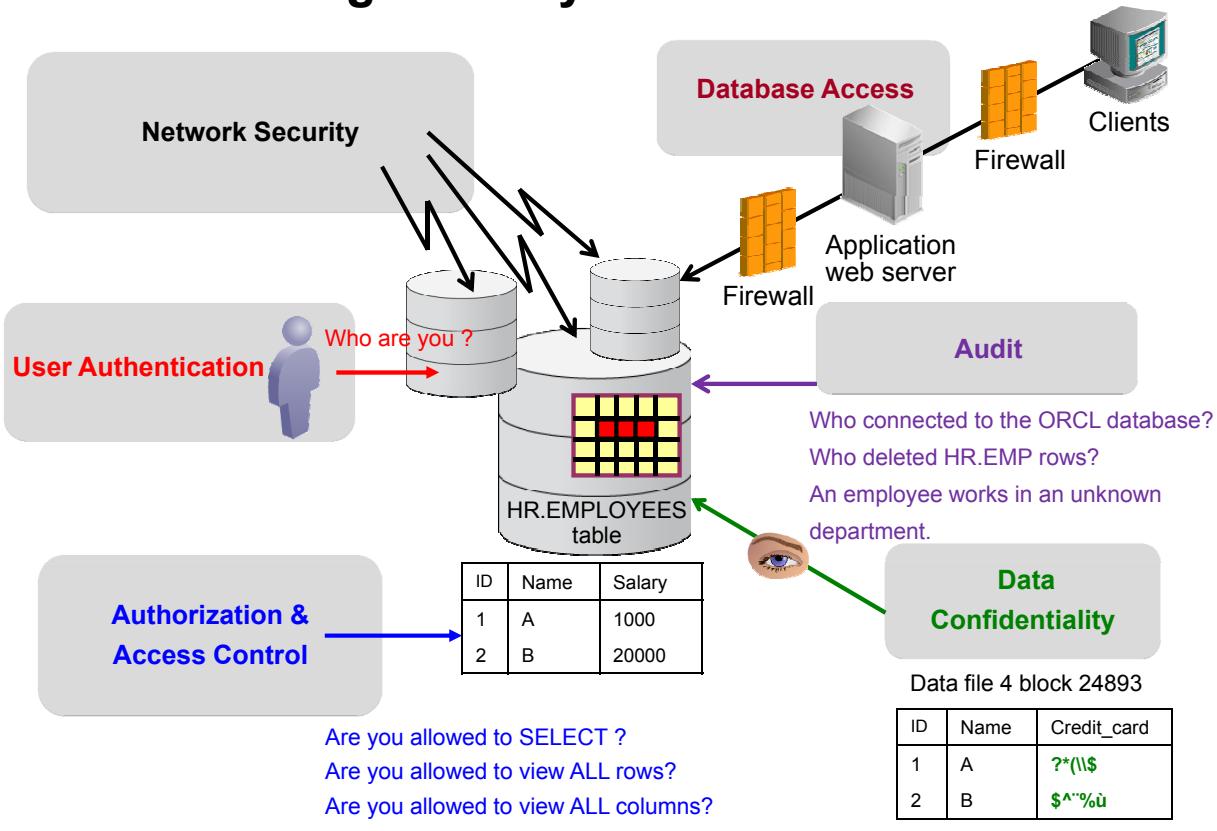
- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a

Enforcing Security at Different Levels



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

Enforcing security means that data is secured at different levels:

- When transmitted over the network
- When the user attempts to access the database
- When the user accessed the database and attempts to access the data in tables
- When the data is controlled by different technologies but is still vulnerable in blocks
- When the data has finally been accessed and requires auditing

Principle of Least Privilege

- Install only the required software on the machine.
- Activate only the required services on the machine.
- Give operating system (OS) and database access to only those users who require access.
- Limit access to the root or administrator account.
- Limit access to privileged database accounts.
- Limit users' access to only the database objects that they require to do their jobs.



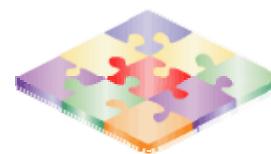
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Apply the principle of “least privilege” starting at the lowest level, and continue at every level. Least privilege is the principle that users should have the fewest privileges necessary to perform their duties. There will always be new security exploits that cannot be anticipated. By applying this principle, the possibility of the exploit is reduced and the damage can be contained.

- **Install only the required software on the machine:** Reduce maintenance, upgrades, the possibility of security holes, and software conflicts by reducing the number of software packages installed.
- **Activate only the required services on the machine:** Fewer services imply fewer open ports and fewer attack vectors.
- **Give OS and database access to only those users who require access:** Having fewer users means requiring fewer passwords and accounts. Reduce the possibility of open or stale accounts. With fewer accounts, it is easier for the administrator to keep the accounts current.
- **Limit access to the root or administrator account:** The administrator account must be carefully guarded, audited, and never shared.
- **Limit access to privileged database accounts:** Any user that can connect with administrative privileges has access to a higher level of privileges. Users who require access with administrative privileges must have their own account, and must be audited.
- **Limit users' access to only the database objects required to do their jobs:** Do they have a need to know? Users who have access to more objects and services than they require have an opportunity for mischief.

Defense in Depth

- Enforce security policies
- Train users
- Harden the operating system
- Use firewalls
- Protect network and media
 - Encrypting network traffic
 - Protect data on disk, disk backup, tape backup
- Protect application data from internal threat
 - DBA access
 - Database access: who, when, where, and how
 - Configuration controls
 - Privilege management
 - Row-level and column-level access
- Control by auditing



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

“Defense in depth” means that you consider every level (OS, network, file system permissions, database, firewall, password protection, user education, and software bug fixes). The defeat of one security feature must not give full access to all data or services. A policy is a roadmap for security implementation, but it must be implemented to be effective. Users must be trained to avoid activities that could easily breach security. You may have a perfect firewall and antivirus checking on all incoming traffic, but if a user working from home downloads a virus or a Trojan, it can infect the network from behind the firewall.

The operating systems on machines that host the application server, the database, the mail server, and other critical services must be hardened. The network services, firewalls, and proxies each add another layer. Then, secure the database. Every layer shown in the slide needs to be in place. Each level complements the others to achieve better security. Defense in depth considers everything.

The list presented in the slide is an outline of best practices. This course deals with database-related items from several of these items.

Refer to the following note: “Defense-in-Depth Guide” in
<http://www.oracle.com/technetwork/database/security/sol-home-086269.html>.

Common Exploits

There are several classes of common exploits:

- Phishing
- Well-known and default accounts
- Backdoors
- Debug code
- Cross-scripting
- SQL injection
 - Tricks the SQL engine into executing unintended commands
 - Exploits a common vulnerability in the application
 - Supplies crafted user-supplied strings



Two main methods of preventing these attacks:

- Educate users about social engineering.
- Use secure coding practices.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

An exploit is a technique used to attack, sabotage, gain unauthorized access to data or services, or to deny service to authorized users. There are several general classes of methods used. Some of the most common are listed.

- **Phishing** is a social engineering method. A carefully crafted email, web page, or even a phone call to unsuspecting end users can be used to obtain personal information that can then be used for identity theft, or to access their accounts—for example, when users receive an email apparently from their bank asking them to connect to a corporate website, and log in, a certain percentage of people will do so. The malicious site then captures their login information.
- **Default accounts:** Many applications have well-known demonstration or default accounts. These accounts should have a method of being secured or deleted.
- **Back doors:** A programmer builds in an undocumented method of bypassing authorization. These should never be allowed to be coded. These can be prevented only by administrative controls and code review.
- **Debug code:** Often, debug code is included in the production code to help during development and later to aid support. This code should have clearly documented methods to be enabled and disabled. Debug code may give additional privileges or bypass authorizations.

- **Cross-scripting:** This is a very common way of gaining information. A crafted URL or script injected into a vulnerable webpage can result in elevated privileges or access to sensitive information, such as session identifiers, cookies, or login credentials. Often, a cross-scripting attack is hidden in a seemingly innocent email or website that invites the recipient to access a URL.
- **SQL injection:** SQL injection makes use of security holes in applications to input and execute crafted SQL statements. SQL injection is a technique for maliciously exploiting applications that use client-supplied data in SQL statements. The flaw is common; a user-supplied input string is concatenated to a dynamic SQL statement, and then executed.
Attackers trick the SQL engine into executing unintended commands by supplying specially crafted string input. This input can allow the attacker to gain unauthorized access to a database to view or manipulate restricted data.
SQL injection techniques may differ, but they all exploit a common vulnerability in the application. That vulnerability is a user-supplied string literal interpreted as code by the SQL engine. These literals may be input through forms, URLs, or hidden fields in webpages. The vulnerability comes whenever the input string is added to a dynamic SQL statement by the application code.
To immunize your code against SQL injection attacks, you must use bind arguments (either automatically with static SQL or explicitly with dynamic SQL), or validate and sanitize all input concatenated to dynamic SQL.
Although a program or an application may be vulnerable to SQL injection, web applications are at a higher risk because an attacker can perpetrate SQL injection attacks without any database or application authentication.

Preventing attacks

There are two methods of prevention.

- The first is educating the end user to limit social engineering attacks such as phishing and cross-scripting. Because of the imagination of the attackers, the attacks will always be changing. The educators can publish warnings and counter measures only after the exploit is discovered.
- The second method is a secure-coding practice, which implements verified methods to prevent attacks. The methods to prevent these exploits are similar across the various exploit types. A few methods to limit SQL injection attacks are presented here.

For more details, see the course titled *Oracle Database PL/SQL Language Reference 12c Release 1* or the *Tutorial on Defending Against SQL Injection Attacks* available under Oracle Learning Library.

Standard Configuration Policies: Enforcing standard configuration policies by scanning is another way to help prevent attacks. Some companies go as far as to have internal hackers attempting to break into systems to test the security policies.

Preventing Exploits

Common exploits may be prevented or mitigated by:

- Reducing the attack surface
 - Use invoker's rights.
 - Reduce arbitrary inputs.
- Limiting privileges to avoid privilege escalation
- Avoiding known vulnerable code constructs
 - Avoid dynamic SQL and use bind arguments.
 - Avoid known weak protocols.
 - Always validate input.
 - Trap and handle exceptions.
- Designing code that is immune to common exploits
- Reviewing and testing code for common exploits



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The principles outlined in the slide can be applied to many types of attacks: SQL injection, cross-scripting, and buffer overflow exploits. These methods are derived from the policies. Reducing the attack surface and limiting privileges are applications of the principle of least privilege.

Normal SQL injection attacks depend, in large measure, on the ability of an attacker to reverse-engineer portions of the original SQL query by using information gained from error messages. Therefore, keep application error messages succinct and do not divulge metadata information (such as column names and table names). Before deploying your application, remove all code tracing and debug messages.

Be aware that trace files, audit logs, and alert logs can contain metadata information. Protect these files by setting the directory permissions to prevent anyone but the instance owner from reading these files. Sometimes, these permissions must be relaxed to allow others access for debugging and performance tuning. Be sure to limit the users carefully and return to original permissions when the need is past.

Note: Use PL/SQL conditional compilation for managing self-tracing code, asserts, and so on. For more information about conditional compilation, see the *Oracle Database PL/SQL Language Reference*.

Applying the methods shown in the slide is covered in more detail in the practices. There are white papers available on OTN that give more details about SQL and PL/SQL coding practices:

- *Doing SQL from PL/SQL: Best and Worst Practices* at
www.oracle.com/technology/tech/pl_sql/pdf/doing_sqlfrom_plsql.pdf
- *How to Write SQL Injection-Proof PL/SQL* at
www.oracle.com/technology/tech/pl_sql/pdf/how_to_write_injection_proof_plsql.pdf
- ORACLE AUDIT VAULT AND DATABASE FIREWALL at
<http://www.oracle.com/us/products/database/security/ds-security-audit-vault-firewall-1883353.pdf?ssSourceSiteId=otnen>

Summary

In this lesson, you should have learned how to:

- Describe fundamental security requirements
- Define the following terms:
 - Least privilege
 - Authentication
 - Authorization
- Describe security policies
- Describe the concept of in-depth security
- Apply these concepts to prevent SQL injection



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Security is not easy. Developing and implementing thorough security policies require management support.

The types of attacks on a secure environment are constantly changing. To counter these risks, sometimes you need to think like a hacker. Maintaining a secure environment often requires a certain level of paranoia. The exploits and attacks are becoming more sophisticated. The average administrator or security officer may not have the time or the background to keep up with new attack vectors. The recommendations presented here are derived from the best industry practices. Applying these recommendations may not prevent an attack, but they can help mitigate the damage, reduce the access, and track the offender.

Practice: Overview

This practice covers the following topics:

- 2-1: SQL injection exploit tutorial
- 2-2: Using invoker's rights procedure
- 2-3: Using static SQL and bind arguments
- 2-4: Avoiding SQL injection through dynamic PL/SQL block
- 2-5: Validating input using the DBMS_ASSERT package



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The practices show how secure-coding practices can be used to reduce or eliminate the possibility of SQL injection exploits. The basic methods used in reducing the possibility of SQL injection can be adapted and applied to other common exploits. Specifics such as removing dynamic SQL would be changed to not allowing certain characters in XML or HTML to prevent cross-scripting. However, general techniques such as peer review and testing are applicable across all type of exploits.

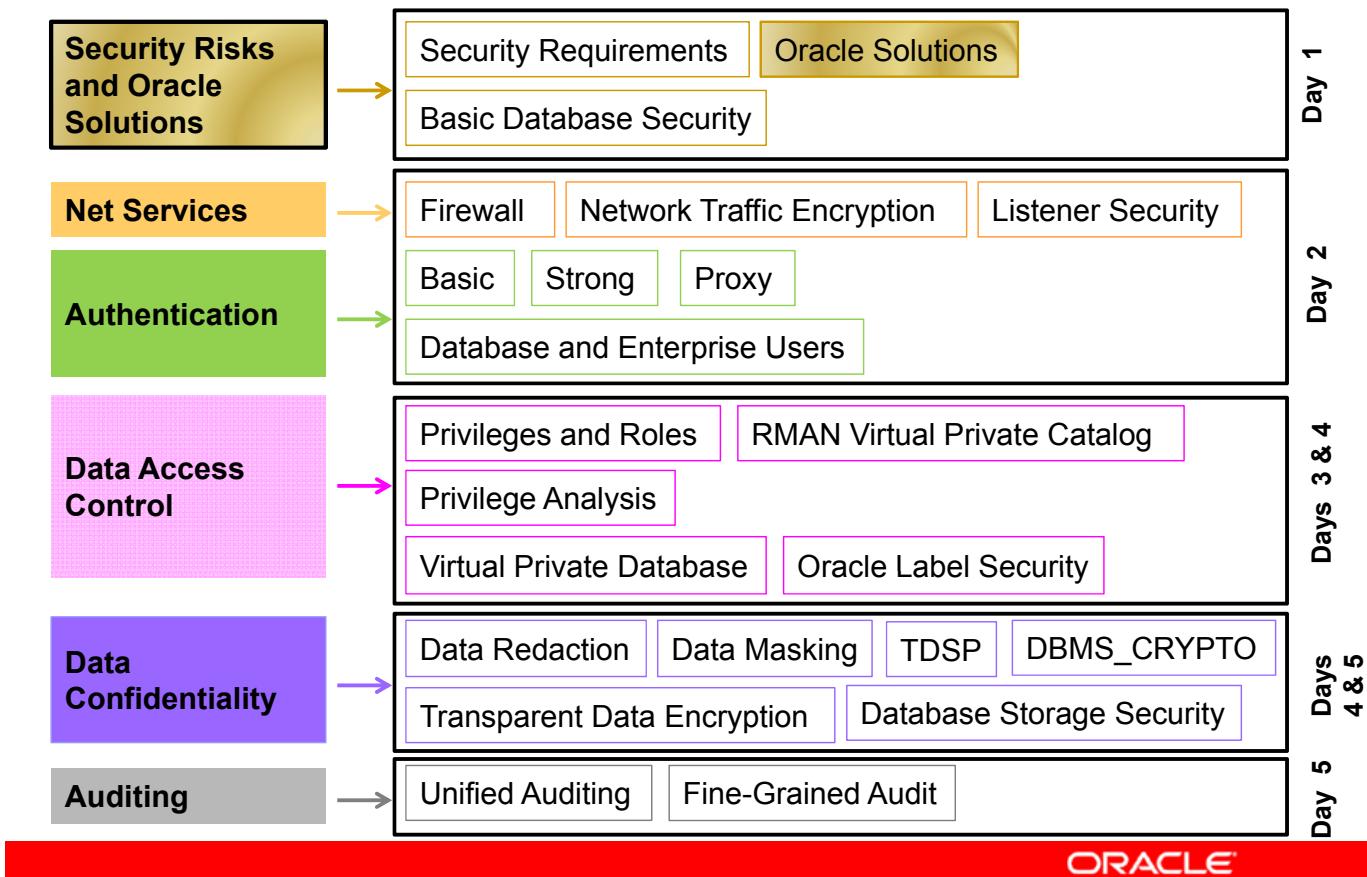
Choosing Security Solutions



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Schedule



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to describe the following recommended solutions to common security concerns:

- Maintaining data integrity
- Protecting data
- Controlling data access



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete understanding of the security and advanced security features, refer to the following guides in the Oracle documentation:

- *Oracle Database Security Guide 12c Release 1 (12.1)*
- *Oracle Database Advanced Security Guide 12c Release 1 (12.1)*

For a complete understanding of the options and packs that require an extra cost, refer to the following guide in the Oracle documentation:

- *Oracle Database Licensing Information 12c Release 1 (12.1)*

Refer to other sources of information available in the following:

- <http://www.oracle.com/database/security>
- <http://www.oracle.com/technetwork/database/security/index.html>

Database Security Solutions

PREVENTION	DETECTION	CONTROL
Authentication	Activity Monitoring	Privilege Analysis
Privileged User Controls	Database Firewall	Sensitive Data Discovery
Redaction, Masking Transparent Sensitive Data Protection	Auditing and Reporting	Configuration Management
Network and Data Encryption		



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Security implies:

- Maintaining data integrity: Financial regulators require assurance of integrity of the data that is used to produce financial reports. Compliance is the term used to describe the actions required by the Sarbanes-Oxley (SOX) regulation in the U.S. and other industry regulations to assure the integrity of data used to produce financial reports. For compliance, you must track who accessed the data, when was it accessed, and how it was changed.
- Protecting data: California Security Breach Laws—CA-SB-1386 and CA-AB-1950—require that personally identifiable information (PII) be protected. The Payment Card Industry Data Security Standard (PCI-DSS) requires that credit card information be protected at several levels. Reasonable care dictates that businesses must protect private information to avoid liability. PII and other sensitive data must be protected.
- Controlling data access

You must follow three guidelines:

- Prevent unauthorized access to the database and data.
- Detect unauthorized access to the database or to application data by using monitoring and auditing products and features.
- Control unauthorized access to the database or to application data by using analysis and configuration tools.

Encrypting Network Traffic



Oracle Net services encryption

- Encrypts network connections
- Encrypts data transmitted over the network
- Prevents decrypting data over the network
- Provides several encryption methods

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

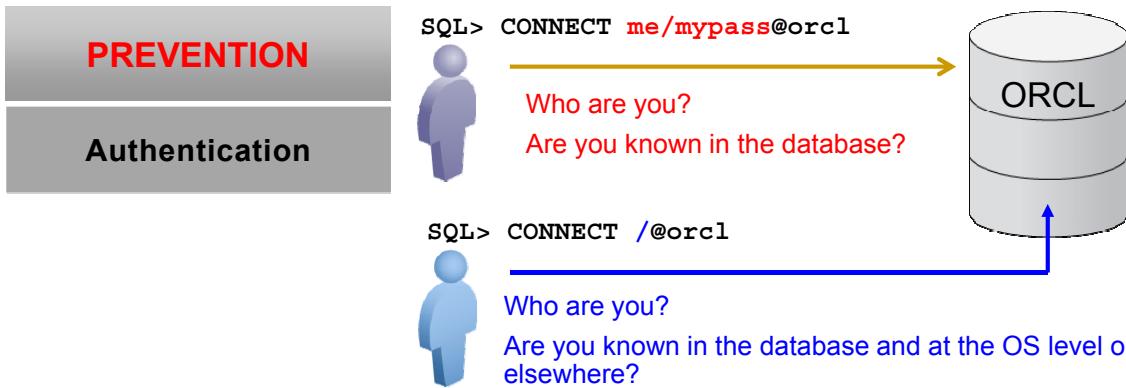
Encrypting Oracle Net Services traffic means:

- When data travels to and from the Oracle database, the network connections can be encrypted by using standard algorithms such as AES (up to 256-bit keys), Triple DES, and RC4.
- You can also configure native Oracle Net Services data encryption and integrity for both servers and clients.
- The purpose of a secure cryptosystem is to convert plaintext data into unintelligible ciphertext based on a key in such a way that it is very hard (computationally infeasible) to convert ciphertext back into its corresponding plaintext without knowledge of the correct key. Both Secure Sockets Layer (SSL) encryption and an Oracle native encryption capability are supported. The support for SSL/TLS follows industry standards. The Oracle native encryption feature provides distinct benefits including the ability to begin encrypting database network connections immediately, without provisioning X.509 certificates.
- Network encryption encrypts all the Oracle Net packets between client computers, databases, and application servers. Network encryption for the client and the server is available only with the Enterprise Edition of Oracle Database.
- Network encryption supports several common encryption methods.

For a complete understanding of the Net Services features, refer to the following guides in the Oracle documentation:

- *Oracle Database Net Services Administrator's Guide 12c Release 1 (12.1)*
- *Oracle Database Net Services Reference 12c Release 1 (12.1)*

Authenticating Users to Database



Authentication to database

- Basic or strong authentication verifies the database user's or enterprise user's identity.
- Identity can be propagated in a three-tier environment.
 - Pass through
 - Proxy user
 - Secure application role

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Authentication verifies the end user and the user's identity.

- Identity is determined by authentication. Users can be authenticated in a number of ways before they are allowed to create a database session.
- Basic authentication relies on the user's password on the operating system.
- Strong authentication allows the use of Smart Cards, biometrics, Kerberos tokens, or certificates to be assured that the end user is correctly identified.
- Centralized identity management such as Enterprise User Security reduces the time and effort required to manage users over multiple databases and applications.
- This identity can be propagated in the three-tier environment by various methods with supported techniques. If your employees or customers log into an application and the application connects to the underlying database as a "big app user," the identities of the individuals are lost, making it more difficult to audit and enforce access controls. Many applications handle the authentication. This is usually expensive and less secure than other solutions. The expense comes with coding the security into each application, and the application developer may not have the experience to code, test, and maintain the security module. Auditing can be a problem in the environment where the user is not known to the database. Both "client identifier" and "proxy authentication" can be used to maintain the identity of the users, enabling the enforcement of audit and access control policies in the database.

Some of the solutions to these problems are:

- **Pass through:** The user provides the database login credentials. The application server passes the credentials to the database. The database server performs the authentication and authorization.
- **Proxy user:** The user has a database account, and the application is granted the privilege to connect on behalf of the user. The connection can be either with or without a user password and may include the privilege to enable some or all of the roles granted to the user.
- **Secure application role:** The application connects as a low-privileged user, and then may enable roles only through a secure package. In this case, the end user may not be known to the database, and the application is responsible for authorizing the user and enabling the required roles.
- **Enterprise User Security (EUS):** The end user may be a schema-independent user, unknown to the database, but authenticated in Oracle Internet Directory (OID). The end user identity is supplied by OID to the database session and is available for audit.

Encrypting Application Data with DBMS_CRYPTO



DBMS_CRYPTO package:

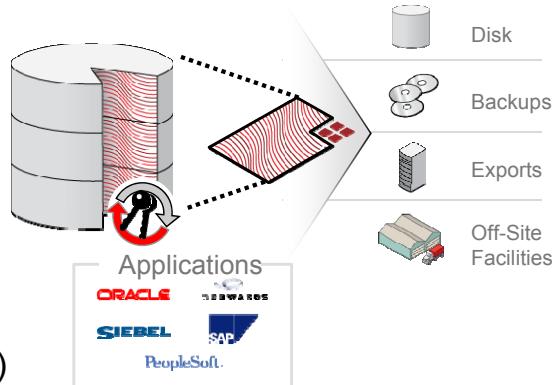
- Prevents reading data in data file blocks
- Prevents decrypting data without encryption key
- Provides PL/SQL function for key management
- Is “Near Zero” overhead with hardware

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The DBMS_CRYPTO package provides an interface to encrypt and decrypt stored data, and can be used in conjunction with PL/SQL programs running network communications. It provides support for several industry-standard encryption and hashing algorithms, including the Advanced Encryption Standard (AES) encryption algorithm. AES has been approved by the National Institute of Standards and Technology (NIST) to replace the Data Encryption Standard (DES).

Encrypting Application Data with TDE



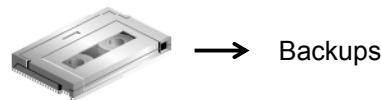
Oracle Advanced Security

Transparent Data Encryption (TDE)

- Prevents reading data in data file blocks
- Prevents decrypting data without encryption key
- Requires no application changes
- Provides built-in two-tier key management
- Is “Near Zero” overhead with hardware
- Is integrated with Oracle technologies (Data Pump, Recovery Manager)

Oracle Secure Backup (OSB)

Oracle Secure Backup Express



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Advanced Security Transparent Data Encryption (TDE) makes encryption of sensitive data simple with no changes to the existing application code.

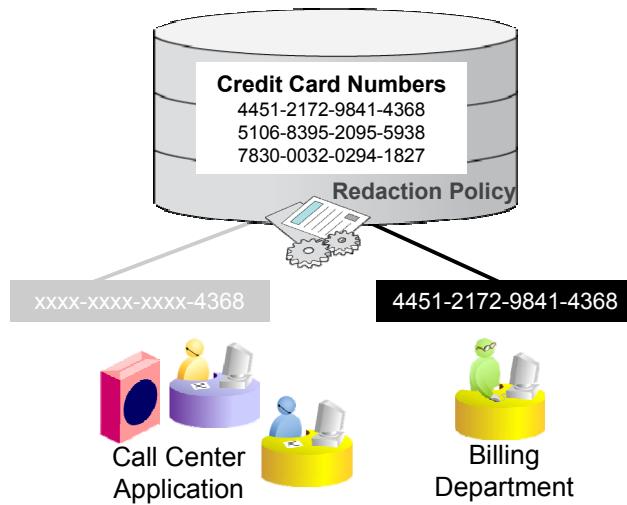
TDE is one of the two components of the Oracle Advanced Security option for Oracle Database 12c Release 1 Enterprise Edition. It provides transparent encryption of stored data to support your compliance efforts. Data is automatically encrypted when it is written to disk and automatically decrypted when accessed by the application. Key management is built-in, eliminating the complex task of creating, managing, and securing encryption keys.

TDE is integrated with the following Oracle technologies:

- TDE tablespace encryption is certified on Oracle Exadata.
- Oracle Data Pump provides the ability to encrypt data as it is written to the export file, providing additional protection for credit card numbers and other sensitive business data.
- Encrypted data can be stored on ASM storage.
- Golden Gate can transfer encrypted data.
- Advanced Compression allows encryption.

Oracle Secure Backup (OSB) delivers centralized tape backup management for the entire IT environment, providing its own encryption features. OSB Express is available free with the Oracle Database and is the same core product as OSB. The Express edition is limited to a single host with one direct-attached tape drive and has some restrictions on advanced feature usage.

Redacting Sensitive Data Displayed



Oracle Advanced Security

Data Redaction:

- Real-time sensitive data redaction based on database session context
- Library of redaction policies and point-and-click policy definition
- Consistent enforcement, policies applied to data
- Transparent to applications, users, and operational activities

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Data Redaction enables you to redact (mask) column data. Data Redaction performs the redaction at run time—that is, the moment that the user tries to view the data. This functionality is ideally suited for dynamic production systems in which the data constantly changes. While the data is being redacted, Oracle Database is able to process all data normally and preserve the back-end referential integrity constraints. Data redaction can help you to comply with industry regulations such as Payment Card Industry Data Security Standard (PCI DSS) and the Sarbanes-Oxley Act.

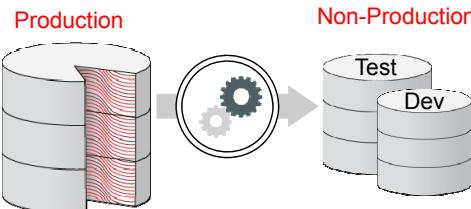
If you need more information, refer to the following articles:

- <http://www.oracle.com/technetwork/database/options/advanced-security/advanced-security-ds-12c-1898873.pdf>
- <http://www.oracle.com/technetwork/database/options/advanced-security/advanced-security-wp-12c-1896139.pdf>

Masking Data for Non-Production Use



LAST_NAME	SSN	SALARY
AGUILAR	203-33-3234	40,000
BENSON	323-22-2943	60,000



LAST_NAME	SSN	SALARY
ANSKEKSL	323-23-1111	60,000
BKJHHHEIEDK	252-34-1345	40,000

Oracle Data Masking:

- Replaces sensitive application data
- Provides extensible template library and formats
- Provides application templates available
- Detects and preserves referential integrity
- Masks and subsets data at source (requires use of Oracle Test Data Management)
- Supports masking data in non-Oracle databases

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A number of regulations mandate that company confidential, sensitive, and personally identifiable data must be protected and access to this data must be restricted. There is often a need to provide production data, or realistic looking data to in-house developers and testing organizations during application development.

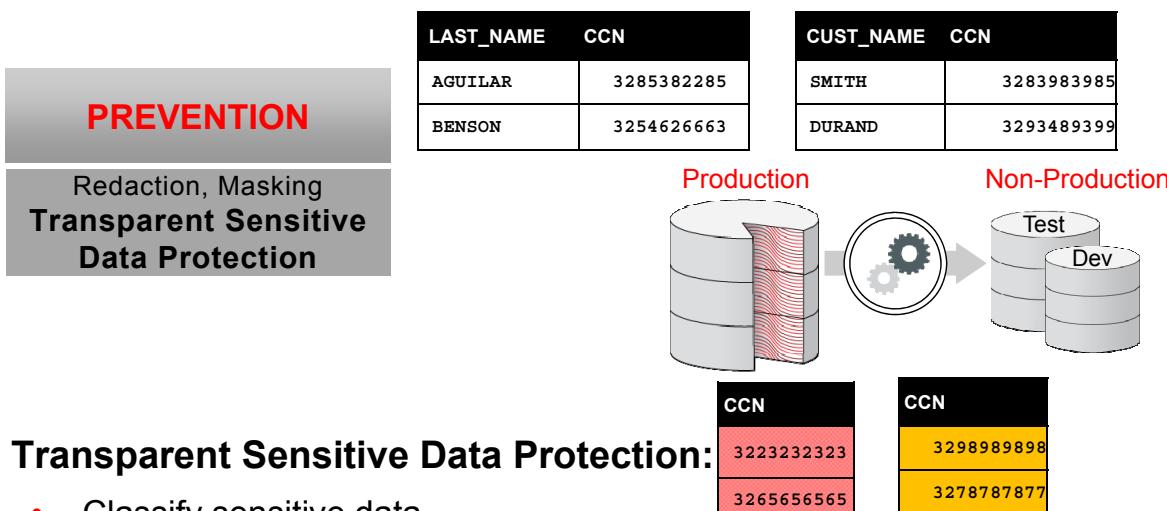
Data masking is a way to meet these two conflicting needs. Data masking is the act of *anonymizing* customer, financial, or company confidential data retaining the original data's properties, such as width, type, and format, and replacing sensitive information copied from production databases to non-production databases with realistic, but scrubbed, data based on masking rules. Data masking is ideal for virtually any situation when confidential data needs to be shared with non-production users. These users may include internal users such as application developers, or external business partners such as offshore testing companies, suppliers, and customers.

The Data Masking Pack is a separately licensed Oracle Enterprise Manager management pack available with Oracle Enterprise Manager Cloud Control 12c.

For a complete understanding of the Data Masking Pack features, refer to the following guides in the Oracle documentation:

- *Oracle Enterprise Manager Licensing Information 12c Release 2 (12.1.0.2)*
- *Oracle Database Testing Guide 12c Release 1 (12.1) - Part "Test Data Management"*

Protecting Multiple Sensitive Columns



Transparent Sensitive Data Protection:

- Classify sensitive data
- Create TSDP policies to protect data as a whole for a given class
- Apply TSDP policy to protect table columns by using either Oracle Data Redaction or Oracle Virtual Private Database settings
- Create a uniform TSDP policy for all of the data that you classify
- Modify the uniform policy as necessary, as compliance regulations change

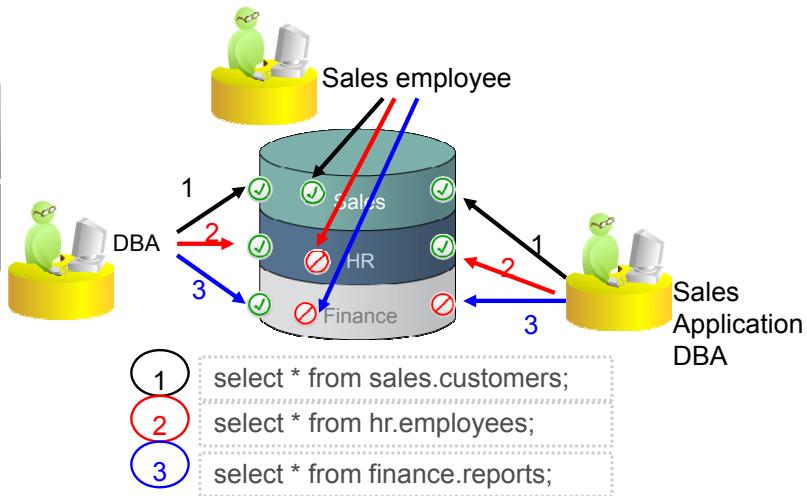
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Transparent sensitive data protection (TSDP) enables you to quickly find all table columns in a database that hold sensitive data (such as credit card or Social Security numbers), classify this data, and then create a policy that protects this data as a whole for a given class. The TSDP policy then protects the sensitive data in these table columns by using either Oracle Data Redaction or Oracle Virtual Private Database settings. The TSDP policy applies at the column level of the table that you want to protect, targeting a specific column data type, such as all NUMBER data types of columns that contain credit card information. You can create a uniform TSDP policy for all of the data that you classify, and then modify this policy as necessary, as compliance regulations change. Optionally, you can export the TSDP policies for use in other databases.

The benefits of TSDP policies are enormous. You can easily create and apply TSDP policies throughout a large organization with numerous databases. This helps auditors greatly by enabling them to estimate the protection for the data that the TSDP policies targets.

Controlling Data Access Using Authorization



Authorization

Privileges and Roles:

- Maintain separation of duties by granting administrative privileges
- Limit user access to application data by granting object and system privileges
- Create roles to group privileges or roles (role hierarchy)
- Grant and revoke privileges and roles to users and roles

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

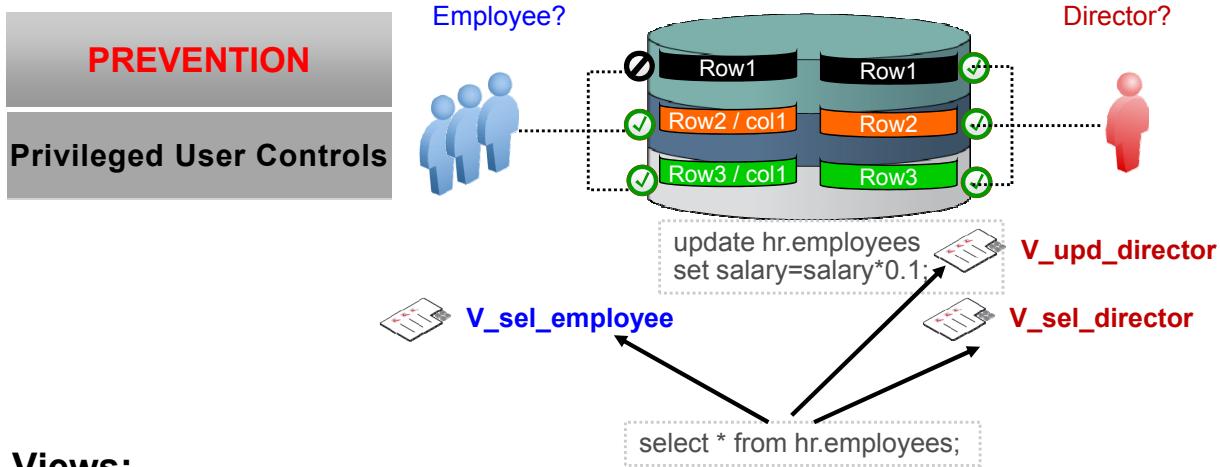
Authorization includes primarily two processes:

- Permitting only certain users to:
 - Access, process, or alter data (object privileges)
 - Execute certain commands (system privileges)
- Applying varying limitations on user access or actions. The limitations placed on (or removed from) users can apply to objects such as schemas, tables, or rows.

A user privilege is the right to run a particular type of SQL statement, or the right to access an object that belongs to another user, run a PL/SQL package, and so on. The types of privileges are defined by Oracle Database.

Roles are created by users (usually administrators) to group together privileges or other roles. They are a way to facilitate the granting of multiple privileges or roles to users.

Controlling Data Access Based on Views



Views:

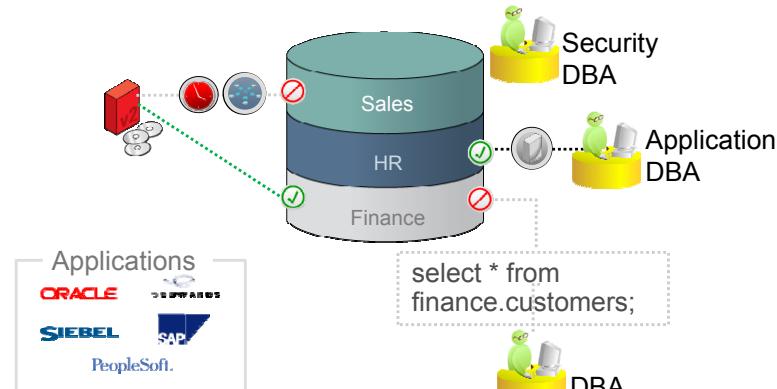
- Enforce row- or column-level access control automatically, transparent to applications
- Restrict access at row level for SELECT and DML statements
- Easy to implement
- Create as many views as different contexts

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

“Views” is the one of the first easy methods to provide a different representation (such as subsets or supersets) of the data that resides within tables. Views are very powerful because they allow you to tailor the presentation of data to different types of users. You will have to create as many views as different selections and projections allowed for the different categories of users.

Controlling Data Access by Privileged Users



Database Vault:

- Limit access to application data to users being granted privileges
- Create realms to protect objects
- Create multi-factor SQL command rules to allow or disallow commands
- Enforce enterprise data governance, least privilege, segregation of duties
- Use out-of-the-box application policies

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database Vault addresses common regulatory compliance requirements and reduces the risk of insider threats by preventing highly privileged user access controls, enforcing separation of duty, and providing controls over who, when, where, and how applications, data, and databases are accessed.

Payment Card Industry (PCI), Sarbanes-Oxley (SOX), EU Privacy Directive, and the Healthcare Insurance Portability and Accountability Act (HIPAA) all require strong internal controls on access, disclosure, or modification of sensitive information that could lead to fraud, identity theft, financial irregularities, and financial penalties.

Oracle Database Vault realms, command rules, factors, and separation of duty can be added to existing application environments without changes to the existing application code. Oracle Database Vault is validated with Oracle PeopleSoft Applications Siebel, SAP, and JD Edwards.

Oracle Database Vault is part of the database kernel, and thus is more secure than a security solution implemented in custom PL/SQL packages.

For a complete understanding of the Database Vault features, refer to the following guides in the Oracle documentation or attend the course:

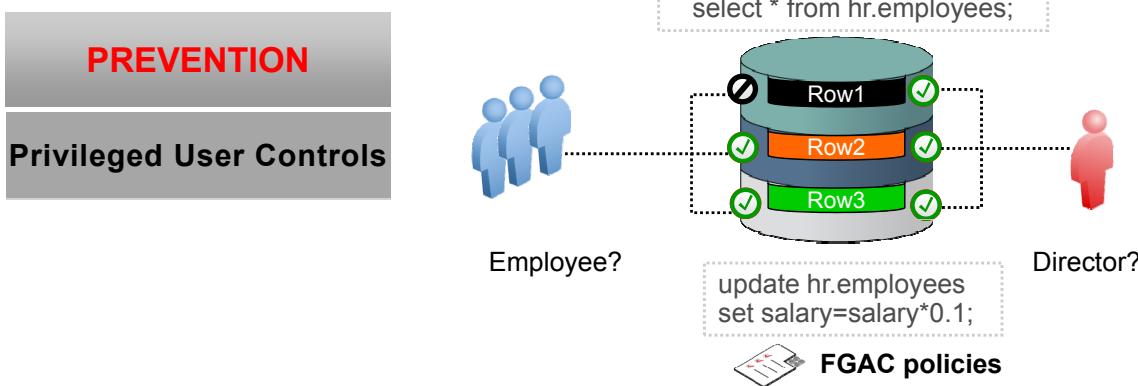
- *Oracle Database Vault Administrator's Guide 12c Release 1 (12.1)*
- *Oracle Database 12c: Implement Database Vault* course

Refer to other sources of information:

<http://www.oracle.com/technetwork/database/security/database-vault-ds-12c-1898877.pdf>

<http://www.oracle.com/technetwork/database/options/database-vault/database-vault-wp-12c-1896142.pdf>

Controlling Data Access Based on Function



Oracle Virtual Private Database (VPD):

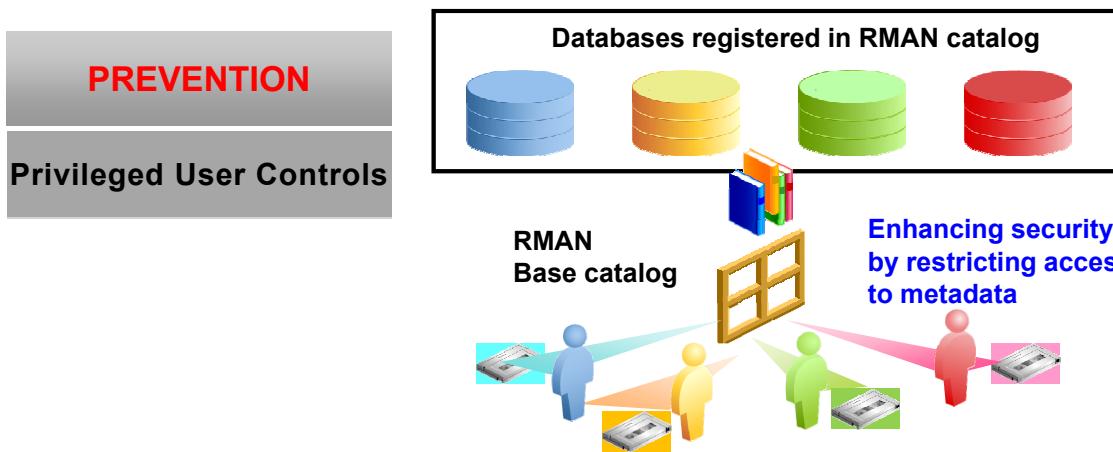
- Enforce row-level access control automatically, transparently to applications
- Create function to restrict access to table rows
- Restrict access at the row level for `SELECT` and `DML` statements
- Use context attribute values during VPD policies evaluation
- Use SQL predicates in other policies (Oracle Label Security)

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Virtual Private Database (VPD) is one of the most popular security features in the database. VPD is used when the standard object privileges and associated database roles or views are insufficient to meet application security requirements providing views that are more complex. VPD policies can be simple or complex depending on your security requirements. VPD can be used in combination with the “application context” feature to enforce sophisticated row-and/or column-level security requirements for privacy and regulatory compliance. A simple VPD example might restrict access to data during business hours and a more complex VPD example might read an application context during a login trigger and enforce row-level security against a table. With “Column Relevance,” VPD can be configured such that the policy is enforced only when a critical column is selected. With “Column Hiding,” VPD allows for the most effective combination of ease-of-use and security: some users still have access to all public information in a table, but confidential information remains hidden.

Controlling Backup Access Based on Privilege



RMAN Virtual Private Catalog (VPC)

- Avoid the inadvertent or malicious destruction of catalog data for other databases
- Keep clear separation of duty between administrators of various databases

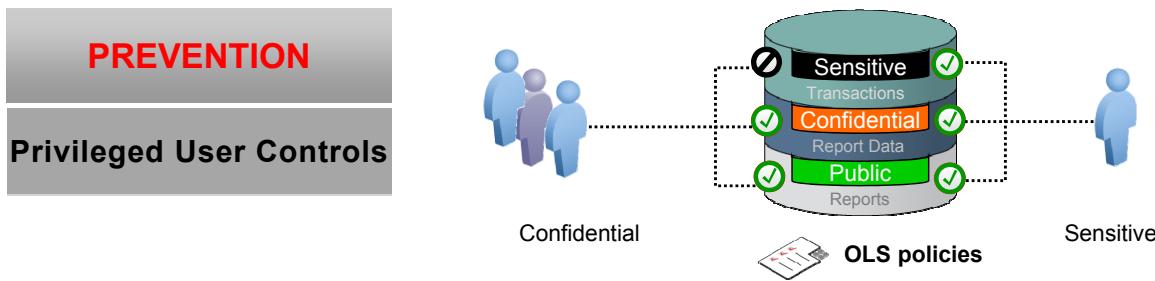
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This feature allows a consolidation of RMAN repositories and maintains a separation of responsibilities, which is a basic security requirement.

By default, all of the users of an RMAN recovery catalog have full privileges to insert, update, and delete any metadata in the catalog. If the administrators of two unrelated databases share the same recovery catalog, each administrator could, whether inadvertently or maliciously, destroy catalog data for the other's database. You can restrict each database administrator to modify only backup metadata belonging to those databases that they are responsible for, while still keeping the benefits of a single, centrally managed, RMAN recovery catalog. This goal can be achieved by implementing virtual private catalogs (VPCs).

Controlling Data Access Based on Label



Oracle Label Security (OLS):

- Chooses your virtual information partitioning
- Classifies users and data using labels
- Creates labels based on business drivers
- Enforces row-level access control automatically, transparent to applications
- Uses labels as factors in other policies (Database Vault)

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Label Security (OLS) was introduced in Oracle8i to provide multi-level security capabilities within the Oracle Database for government and defense organizations. Since then, the use of label-based access control technology has expanded to commercial organizations addressing privacy and regulatory compliance requirements.

Oracle Label Security has the ability to:

- Control access based on data classification, adding a powerful dimension to the access control decision process
- Enforce traditional multi-level security (MLS) policies for government and defense applications

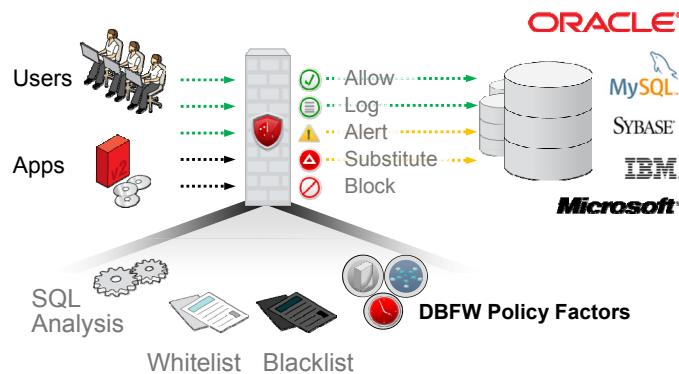
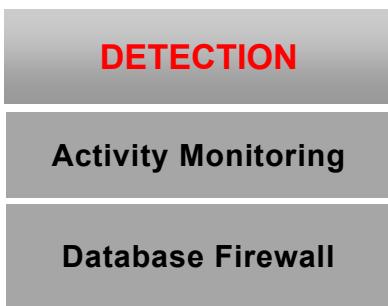
In addition, OLS user authorizations such as "Sensitive:PII" can be used within Database Vault command rules as factors or within VPD policies to determine need-to-know. For government and defense applications, OLS provides multi-level security capabilities to protect classified information.

Note: Temporal Validity and In-Database Archiving are two features that also have the effect of hiding rows during SELECT and DML statements. The user can at any time decide to display the hidden rows by an ALTER SESSION statement. Rows in a table may be hidden according to their ageing but are still controllable by users. They are not hidden for security purposes. These features belong to Information Lifecycle Management rather than to security management.

If you need more information, refer to the following article:

- <http://www.oracle.com/technetwork/database/security/label-security-ds-12c-1898878.pdf>

Monitoring Database Activity



Database Firewall

- Monitors network traffic and detects and blocks unauthorized activity
- Performs a highly accurate SQL grammar analysis
- Can detect or stop SQL injection attacks to Oracle and non-Oracle databases by using:
 - A **whitelist** of approved SQL statements
 - A **blacklist** of SQL statements disallowed from specific users (DB, OS), of client IP addresses, of times of the day, or of specific types

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Database Firewall provides two features for databases:

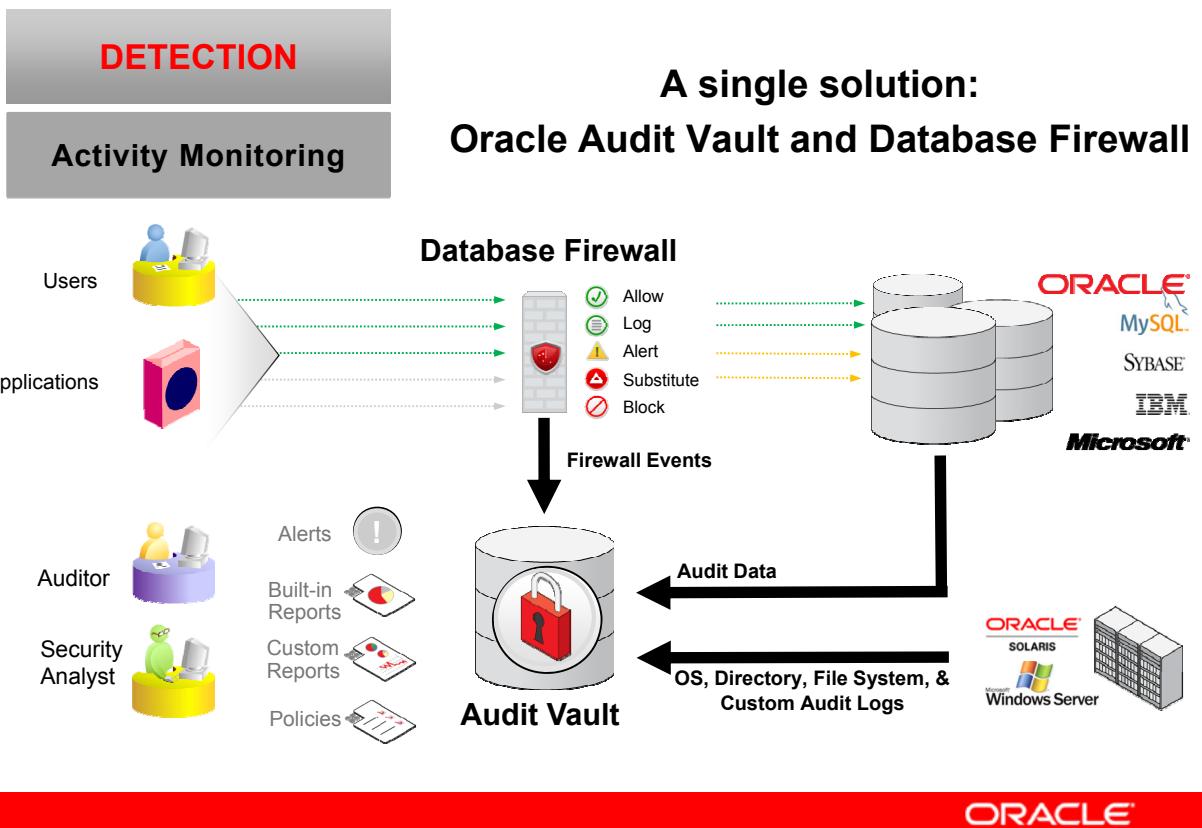
- Allows logs, alerts, substitutes, and blocks on SQL statements on the network
- Uses a SQL grammar analysis engine for high performance and accuracy, an approach that is superior to first generation database firewalls that relied on regular expressions

The Database Firewall is a dedicated server that runs the Database Firewall software. Each Database Firewall monitors SQL traffic on the network from database clients to secured target databases. The Database Firewall then sends the SQL data to the Audit Vault Server to be analyzed in reports.

An auditor can create firewall policies that define rules for how the Database Firewall handles SQL traffic to the database secured target. The firewall policy specifies the types of alerts to be raised in response to specific types of SQL statements. The policy also specifies when to block potentially harmful statements, and optionally substitute harmless SQL statements for blocked statements. The Database Firewall can operate in one of two modes:

- Database Policy Enforcement mode: The Database Firewall applies rules in a firewall policy to monitor SQL traffic to your secured target database and block traffic, and/or substitute benign SQL statements for potentially destructive ones.
- Database Activity Monitoring mode: The Database Firewall applies rules in a firewall policy to monitor and raise alerts about potentially harmful SQL traffic to your secured target database, but it does not block or substitute SQL statements.

Oracle Audit Vault and Database Firewall



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A single solution, Oracle Audit Vault and Database Firewall (AVDF), provides a first line of defense for databases and consolidates audit data from databases, operating systems, and directories. Effective monitoring and auditing can alert and block attempted policy violations, as well as provide comprehensive reports for compliance.

Key features of AVDF include the following:

- Monitor and control database activity on the network.
- Prevent SQL injections, unauthorized database access, misuse of database privilege.
- Capture and log database interactions on the network for forensic analysis and compliance reporting.
- Set audit policies and provision them from the Audit Vault Server console.
- Consolidate all audit data from Oracle and non-Oracle into secure centralized repository.
- Detect and alert on suspicious activities, including privileged user.
- Provide out-of-the box compliance reports for SOX, PCI, and other regulations.
- Streamline audits: Report generation, notification, attestation, archiving.

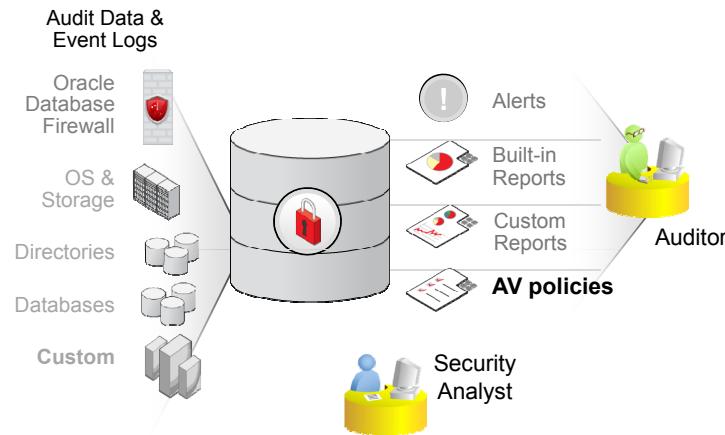
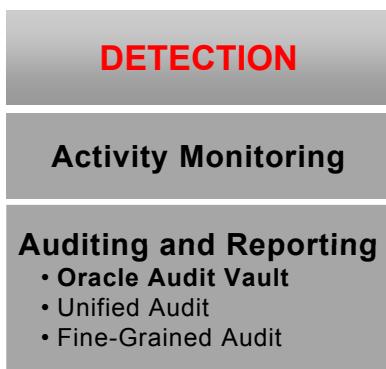
For detailed information of the AVDF features, refer to the following guides in the Oracle documentation:

- *Oracle Audit Vault and Database Firewall Administrator's Guide Release 12.1.1*
- *Oracle Audit Vault and Database Firewall Auditor's Guide Release 12.1.1*

If you need more information, refer to the following articles:

- <http://www.oracle.com/us/products/database/security/ds-security-audit-vault-firewall-1883353.pdf>
- <http://www.oracle.com/technetwork/products/audit-vault-and-database-firewall/audit-vault-and-firewall-wp-1896141.pdf>

Auditing and Alerting in Real-Time



- Database audit streamline
- Powerful detection and alert of suspicious activities
- Out-of-the box compliance and custom reports
- Consolidated multi-source reporting
- Built-in segregation of duties
- Centralized secure repository

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

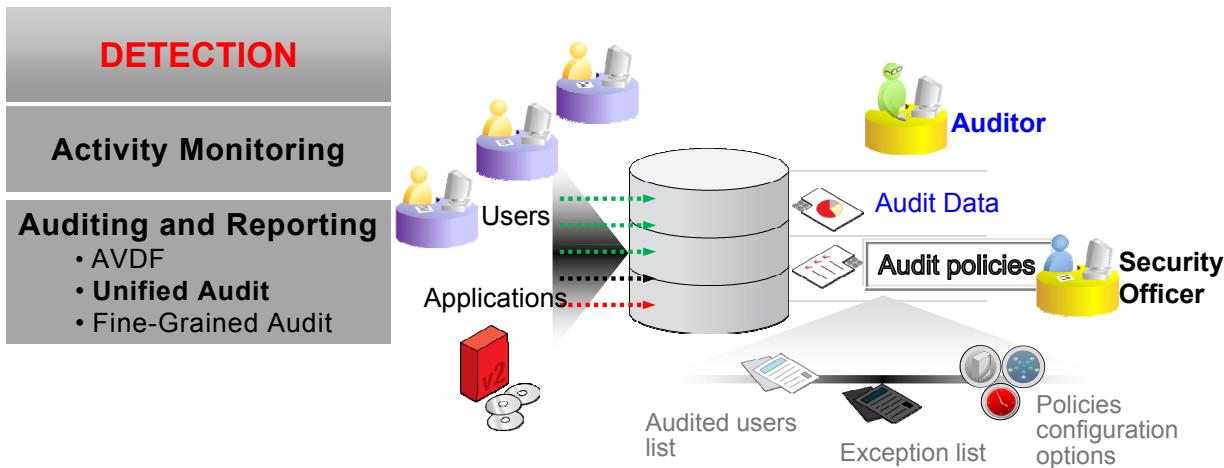
Compliance regulations and laws require businesses to secure business and personal data for customers, employees, and partners by auditing users, activities, and associated data.

Key benefits of Oracle Audit Vault include the following:

- Transparent collection and consolidation of audit data and logs from databases, operating systems, directories, file systems, and custom sources
- Detection and alerts of suspicious activity
- Built-in reports and custom reports. In addition, Oracle Audit Vault provides an open audit warehouse schema that can be accessed from Oracle BI Publisher, Oracle Application Express, or any third-party reporting tools.
- A secure and scalable audit warehouse built on Oracle's data warehousing technology and secured with Oracle's database security products, including Oracle Database Vault and TDE. Oracle Audit Vault includes Oracle Partitioning to improve manageability and performance.
- Centralized management of database audit settings (policies). This makes it easier for IT security officers and internal auditors to do their jobs.
- Oracle Audit Vault administrative activities through Oracle Enterprise Manager Cloud Control 12c.

For a complete understanding of the Audit Vault features, refer to the following guide in the Oracle documentation: *Oracle Audit Vault Administrator's Guide Release 10.3*

Unified Auditing



Unified Auditing

- Simplicity: Grouping audit options into a simple audit policy
- Consolidation: Merging all audit trails into a single unified audit trail table
- Security: Relying on a read-only audit trail table
- Performance: Enabling auditing with negligible overhead

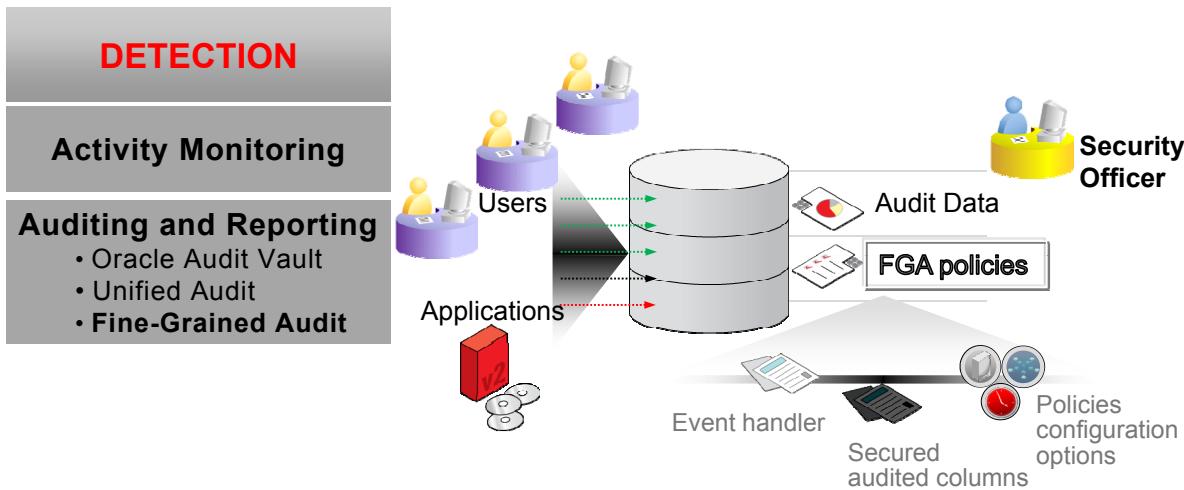
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The unified auditing facility needs to address the following challenges:

- **Simplicity:**
 - Grouping audit options into a simple audit policy
 - Allowing simpler action-based audit configurations
 - Setting condition-based audit configurations
 - Exempting users from being audited
- **Consolidation:** Merging all audit trails into a single unified audit trail table
- **Security:**
 - Relying on a read-only audit trail table
 - Auditing any operation related to audit configuration
 - Auditing any SYS user auditable action
 - Separating audit administration duties with audit administration roles, AUDIT_ADMIN and AUDIT_VIEWER
- **Performance:**
 - Oracle Database 12c auditing provides the ability to use System Global Area (SGA) queues for accumulating audit records, enabling auditing to be turned on with negligible overhead.

Fine-Grained Auditing



Fine-Grained Auditing (FGA):

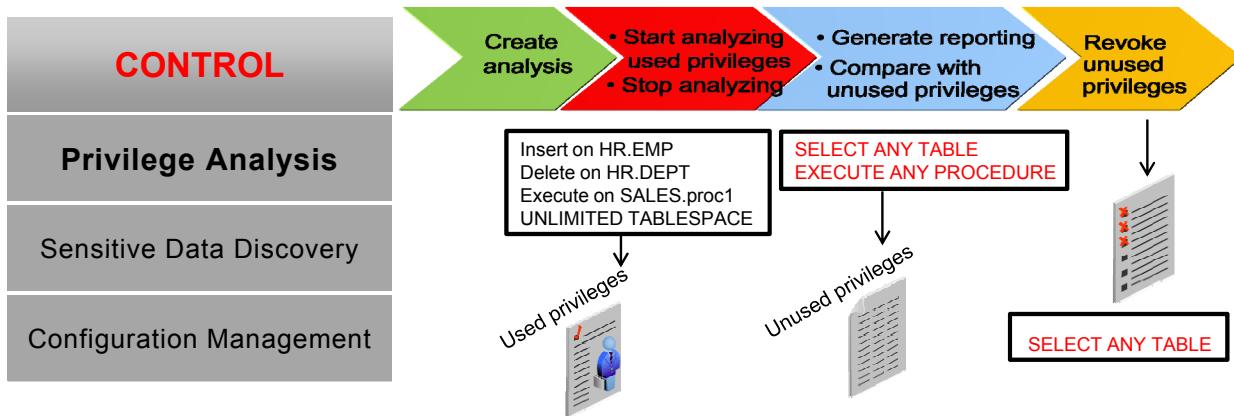
- Monitors data access based on content
- Audits `SELECT` and `DML` statements
- May fire an event handler procedure

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Fine-grained auditing (FGA) enables you to create policies that define specific conditions that must take place for the audit to occur. The specific conditions allowed by FGA are not those of unified audit. This enables you to monitor data access based on content. FGA options can be focused on individual columns within a table or view.

Discovering Use of Privileges and Roles



Oracle Database 12c Enterprise Edition Privilege Analysis

- Turns on privilege capture mode
- Reports on actual privileges and roles used in the database
- Helps revoke unnecessarily granted privileges
- Enforces least privilege and reduces risks
- Increases security without disruption

ORACLE

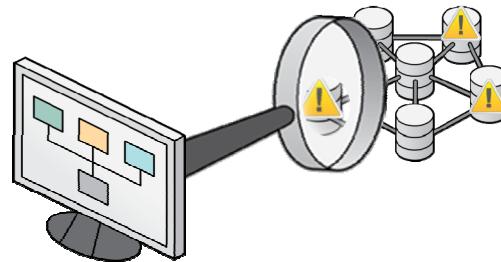
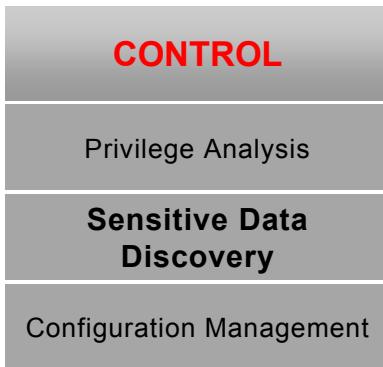
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Analyzing privilege use is beneficial in the following scenarios:

- Finding over-privileged users
- Finding privileges that are unnecessarily granted to the PUBLIC role
- Finding unnecessarily granted privileges of application database users

You can perform privilege analysis to find information about privilege usage for a database according to a specified condition, such as privileges to run an application module or privileges used in a given user session. The privilege analysis includes both system privileges and object privileges. When a user performs an action and you want to monitor the privileges that are used for this action, you can create and enable a privilege analysis policy. Afterward, you can generate a report that describes the used privileges.

Discovering Sensitive Data



Oracle Enterprise Manager 12c Cloud Control

Sensitive Data Discovery

- Scans Oracle for sensitive data
- Provides built-in, extensible data definitions
- Discovers application data models (ADM)
- Protects sensitive data appropriately: encrypt, redact, mask, audit, and so on

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Test data management features use Enterprise Manager's Data Discovery and Modeling (DDM) capability to enable operations, such as sensitive data discovery, data subsetting, and data masking. DDM enables scanning and tagging of sensitive data and modeling of data relationships incorporated within an ADM. Discovering what sensitive data you have is important for masking data. Data masking works on top of an ADM to scan your database for columns that are known to be sensitive, such as those that store credit card numbers, or those that you define as being sensitive.

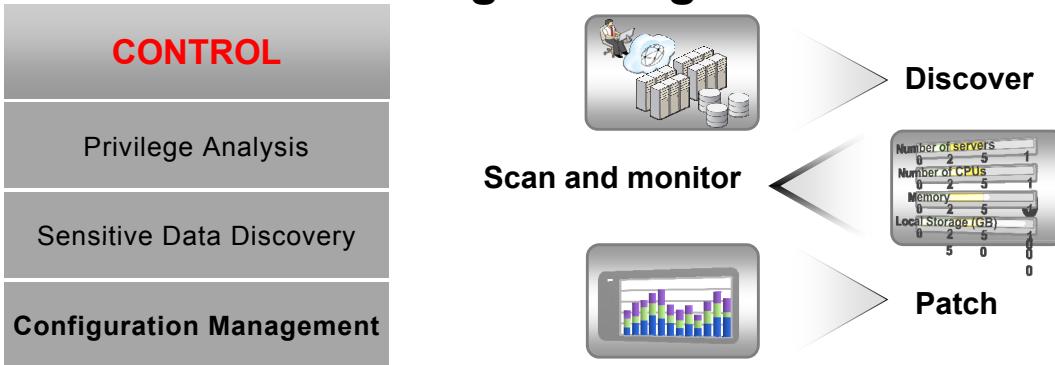
Sensitive data from your production system can be replaced with fictitious data in a development system during testing using Oracle Data Masking.

The process of identifying the sensitive columns so that they get included in the ADM will be explained in the lesson titled “Application Data Model and Oracle Data Masking.”

For a complete understanding of the Data Discovery and Modeling capabilities, refer to the following guide in the Oracle documentation:

- *Oracle Database Testing Guide 12c Release 1 (12.1) - Part “Test Data Management”*

Managing Configuration, Compliance, and Change Management



Oracle Enterprise Manager 12c Cloud Control Configuration, Compliance, Change Management

- Discover and classify databases, and collect configuration information
- Scan for best practices and compliance standards
- Detect unauthorized changes
- Automate remediation
- Patch and provision

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Configuration Management

Enterprise Manager Cloud Control collects configuration information for all managed targets across the enterprise and enables you to view, save, track, compare, search, and customize collected configuration information for all managed targets known to Enterprise Manager.

Compliance Management

Compliance Management provides the ability to evaluate the compliance of targets and systems as they relate to business best practices for configuration, security, and storage. This is accomplished by defining, customizing, and managing compliance frameworks, compliance standards, and compliance standard rules. In addition, Compliance Management provides advice on how to change configuration to bring your targets and systems into compliance.

Change Management

For example, for production databases, it is essential to ensure adherence to proper production control procedures. It is vital that administrators have the tools to detect unauthorized changes, such as an index being dropped without the change approvals. It then becomes vital to monitor changes to production databases day over day or week over week.

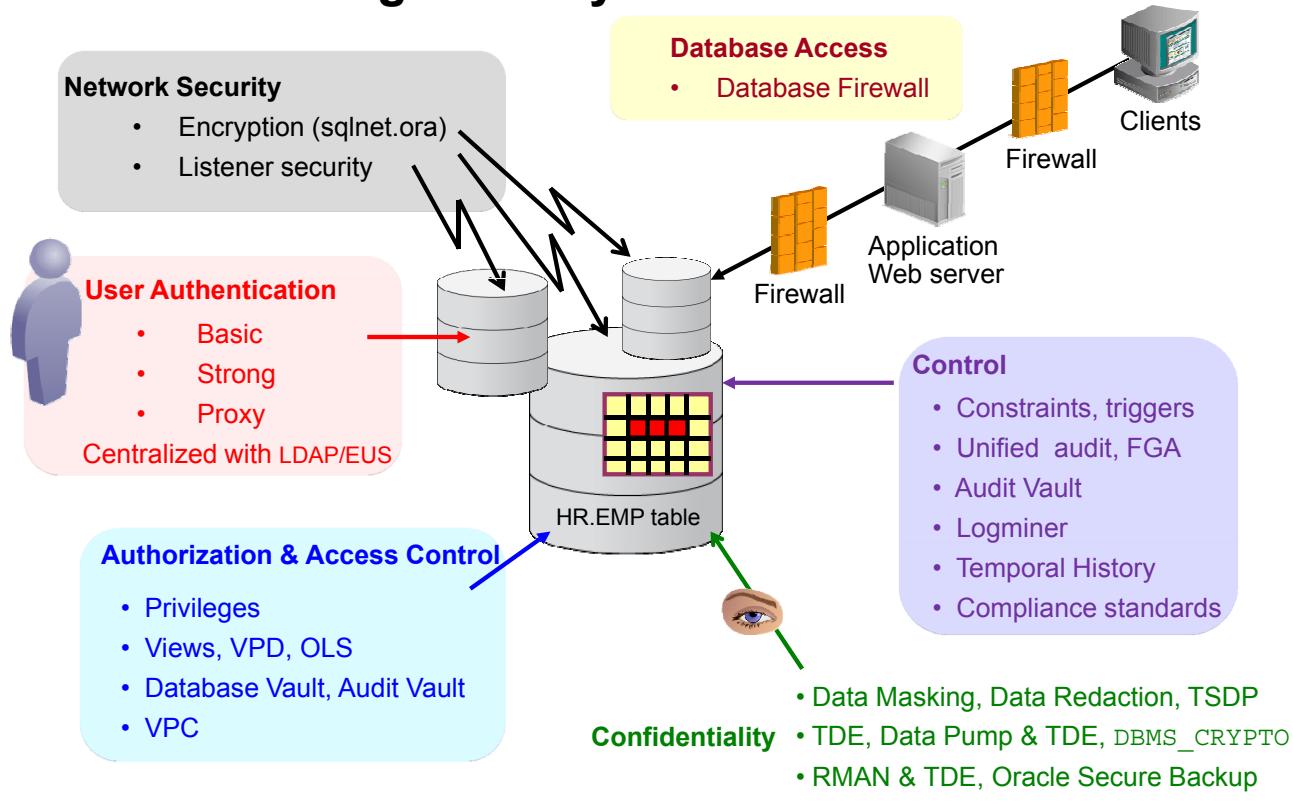
For a complete understanding of the configuration, compliance and change management capabilities, refer to the following guide in the Oracle documentation:

- *Oracle Enterprise Manager Lifecycle Management Administrator's Guide 12c Release 2 (12.1.0.2) - Part VIII "Configuration, Compliance, and Change Management"*

If you need more information, refer to the following article:

- <http://www.oracle.com/technetwork/database/security/security-compliance-wp-12c-1896112.pdf>

Enforcing Security at Different Levels



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The slide summarizes the Oracle solutions found for each level to maintain data integrity, ensure data protection in an Oracle Database, and control who accesses or modifies data in a database and when and how the operation can be performed.

Other Oracle solutions exist like *Oracle Real Application Security*. They are not covered in this course but in other courses. This course focuses on the security of the database. For example, *Oracle Real Application Security* manages **application security** for **application users** rather than database users and enables developers to manage **security for application-level tasks** rather than security for database-level tasks.

Quiz

Which of the following are used to implement and track *authorization*?

- a. Privilege analysis
- b. Oracle Database system privileges
- c. Proxy authentication



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a, b

Summary

In this lesson, you should have learned how to describe the following recommended solutions to common problems:

- Maintaining data integrity
- Protecting data
- Controlling data access



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- 3-1: Choosing a security solution
- 3-2: Configuring monitoring credentials for your database using Enterprise Manager Cloud Control
- 3-3: Viewing the PCI DSS 1.2 compliance framework
- 3-4: Implementing constraints to maintain integrity within application
- 3-5: Using triggers to maintain integrity within application
- 3-6: Using views to control the access to confidential data
- 3-7: Creating Database Vault realms to disallow a user from viewing his or her own schema objects



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

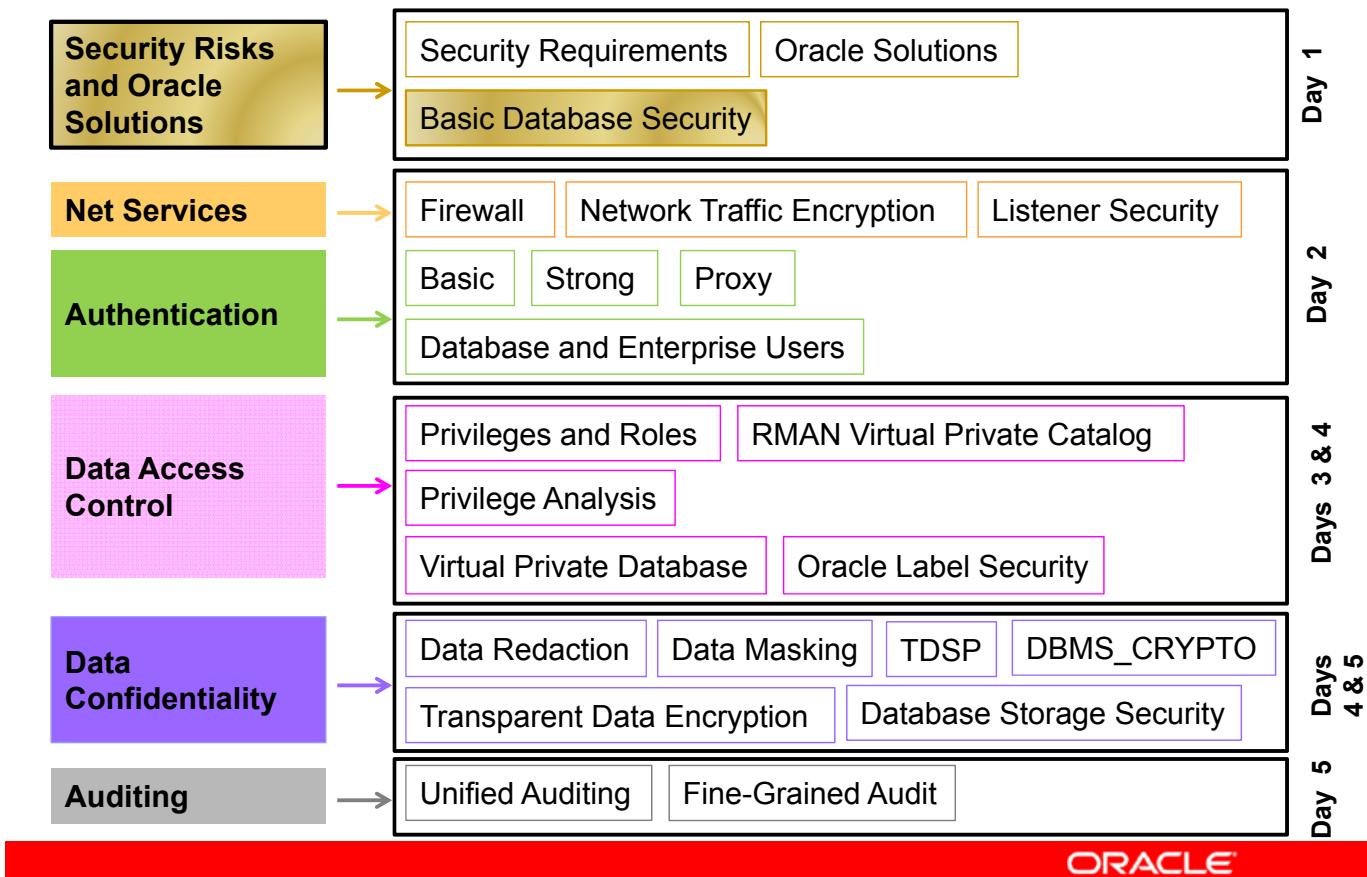
Implementing Basic Database Security

4



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Schedule



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Apply the principle of least privilege to the database
- Lock and expire default user accounts
- Change default user passwords
- Create strong passwords
- Enforce password management
- Protect the data dictionary



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

More Information

This lesson also explains how to configure Oracle Database 12c in a secure manner by adhering to industry standard “best security practices” for operational database deployments. Details about specific database-related tasks and actions can be found in the following:

- Other lessons in this course
- Other courses, including the following:
 - *Oracle Database 12c: Administration Workshop*
 - *Oracle Database 12c: Backup and Recovery Workshop*
- Oracle Database 12c documentation set

For a detailed explanation of the topics covered in this lesson, refer to the *Oracle Database Security Guide 12c Release 1*.

Database Security: Checklist

- Harden the operating system (OS).
- Apply the principle of least privilege:
 - Install only what is required.
 - Apply security patches.
 - Enforce password management.
 - Manage default user accounts.
 - Limit users with administrative privileges.
 - Protect the data dictionary.
 - Restrict the directories accessible by the user.
- Use available database security features.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Oracle Database server leads the industry in security. However, to fully maximize the security features that are offered, it is imperative that the database itself is well protected.

Furthermore, proper use of the security features and adherence to basic security practices help protect against database-related threats and attacks, and provide a more secure operating environment for the database.

Hardening the OS

Follow best practices of your operating system. For example, protect sensitive files with proper file permissions. Give the OS user only the privileges and permissions needed to do his or her job.

Practicing the Principle of Least Privilege

The principle of least privilege means that a user must be given only those privileges that are required to efficiently complete a task. This reduces the risk of a user, either accidentally or maliciously, modifying the data that he or she does not have the privilege to modify.

Use Available Database Security Features

The lesson titled “Choosing Security Solutions” displayed the list of available database security features, options, and products. Many of them are detailed in further lessons.

Reducing Administration Effort

- Use roles to grant users the same set of rights.
- Enable and disable roles as appropriate.



- Centralize the management of user-related information, including authorizations, in an LDAP-based directory service.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Create roles and grant privileges to the roles. Then grant the same role to users with the same security requirements. Normally, roles are organized around the job function of the employee. These roles are independent of the user. Therefore, if another user needs the same privileges, he or she can be assigned those privileges in a role. If a user is dropped, any directly granted privileges are also dropped. Assigning privileges through roles allows the role definition to exist without being assigned to a user.

As users change jobs, roles can be enabled or disabled for each user. When the job responsibilities change or the process changes, different privileges may be required. Then, privileges can be added to or removed from a role.

In sites with large numbers of users or a large number of systems, a significant effort is required just to keep user accounts current on each system so that users can continue to work. Often, administrators do not have the time or the information required to purge accounts of users who no longer have a need for access or users who are no longer authorized.

Assume a system with hundreds of users, each with accounts on many systems. When a user is removed, the administrator may not know all the accounts that need to be locked.

A central authentication system, which manages the user's credentials and authorizations, reduces the effort required to manage (create, update, and purge) accounts. This centralized management enables the creation of enterprise users and enterprise roles. Enterprise users are defined and managed in the directory. An enterprise role consists of one or more global roles, and might be thought of as a container for global roles.

Installing Only What Is Required

- Install only the required products and features.
- Plan your installation:
 - Determine what needs to be installed.
 - Determine whether a custom installation is required.
- You can install additional products later.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Oracle Database software contains many products in addition to the database server. Install additional products and options only as necessary. There is no need to maintain the additional products and options if they are not being used. They can always be properly and easily installed as required.

By reducing the number of products installed, you can reduce the possibility of patches being required and also the possibility of software conflicts. In addition, typically, you can reduce the cost of licenses and maintenance. These principles apply to all software products.

Note: The Oracle inventory directory keeps a record of installed software, and patches are applied only to the installed software. In many cases, if a new product or option is installed from the prepatched base distribution, the patches must be reapplied.

Applying Security Patches

- Use the Critical Patch Update process.
- Apply all security patches and workarounds.
- See the Oracle Security Products website.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Critical Patch Update (CPU) Process

Oracle Corporation uses a CPU process that bundles together critical patches on a quarterly basis. These patches are cumulative and include commonly requested and required prerequisite patches. The quarterly patch release comes with a risk assessment matrix to enable you to determine for your site the impact and security risks. See the “Critical Patch Updates, Security Alerts and Third Party Bulletin” page in Oracle Technology Network at <http://www.oracle.com/technology/deploy/security/alerts.htm>.

This page lists announcements of security fixes made in Critical Patch Update Advisories and Security Alerts, and it is updated when new CPU Advisories and Security Alerts are released. It is possible to receive notification of new announcements by email, as explained on the page.

You must subscribe to My Oracle Support to receive these updates.

Applying All Security Patches and Workarounds

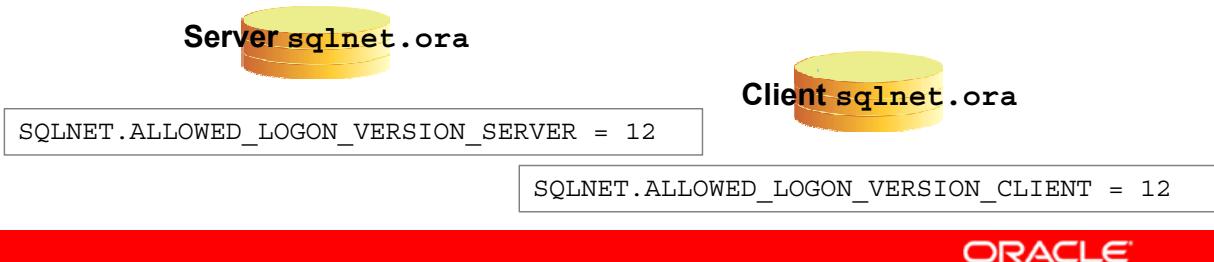
Always apply all relevant and current security patches for both the operating system on which the database resides and the Oracle software, and for all installed options and components. Read the Technical White Paper, “Critical Patch Update Implementation Best Practices.”

Referring to the Oracle Security Products Website

If you find a security vulnerability in the Oracle software, follow the instructions provided on the Security Alerts page at <http://www.oracle.com/technology/deploy/security/alerts.htm>.

Secure Password Support

- A stored hash value of a password is more secure—it:
 - Is case-sensitive since 11g by default
 - Uses secure hash algorithm (SHA-1)
 - Uses salt in addition to the hash algorithm
- You can configure Oracle Database to use the SHA-1 exclusively by setting `sqlnet.ora` parameters.
 - Ensures that new passwords are treated in a case-sensitive fashion (`SEC_CASE_SENSITIVE_LOGON` is deprecated)
 - Excludes the use of the 10g password version



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You must use more secure passwords to meet the demands of compliance to various security and privacy regulations. Passwords that are very short and formed from a limited set of characters are susceptible to brute force attacks. Longer passwords with more different characters allowed make the password much more difficult to guess or find.

- Passwords are case-sensitive.
- Passwords may contain multibyte characters without quoting. Of the special characters, only "\$," "_" and "#" are allowed in the password without quoting the password.
- Passwords are always passed through a hash algorithm, and then stored as a user credential. When the user presents a password, it is hashed, and then compared to the stored credential. In Oracle Database 12c, you can configure Oracle Database to use the SHA-1 hash algorithm exclusively. Set

`SQLNET.ALLOWED_LOGON_VERSION_SERVER` and
`SQLNET.ALLOWED_LOGON_VERSION_CLIENT` in the `sqlnet.ora` server and client files. The default value is 11 for Oracle Database 11g authentication protocols.

- Passwords always use salt. The SHA-1 function adds salt to the password when it is hashed, which provides additional protection. This enables your users to create passwords that are far more complex and, therefore, harder to hack. Salt is a unique (random) value that is added to the input to ensure that the output credential is unique.

Note: Usernames and passwords are still limited to 30 bytes.

Automatic Secure Configuration

- Default password profile enabled
 - Account is locked after 10 failed login attempts.
 - The `ora12c_verify_function` complexity check function is **enforced** in the `DEFAULT` default profile for non-SYS new users.
 - Three verification functions are available:
 - `verify_function_11g`
 - `ora12c_verify_function`
 - `ora12c_strong_verify_function`
 - No profile, including the `DEFAULT` profile, has any effect on the `SYS` user.
- Default auditing of certain privileges
- Last successful logon time recorded for non-SYS users even if not audited in the `DBA_USERS.LAST_LOGIN` column



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The secure configuration of Oracle Database 12c provides the following primary features:

- **Default password profile:** When creating a custom database by using the Database Configuration Assistant (DBCA), you can specify the Oracle Database 12c default security configuration. By default, if a user tries to connect to an Oracle instance multiple times using an incorrect password, logins are delayed after the third try. This protection applies for attempts made from different IP addresses or multiple client connections. Afterwards, it gradually increases the time elapsed before the user can try another password, up to a maximum of about 10 seconds. The basic password management applies rules provided by the database, such as password length, history, and complexity, to all user passwords. Ensure that all users must change their passwords periodically. The password management rules may be applied in the database to manage the database user, or in Oracle Internet Directory to enforce the same rules on enterprise users.
- **Built-in password complexity checking:** Three functions are provided and included in the `utlpwdmg.sql` file. The `ora12c_verify_function` function is enforced in the `DEFAULT` profile checking that the passwords contain no fewer than eight characters and do not exceed 256 characters. It is enforced by default for non-SYS users. Password complexity check is only enforced for `SYS` users when they try to change their password using `ALTER USER SYS IDENTIFIED BY <password>`. Password complexity check for existing users is enforced only when the password is being changed otherwise existing users can continue to use old passwords even if they were weak.

Locking, Expiring, or Changing Passwords of Default Accounts

- The Database Configuration Assistant (DBCA) expires and locks all accounts, except:
 - SYS, SYSTEM
 - SYSMAN, DBSNMP, MGMT_VIEW
- Default accounts provide easy access to the database.
 - For a manual database creation, lock and expire accounts by using:



```
SQL> ALTER USER hr PASSWORD EXPIRE ACCOUNT LOCK;
```

- Change the password on any account that has not been locked.
- Prefer PASSWORD command to ALTER USER command.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Locking and Expiring Default User Accounts

Oracle Database installs a number of default, preset database server user accounts. During the successful installation of the database server, the DBCA tool automatically locks and expires all default database user accounts, except the following:

- SYS, SYSTEM
- SYSMAN, DBSNMP, MGMT_VIEW

The MGMT_VIEW account, which is used for Oracle Enterprise Manager, has been granted the MGMT_USER role with only select privileges on some SYSMAN views.

Refer to My Oracle Support for notes about changing the passwords for the default user accounts under special circumstances.

If you create the database manually (that is, without using DBCA), none of the default database users are locked on the successful creation of the database. If left open in their default states, these user accounts may be exploited to gain unauthorized access to data or to disrupt database operations. After creating a database without using DBCA, lock and expire all default database user accounts except those in the preceding list. Use the appropriate SQL statements to perform such operations.

Changing Default Account Passwords

The other default accounts are created with a default password that is exactly the same as the username. For example, the MDSYS user is created with the MDSYS password. If any of the other default user accounts that were locked and expired upon installation need to be activated, assign a new password to that user account. If the patch process changes the schema objects of an account that is locked, you may be required to unlock the account for the patch to succeed.

The most trivial method by which the database can be compromised is by connecting to a default user account that still has a default password. Several exploits require access only to a low-privilege account.

Changing the Password Securely

It is important that passwords not be entered on the command line, because they show up in logs, shell history files, and in the process listing on UNIX.

It is just as important not to change a password with the `ALTER USER` command, because if you are not running an encrypted shell or Oracle native encryption, it passes over the network in clear text and can easily be intercepted by a network-sniffing program. The `SQL*Plus PASSWORD` command does not echo the password and encrypts the password before transmitting it over the network.

Even if you are only performing a quick action such as unlocking an account for patching, do not use a “standard” or weak password. Attackers will know patching windows due to announced outages, and it only takes a small window of time for an attacker to run a script that will loosen security settings or create a backdoor that he or she can use later to escalate his privileges.

Always use a secure connection and strong passwords.

Changing Default Passwords of Administrative Users

Change the default passwords associated with the `SYS` and `SYSTEM` users immediately upon installation and initial configuration of the database server. The DBCA and Oracle Universal Installer (OUI) tools force you to change the `SYS`, `SYSTEM`, `SYSMAN`, and `DBSNMP` passwords when creating new databases.

Changing the DBSNMP Password

Oracle Corporation recommends that the database instance be managed through Enterprise Manager Cloud Control.

For many sites, it is a common requirement to change passwords periodically. Changing the `DBSNMP` password is a special case because it is used by the agent to monitor the database.

Changing the MGMT_VIEW Password

The `MGMT_VIEW` password should not be changed. Changing the password will have a serious effect on many pages displayed by Oracle Enterprise Manager Database Express and Enterprise Manager Cloud Control. The password is a long, random string, generated from a combination of the user-provided `SYSMAN` password and the random output from `java.security.SecureRandom` hashed together. The `MGMT_VIEW` account has only read access to the tables that contain management data such as the percent CPU usage over time for servers in the data center.

The `MGMT_VIEW` user is assigned to the `DEFAULT` profile. Any changes to the `DEFAULT` profile that forces password expiration and changes could cause this account to become unusable.

Best Practice Tip: Create a profile specifically for the `MGMT_VIEW` account, in which the password never expires.

Privileged Accounts

- The **SYS** account:
 - Has all privileges with ADMIN OPTION
 - Can start up, shut down, and perform all maintenance commands
 - Owns the data dictionary
- The **SYSTEM** account is granted the **DBA** role only and none of the administrative privileges (SYSDBA, SYSOPER, SYSBACKUP, SYSASM, SYSKM, SYSDG).
- These accounts are *not* for normal use.

Note: “Oracle Created Database Users: Password, Usage and Files References” [ID 160861.1]



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

SYS and SYSTEM Accounts

The **SYS** account has all privileges with **ADMIN OPTION** and owns the data dictionary. To connect to the **SYS** account, you must use the **AS SYSDBA** clause. Any user who is granted the **SYSDBA** privilege can connect to the **SYS** account by using the **AS SYSDBA** clause.

Only users who are granted the **SYSDBA** or other administrative privileges are allowed to start up and shut down the database instance. These privileges are detailed in a later lesson about privileges.

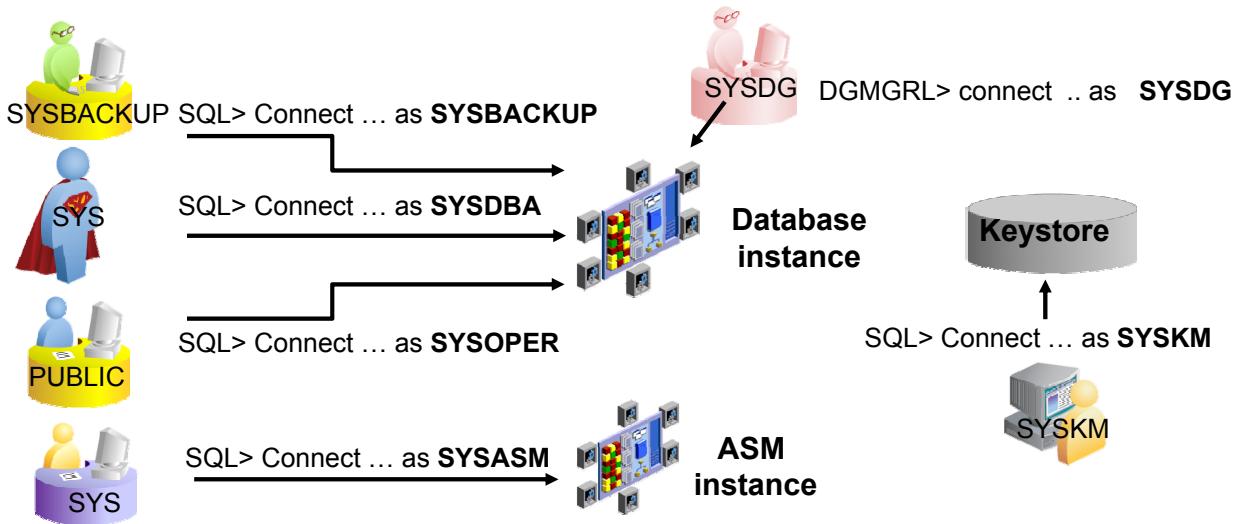
The **SYSTEM** account is granted the **DBA** role by default, but not the **SYSDBA** privilege.

Applying the principle of least privilege, these accounts are not used for routine operations. Users who need the **DBA** privileges have separate accounts with the required privileges granted to them. For example, Jim has a low privileged account called **JIM** and a privileged account called **JIM_DBA**. This method enables the principle of least privilege to be applied, eliminates the need for account sharing, and enables individual actions to be audited.

The **SYS** and **SYSTEM** accounts are required accounts in the database. They cannot be dropped.

Note: Refer to the following article available under My Oracle Support “Oracle Created Database Users: Password, Usage and Files References”[ID 160861.1].

Administrative Privileges



Allow remote database administration to start up and shut down from a remote node:

- Create a password file.
- GRANT <priv> TO <user>
- CONNECT sys/<pw>@service AS <priv>



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The six database administration privileges contain specific system privileges permitting specific database or instance or keystore or Data Guard administration tasks like STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, BACKUP, RECOVER, and ADMINISTER KEY MANAGEMENT commands. Details about the database administration privileges are covered in the lesson titled “Using Privileges and Roles.”

When you request to start up or shut down the database using Enterprise Manager Cloud Control, a host connection is made to the database server target machine as an OS user through the agent running on the target machine, so it is a “local” connection and the password file is not needed. Enterprise Manager Database Express always connects through the listener, so any AS SYSDBA connection must use the password file.

To shut down or start up a database remotely, with a utility such as SQL*Plus, the database must be configured to allow remote connections by privileged users. REMOTE_LOGIN_PASSWORDFILE must be set to EXCLUSIVE. The password file must be created with the orapwd utility, and the users must be added to the file.

List the users who have been added to the password file by selecting from the V\$PFILE_USERS view. Any user that has been granted SYSDBA, SYSOPER, SYSASM, SYSBACKUP, SYSKM, or SYSDG is listed; SYS is always listed. The password file is created on a per-instance basis; so in a RAC instance, the GV\$PFILE_USERS view may show different users for each instance ID.

Limiting Users with Administrative Privileges

- Restrict the following types of privileges:
 - Grants of system and object privileges
 - SYS-privileged connections: SYSDBA, SYSOPER, SYSBACKUP, SYSKM, SYSDG, and SYSASM
 - DBA-type privileges, such as DROP ANY TABLE
- Restrict run-time permissions.
- Example: List all users with the DBA role:

```
SQL> SELECT grantee FROM dba_role_privs  
      WHERE granted_role = 'DBA';
```

GRANTEE
SYS
SYSTEM



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Do not provide database users more privileges than necessary. To implement the least privilege, restrict the following types of privileges:

- Grants of system and object privileges
- SYS-privileged connections to the database, such as SYSDBA and SYSOPER
- Other DBA-type privileges, such as DROP ANY TABLE

Use the administrative privileges instead of SYSDBA as much as possible. Administratively, limit access to the SYSDBA password.

Create separate user accounts with DBA-like privileges, but limit the privileges to just those that are needed. Create roles similar to the DBA role but with only the required privileges for a job function. Assign the role to all users with that job function.

Restricting Permissions on Run-Time Facilities

Do not assign all permissions to any database server run-time facility, such as the Oracle Java Virtual Machine. Grant specific permissions to the explicit file paths for such facilities that may execute files and packages outside the database server.

The following example of a vulnerable run time gives SCOTT read access to all the files and directories below the root directory:

```
call dbms_java.grant_permission (
    'SCOTT', 'SYS:java.io.FilePermission', '/-', 'read' );
```

The following example of a better and more secure run-time call restricts SCOTT to reading all the files in the HR directory:

```
call dbms_java.grant_permission (
    'SCOTT', 'SYS:java.io.FilePermission', '/hr/*', 'read' );
```

Example: The example in the slide lists all users who have the DBA role granted to them. The two users displayed are the built-in users with the DBA role granted.

Separation of Responsibilities

- Users with DBA privileges must be trusted, but separation of responsibilities can:
 - Prevent abuse of trust
 - Allow audit trails to protect the trusted position
- To implement separation of responsibility:
 - DBA responsibilities must be shared
 - Accounts must never be shared
 - DBA and system administrator must be different people
 - SYSOPER and SYSDBA responsibilities must be separated
 - Security officer is an additional protection
 - Oracle Database Vault provides separation of duty features



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

These are the main requirements for a satisfactory separation of duties. Many small companies do not have enough people to fulfill all these requirements. For more stringent requirements, Oracle Database Vault enforces separation of duties.

- **DBAs must be trusted:** It is hard to restrict a DBA. To do their job, DBAs require high-level privileges. A DBA has a position of trust and must be thoroughly vetted. Even a trusted DBA must have accountability. A policy of separation of responsibility can:
 - **Prevent abuse of trust:** A DBA can view the encrypted passwords in the DBA_USERS table. The DBA can save any user's encrypted password, change the password, and connect as that user. When finished, the DBA can replace the user's original password by using the following command:

```
ALTER USER username IDENTIFIED BY VALUES 'encrypted_password';
```

The DBA need not know the original password. This action is not traceable unless auditing for the ALTER ANY USER privilege is turned on. If the password profile PASSWORD_REUSE_MAX is set to a number of days, the password cannot be set back to the original value for a number of days (of course, a DBA can also change the profile during this exploit). In Database Vault, by default, the DBA user cannot issue an ALTER USER IDENTIFIED BY... command. Only the database account manager would be allowed to do that.

- **Allow audit trails to protect the trusted position:** When auditing is carefully implemented and the guidelines have been followed, the audit trail can show that the particular person did not violate procedures or commit a damaging act. If a malicious user tries to cast suspicion on a trusted user, well-designed audit trails detect the act. In the lesson titled “Auditing Database Users, Privileges, and Objects,” you implement auditing on SYSDBA users.
- **DBA responsibilities must be shared:** In any job that requires trust, the principle of separation of responsibilities protects all involved. A single DBA does not have the oversight of another to correct errors, ask questions, and challenge actions. Two or more DBAs can hold each other accountable.
- **Accounts must never be shared:** This principle applies from the least-privileged account to the highest. When users share accounts (and passwords), no single user can be held accountable. With shared accounts, audit records can narrow the source but not confirm the actual user. It is especially important that root and DBA accounts are not shared. When multiple users require SYSDBA privileges, they can be added to the OSDBA group. Each user can have his or her own account and connect by using the AS SYSDBA clause.
- **The DBA and the system administrator must be different people:** The DBA must not have root access. System administrators with root access can always give themselves SYSDBA privileges. In very secure environments, even most system administrators are not given root access on production servers; instead, root privileges are handled via audited mechanisms such as sudo. By separating duties, the DBA and the system administrator can avoid situations where one person can easily compromise security. Even with trusted people in these positions, audit logs must be maintained and protected by yet another person to allow accountability. Historically, banks have required officers to take two consecutive weeks of vacation to prevent embezzlement and consequent concealment. Rotation of duties of system administrators, DBAs, and security officers adds another layer of accountability. If someone always administers the same task with no oversight, he or she could be hiding improper actions. Rotating duties increases the likelihood of catching an embezzler and preventing abuse of trust violations.
- **SYSOPER and SYSDBA responsibilities should be separated:** Do not grant SYSDBA when SYSOPER provides sufficient privileges. The SYSOPER role can do almost all of the day-to-day operations that SYSDBA can, but cannot see all the data. SYSOPER has only the privileges that are granted to the PUBLIC user. For example, the PUBLIC user does not have the SELECT ANY TABLE or EXECUTE ANY PROCEDURE privileges by default. The SYSDBA role has the privileges granted to SYS.
- **Security officer is an additional protection:** Additional roles such as security officer and auditor can provide another level of oversight. The security officer role could be responsible for account activations and grants. Auditing activities such as enabling, disabling, and viewing audit records can be included in the auditor role.
- **Oracle Database Vault enforces separation of duties:** Oracle Database Vault enforces separation of duties by default. Privileges of the SYSDBA user are limited. A separate user is required by Oracle Database Vault to create and alter users. With Oracle Database Vault, you have the ability to create new administrative roles with specific privileges tailored to the job and no more. Refer to *Oracle Database Vault Administrator’s Guide 12c Release 1 (12.1)* for detailed information about separation of duties.

Protecting the Data Dictionary

- The data dictionary is protected, by default, with the O7_DICTIONARY_ACCESSIBILITY initialization parameter set to FALSE.
- This configuration prevents users with the ANY system privileges from accessing the data dictionary base tables.
- The SELECT ANY DICTIONARY system privilege allows read access to **some** of the dictionary tables.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

By default, the data dictionary is protected to prevent users having the ANY system privileges from using such privileges on the data dictionary. The O7_DICTIONARY_ACCESSIBILITY initialization parameter controls this behavior.

The default value for O7_DICTIONARY_ACCESSIBILITY is FALSE. Setting the O7_DICTIONARY_ACCESSIBILITY parameter to FALSE may impact existing Oracle and third-party application environments that depend on dictionary access. Customers must refer to application-specific documentation for security guidelines.

This configuration does not restrict users who have the SYSDBA privilege.

Example

If this parameter is set to TRUE, any user with a DROP ANY TABLE system privilege can drop parts of the data dictionary either maliciously or accidentally.

If a user requires view access to the data dictionary, it is permissible to grant that user the SELECT ANY DICTIONARY system privilege. This configuration allows the user to read the data dictionary tables except DEFAULT_PWD\$, ENC\$, LINK\$, USER\$, USER_HISTORY\$, and XS\$VERIFIERS. Only user SYS has access to these tables, but user SYS can grant object privileges (such as GRANT SELECT ON USER\$ TO sec_admin) to other users, but it does not give the user access to objects owned by other schemas.

Limiting Privileges

Applying the principle of least privilege:

- Limit object and system privileges to actual needs.
- Do not use GRANT ALL ON object.
- Selectively grant specific privileges to users.
- Encapsulate sensitive objects.
- Do not use the UTL_FILE_DIR parameter.
- Use DIRECTORY objects:
 - Allows granular access to directory paths by the user
 - Allows read or write access separately
 - Is limited to the directories accessible by Oracle process

```
SQL> CREATE DIRECTORY local AS '/user/local/dbs';
SQL> GRANT READ, WRITE ON DIRECTORY local TO scott;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Analyze your system while keeping the principle of least privilege in mind. This requires more effort, but it can prevent misuse. There will always be bugs, and ingenious individuals will find new ways of exploiting weaknesses in the very complex business systems that have been deployed. Although the next exploit cannot be predicted, if you put the principle of least privilege into practice, the number and severity of possible exploitations can be reduced.

Grant only the privileges required. The GRANT ALL... command gives a user the ability to drop or alter a database object. These privileges must be reserved for the DBA, application administrator, and the object owner, except in special cases. Granting the DBA role must be limited to the actual DBA. At many sites, even the DBA role has more privileges than required, and a new role must be created with a subset of the DBA role privileges. The CONNECT, RESOURCE, and DBA roles are provided in the database for backward compatibility. Create new roles that are customized to the requirements of your site.

The RESOURCE role does not contain the UNLIMITED TABLESPACE anymore.

With Privilege Analysis, the security officer is able to analyze the privileges granted and used and compare them to those unnecessarily granted because unused.

Encapsulating Objects

Encapsulate objects containing sensitive data and access them only through stored procedures and packages. Sensitive data is not just data of a confidential nature, but data and objects that must be kept correct to prevent business disruption. The stored procedure can apply stringent controls on what can be updated easily. These procedures can also provide auditing mechanisms beyond the capability of database auditing features.

Restricting the Directories Accessible by the User

The `UTL_FILE_DIR` configuration parameter has been deprecated in favor of the `DIRECTORY` object. By default, no directories can be accessed. With `UTL_FILE_DIR`, all PL/SQL users can read from or write to all files in the directories specified by this parameter. Thus, all PL/SQL users must be trusted with the information in the directories specified by this parameter.

Never set `UTL_FILE_DIR = *` because this enables access to all directories accessible by the Oracle instance, including the data file and redo log directories.

Directory Objects

The recommended method of controlling database-user access to OS directories is through directory objects. Any OS path that is accessible to the Oracle software owner can be accessed through a directory object. A `DIRECTORY` object is created with a name and a path, as shown in the slide. All directories are created in a single namespace and are not owned by an individual schema. You must have the `CREATE ANY DIRECTORY` system privilege to create directories.

When a user creates a directory, that user is automatically granted the `READ` and `WRITE` object privileges on the directory with the `GRANT` option. The DBA can also grant these privileges to other users and roles.

Best Practice Tip: Avoid allowing different database instances access to the same directories. Differing security requirements could allow one to read or write sensitive data from or to the other. In addition, one instance could cause a Denial of Service (DoS) for the other instance by completely filling a common file system.

Limiting External Procedures Privileges

External procedures are executed from a library:

- No longer requires a network listener
- Oracle Database directly spawns an extproc process.
- The extproc runs with the privileges of another [OS user](#):

1. Create credential:

```
SQL> EXEC DBMS_CREDENTIAL.CREATE_CREDENTIAL ( -  
      credential_name => 'smith_cred', -  
      user_name => 'tjones', password => 'pass')
```

2. Define credentials in libraries:

```
SQL> CREATE OR REPLACE LIBRARY ext_lib AS 'ddl_1'  
      IN ddl_dir  
      CREDENTIAL smith_cred;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

CREATE LIBRARY Privilege

The CREATE LIBRARY privilege allows the user to create a library and run modules from the library. The modules in the library are called external procedures. The user can grant the EXECUTE privileges on the external procedures by creating a PL/SQL program unit for the external procedure and granting EXECUTE on the PL/SQL program unit.

External procedures no longer run with the privileges of the listener process. For safety reasons, Oracle external procedures run in a process that is spawned by the database and physically separate from the database. In most cases, you configure this process to execute as a user other than the Oracle software account.

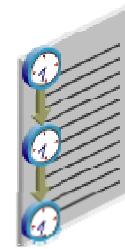
Only administrators and trusted users should have the CREATE LIBRARY privilege.

The libraries calling external procedures should be declared with credentials.

1. Create a credential using the DBMS_CREDENTIAL.CREATE_CREDENTIAL PL/SQL procedure where you define the following parameters:
 - `user_name`: Enter a valid operating system username to be used to run as the user.
 - `password`: Enter the password for the `user_name` user.
2. Create a library using the CREATE LIBRARY statement. Associate the credential with the library to run the extproc agent as a particular operating system user.

Managing Scheduler Security

- Database jobs execute with the privileges of the owner.
- The CREATE ANY JOB privilege allows a job to be created in any schema.
- The CREATE EXTERNAL JOB privilege is required to create an OS job.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Scheduler is a very powerful manageability feature. You can create jobs from PL/SQL blocks, stored procedures, programs, or external commands. Privileges of these jobs fall into two categories: database jobs and external jobs. External jobs are OS commands.

Database jobs are run with the privileges that are granted to the job owner directly or indirectly through default logon roles. External OS roles are not supported. Given sufficient privileges, users can create jobs in schemas of other users. The creator and the owner of the job can, therefore, be different. For example, if user JIM has the CREATE ANY JOB privilege and creates a job in the SCOTT schema, the job will run with the privileges of SCOTT.

You should grant the CREATE JOB system privilege to regular users who need to use the Scheduler to schedule and run jobs. You should grant MANAGE SCHEDULER to any database administrator who needs to manage system resources. Granting any other Scheduler system privilege or role should only be done with great caution. In particular, the CREATE ANY JOB system privilege and the SCHEDULER_ADMIN role, which includes the CREATE ANY JOB privilege, are very powerful because they allow execution of code as any user.

External Jobs

- An external job on UNIX:
 - Is named the `extjob` process
 - Is owned by `root`
 - Runs as the user specified in the `externaljob.ora` file
- An external job on Windows:
 - Is named the `OracleJobSchedulerSID` Windows service
 - Runs as `LocalSystem` or a named user



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A particularly important issue from a security point of view is handling external jobs. Only users that need to run jobs outside of the database should be allowed to do so. Grant the `CREATE EXTERNAL JOB` system privilege only to those users.

When upgrading from Oracle Database 10g Release 1 to 10g Release 2 or higher, `CREATE EXTERNAL JOB` is automatically granted to all users and roles that have the `CREATE JOB` privilege. It is recommended that you revoke this privilege from users that do not need it.

In Oracle Database 10g Release 2 and later, the security model allows you to edit the `external_job.ora` file in the `$ORACLE_HOME/rdbms/admin` directory to specify which user the `extjob` should run as. The `external_job.ora` file is owned by the `root` user and can be modified only by `root`.

In Oracle Database 11g on UNIX and Linux, the `extjob` process is owned by `root` with the setuid bit set, but the external jobs started by `extjob` run as the OS user `nobody` by default or the user specified by the `externaljob.ora` file.

On Windows, the jobs will run as the user specified for the `OracleJobScheduler<SID>` Windows service: either `LocalSystem` or a named user (local or domain).

Summary

In this lesson, you should have learned how to:

- Apply the principle of least privilege to the database
- Lock and expire default user accounts
- Change default user passwords
- Create strong passwords
- Enforce password management
- Protect the data dictionary



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To get a rather complete security checklist, you can refer to the note in My Oracle Support titled “*Security Check List: Steps to Make Your Database Secure from Attacks*” [ID 131752.1].

Practice: Overview

This practice covers the following topics:

- 4-1: Creating a security officer account performing various basic security checks
- 4-2: Managing secure passwords
- 4-3: Protecting the data dictionary
- 4-4: Investigating the compliance violations



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice 4-1: Create a security officer for later practices and performing various basic security checks.

Practice 4-2: You will enforce the password policy for this particular user. You will check the default password profile and apply one of the three predefined password verify functions.

Practice 4-3: Protect the data dictionary.

Practice 4-4: Investigate the compliance violations in your enterprise against the Basic Security Configuration for Oracle Database compliance standards.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

Module

Network Security



ORACLE

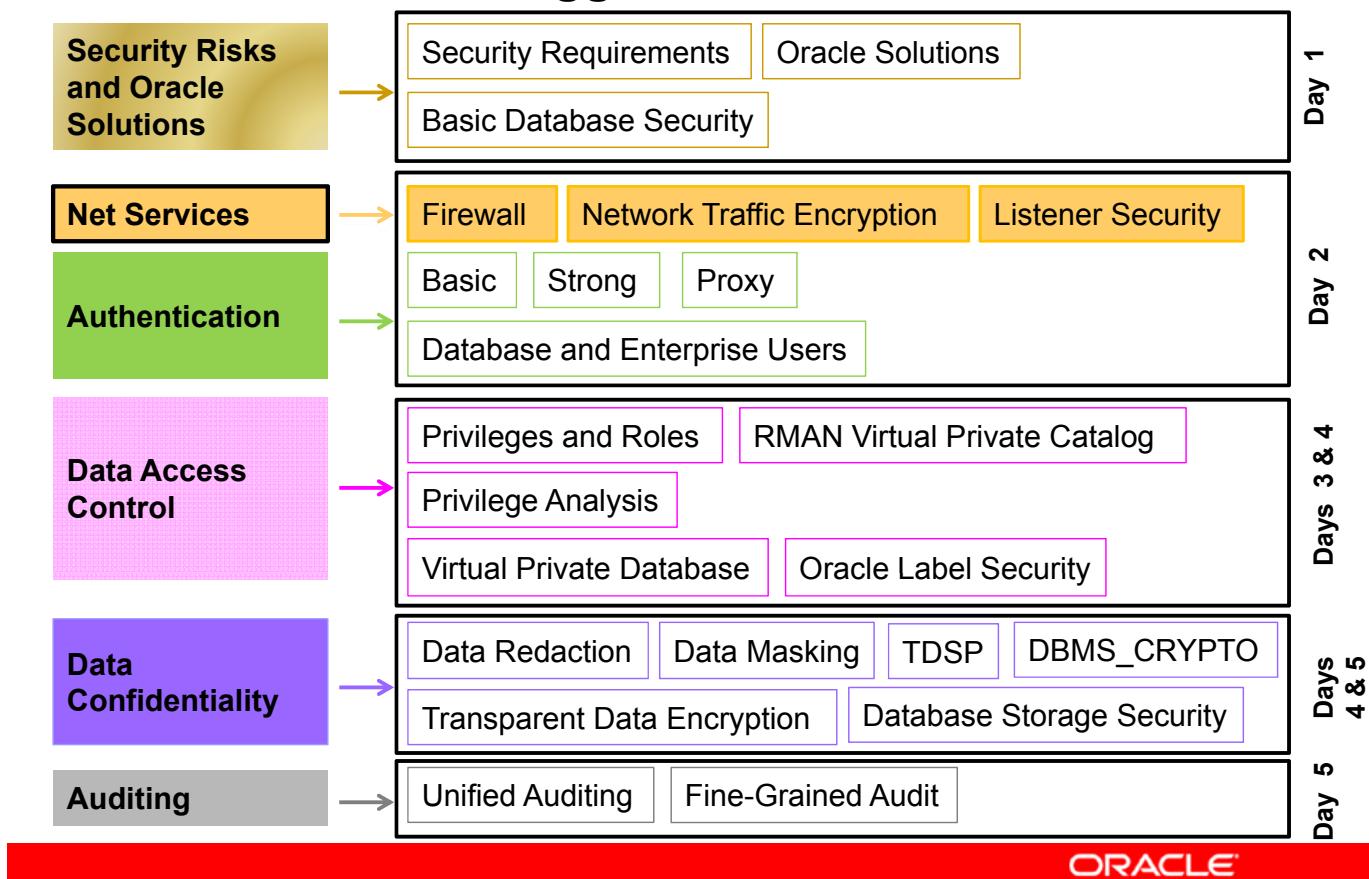
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Securing Network Services

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Schedule



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Restrict IP addresses
- Restrict node registration
- Administer the listener securely
- Monitor listener activity
- Manage fine-grained access to external network services



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Network Security: Checklist

- Use a firewall.
- Restrict IP addresses.
- Encrypt network traffic.
- Use network log files to monitor connections.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Data access and secure transfer of data are important considerations when deploying Oracle Database.

A firewall is a single point of control on a network. It is used to prevent unauthorized client computers from reaching the server. It acts as a filter, screening out unauthorized network users from using the Internet. It does this by enforcing access control on the basis of the contents of the packets of data that are being transmitted. It can thus protect against attacks on individual protocols or applications.

Firewalls are rule-based. They have a list of rules that define which client computers can connect and which cannot, and which SQL statements are allowed and which are not. They can compare the client computer's host name or IP name with the rules, and either grant the client computer access or not. They can analyze the client SQL statement with the rules, and either allow the client to execute the statement or not.

A firewall protects the perimeter from threats. Many organizations partition their networks with departmental firewalls to further limit privileges.

For more information about encryption on network traffic, see the appendix titled "Encrypting Network Traffic." As of the release of Oracle Database 12c, network encryption is no longer part of Oracle Advanced Security and is available in all licensed editions of the Oracle database.

Restricting Network IP Addresses: Valid Node Checking

- Limit access to the database by restricting client-computer IP addresses.
- Set the following `sqlnet.ora` parameters:
 - Turn on the feature:

```
tcp.validnode_checking = YES
```

- Deny access from these nodes:

```
tcp.excluded_nodes = (135.245.234.44)
```

- Allow access from these nodes:

```
tcp.invited_nodes =
(144.198.58.146, 144.198.58.147)
```

Note: Wildcards are allowed for IPv4 addresses.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can use the Oracle Net “valid node checking” security feature to allow or deny access to Oracle server processes from network client computers with specified IP addresses. By restricting client-computer IP addresses, you limit access to the database. The listener is responsible for performing this check. To use this feature, set the following `sqlnet.ora` parameters:

- `TCP.VALIDNODE_CHECKING`: If this parameter is set to `YES`, Oracle Net Services checks the `TCP.EXCLUDED_NODES` and `TCP.INVITED_NODES` parameters to determine which client computers must be allowed access to the database. The default value is `NO`.
- `TCP.EXCLUDED_NODES`: This specifies which client computers that use TCP/IP protocol are denied access to the database.
- `TCP.INVITED_NODES`: This specifies which client computers that use the TCP/IP protocol are allowed access to the database.

Include the listener node in the `INVITED_NODES` parameter. If you forget the listener node, this prevents start, stop, and administration commands from being performed.

If there are invalid host names listed in the list, the listener cannot be started. If an invalid entry is entered while the listener is running and reload is done then, the new list is ignored and the listener continues with the old list. `LSNRCTL` reports an error as the output of reload command.

`TCP.INVITED_NODES` and `TCP.EXCLUDED_NODES` cannot be used together. If both are specified, the `TCP.INVITED_NODES` list takes precedence.

Restricting Network IP Addresses: Guidelines

Network IP restrictions can help secure access to your server.
Consider the following guidelines:

- Do not use IP restrictions as your only security. IP addresses can be spoofed.
- Use listener node registration lists.
- Limit access by protocol: TCPS is a secure protocol and can be used.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

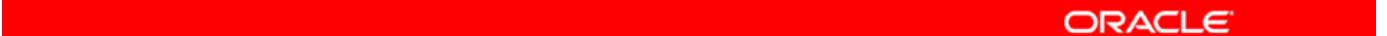
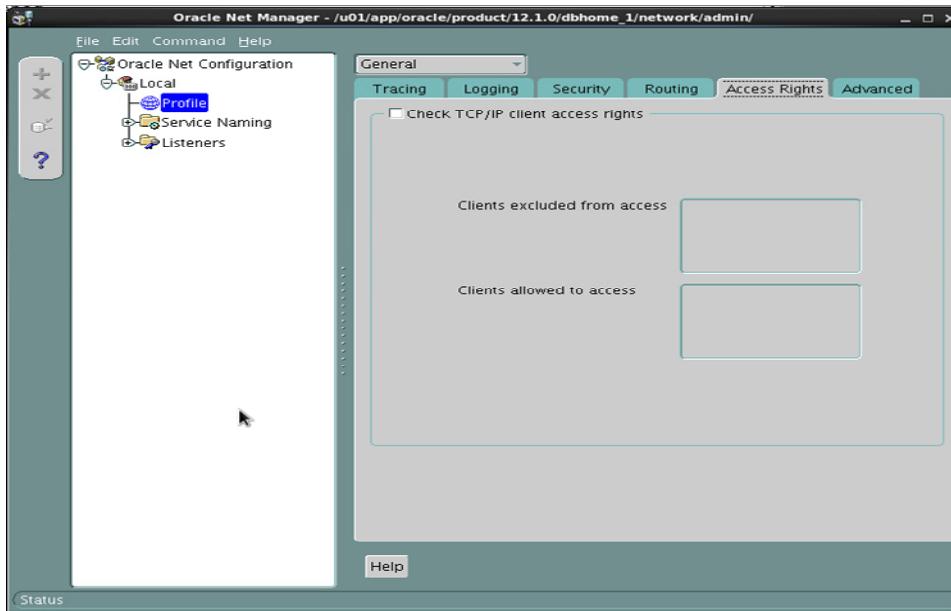
IP Address Spoofing

Because IP addresses can be spoofed, do not use this technique as your primary method for authorizing users.

Limiting Access by Protocol

The protocols included in the `listener.ora` file limit which protocol can access the database. However, because most networks use the TCP/IP protocol, this restriction is not as important as it once was.

Configuring IP Restrictions with Net Manager

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red background.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To start Oracle Net Manager on a UNIX or Linux system, execute the following command:

```
$ netmgr
```

1. In the navigator pane, select Local > Profile.
2. From the drop-down list in the right pane, select General.
3. Click the Access Rights tab. The window is shown in the slide.
4. Ensure that the “Check TCP/IP Client Access Rights” check box is selected.
5. In the two list boxes, enter either a host name or an IP address for a client computer that you want to include or exclude, using commas to delimit entries placed on the same line.

Quiz

You can limit access to the Oracle instance from specific IP addresses by setting only TCP.VALIDNODE_CHECKING = YES.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

Listener Security: Checklist

- Limit the privileges of the listener.
- Restrict node registration.
- Move the listener to a nondefault port.
- Secure administration.
- Protect against denial-of-service (DoS) attacks.
- Monitor listener activity.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Because the listener acts as the gateway to the network, some of these checklist items are closely related to network checklist items.

The items listed in the slide are discussed on the following pages.

Restricting Nodes Registration: Valid Node Checking Registration (VNCR)

- Limit access to databases by restricting node registration.
- Set the following `listener.ora` parameters:
 - Deny access from these nodes:

```
registration_excluded_nodes_listener = (10.1.35.*,
10.1.34.0/24, 2003::216:3eff:fe38:7307, node1)
```

- Allow access from these nodes:

```
registration_invited_nodes_listener = (10.1.34.*, node2)
```

- Reload the listener:

```
LSNRCTL> RELOAD listener_name
```

- Turn off the feature:

```
valid_node_checking_registration_listener_name = OFF
```

Note: Wildcards are allowed for IPv4 and IPv6 addresses.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can use the new 12c Oracle Net “valid node checking registration” (VNCR) security feature to restrict the set of nodes that can register with the listener. By restricting nodes registration, you limit access to the databases. The administrator can define the nodes that can register via:

- An invited list that explicitly enumerates nodes and subnets, which can register with the listener.
- An excluded list that enumerates the set of nodes, which cannot register with the listener.

Both invited and excluded lists for VNCR cannot be specified together. If both are specified, then the invited list takes precedence.

The administrator has the ability to switch off the VNCR feature by setting the `listener.ora VALID_NODE_CHECKING_REGISTRATION_list_name` parameter to the following values:

- `off` or `0` to specify VNCR is off, and no checking is performed.
- `on` or `1` or `local` to specify valid node checking registration is on, and all local IP addresses can register. If a list of invited nodes is set, all IP addresses, host names, or subnets in the list as well as local IP addresses are allowed.
- `subnet` or `2` to specify valid node checking registration is on, and all machines in the local subnets are allowed to register. If a list of invited nodes is set, all nodes in the local subnets as well as all IP addresses, host names, and subnets in the list are allowed.

The default value is `ON`.

Moving the Listener to a Nondefault Port

Make the detection of the listener reasonably difficult.

- Change the listening port number.
- Register the database by using LOCAL_LISTENER.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

It is very common to leave the listener in the default configuration listening on port 1521. You must assume that any hacker, internal or external, will know this port number. Because most remote connects require that the listener set up the connection, you can reduce illegitimate connections by hiding the listener. This is by no means a serious barrier, but will help prevent scripted attacks, viruses that make use of default ports, and unsophisticated attacks that rely on published information.

When you move the listener to a nondefault port, you must give the database the listener port number with the LOCAL_LISTENER initialization parameter. The database will automatically register with a listener on the local machine if it is using port 1521, and with a listener described by the LOCAL_LISTENER parameter.

Administering the Listener

- Password-protecting the listener is no longer supported.
- Local listener administration is secured through local operating system authentication.

⇒ Local listener administration is restricted to the user who started the listener.

- By default, remote listener administration is disabled.
- Remote listener administration allows all commands except START.

⇒ Start the listener on the same computer on which the utility is running.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 12c, the listener password feature is no longer supported. This does not cause a loss of security because authentication is enforced through local operating system authentication.

Oracle recommends that you perform listener administration in the default mode (local listener administration), and access the system remotely using a remote login. When you administer the listener remotely, use Oracle Enterprise Manager Cloud Control or Secure Shell (ssh) to access the remote host.

Note: The LSNRCTL utility may prompt for a password on Microsoft Windows if the database was installed with the Oracle Home User. The password is the operating system password for the Oracle Home User. The prompt is displayed only if the listener service does not exist and needs to be created as part of starting the listener.

Administering the Listener Securely

- Use TCP/IP for SSL (**TCPS**) when administering over a nonsecure network.
- Use **COST** parameters to define which transports are considered secure for:

```
Listener1 =
  (DESCRIPTION = (ADDRESS_LIST =
    (ADDRESS =
      (PROTOCOL = tcps)
      (HOST     = remote_server1.oracle.com)
      (PORT     = 8281)))
  SECURE_CONTROL_listener1 = tcps           Control commands
  SECURE_PROTOCOL_listener1 = ipc            Control & registration
                                                commands
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

If you are administering the listener remotely over a nonsecure network and require maximum security, configure the listener with a secure protocol address that uses TCP/IP for secure sockets layer (TCPS). To use TCPS, configure `listener.ora` with the `PROTOCOL=TCPS` parameter as shown in the slide.

The class of secure transports (COST) parameters specify a list of transports that are considered secure for administration and registration of a particular listener. The COST parameters identify which transports are considered secure for that installation and whether the administration of a listener requires secure transports. Configuring these parameters is optional.

- `SECURE_CONTROL_listener_name`: To specify the transports on which control commands are to be serviced. The `SECURE_CONTROL_listener1 = (TCPS,IPC)` example allows administration requests of `listener1` only on TCPS and IPC transports.
- `SECURE_PROTOCOL_listener_name`: To specify the transports on which administration and registration requests are accepted for `listener1`. The `SECURE_PROTOCOL_listener1 = IPC` example allows control commands and service registrations only on the IPC channel.

COST parameters can be used in combination to control which transports accept service registration and control commands. In the example in the slide, control commands are accepted only on the TCPS transport, and service registrations are accepted only on the IPC channel.

Preventing Online Administration of the Listener

- Listener configuration cannot be changed online when the `listener.ora` file contains the following parameter:

```
ADMIN_RESTRICTIONS_LISTENER=ON
```

- To change the configuration, you must:
 - Make the changes in the `listener.ora` file manually
 - Reload the configuration (not stop, not start)

```
LSNRCTL> RELOAD listener_name
```

- This configuration requires the administrator to have write privileges on the `listener.ora` file.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can restrict run-time administration of the listener by using the `ADMIN_RESTRICTIONS_listener_name` parameter. When this parameter is set, the listener refuses to accept the `SET` commands that alter its parameters. This feature provides additional security on the locally managed listener. To change any of the listener parameters, including `ADMIN_RESTRICTIONS_listener_name`, modify the `listener.ora` file and use the `RELOAD` command to reload the parameters. The `RELOAD` command enables the new changes without explicitly stopping and restarting the listener.

To turn the parameter on for the default listener, enter the following `listener.ora` parameter:

```
ADMIN_RESTRICTIONS_LISTENER=ON
```

To turn the parameter on for a listener named `PAY_LSNR`, include the following parameter:

```
ADMIN_RESTRICTIONS_PAY_LSNR=ON
```

To change `listener.ora`, the user must have the operating system write privilege on the file.

INBOUND_CONNECT_TIMEOUT

- Protect the listener from DoS attacks with the following network parameters:
 - SQLNET.INBOUND_CONNECT_TIMEOUT
 - INBOUND_CONNECT_TIMEOUT_listener_name
- These parameters:
 - Set the time allowed for a connection to complete authentication
 - Log failures with source IP addresses



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

It is possible for connections without authentication to stay open indefinitely. Malicious clients can attempt to flood database servers with connect requests that consume resources. This is a possible scenario for a DoS or distributed DoS attack.

To prevent this scenario, use the SQLNET.INBOUND_CONNECT_TIMEOUT and INBOUND_CONNECT_TIMEOUT_listener_name parameters in conjunction. In the sqlnet.ora file, SQLNET.INBOUND_CONNECT_TIMEOUT is set to a value in seconds (default value is 60 seconds) and it determines how long a client has to provide the necessary authentication information to a database. Set the INBOUND_CONNECT_TIMEOUT_listener_name parameter in the listener.ora file. INBOUND_CONNECT_TIMEOUT_listenername is set to a value in seconds and it determines how long a client has to complete its connect request to the listener after the network connection has been established.

If the client fails to establish a connection and complete authentication in the time specified, the database server terminates the connection. The database server logs the IP address of the client and an ORA-12170 : TNS : Connect timeout occurred error message to the sqlnet.log file.

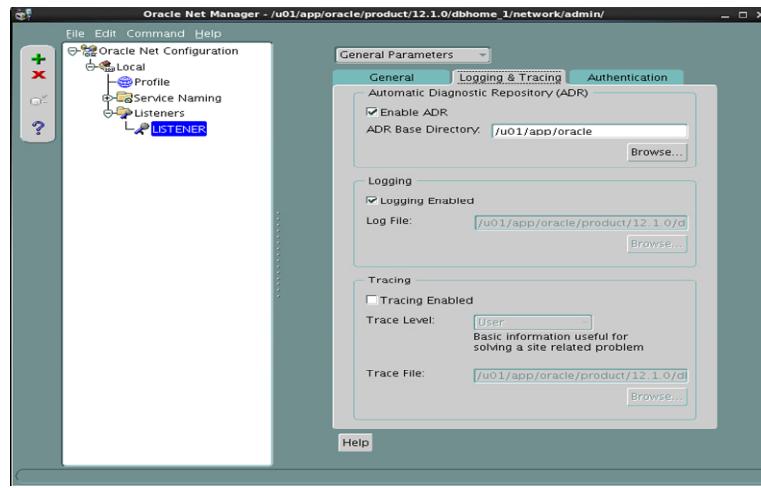
To protect both the database server and the listener, Oracle Corporation recommends setting both these parameters in combination. When specifying values for these parameters, consider the following recommendations:

- Set both parameters to an initial low value.
- Set the value of the `INBOUND_CONNECT_TIMEOUT_listener_name` parameter to a lower value than that of the `SQLNET.INBOUND_CONNECT_TIMEOUT` parameter.

For example, you can set `INBOUND_CONNECT_TIMEOUT_listener_name` to two seconds and `INBOUND_CONNECT_TIMEOUT` to three seconds. If clients are unable to complete connections within the specified time due to system or network delays that are normal for a particular environment, increment the time as needed.

Setting Listener-Logging Parameters

- In the `listener.ora` file:
 - `DIAG_ADR_ENABLED_listener_name`
 - `LOG_DIRECTORY_listener_name`
 - `LOG_FILE_listener_name`
- With Oracle Net Manager:



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 12c, by default, the listener log is sent to the Automatic Diagnostic Repository (ADR). The ADR is a directory specified by the `ADR_BASE` initialization parameter.

On Linux, with a default installation, the listener log defaults to:

`$ORACLE_HOME/diag/tnslsnr/hostname/listener/alert/log.xml`

You can set the following logging parameters in the `listener.ora` file:

- `DIAG_ADR_ENABLED_<listener>=OFF` is used to have the log file go to another location. The following log parameters are ignored if this parameter is ON.
- `LOG_DIRECTORY_listener_name` is the destination directory for the log file. The default directory on UNIX is `$ORACLE_HOME/network/log` and on Windows is `%ORACLE_HOME%\network\log`.
- `LOG_FILE_listener_name` is the file name for the log file. The default name is `listener.log`.

Setting Logging in the Listener

To set the logging parameters in Oracle Net Manager, perform the following steps:

1. In the navigation pane, select the listener that you want to configure.
2. Select General Parameter from the drop-down list.
3. Click the “Logging & Tracing” tab.
4. Deselect “Enable ADR.”
5. Select “Logging Enabled” and specify the location.
6. To save the configuration by using the menu options, select File > Save Network Configuration.

Setting Logging During Run Time

If the `DIAG_ADR_ENABLED_<listener>` parameter is set to OFF, you can also control the listener logging by using the `SET` command in the Listener Control utility with the following parameters:

- `LOG_DIRECTORY` is the destination directory for the listener log file.
- `LOG_FILE` is the name of the log file for the listener.

These changes remain in effect until the listener is shut down.

Analyzing Listener Log Files

The listener log contains the following information:

- Listener log audits:
 - Client connection request
 - Listener Control utility commands
- Listener service registration events:
 - service_register
 - service_update
 - service_died
- Listener direct hand-off information



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Listener Log Audit Trail

The listener log file contains audit trail information that helps you analyze network usage. The following information is recorded:

- A client connection request
- A SHOW, SET, RELOAD, START, STOP, STATUS, or SERVICES command issued by the Listener Control utility

The audit trail record has the following format:

Timestamp * Connect Data * [Protocol] * Event * [SID|Service] *
Return Code

The audit trail fields have the following properties:

- Protocol address information and system identifier (SID), or service name information, appear only when a connection is attempted.
- A successful connection or command returns a code of zero.
- A failed connection or command includes a return code that maps to an error message.

When diagnosing security problems, you primarily use the listener audit records.

Listener Service Registration Event

Service registration events are recorded in the `listener.log` file as follows:

- `service_register` indicates that the listener received registration information for an instance.
- `service_update` indicates that the listener received updated registration information for a particular instance.
- `service_died` indicates that the listener lost its connection to the `PMON` background process.

The service registration records have the following format:

`Timestamp * Event * Instance Name * Return Code`

The service registration fields have the following properties:

- It is normal for the events to appear multiple times in a row for one instance.
- A successful registration returns a code of zero—that is, the client can connect to the instance.
- A failure produces a code that maps to an error message.

Frequent starts and stops of the database instance may indicate a problem, such as abuse of privileges.

Listener Direct Hand-Off

Information concerning the direct hand-off to dispatchers is recorded in the `listener.log` file.

The direct hand-off event records have the following format:

`Timestamp * Presentation * Handoff * Error Code`

The direct hand-off fields have the following properties:

- A successful connection or command returns a code of zero.
- A failure produces a code that maps to an error message.

Listener Log Connect: Examples

```
26-JUN-2013 01:55:45 *
(CONNECT_DATA=(SERVICE_NAME=orcl) (CID=(PROGRAM=sqlplus) (HOST=server1) (USER=oracle))) *
(ADDRESS=(PROTOCOL=tcp) (HOST=127.0.0.1) (PORT=16694)) )
* establish * orcl * 0
...
26-JUN-2013 02:58:33 *
(CONNECT_DATA=(SERVICE_NAME=p0orcl) (CID=(PROGRAM=sqlplus) (HOST=server2) (USER=oracle))) *
(ADDRESS=(PROTOCOL=tcp) (HOST=10.100.10.110) (PORT=58323)) * establish * p0orcl * 12514
TNS-12514: TNS:listener does not currently know of
service requested in connect descriptor
...
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Each record in the `listener.log` file is on a single line. An asterisk (*) is used to delimit the fields in the log record. These examples are formatted for readability.

The example in the slide contains the following records:

- The first record shows a successful connect request. It includes the following components:
 - The current date and time:

26-JUN-2013 01:55:45

- The computer and user that originated the request:

(CONNECT_DATA=(SERVICE_NAME=orcl) (CID=(PROGRAM=sqlplus) (HOST=server1) (USER=oracle)))

- The destination requested:

(ADDRESS=(PROTOCOL=tcp) (HOST=127.0.0.1) (PORT=16694))

- The type of request:

establish

- The service requested:

orcl

- The return code:

0

The zero return code indicates that the listener found the service. It does not indicate that the user made a successful connection. For example, if the user enters an invalid password, the listener still shows a return code of zero. You can audit connection attempts to get information about failed database connections.

- The second record shows an unsuccessful connect request. It includes the following components:

- The current date and time:

26-JUN-2013 02:58:33

- The computer and user that originated the request:

(CONNECT_DATA= (SERVICE_NAME=p0orcl) (CID= (PROGRAM=sqlplus) (HOST=server2) (USER=oracle)))

- The destination requested:

(ADDRESS= (PROTOCOL=tcp) (HOST=10.100.10.110) (PORT=58323))

- The type of request:

establish

- The service requested:

p0orcl

- The return code:

12514

- Because the return code is nonzero, the next line displays the associated error message:

TNS-12514: TNS:listener could not resolve SERVICE_NAME given in connect descriptor

The error message is truncated in the slide.

Parsing Security Breaches

If your listener is being probed, it shows up as a series of failed connection attempts from the same source but with different destinations. A DoS attack appears as multiple connections from the same source that may or may not fail. Probing may be sufficient for a DoS attack, so a successful connection may not be required.

26-JUN-2013 02:58:33 * *

(ADDRESS= (PROTOCOL=tcp) (HOST=10.100.10.110) (PORT=58323)) *

establish * p0orcl * 12514

TNS-12514: TNS:listener does not currently know of service requested in connect descriptor

Listener Log Command: Examples

```
TNSLSNR for Linux: Version 12.1.0.1.0  ...
...
System parameter file is ...
...
Started with pid=21919
Listening on: ...
...
26-JUN-2013 23:09:29 * ... * status * 0
...
26-JUN-2013 23:09:40 * ... * reload * 0
...
No longer listening on: ...
...
26-JUN-2013 23:13:55 * ... * stop * 0
...
```

1

2

3

4



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The examples in the slide contain the following records:

1. The first record shows a successful listener startup:

```
TNSLSNR for Linux: Version 12.1.0.1.0 - Production on 26-JUN-2013
01:49:32
```

```
System parameter file is /home/oracle/labs/NET/listener.ora
Log messages written to /home/oracle/labs/NET/listen1.log
Trace information written to
/u01/app/oracle/product/12.1.0/dbhome_1/network/trace/
listen1.trc
Trace level is currently 0
```

```
Started with pid=21919
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=server1)(PORT=13001)))
Listener completed notification to CRS on start
```

2. The next record shows a listener status command:

```
TIMESTAMP * CONNECT DATA [* PROTOCOL INFO] * EVENT [* SID] *
RETURN CODE
26-JUN-2013 23:09:29 *
(CONNECT_DATA=(CID=(PROGRAM=) (HOST=server1)
(USER=oracle)) (COMMAND=status) (ARGUMENTS=64) (SERVICE=LISTENER)
(VERSION=185599488)) * status * 0
```

3. The third record shows a successful listener reload:

```
26-JUN-2013 23:09:40 *
(CONNECT_DATA=(CID=(PROGRAM=) (HOST=server1)
(USER=oracle)) (COMMAND=reload) (ARGUMENTS=64) (SERVICE=LISTENER)
(VERSION=185599488)) * reload * 0
```

4. The last record shows a successful listener shutdown:

```
No longer listening on:
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc) (KEY=EXTPROC1)))
No longer listening on:
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (server1) (PORT=12001)))
Listener completed notification to CRS on stop
26-JUN-2013 23:13:55 *
(CONNECT_DATA=(CID=(PROGRAM=) (HOST=server1)
(USER=oracle)) (COMMAND=stop) (ARGUMENTS=64) (SERVICE=LISTENER)
(VERSION=185599488)) * stop * 0
```

Managing Fine-Grained Access to External Network Services

Control the access to external network services:

- Limit access external network services from the database.
- Specify groups of users who can connect to one or more host computers, based on privileges granted.

```
SQL> EXEC DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE ( -  
    host => 'host_name', -  
    lower_port => null|port_number, -  
    upper_port => null|port_number, -  
    ace => xs$ace_type( -  
        privilege_list => xs$name_list('privilege'), -  
        principal_name => 'user_or_role', -  
        principal_type => xs$ace_type_user), -  
        granted      => true);
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The network utility PL/SQL packages such as UTL_TCP, UTL_INADDR, UTL_HTTP, UTL_SMTP, and UTL_MAIL and also the DBMS_LDAP PL/SQL package allow Oracle users to make network callouts from the database by using raw TCP or higher-level protocols built on raw TCP. By default, the execute privilege on these packages is granted to PUBLIC. Access to network resources through these packages is controlled by access control lists (ACLs) containing entries (ACE for Access Control Entry).

Use the DBMS_NETWORK_ACL_ADMIN package to create an ACL, which includes the following arguments:

- Network resources that can be specified as IP addresses, machine or domain names, or URLs. Wildcard characters are allowed for the host name to assign the ACL to all the hosts of a domain. The use of wildcards affects the order of precedence for the evaluation of the ACL. Fully qualified host names with ports are evaluated before hosts with ports. Fully qualified host names are evaluated before partial domain names, and subdomains are evaluated before the top-level domain level.
- Optionally, a port or range of ports
- The principal, which is either a user or a role
- The list of privileges granted or denied (denied with the GRANTED argument set to false)
- The principal type can be either XS_ACL.PTYPE_DB for a database user or role or XS_ACL.PTYPE_XS for an Oracle Database Real Application Security application user.

Summary

In this lesson, you should have learned how to:

- Restrict IP addresses
- Restrict node registration
- Administer the listener securely
- Monitor listener activity
- Create fine-grained access to external network services



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- 5-1: Preventing online administration
- 5-2: Reviewing and analyzing the listener log file



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Module

Authentication Models

A solid red horizontal bar spanning most of the page width.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

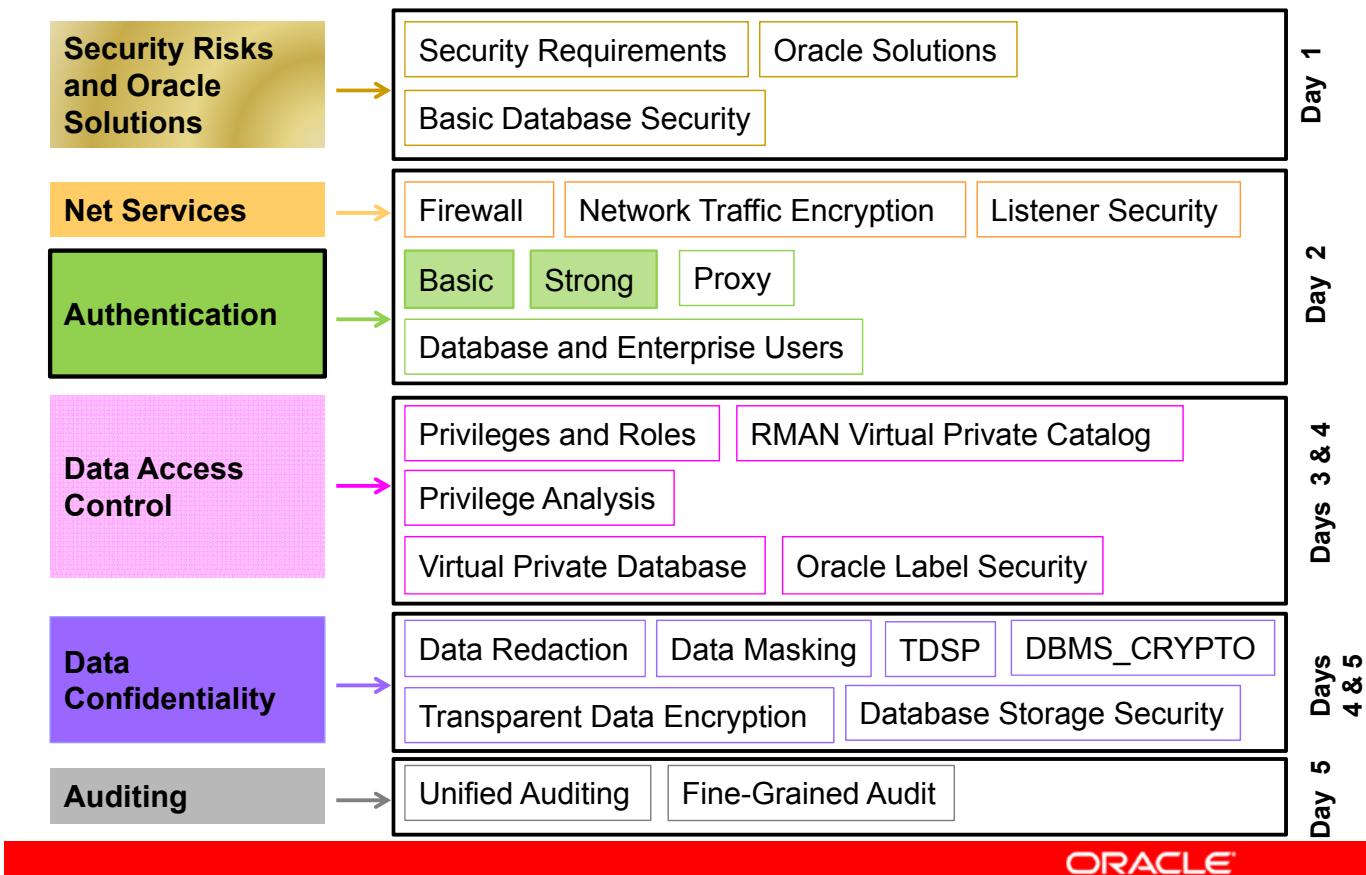
Using Basic and Strong User Authentication



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Schedule



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe different authentication methods
- Authenticate users with passwords
- Protect passwords in a secure external password store
- Authenticate users with the operating system (OS)
- Describe strong authentication with certificates or Kerberos or RADIUS
- Describe a setup for strong authentication
- Protect database link passwords

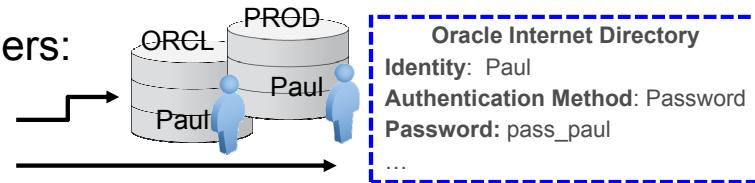


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

User Authentication

Identify two types of users:

- Database users
- Enterprise users



Identify the user in the following ways:

- Basic authentication
 - Database user identified by a password
 - Database user identified by the operating system
- Strong authentication
 - Certificates
 - Kerberos
 - RADIUS
- Proxy authentication



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Types of Users

Database users are managed by the database where they have been created.

Enterprise User Security (EUS) is a security feature of Oracle Database.

- Enabling database user accounts to be managed centrally in an LDAP directory instead of in individual databases
- Allowing end users to authenticate their usernames and authentication methods (passwords, SSL certificate, Kerberos) to a database
- Managing authorizations for access to the database through enterprise roles

EUS simplifies management of accounts and roles and improves security by reducing the requirement to have shared passwords.

User Authentication Methods

A basic security requirement is that you must know your users. You must identify them before you can determine their privileges and access rights, so that you can audit their actions on the data.

Users can be authenticated in a number of different ways before they are allowed to create a database session.

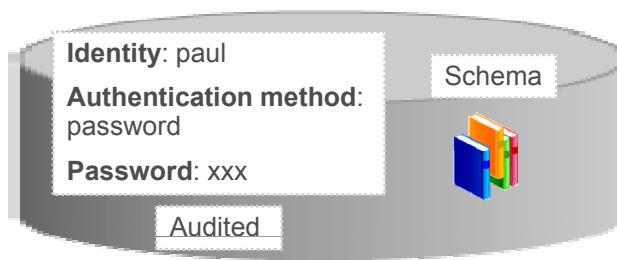
- **Basic authentication:**
 - With database authentication, you define users such that the database performs the authentication of users.
 - With operating system (OS) authentication, you define users such that authentication is performed by the OS or an OS-based network service.
- **Strong authentication:** You can define users such that they are authenticated by strong authentication methods: certificates, smart cards, and Kerberos.
- **Proxy users:** You can specify users who are allowed to connect through a middle-tier server. The middle-tier server authenticates and assumes the identity of the user and is allowed to enable specific roles for the user. This is called proxy authentication.

Basic User Authentication by Password

A database user:



`CONNECT paul/xxx`

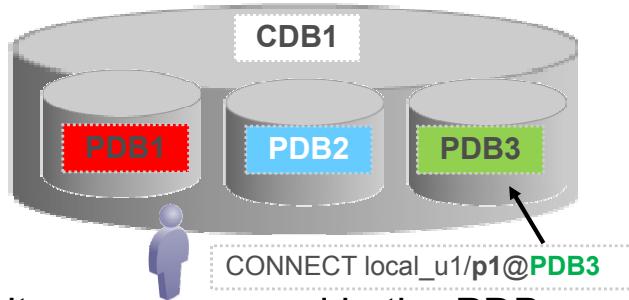


```
SQL> CREATE USER username IDENTIFIED BY password;
```

- A common user connects with the same password in all containers of a CDB:



`CONNECT c##u1/xxx@PDB1`
`CONNECT c##u1/xxx@PDB2`



- A local user connects with its own password in the PDB.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

An IDENTIFIED BY PASSWORD user is always authenticated by the database. This user has a schema. The schema may be empty but it does exist. Because the identity of the user is authenticated by the database, the user identity is available for audit records. The authentication is based on a password. The advantages are clear: simplicity, local authentication, and auditing, and the DBA has complete control.

However, some of the advantages can also be viewed as disadvantages. Each database user is authenticated in each database. This can result in a user having many different passwords. Each database can have its own password policy. Conflicting policies often lead to multiple passwords per user. When there is a change in user status, the user accounts must be updated in all databases where the user accounts exist. Each database user has a schema and, by implication, can create certain objects, even when most end users in a production database have no need for a schema.

When applications create a single local database user with password identification that is used to establish a connection pool, end users can have lightweight sessions that increase performance, but the application is then responsible for authenticating the end user. When multiple databases can be accessed through database links, the database with the weakest password protection could become a point of entry to all.

Best practice tip: Local database authentication should include a password profile that enforces password policies and users should periodically be required to justify the existence of their accounts.

Common Users in Container Databases

A common user is a user that has the same username and authentication credentials across multiple PDBs, unlike the local user, which exists in one and only one PDB.

A common user cannot have the same name as any local user across all of the PDBs. A common user can be created in the root container, and only in the root container. A common user is a database user that has the same identity and password in the root and in every existing and future PDB. Names of the common users are either system defined or begin with C## if user defined.

Local Users in Container Databases

A local user exists in one and only one pluggable databases (PDBs). Even if multiple PDBs have local users with the same name and credentials, each local user is distinct. A local user is identified and authenticated in one PDB only.

Protecting Passwords

- A password must never be embedded in files.
- Beware that passwords entered on the command line are visible when executing the `ps` command.
- Beware that passwords placed in an environment variable are visible when executing the `ps` command.

Solutions:

- Use Secure External Password Store.
- Use another authentication method: externally authenticated users for server scripts and tools.



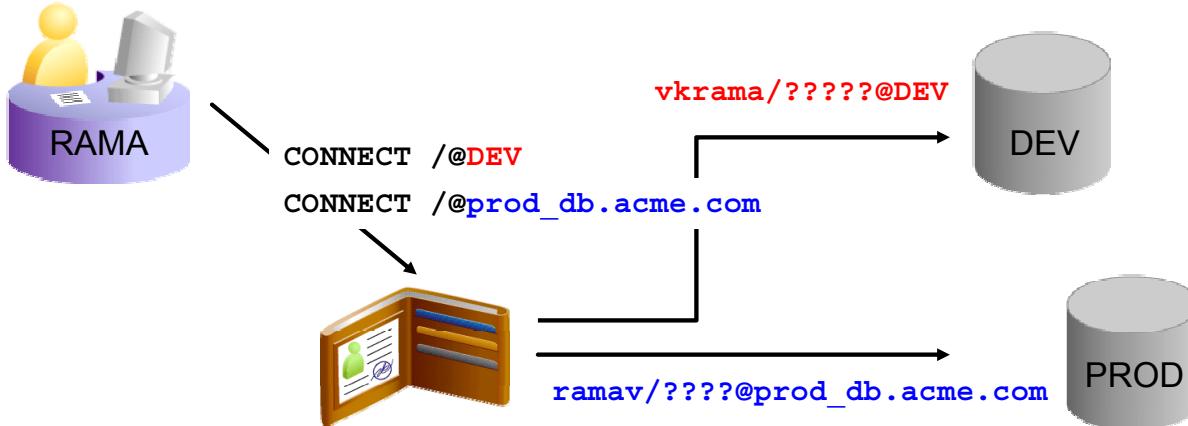
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Passwords used in scripts, job schedulers, and command lines are vulnerable. A password must never be placed in script files in clear text. Even passwords embedded in executable files are vulnerable to the `strings` command or a binary file editor.

Entering the password on the command line may seem safe if no one sees you type it. However, on some UNIX systems, it is possible to view the full command line, including the username and password, by using the `ps` command with the appropriate switches. If an OS account is compromised, someone viewing the shell command history file may find passwords to privileged database accounts. If a script prompts for a password to be input, this is not visible with the `ps` command.

Using environment variables to hold a password is also visible with the `ps` command and the `show environment` switch—for example, `ps -ef e` shows a full listing of every process followed by the environment variables.

Secure External Password Store



Use the `mkstore` command to:

- Create a wallet
- Add, modify, list, or delete database login credentials in an existing client wallet

```
$ mkstore -wrl $HOME/admin/orcl/wallets
      -createCredential DEV vkrama [pass1]
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

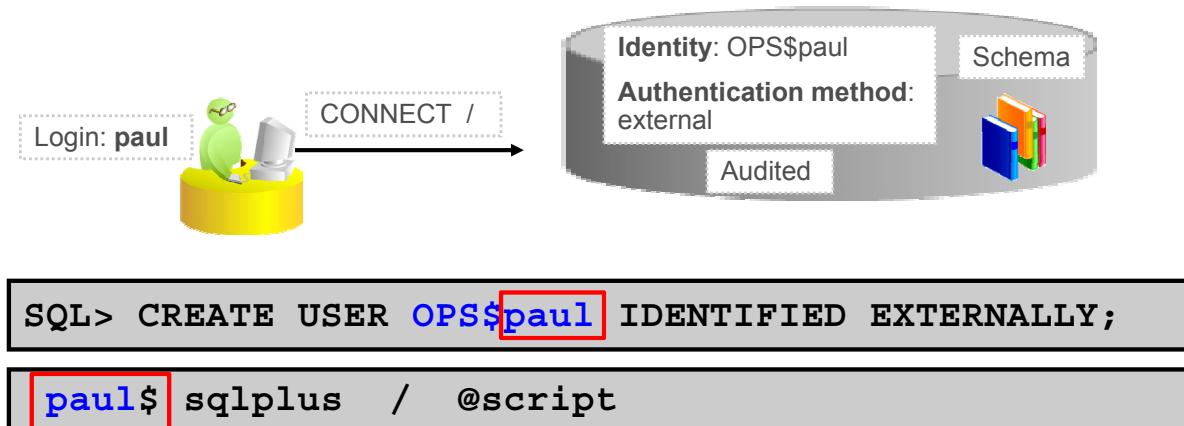
The password store is an Oracle wallet created to provide a secure method of storing user login passwords. By default, this is an auto login wallet, so a password is not needed to open the wallet. The OS authenticates the user, and file system permissions control access to the wallet. The credentials to access the database are extracted from the wallet for the user. In the example, the OS user, Rama, owns a wallet that holds different passwords for each database. With this feature, Rama may connect with `CONNECT /@DEV` or `CONNECT /@prod_db.acme.com`. The username and password are stored in the wallet and retrieved based on the connect string. The connect string can be a `tnsnames.ora` file entry as shown for `DEV`, a hostname to be resolved with DNS, or a `hostname:port:sid` string such as `dlsun1:1521:dev`.

This feature provides a secure way to run applications or command files without embedding the password in OS files. An application server can use this method to connect and establish the connection pool without requiring a password in the application.

Note: The wallet for the external password store is managed with the `mkstore` utility. The PKI Wallet managed by Oracle Wallet Manager is intended for use by the database. If the user that owns the database wallet wants to also use the external password store, the `mkstore` utility must be used to configure the external password store. The external password store uses a separate section of the database wallet.

Basic User Authentication by Operating System

- Use externally authenticated users for server scripts and tools:



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The externally authenticated database user account is a better way to protect the password. Suppose Paul is a database administrator (DBA) and needs to run batch jobs on the server machine. An OS user with a login of `paul` can now connect to the database with “`/`” and run a script without a password. This method depends on the OS accounts being secure.

The `IDENTIFIED EXTERNALLY` user has all the same advantages and disadvantages of the `IDENTIFIED BY PASSWORD` user, with the major difference that the authentication is performed by the operating system. OS authentication is not supported with PDBs.

An `IDENTIFIED EXTERNALLY` user can connect to the database with a “`/`”. This type of connection, by default, requires that the user have an account on the machine where the database resides. The externally identified user account is used primarily for scripting and batch programming, so passwords would not be embedded in files.

The `OS_AUTHENT_PREFIX` initialization parameter determines the relationship between the OS account name and the database account name. The default value for this parameter is `OPS$`. Users connecting as an OS-authenticated user do not supply a password to log in; “`/`” is sufficient. Unless the `REMOTE_OS_AUTHENT` initialization parameter is also set the same, users would supply a password on remote connections—that is, using `@service_name`.

The `REMOTE_OS_AUTHENT` initialization parameter allows the externally identified user to be authenticated remotely. The default value is `FALSE`. This parameter was deprecated in Oracle Database 11g and should not be used.

Quiz

A user that is authenticated externally can execute a SQL*Plus script as follows: `sqlplus / @<script_name>`

- a. True
- b. False



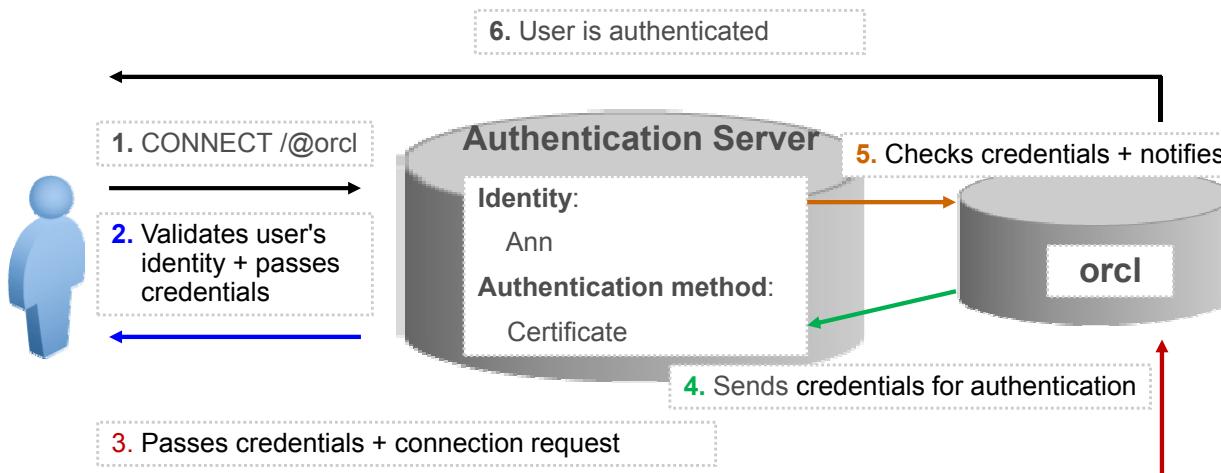
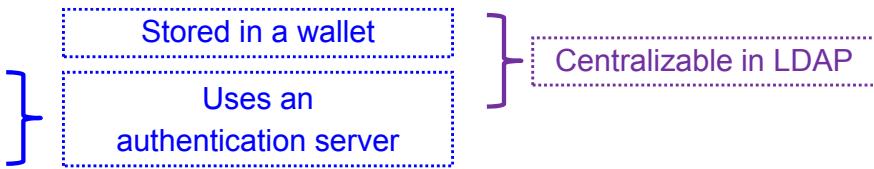
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a

Strong User Authentication

Identify the user strongly in the following ways:

- Certificates
- Kerberos
- RADIUS



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

User authentication is a basic security requirement. The user's identity must be confirmed before granting privileges and access rights. Auditing of user actions on data is essential, and user authentication is the first step to having a valid audit trail of user actions on the data.

Authentication methods can be classified as:

- Something you know (password)
- Something you are (biometric)
- Something you have (smart card)

Strong user authentication is a way of confirming the identity of the user with something other than a password or the OS. Smart cards, biometrics, certificates, and Kerberos tokens provide strong user authentication. Some of these methods are known as two-factor or multifactor authentication. Two-factor authentication requires something that the user knows plus something that the user has. Several of the strong authentication options use centralized authentication, which gives you high confidence in the identity of users, clients, and servers in distributed environments. These centralized servers may also include single sign-on.

Strong User Authentication

Strong authentication:

- Is stronger than password authentication
- Often includes the single sign-on functionality
- Is supported by the following authentication technologies:
 - Certificates, public key infrastructure (PKI)
 - RADIUS, token, and smart cards
 - Kerberos
- Integrates with Oracle Net Services



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Certificates

Certificates are digital documents that are used to provide proof of identity. Certificates can use PKI or a Web of trust to authenticate a user or a machine. They are used in one-way authentication to authenticate a server to a user and in two-way authentication that additionally authenticates the user to the server. The process for exchanging certificates is based on TCP for the secure sockets layer (SSL)—enabled protocol called TCPS. SSL has been superseded by Transport Layer Security (TLS), but is still referred to as SSL. The certificates require the use of SSL libraries.

Authentication Adapters

Support for authentication services is provided through authentication adapters, which are very much like the existing Oracle Net protocol adapters. Authentication adapters integrate below the Oracle Net interface so that existing applications can take advantage of new authentication systems transparently, without any changes to the application.

RADIUS Authentication

Remote Authentication Dial-In User Service (RADIUS) is a flexible, lightweight protocol that provides authentication, authorization, accounting, and centralized user information.

RADIUS provides two major benefits:

- It enables support for authentication technologies, including token cards, smart cards, and challenge-response.
- It readily integrates into existing systems by making the Oracle database a RADIUS client, thus capitalizing on the infrastructure and investment that organizations have already made.

Supported RADIUS Services

With RADIUS, you can choose virtually any mechanism available to authenticate network users. Many token and smart card manufacturers support RADIUS. Any RADIUS-compliant device can authenticate database users with little modification required by the authentication provider. Because many organizations have implemented RADIUS for remote access to their networks, the Oracle server easily integrates into existing systems and takes advantage of the investments that an organization has already made.

Kerberos Adapters

Kerberos is a trusted third-party authentication system that relies on shared secrets. It assumes that the third party is secure. It provides the following:

- Single sign-on capabilities
- Centralized password storage
- Database-link authentication
- Enhanced PC security

Two adapters for the Kerberos authentication service are provided:

- Kerberos Authentication Adapter (essentially a Kerberos client). The Kerberos server can be obtained from third-party vendors. The Massachusetts Institute of Technology Key Distribution Center (MIT KDC) is an open source.
- Microsoft provides a slightly different version of KDC that works with Active Directory on the Windows servers. The Windows and UNIX versions of Kerberos are interoperable.

Note: As of the release of Oracle Database 12c, strong authentication services (Kerberos, PKI, and RADIUS) are no longer part of Oracle Advanced Security and are available in all licensed editions of the Oracle database.

Public Key Infrastructure (PKI) Tools

The Oracle Database PKI implementation includes:

- Components:
 - Oracle wallet
 - Oracle Identity Management Infrastructure
- Management tools:
 - Oracle Wallet Manager



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The authentication systems based on the Public-Key Cryptography Standards (PKCS) issue digital certificates to users. The certificates are used to authenticate users directly to servers in the enterprise without the direct involvement of an authentication server. Oracle Database 12c provides PKI tools for using public keys and certificates. Some of its features and tools are as follows:

- Oracle wallet is a data structure used to store and manage security credentials for an individual entity. These credentials can be passwords or certificates.
- Oracle Wallet Manager is an application that is used to manage and edit the security credentials in Oracle wallets. Oracle wallets are data structures that contain a user private key, a user certificate, and a set of trust points (the list of root certificates that the user trusts). Oracle Wallet Manager uses the Public-Key Cryptography Standards (PKCS) #12 specification for the wallet format, and PKCS #11 for certificate requests.
- Integration with Entrust PKI provides full-certificate life-cycle management and certificate revocation list (CRL) checking.
- An Oracle Database environment supports hardware devices by providing APIs that conform to the RSA Security, Inc., PKCS #11 specification.

Certificates

- Certificates:
 - Are digital documents
 - Provide proof of identity
 - Are stored in Oracle Wallets
- Certificate authority:
 - Is a trusted organization (trust point)
 - Attests the identity of the certificate
 - Issues trusted certificates X.509 v3
- Certificate use:
 - Requires a secure sockets layer (SSL)
 - Requires a level of trust in the signing authority



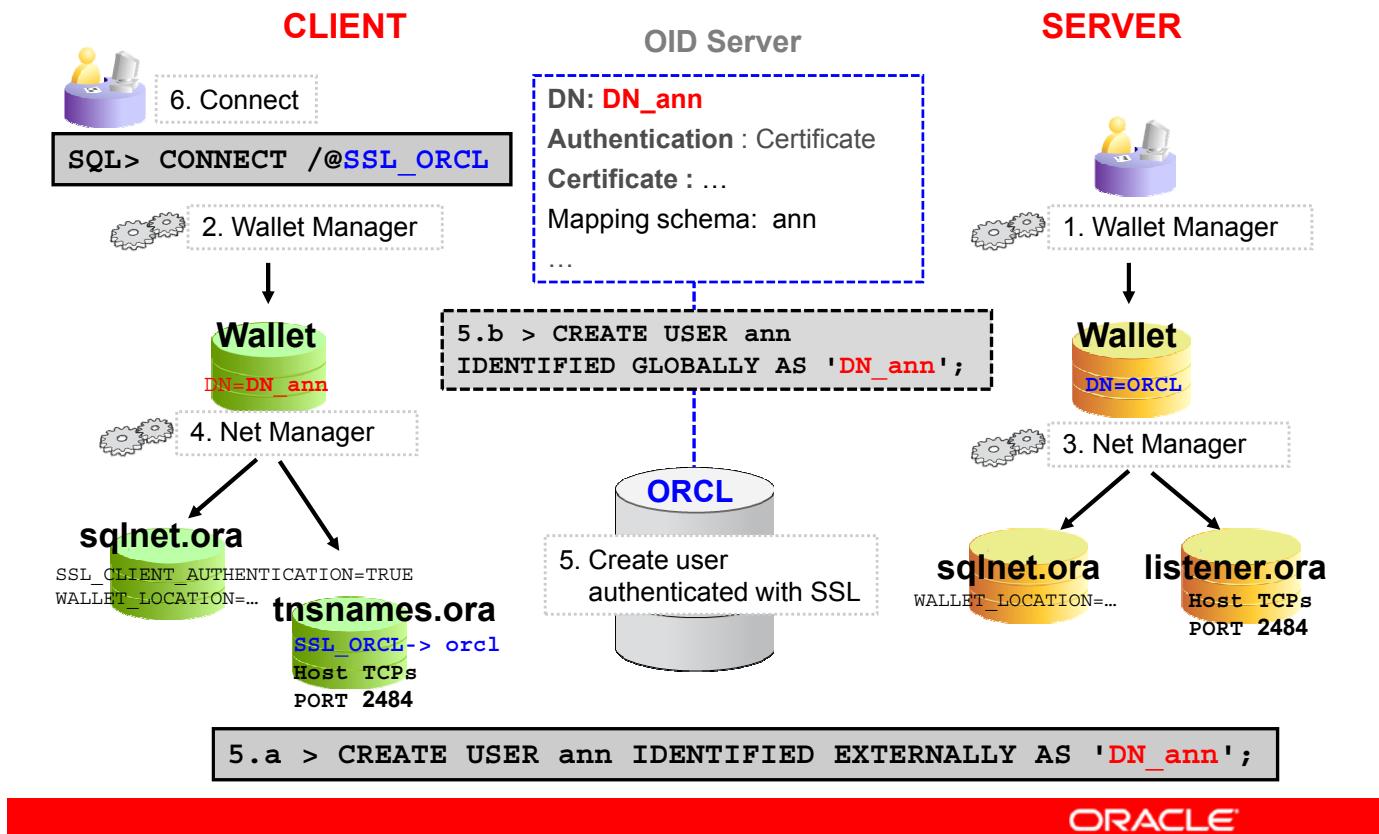
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Certificates are digital documents that are used to provide proof of identity. For example, a certificate from a bank's URL convinces bank customers that they have connected to the bank and not a spoofed site that is designed to collect their account numbers. When used in a PKI, each certificate includes an issuing certificate authority (CA). The CA asserts that it is a valid certificate.

The CA is also known as a trust point and is hierarchical. If George connects to a website that presents a certificate signed by a CA that George does not recognize, he can check the signer of the CA certificate and so on. The certificate contains identity credentials: name, public key, expiration time, and a URL of a signing authority. Each signing authority provides a certificate of another signing authority at a higher level. This chain of certificates repeats to a self-signed root certificate provided by one of the root certificate authorities such as Verisign. This chain of trust is the PKI.

For SSL sessions, certificates are exchanged, and the public keys are used to pass a shared key. Then the much faster symmetric encryption algorithms are used for the session messages. Certificates may be used between a client and a server without any centralized server. Note that only the client and the server are involved in the authentication after the certificates have been issued.

How to Use Certificates for Authentication



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

If you plan to centralize users' authentication, you can install the required components. Oracle Identity Management, although not required for SSL, provides Oracle Internet Directory (OID) and directory management tools.

1. Configure the server for SSL: Configure a wallet for the node using Oracle Wallet Manager.

- To obtain a certificate for the server database instance, you must generate a certificate request and send the request to a CA. You must import the certificates that you receive into the wallet. Oracle Wallet Manager is used to create the wallet, generate the request, and import the certificates into the wallet. The user certificate refers to a certificate that matches the certificate request—in this case, a database. The certificates returned by the CA are the trusted certificate and the user certificate. The trusted certificate must be imported first. It is important that the distinguished name (DN) on the user certificate be the same as the database service name.
- The wallet can be stored in the system's default location, in a specified directory, or in the Windows registry. The wallet has a password. The Oracle Wallet Manager auto login feature creates an obfuscated copy of the wallet. File system permissions provide the necessary security for auto login wallets. When auto login is enabled for a wallet, it is available only to the operating system user who created that wallet. You must enable auto login if you want single sign-on access to multiple Oracle databases, which is disabled by default. Sometimes, these wallets are called "SSO wallets" because they provide the single sign-on capability.

- 2. Configure the server-side Oracle Net files:** Use Net Manager to configure the `sqlnet.ora` and `listener.ora` files to use SSL.

- Configure `sqlnet.ora` as the following:

```
WALLET_LOCATION = (SOURCE= (METHOD=File)
(METHOD_DATA= (DIRECTORY=<wallet_location>)))
```

- In the `listener.ora` file, set the listener to listen on an end point for the TCPS protocol. The recommended port is 2484:

```
LISTENER =
(DESCRIPTION_LIST = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCPS)
(HOST = <server_name>) (PORT = 2484))))
```

- 3. Configure the client for SSL:** Use Oracle Wallet Manager to configure a wallet for the user. This process is almost identical to configuring SSL on the server. On the server, the certificate request is for the server machine or database. On the client, the certificate request is for a user and is requested with the DN of that user. The DN of the user is used to query the directory to obtain the user authorizations.

- 4. Configure the client-side Oracle Net files:** Use Net Manager to configure the `sqlnet.ora` and `tnsnames.ora` files to use SSL.

- In the `sqlnet.ora` file, specify the wallet location, set `SSL_CLIENT_AUTHENTICATION` to TRUE to require the client to use SSL authentication, and set `SSL_SERVER_DN_MATCH` to ON to require the server service name to match the distinguished name.

```
wallet_location =
(SOURCE= (METHOD=File) (METHOD_DATA= (DIRECTORY=<wallet_location>)))
SSL_CLIENT_AUTHENTICATION =TRUE
SSL_SERVER_DN_MATCH=(ON)
```

`SSL_SERVER_DN_MATCH=ON` requires that the DN of the server matches the service name listed in the `tnsnames.ora` entry.

- Set an entry in the `tnsnames.ora` file to reference the port for the TCPS protocol:

```
SSL_ORCL =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCPS) (HOST = <server_name>) (PORT = 2484)))
(CONNECT_DATA = (SERVICE_NAME = orcl.us.oracle.com)))
```

- 5. Create an external user that is authenticated with a certificate:** Use the distinguished name (DN). The DN is the distinguished name in the user's PKI certificate in the user's wallet. The user's authentication can be stored in an LDAP and this is the example described in 5.b. Centralizing the user's authentication is covered in the lesson titled "Using Global User Authentication."

- 6. Connect to the database:** The user that owns the client wallet connects to the database without a password. The connect string, `SSL_ORCL` in the example that named the SSL listener port and the distinguished name (DN) of the server, authenticates a user by using only the certificate. The Oracle Database server may check the directory by using the user-distinguished name (DN) for the user authorizations if the user has been created as a global user.

- The wallet is accessed and the certificate information is passed to the server.
- The server certificate information is passed to the client as encrypted text.

Quiz

In which of the following do you store certificates?

- a. listener.ora file
- b. sqlnet.ora file
- c. Oracle Internet Directory
- d. Oracle wallet



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: c, d

How to Use Kerberos for Authentication

1. Install Kerberos.
2. Configure a service principal for the Oracle Database server.
3. Extract a service key table from Kerberos.
4. Install an Oracle Database server and a client.
5. Install Oracle components.
6. Configure Oracle Net Services and Oracle Database.
7. Configure Kerberos authentication.
8. Create a Kerberos user.
9. Create an externally authenticated Oracle user.

```
SQL> CREATE USER ann IDENTIFIED EXTERNALLY;
```

10. Get an initial ticket for the Kerberos and Oracle user.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To use Kerberos authentication, you must perform the following steps: (Additional details are available in *Oracle Database Security Guide 12c Release 1*.)

1. **Install Kerberos:** Install it on the machine that functions as the authentication server.
2. **Configure a service principal for an Oracle Database server:** Create a name in Kerberos to identify the Oracle database. This name has the following format:

kservice/kinstance@REALM

where:

- `kservice` is a case-sensitive string. It may be the same as the database service name.
- `kinstance` is typically the fully qualified name of the system on which the Oracle database is running.
- `REALM` is the domain name of the database server. `REALM` must always be in uppercase and is typically the DNS domain name.

Example: orcl/testp1.us.acme.com@US.ACME.COM

3. **Extract a service key table from Kerberos:** In Kerberos authentication, a service key table is a list of service principals that exist on `kinstance`. This information must be extracted from Kerberos and copied to the database server before Kerberos can be used by the Oracle instance.

4. **Install an Oracle Database server and an Oracle client.**

5. **Install Oracle components:** The components are Oracle Net Services.

6. **Configure Oracle Net Services and Oracle Database.**

7. **Configure Kerberos authentication:** Use Oracle Net Manager to configure the Kerberos authentication service parameters on the client and on the database server. Set the Kerberos method for authentication and provide the Kerberos service name. The `sqlnet.ora` file parameters to define Kerberos authentication are as follows:

```
SQLNET.AUTHENTICATION_SERVICES=(KERBEROS5)
SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=kservice
```

8. **Create a Kerberos user:** To create Oracle users that Kerberos can authenticate, perform this task on the Kerberos authentication server where the administration tools are installed. The realm must already exist. On UNIX, run `/krb5/admin/kadmin.local` as root to create a new Kerberos user, such as `krbuser`:

```
# ./kadmin.local
kadmin.local: addprinc krbuser
Enter password for principal: "krbuser@SOMEKO.COM":
(password does not display)
Re-enter password for principal: "krbuser@SOMEKO.COM":
(password does not display)
kadmin.local: exit
```

9. **Create an externally authenticated Oracle user.**

```
SQL> CONNECT / AS SYSDBA;
SQL> CREATE USER "KRBUSER@SOMEKO.COM" IDENTIFIED
EXTERNALLY;
SQL> GRANT CREATE SESSION TO "KRBUSER@SOMEKO.COM";
```

10. **Get an initial ticket for the Kerberos and Oracle user:** In Kerberos authentication, an initial ticket or ticket granting ticket (TGT) identifies the user as having the right to ask for additional service tickets. No tickets can be obtained without an initial ticket. An initial ticket is retrieved by running the `okinit` program and providing a password. Before you can connect to the database, you must ask the Key Distribution Center (KDC) for an initial ticket. To do so, run the following on the client:

```
$ okinit username
```

At this point, the `krbuser` user can connect to any database with a `kservice` name and `krbuser` defined with the string:

```
$ sqlplus /@orcl
```

How to Use KDC with Windows 2000 for Authentication

1. Configure an Oracle Kerberos client to interoperate with a Windows 2000 domain controller KDC.
2. Configure a Windows 2000 domain controller KDC to interoperate with an Oracle client.
3. Configure an Oracle database to interoperate with a Windows 2000 domain controller KDC.
4. Get an initial ticket for the Kerberos and Oracle user.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Windows 2000 Server includes a version of Kerberos called KDC that configures Kerberos to operate with Windows 2000 domain controllers. This is a high-level overview of the configuration process. For details, see the *Oracle Database Security Guide*.

1. **Configure the Oracle Kerberos client to interoperate with a Windows 2000 domain controller KDC.**

On the computers that use the Oracle client, perform the following steps:

- a. Create the Kerberos client configuration files: `krb.conf`, `krb5.conf`, and `krb5.realms`.
- b. Specify the Oracle Net configuration parameters in the `sqlnet.ora` file. These may be specified with Net Manager.
- c. For UNIX clients, specify the listening port number. Set the port number in the `/etc/services` file to UDP/TCP port 88.

2. Configure a Windows 2000 domain controller KDC to interoperate with an Oracle client.

Perform the following steps on the Windows 2000 domain controller:

- a. Create a new user for the Oracle client in Microsoft Active Directory. This is the name of the Kerberos user that connects to the database.
- b. Create a new user for the Oracle database in Microsoft Active Directory. For example, if the Oracle database runs on the sales3854.us.acme.com host, use Active Directory to create a user with the username sales3854.us.acme.com and the password oracle.
- c. Use the Ktpass command-line utility to extract the keytab file.
- d. Copy the extracted keytab file to the host computer where the Oracle database is installed.

3. Configure an Oracle database to interoperate with a Windows 2000 domain controller KDC.

On the computer hosting the Oracle database, perform the following steps:

- a. Set the sqlnet.ora parameters.
- b. Create an externally identified user. Ensure that the username is created in all uppercase characters (for example, ORAKRB@SALES.US.ACME.COM).

4. Get an initial ticket for the Kerberos and Oracle user.

In Kerberos authentication, an initial ticket or TGT identifies the user as having the right to ask for additional service tickets. No tickets can be obtained without an initial ticket.

With Active Directory, do not use `okinit`. Instead, use the Windows in-memory credential cache by setting `SQLNET.KERBEROS5_CC_NAME=OSMSFT://`.

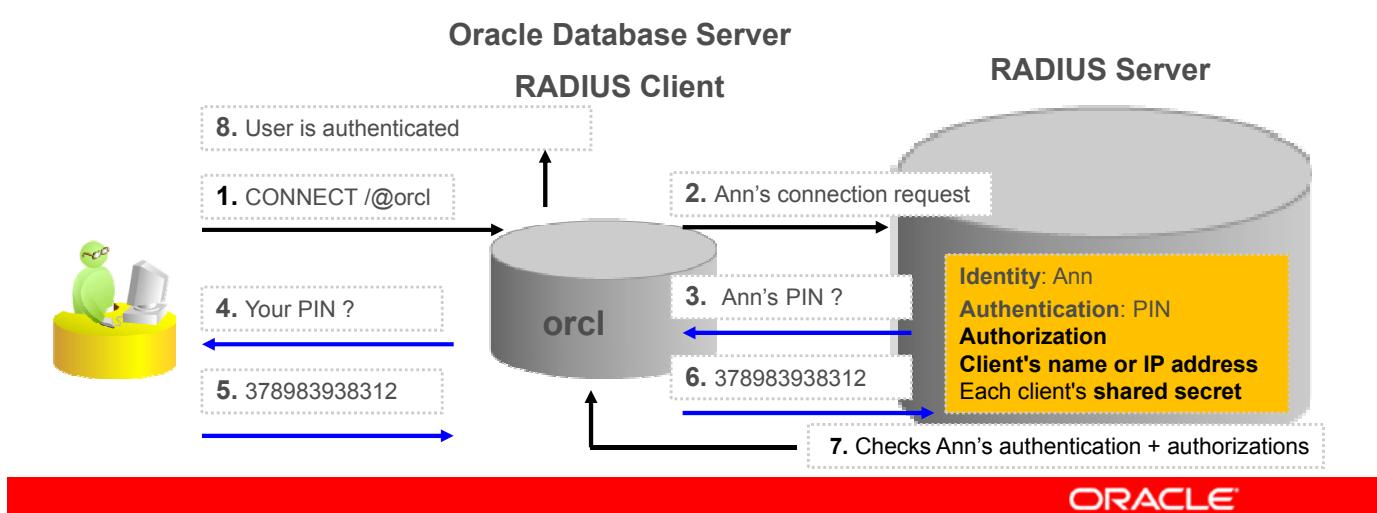
At this point, the Kerberos user can connect to the database configured in step 3 with the following string:

```
$ sqlplus /@orcl
```

RADIUS Authentication: Overview

- RADIUS is a protocol for remote authentication and access.
- The user is an externally authenticated Oracle user.

```
SQL> CREATE USER ann IDENTIFIED EXTERNALLY;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

RADIUS is a client/server security protocol that is widely used to enable remote authentication and access. Oracle Security uses this industry standard in a client/server network environment. You can enable the network to use any authentication method that supports the RADIUS standard, including token cards and smart cards, by installing and configuring the RADIUS protocol. For details about enabling RADIUS authentication, see the *Oracle Database Security Guide 12c Release 1*.

From the user's perspective, the entire authentication process is transparent:

1. The user seeks access to an Oracle database server.
2. The Oracle database server (acting as the RADIUS client) notifies the RADIUS server.
3. The RADIUS server looks up the user's security information sending a challenge, such as a random number authentication data or a PIN code to the Oracle database server.
4. The Oracle Database server passes it to the Oracle client.
5. The user sends a response to the challenge to the Oracle Database server.
6. The Oracle Database server passes the answer to the RADIUS server.
7. The RADIUS server passes authentication and authorization information to the Oracle database server.
8. The Oracle database server grants the user access to the Oracle database server.

Fixed User Database Links

Types of database links

Public fixed user database link

Connected user database link

Current user database link

Shared database link

Create a public fixed user database link:

```
SQL> CREATE PUBLIC DATABASE LINK test
      CONNECT TO scott IDENTIFIED BY tiger
      USING 'FINANCE';
```

Use a public database link:

```
SQL> SELECT * FROM emp@TEST;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To create a fixed user database link, you embed the credentials (in this case, a username and password) required to access the remote database in the definition of the link. The credentials are stored in the data dictionary in an encrypted form.

The following statement creates a public link to the remote SUPPLY database by using the global database name. The link connects to the remote database with the user ID and password as SCOTT and tiger, respectively:

```
CREATE PUBLIC DATABASE LINK supply.us.acme.com
      CONNECT TO scott IDENTIFIED BY tiger;
```

The following statement creates a private fixed user link called JANE_FINANCE to the database with the service name, FINANCE. The link connects to the remote database with the user ID and password as JANE and doe, respectively:

```
CREATE DATABASE LINK jane_finance
      CONNECT TO jane IDENTIFIED BY doe USING 'finance';
```

When an application uses a fixed user database link, the local server always establishes a connection to the specified schema in the remote database. The local server also sends the fixed user's credentials across the network when an application uses the link to access the remote database.

Anyone who is granted access to a fixed database link has access to everything in the remote schema.

Database Links Without Credentials

To avoid storing passwords in the data dictionary:

- Create a connected user database link:

```
SQL> CREATE DATABASE LINK sales.div.com USING 'sales';
```

- Create a current user database link:

```
SQL> CREATE DATABASE LINK sales
      CONNECT TO CURRENT_USER USING 'sales';
```

To limit the number of network connections between the local server and the remote server, create a shared database link:

```
SQL> CREATE SHARED DATABASE LINK link2sales
      CONNECT TO scott IDENTIFIED BY tiger
      AUTHENTICATED BY linkuser IDENTIFIED BY p1
      USING 'sales';
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Users can connect to remote databases without having to embed a password in the database link definition. The connected user and current user database links do not include credentials in the definition of the link. The credentials that are used to connect to the remote database change depending on the user that references the database link and the operation performed by the application.

Creating a Connected User Database Link

To create a connected user database link, omit the CONNECT TO clause. The syntax in the slide creates a connected user database link, where sales.div.com is the name of the link and USING 'sales' is an optional connect string. This type of database link requires that the user on the local database have an account on the remote database with the same username and password. The local instance passes the credentials of the current user to the remote instance.

Creating a Current User Database Link

To create a current user database link, use the CONNECT TO CURRENT_USER clause in the link creation statement. The syntax in the slide creates a current user database link, where sales is the name of the link and USING 'sales' is an optional connect string. To use a current user database link, the current user must be a global user registered with a Lightweight Directory Access Protocol (LDAP) directory service and be authorized to connect to both the databases that are involved in the link.

Shared Database Link

Every application that references a remote server using a non-shared database link establishes a connection between the local and remote databases. Many users running applications simultaneously can cause a high number of connections between the local and remote databases. Shared database links enable you to limit the number of network connections required between the local and remote servers.

Use shared database links when the number of users accessing a database link is expected to be much larger than the number of server processes in the local database.

Whenever you use the keyword SHARED, the clause AUTHENTICATED BY is required. The schema specified in the AUTHENTICATED BY clause must exist in the remote database and must be granted at least the CREATE SESSION privilege. The credentials of this schema can be considered the authentication method between the local database and remote databases. These credentials are required to protect the remote shared server processes from clients that masquerade as a database link user and attempt to gain unauthorized access to information. After a connection is made with a shared database link, operations on the remote database proceed with the privileges of the CONNECT TO user or CURRENT_USER, not the AUTHENTICATED BY schema.

The example in the slide creates a fixed user, shared link to database sales, connecting as scott and authenticated as linkuser.

Controlling Access to Remote Databases

In many distributed applications, you do not want a user to have privileges in a remote database. One simple way to achieve this result is to embed a fixed user or current user database link within a procedure. This procedure performs the needed SQL operation with definer's rights. Thus, the user accessing the procedure temporarily assumes the privileges of the user defined in the link. The fixed link connects to the remote database as the user defined in the link. The current user link connects as the owner of the procedure.

In this definer's rights example, the HR user creates a database link and a procedure, and then grants EXECUTE on the procedure to a role: HR_CLERK. Anyone with the role can execute the update_hr_salary procedure, but cannot access the remote HR schema through the database link except through the procedure.

```
CONNECT HR
PASSWORD : *****
CREATE DATABASE LINK hremp
CONNECT TO hr IDENTIFIED BY oracle_1 USING 'p0orcl';
CREATE OR REPLACE PROCEDURE update_hr_salary
    (p_employee_id NUMBER, p_salary NUMBER)
IS
BEGIN
    UPDATE employees@hremp SET salary = p_salary
    WHERE employee_id = p_employee_id;
END;
GRANT execute on update_hr_salary to HR_CLERK;
```

See the dblink_proc.sql demonstration script in the /home/oracle/labs/demos directory.

Database Links and Changing Passwords

Good practice requires:

- Passwords to change:
 - Cascade password changes to database links.
 - Current user links: User credentials are in the directory.
 - Connected user links: Find links to be changed by username.
 - Fixed links: Find links to be changed by username.
- Consistent naming conventions for databases and links in a networked environment:
 - GLOBAL_NAMES instance parameter set to TRUE specifies that a database link is required to have the same name as the database to which it connects.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Good practice demands that all passwords be changed periodically. There is no automatic way to find the database links on remote databases that depend on the password on a local database. Password changes on one machine can cause the database links on multiple machines to fail.

For ease of maintenance, current user links are the best. The user credentials are kept in a directory. The password is changed in only one location.

For connected user links, the administrative practice of keeping all passwords the same on all databases mitigates the troubleshooting that is required when passwords are not synchronized. This practice depends on users remembering to drop and create all links on all machines that depend on the password when the password must be changed on the links.

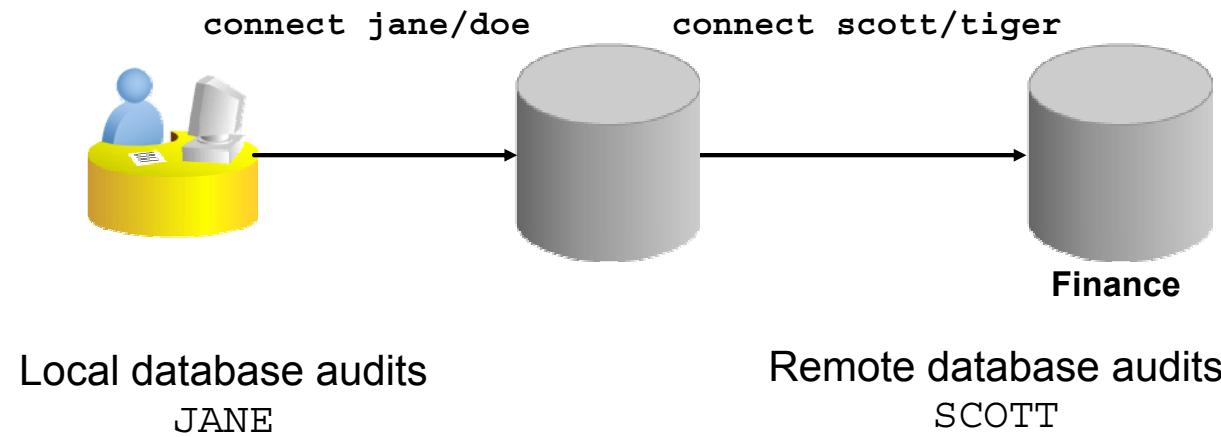
For fixed links, keep a record of all database links by auditing the CREATE and DROP LINK commands, and audit ALTER USER to detect password changes.

```
SELECT owner, db_link, username, host FROM DBA_DB_LINKS  
WHERE username not in ('SYS', 'SYSTEM', 'PUBLIC');
```

This query reports the user that is named in the CONNECT TO clause, and the host where the user account exists.

If you use or plan to use distributed processing, Oracle recommends that you set the GLOBAL_NAMES parameter to true to ensure the use of consistent naming conventions for databases and links in a networked environment.

Auditing with Database Links



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You must always perform auditing operations locally. That is, if a user acts in a local database and accesses a remote database through a database link, the local actions are audited in the local database, and the remote actions are audited in the remote database, provided that appropriate audit options are set in the respective databases.

The remote database cannot determine whether a successful connection request and subsequent SQL statements come from another server or a locally connected client. For example, assume the following:

- A fixed user link named `hq.acme.com` connects the local user JANE to the remote HQ database as the remote user SCOTT.
- The SCOTT user is audited on the remote database.

Actions performed during the remote database session are audited as though SCOTT were connected locally to HQ and performing the same actions there. You must set audit options in the remote database to capture the actions of the user (in this case, SCOTT on the HQ database) that is embedded in the link if the desired effect is to audit what Jane does in the remote database. You cannot set local auditing options on remote objects. Therefore, you cannot audit the use of a database link although access to remote objects can be audited on the remote database.

Note: You can audit the global username for global users. Global users are identified in the directory.

Restricting a Database Link with Views

Creating views to access data through a database link:

- Allows a private database link owned by the application owner
- Limits the access to the remote schema
- Limits changes to a database link to a view rebuild
- Simplifies auditing of database link access



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When a database link is exposed to an end user, that user has access to every table in the schema authorized by the database link. Auditing the access of the database link is not possible on the local machine.

A possible solution is to create a view in the schema of the application owner that uses the database link. The database link is owned by the application owner, and not exposed to other users. The application owner grants access only to the views. Typically, the application owner account would be locked in the production environment.

The solution of putting the database link behind a view provides an extra layer of security and also minimizes the dependency on the database link. If something about the remote database changes, the database can be changed in one place and the views that need to be rebuilt are limited to the schema of the application owner. In many databases, the policies require that passwords must be changed on a regular basis.

The view over a database link makes it impossible for the developers of the middle-tier application to reference the database link directly.

The grantees to the views get privileges only to the views, even if the link offers more. Access of the views can be audited easily.

Example:

In this example, the HR user creates a database link and a view, and then grants privileges on the view to the HR_MGR role. Anyone with the role can change the EMPLOYEES table but not any other table in the HR schema, nor can they access the remote HR schema through the database link except through the view.

```
CONNECT HR
PASSWORD: *****
CREATE DATABASE LINK hrviewlink
CONNECT TO hr IDENTIFIED BY oracle_1 USING 'p0orcl';

CREATE OR REPLACE VIEW employees_vw
AS SELECT * FROM employees@hrviewlink;

GRANT select, insert, update, delete on EMPLOYEES_VW to HR_MGR;
```

See the `dblink_view.sql` demonstration script in the `/home/oracle/labs/demos` directory.

Summary

In this lesson, you should have learned how to:

- Describe different authentication methods
- Authenticate users with passwords
- Protect passwords in a secure external password store
- Authenticate users with the operating system (OS)
- Describe strong authentication with certificates, Kerberos, or RADIUS
- Describe a setup for strong authentication
- Protect database link passwords



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- 6-1: Implementing an OS-authenticated user connecting locally without a password
- 6-2: Observing the stored password for a fixed database link
- 6-3: Restricting a database link with views
- 6-4: Configuring the external secure password store
- 6-5: Logging into a CDB or PDB as a local user or a common user



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To access the tutorial of practice 6-6, refer to the following Oracle documentation:

Oracle Database Security Guide 12c Release 1 (12.1)

Chapter 6 Managing Fine-Grained Access in PL/SQL Packages and Types

Configuring Access Control to an Oracle Wallet

Example: Access Control Configuring for Passwords in a Non-Shared Wallet

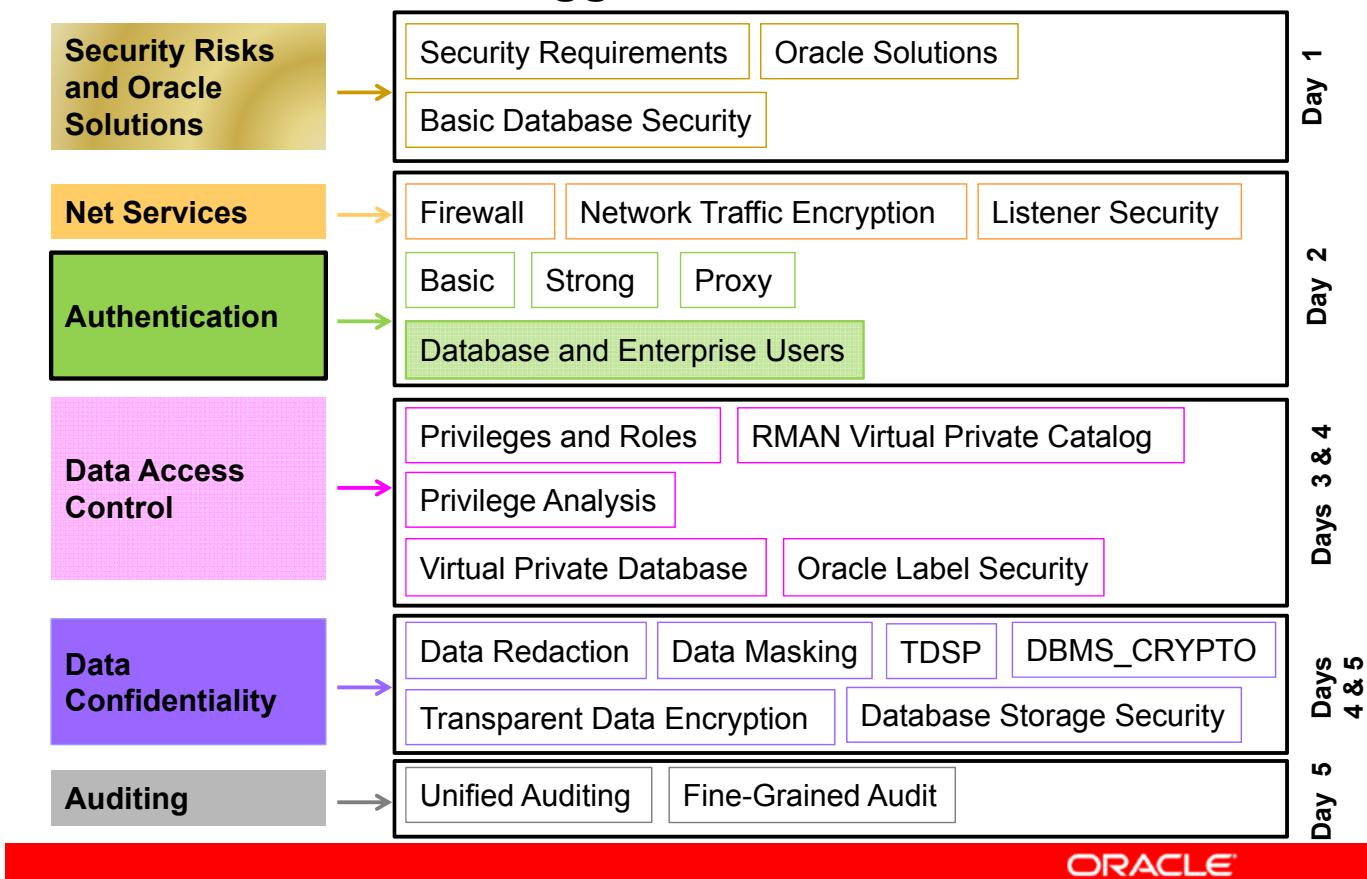
Using Global User Authentication



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Schedule



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe Enterprise User Security
- Set up Enterprise User Security
- Create an enterprise user
- Use global shared and private schemas
- Implement enterprise roles
- Audit enterprise users
- Migrate users from the database to Oracle Internet Directory (OID)



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This lesson describes the basic components of Enterprise User Security (EUS). It shows the architecture of this feature and describes the installation process. The components required to create and manage an enterprise user are discussed. In addition, the lesson explains the techniques for using the enterprise user in the context of the database server and integrating the enterprise user with familiar security policies and auditing.

More Information

For a detailed explanation of the topics covered in this lesson, refer to the following documentation guide:

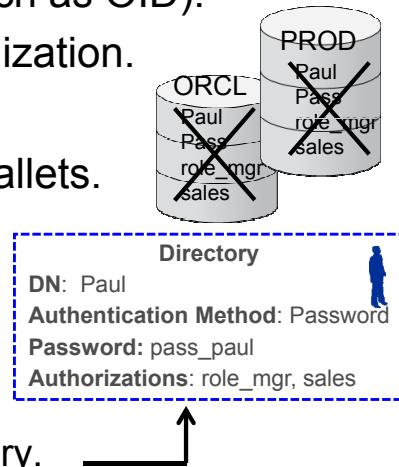
- *Oracle Database Enterprise User Security Administrator's Guide 12c Release 1*

In addition, refer to other courses, including the following:

- *Oracle WebLogic Server 11g: Administration Essentials*
- *Oracle Directory Services 11g: Essential Concepts*
- *Oracle Virtual Directory 11g: Administration*

Enterprise User Security

- EUS stores user credentials and authorizations in a central location (LDAP-compliant directory such as OID).
- It eases administration through centralization.
- It enables single-point authentication.
- It eliminates the need for client-side wallets.
- Example:
 - User changes job roles.
 - Security administrator changes the user roles in Oracle Virtual Directory.
 - No changes are made to the services that the user accesses.



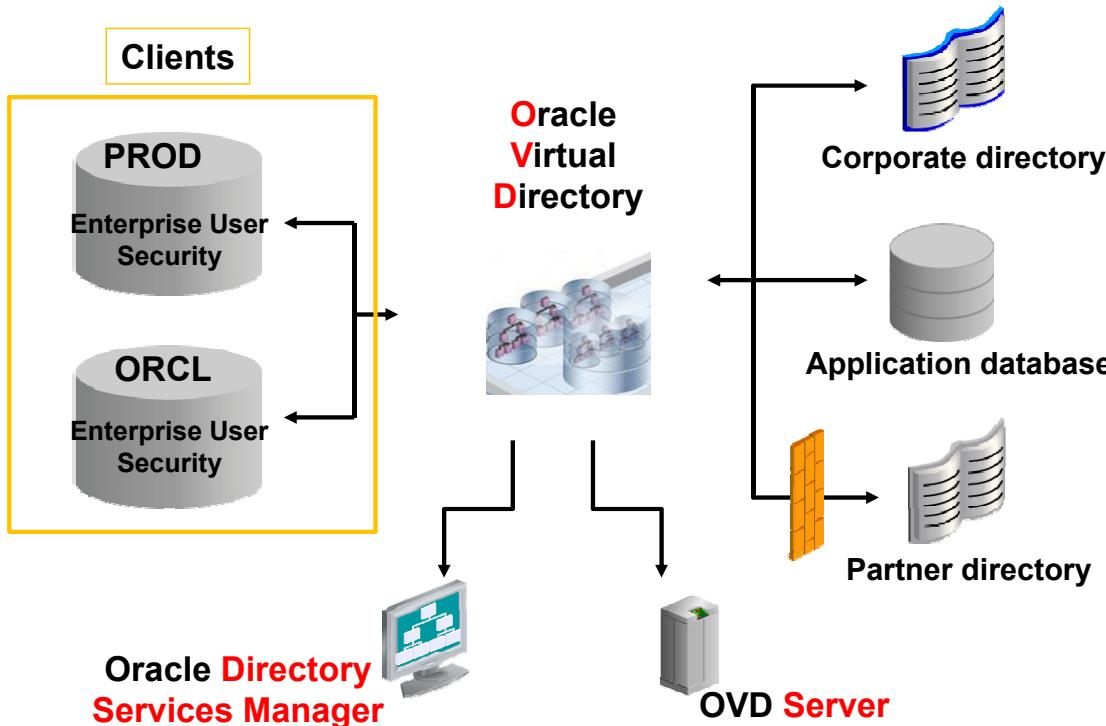
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

EUS is an important component of Oracle Database 12c Release 1 (12.1) Enterprise Edition. Enterprise users are those users that are defined in a directory. Their identity remains constant throughout the enterprise. EUS addresses the user, the administrative, and the security challenges by centralizing storage and management of user-related information and relying on the identity management services supplied by Oracle Virtual Directory (OVD).

When an employee changes jobs in such an environment, the administrator needs to modify information only in one location (the directory) to make effective changes in multiple databases and systems. This centralization can substantially lower administrative costs while materially improving enterprise security.

Oracle Identity Management Software



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

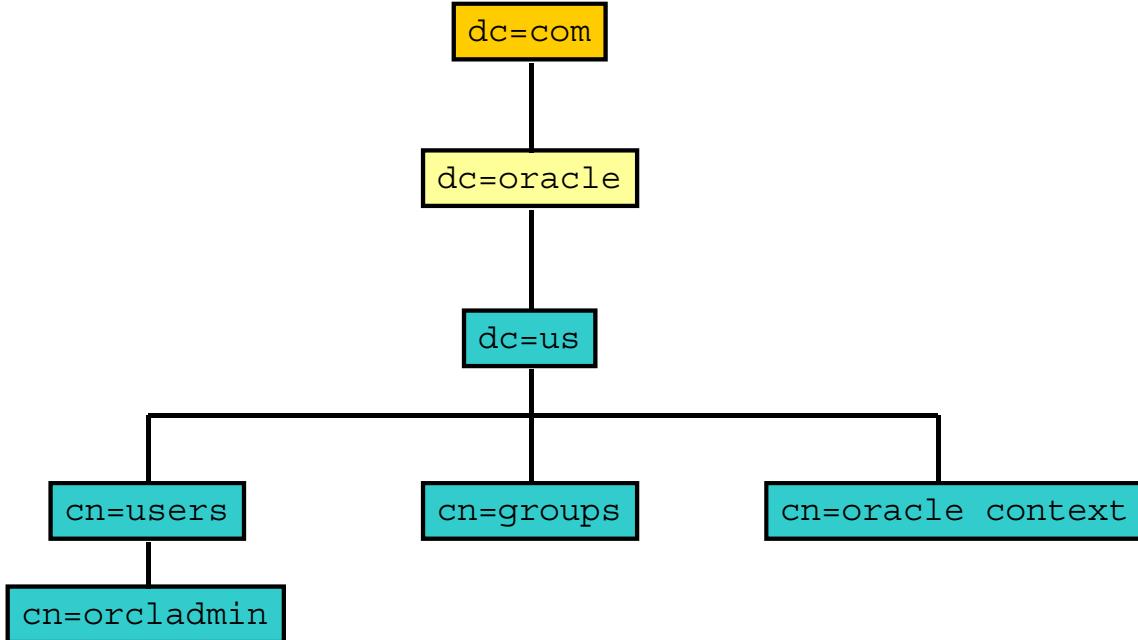
EUS relies on Oracle Identity Management infrastructure, including OID and OVD, which in turn uses an LDAP-compliant directory service to centrally store and manage users. Before you install and configure OVD, which comprises OVD Server and Oracle Directory Services Manager, make sure that you have both Oracle Database and Oracle WebLogic Server installed and configured.

EUS used to be integrated with OID. Because most enterprises already have deployed enterprise directories, they prefer EUS implementation to leverage their existing directory infrastructure. To realize this requirement without using OID, OVD acts like an interpreter for Oracle database information requests. The databases communicate with OVD in the same way they do with OID without knowing that the information they receive is actually stored in a third-party directory. OVD is certified with EUS to support user identities in OID, Active Directory, and Novell eDirectory.

OVD is composed of two applications:

- **Oracle Virtual Directory Server:** OVD Server is a back-end application used to integrate with internal and external data repositories. It also serves as a gateway between Oracle Directory Services Manager and the data repositories.
- **Oracle Directory Services Manager:** Oracle Directory Services Manager is a web-based client interface for OVD. It communicates with OVD Server through standard web services operations. Oracle Directory Services Manager is also a single interface for unified identity services. As a result, administrators can:
 - Unify existing identity data without consolidating it
 - Reuse this data without copying it

Directory Structure: Overview



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The structure in a directory is a tree structure following LDAP standards. Each level is a called a realm.

There is an `orcladmin` defined at the highest level (not shown) that is the administrator of the entire structure.

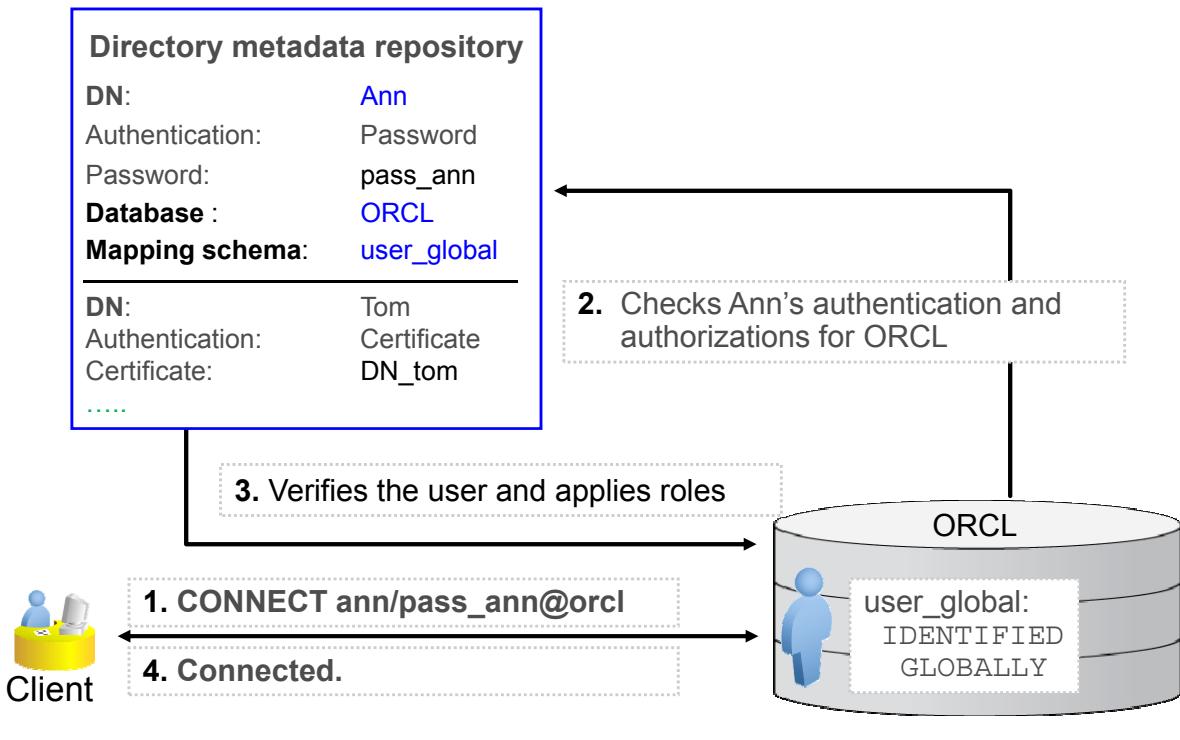
The `cn=orcladmin, cn=users, dc=us, dc=oracle, dc=com` (shown at the lower left) is the administrator that is used to manage the `us.oracle.com` realm.

A directory object in the hierarchy is uniquely named by a distinguished name (DN). A DN is structured with several attributes. The attributes appear in order beginning at the left with the name of the leaf and ending at the right with the name of the root.

For example: `cn=j smith, cn=users, ou=promotions, ou=marketing, dc=us, dc=oracle, dc=com`

- CN: Common Name => groups, users
- OU: Organizational Unit => Promotions, Marketing
- O: Organization => Oracle, non-Oracle
- C: Country => UK, US, FR
- DC: Domain Component => mydomain, us, oracle, com

Oracle Database: Enterprise User Security Architecture



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The EUS architecture is transparent to the end user. In the example shown in the slide, a client can submit the same connect command, whether connecting as a database user or an enterprise user. The enterprise user has the additional benefit of allowing the use of a shared schema.

The authentication process is as follows:

1. The user presents a username and password (or other credentials).
2. The directory returns the authorization token to the database.
3. The schema is mapped from the directory information.
4. The directory supplies the global roles for the user. Enterprise roles are defined in the directory and global roles are defined in the database (non-CDB or PDB). The mapping from enterprise roles to global roles is in the directory.
5. The directory can supply the application context. An application context supplied from the directory is called a global context.

Note: Each PDB has its own EUS metadata, such as global users, global roles, and so on. Each PDB should have its own identity in the directory.

Authenticating Enterprise Users

EUS supports the following authentication methods:

- Password authentication
- Secure sockets layer (SSL) authentication
- Kerberos authentication



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

EUS supports three authentication methods. Each authentication method has advantages and disadvantages. These determine which authentication method is best for your EUS implementation. Configuring password authentication is described in this lesson. Configuration details for the other methods can be found in *Oracle Database Security Guide 12c Release 1*.

All three methods provide:

- Centralized user and credential management
- A user identity that can be used in two-tier or multitier applications
- The methods to support current user database links if the connection between databases is over the secure sockets layer (SSL)

Password authentication:

- Is password-based authentication
- Requires separate authentication for each database connection
- Retains users' current authentication methods
- Supports Oracle Release 7.3 (and later) clients with Oracle Database 10g and later

SSL authentication:

- Provides strong authentication over SSL
- Supports single sign-on by using SSL
- Supports Oracle8*i* (and later) clients with Oracle Database 10g and later

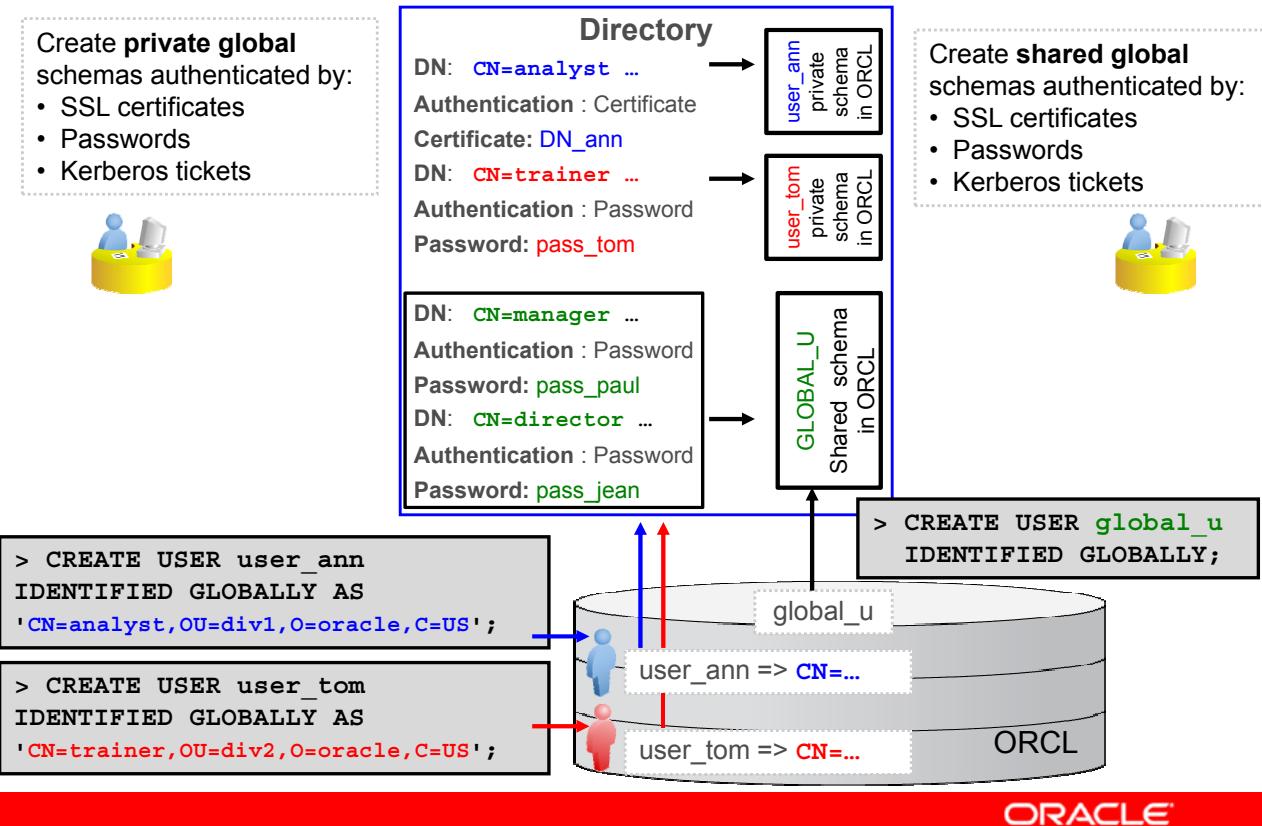
Initial configuration may be more difficult because public key infrastructure (PKI) credentials must be generated for all users. (The degree of difficulty depends on administrators' PKI knowledge.)

Kerberos authentication:

- Provides strong authentication by using Kerberos (version 5) tickets
- Supports single sign-on by using Kerberos (version 5) encrypted tickets and authenticators, and authentication forwarding
- Supports Oracle Database 10g (and later) clients with Oracle Database 10g and later

Initial configuration may be more difficult because Kerberos must be installed and configured to authenticate database users.

Enterprise Users



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Authentication methods and certificates of users can be centralized in the directory. A user can connect to the database in two different ways.

- The first way is a **global private schema** in the database that has a one-to-one schema mapping in the directory. This method requires that the user be created in every database where the end user requires access. The following command creates a database user identified by a distinguished name. The DN is the distinguished name in the user's PKI certificate in the user's wallet.

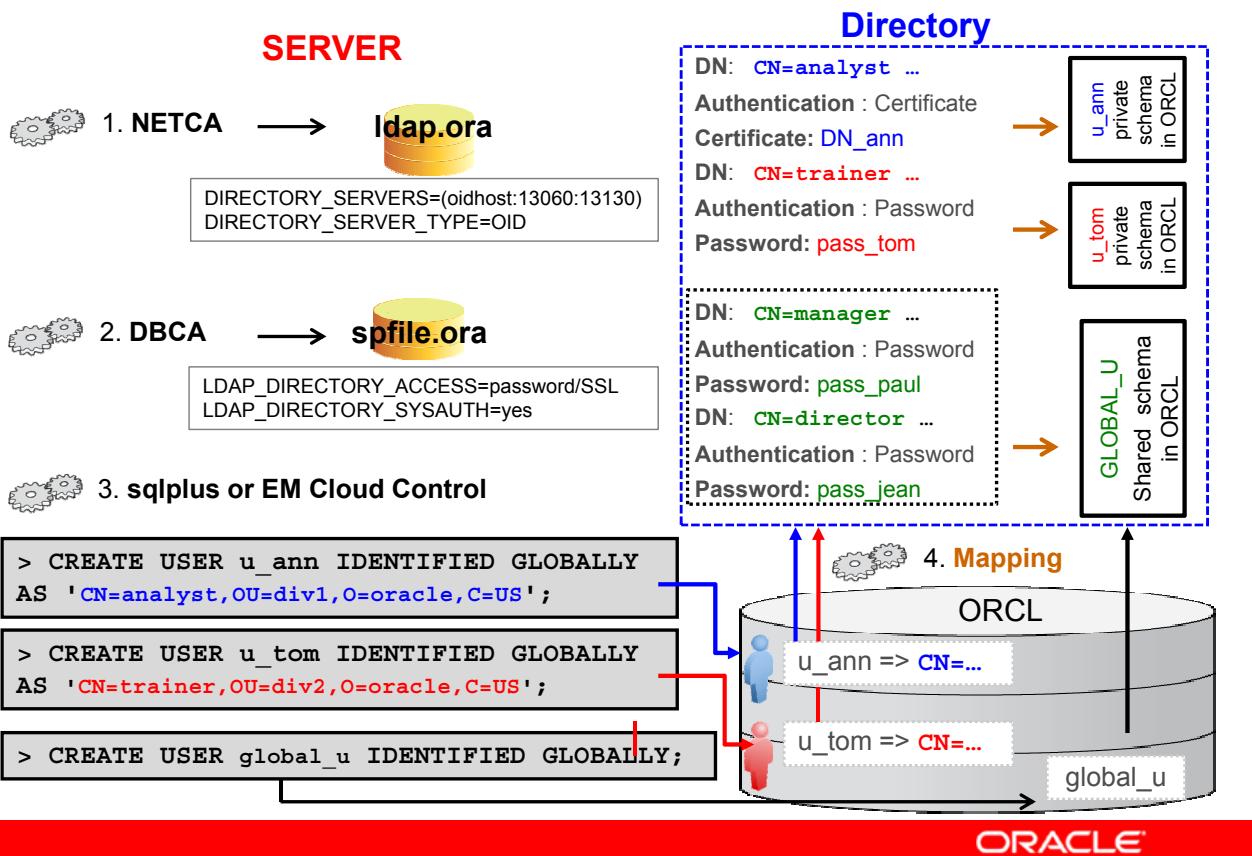
```
CREATE USER global_ann
IDENTIFIED GLOBALLY AS 'CN=analyst,OU=division1, O=oracle,
C=US';
```

- The second way is a **global shared schema** in the database that has a shared schema mapping in the directory. Any user identified to the directory can be mapped to the shared global schema in the database. The mapped user will be authenticated by the directory and the schema mapping will provide the privileges. The following command creates the global shared schema:

```
CREATE USER global_u IDENTIFIED GLOBALLY;
```

No one connects directly to the GLOBAL_U schema. Any user that is mapped to the GLOBAL_U schema in the directory can connect to this schema.

Setting Up Enterprise User Security



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A directory must be installed to use EUS. The directory requires a metadata repository in an Oracle database. The simplest way to meet these requirements is to use the default installation of Oracle WebLogic Server with a metadata repository from Oracle Universal Installer.

The steps required to set up Enterprise User Security are the following:

1. Configure the database to use an LDAP directory. Use the Network Configuration Assistant to configure the database. An `ldap.ora` file contains the directory server hostname and ports accessible by TCP or TCPS protocols.
2. Register the database in the directory with the Database Configuration Assistant (DBCA).
3. Create an enterprise user in the `IDENTIFIED GLOBALLY` database.
4. Create a schema mapping object in the directory between one directory entry and a private global enterprise user of the database registered in the directory. You can create a schema mapping object in the directory between several directory entries and a shared global enterprise user of the database.

Identifying the Enterprise User

Who is the real user?

- Show the current shared schema:

```
SQL> SELECT user FROM dual;
```

- Show the real user:

```
SQL> SELECT sys_context ('userenv' , 'external_name')
FROMdual;

SYS_CONTEXT('USERENV','EXTERNAL_NAME')
-----
cn=Scott Taylor,cn=Users,dc=us,dc=oracle,dc=com
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The enterprise user who is mapped to a shared schema is unknown to the database. When the directory is used to authenticate the user to the database, the “real” name of the enterprise user can be found in his or her login session by using the SYS_CONTEXT function. This name is held in the EXTERNAL_NAME attribute of the USERENV context. The shared schema is shown by:

```
$ sqlplus staylor
Password: *****
SQL> SELECT user FROM dual;
```

USER

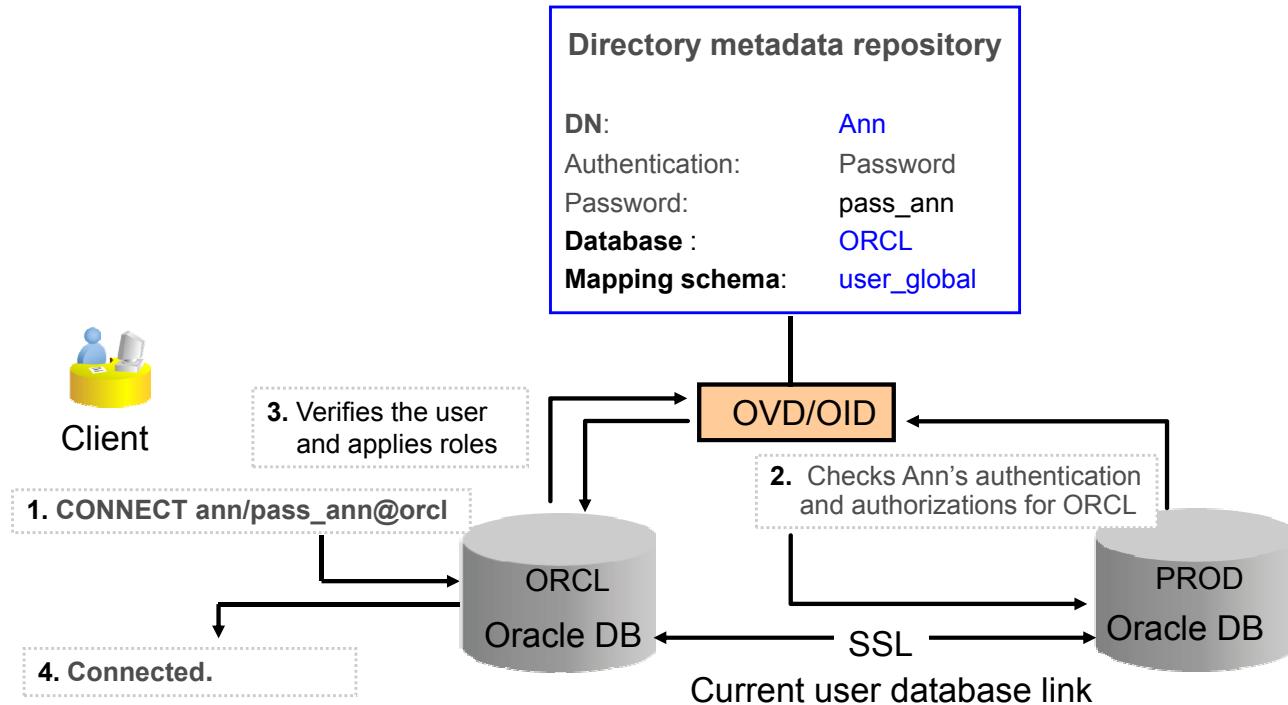
GUEST

The real user is shown by checking the EXTERNAL_NAME attribute of the USERENV context:

```
SQL> select sys_context ('userenv' , 'external_name') from dual;

SYS_CONTEXT('USERENV','EXTERNAL_NAME')
-----
cn=Scott Taylor,cn=Users,dc=us,dc=oracle,dc=com
```

Enabling Current User Database Links



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Current user database links require SSL-enabled network connections between the databases. Before you can enable current user database links, you must enable SSL, create Oracle wallets, and obtain PKI credentials for all databases involved. The directory and Oracle Identity Management metadata repository are not required, but some form of global authentication is required (SSL or Kerberos).

Current user database links enable you to connect to another database as the user you are logged in as, or as another user when used from within a stored procedure owned by that user. Access is limited to the scope of the procedure.

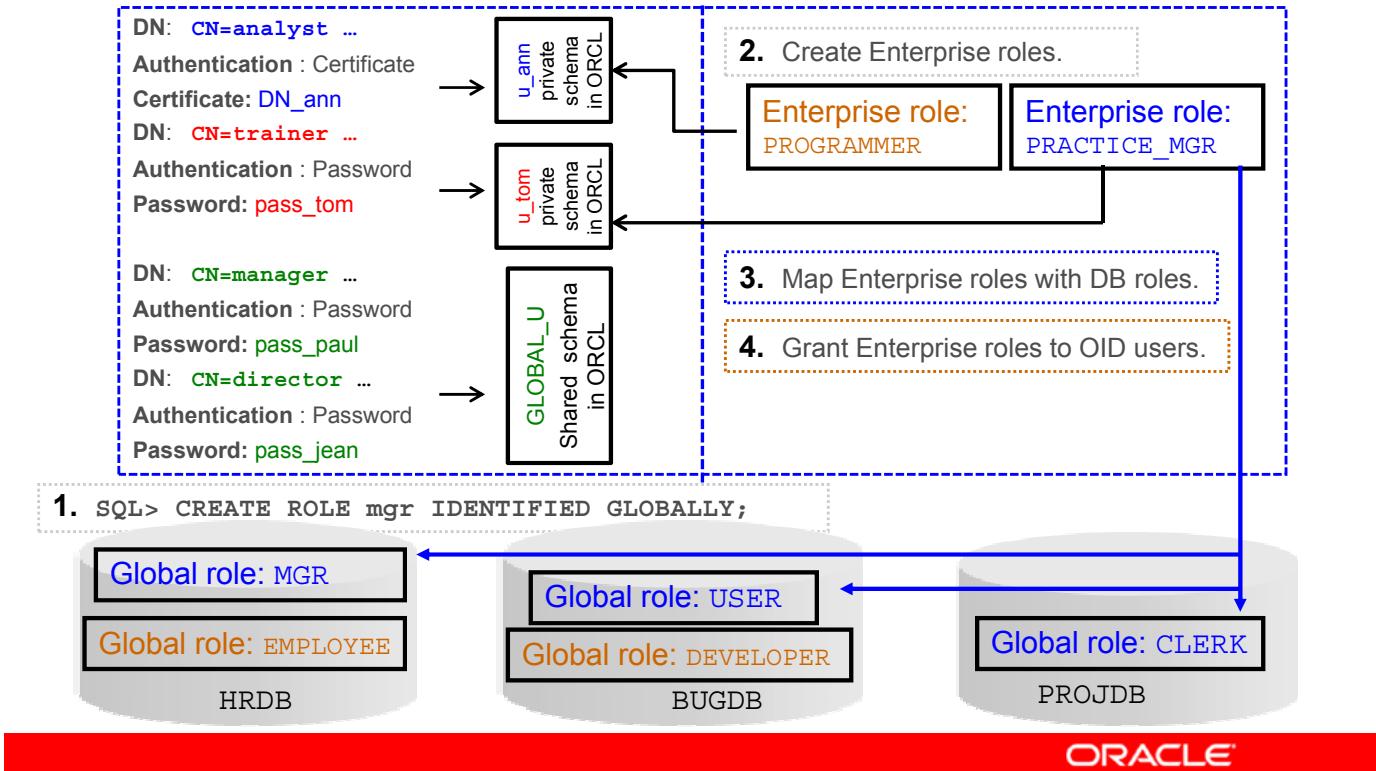
Use the Configure Domain page of EUS in Enterprise Manager Cloud Control to enable current user database links between databases within the enterprise domain in the directory by performing the following steps:

1. On the Enterprise User Security page, click Manage Enterprise Domains.
2. On the Manage Enterprise Domains page, select a specific enterprise domain and click Configure.
3. On the Configure Domain page, click the Configuration tab, select Enable Current User Database Links, and click OK.

Note: Each user accessing a current user database link must be a global user.

Using Enterprise Roles

Enterprise roles can be used like job descriptions.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

An enterprise role allows the administrator to collect all the access requirements for a job role or function into a named role that can span the enterprise. Roles defined in each database are tailored to the applications and data residing in that database. Enterprise roles require the use of a directory.

Example: The HRDB database has an MGR role that allows a manager to modify certain fields of the employee records that are assigned to his or her organization. The BUGDB database has a USER role that allows a user to create bugs and update bugs that he or she created. The PRACTICE_MGR enterprise role is assigned a set of global roles: MGR in the HRDB database, USER in the BUGDB database, and CLERK in the PROJDB database. Each global role is defined in the local database, so the practice manager gets different privileges in each database.

The HRDB database has an EMPLOYEE role that allows every employee to view and modify certain records, such as a profile. The BUGDB database has a DEVELOPER role that is allowed to update any bug that is assigned to him or her. The PROGRAMMER enterprise role is assigned EMPLOYEE and DEVELOPER global roles.

Note: All users assigned the same enterprise role receive the same privileges.

User Migration Utility

The user migration utility is a command-line tool. It migrates users:

- From the database to OID:
 - External users
 - Local users
 - A supplied list of users
- In a two-phase process:
 - Phase 1 populates a table in the database with user information. The DBA is allowed to modify the table data as required.
 - Phase 2 updates the directory and the database.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The user migration utility is a command-line utility invoked with the `uumu` command, which is used to move users from a local database model to an enterprise-user model. This utility makes it easy to migrate local and external database users to an enterprise-user environment in an LDAP directory. It uses the Oracle JDBC Oracle Call Interface (OCI) driver to connect to the database.

Enterprise-user administrators can select for migration any combination of the following user subsets in a database:

- List of users specified on the command line or in a file
- All external users
- All local users

In addition, enterprise-user administrators can specify values for utility parameters that determine how the users are migrated, such as:

- Where to put the migrated users in the LDAP directory tree
- Mapping a user with multiple accounts on various databases to a single directory user entry

Step 1: Preparing for the Migration (Phase One)

In the first part of the migration process, the ORCL_GLOBAL_USR_MIGRATION_DATA interface table is populated with information about the users from the database and the directory. The command-line options are used determine what information populates this table.

Step 2: Verifying User Information

This is an intermediate step to allow the enterprise-user administrator to verify that the user information is correct in the interface table before committing the changes to the database and the directory.

Step 3: Completing the Migration (Phase Two)

After the user information in the interface table is checked, in phase two, the utility retrieves the information from the table and updates the directory and the database.

Depending on whether directory entries exist for migrating users, the utility creates random passwords as follows:

- If migrating users are being mapped to newly created directory entries, the utility generates random passwords, which are used as credentials for both the database and the directory.
- If migrating users are being mapped to existing directory entries with unset database passwords, the utility generates random database passwords only.

In either case, after generating the required random passwords, the utility stores them in the DBPASSWORD and DIRPASSWORD interface table columns. The enterprise-user administrator can read these passwords from the interface table and inform migrating users.

The umu utility produces a list of the allowable parameters for each phase with:

```
umu HELP=YES
```

For more information about the user migration utility, see the *Oracle Database Enterprise User Administrator's Guide*.

Enterprise-User Auditing

Unified audit : UNIFIED_AUDIT_TRAIL view

```
DBUSERNAME      -----> GLOBAL_USER  
EXTERNAL_USERID -----> uid=user.0,ou=People,dc=example,dc=com  
GLOBAL_USERID   -----> ad55a34a763f358f93f9da86f9ecd9e4  
AUTHENTICATION_TYPE -----> (TYPE= (DIRECTORY PASSWORD) )
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

If auditing is turned on, the Oracle Database server captures the identity of enterprise users in the audit trail. The directory can store additional attributes for each user to help identify both authorized and unauthorized users.

- The DBUSERNAME column shows the user identity in the database.
- The EXTERNAL_USERID column shows the user's global identity from the directory.
- The GLOBAL_USERID column shows the global user identifier for the user, for a user logged in as an enterprise user.

Quiz

Enterprise User Security does not require Oracle Internet Directory when other LDAP-compliant directory services are used to define enterprise users.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a

Enterprise User Security can use OVD. OVD is certified with EUS to support Active Directory and OID for deployments.

Summary

In this lesson, you should have learned how to:

- Describe Enterprise User Security
- Set up Enterprise User Security
- Create an enterprise user
- Use global shared and private schemas
- Implement enterprise roles
- Audit enterprise users
- Migrate users from the database to Oracle Internet Directory



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice or Demo: Overview

This practice covers the following topics:

- 7-1: Setting up EUS authentication
- 7-2: Creating a global private schema in the database and mapping with a single directory entry
- 7-3: Creating a global shared schema in the database and mapping with multiple directory entries
- 7-4: Implementing enterprise roles



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

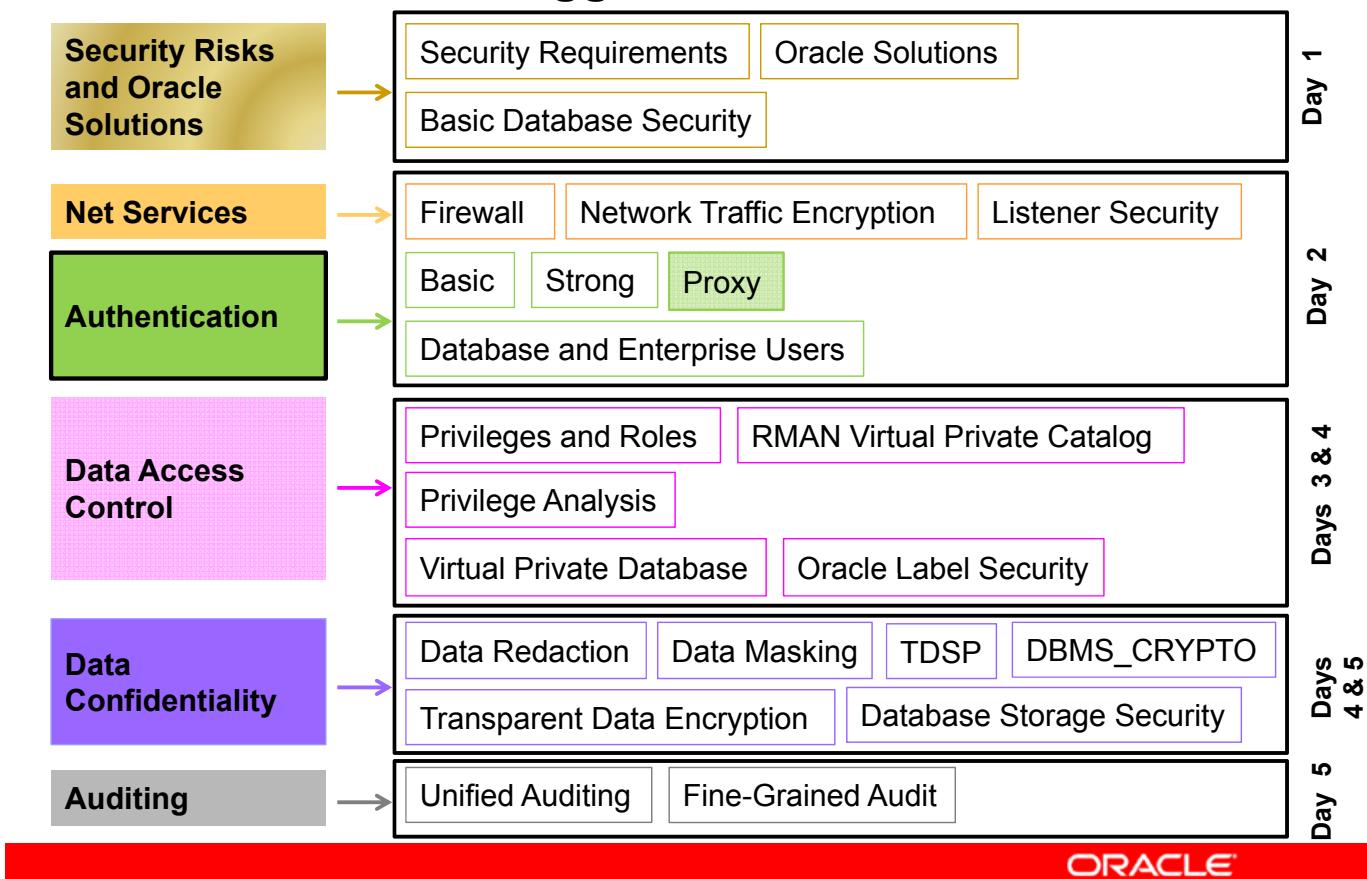
8

Using Proxy Authentication

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Schedule



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe how proxy authentication works
- Manage users authenticated by proxy authentication
- Audit users authenticated by proxy

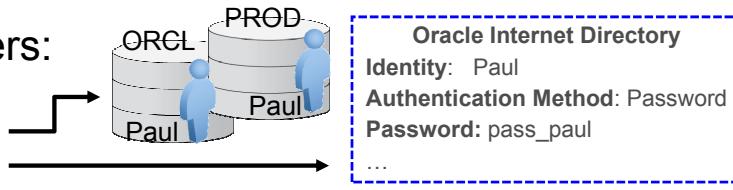


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

User Authentication

Identify two types of users:

- Database users
- Enterprise users



Identify the user in the following ways:

- Basic authentication
 - Database user identified by a password
 - Database user identified by the operating system
- Strong authentication
 - Certificates
 - Kerberos
 - RADIUS
- Proxy authentication



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A basic security requirement is that you must know your users. You must identify them before you can determine their privileges and access rights, and so that you can audit their actions on the data. Knowing the end user allows you to specify users who are allowed to connect through a middle-tier server. In many cases, the middle-tier server authenticates and assumes the identity of the user and is allowed to enable specific roles for the user. This is called proxy authentication.

Note: The term “application” or “application server” is used in the rest of this lesson to refer to a generic application program or application server that may be a custom application or a third-party application. It is not necessarily an Oracle Application Server.

Security Challenges of Three-Tier Computing

- Identify the real user:
 - To control access to the database
 - To audit the real user actions in the database
- Authenticate the end user to the database securely
- Restrict the privileges of the middle tier

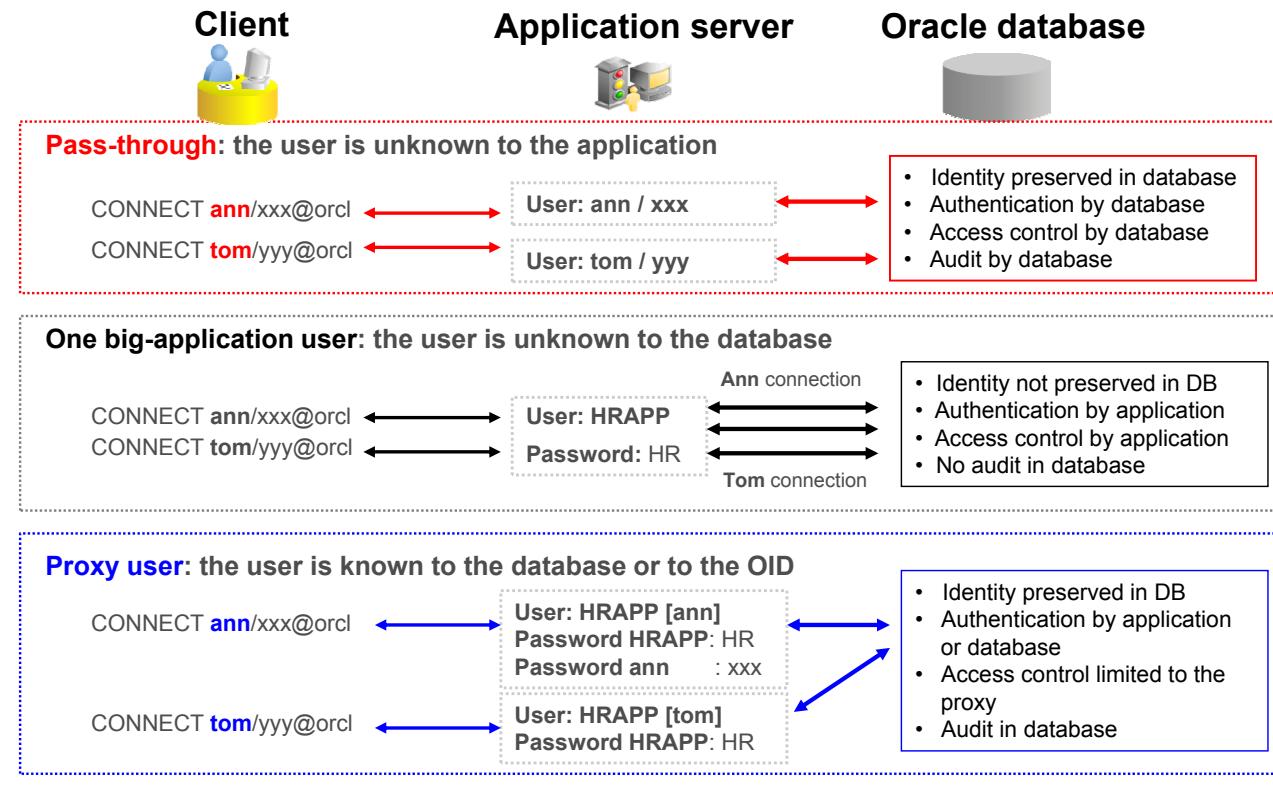


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Although three-tier computing provides many benefits, it raises a number of security issues.

- Identify the real user.
 - Who is the real user? Database-level access control and auditing depends on being able to identify the end user.
- Authenticate the end user to the database.
 - In multitier computing, authenticating the end user to the database securely becomes a challenge.
- Restrict the privileges of the middle tier.
 - For many applications, the security model gives the proxy application user excessive privileges. The challenge is to allow the session created or used by the middle tier to have privileges that are appropriate to the real end user.

Common Implementations of Authentication



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There are several implementations of three-tier authentication methods and variations of those methods. Most of these methods can be grouped into the following categories:

- **Pass through (the user is unknown to the application):** In this model, the application queries the user for credentials and passes the response to the database server for authentication. This is a traditional model requiring each user to have a database schema and login. This method can be moved to an enterprise-user model with no changes to the application. The application is not responsible for authentication, authorization, or access control. Auditing can be handled by the database server because the real user is obvious. This method is no different from the client/server model, where the user connects directly to the database.
- **One big-application user (the user is unknown to the database):** In this model, the application is responsible for authentication, authorization, and access control. All end users present their credentials to the application, and the application user connects to the database. Often, the application user is the application owner with all the privileges to all application objects. This method is easy to code, but violates the principle of least privilege.

When the authentication process is determined by the application, it can use an LDAP directory, local files, or some other method to validate the user. The application can make use of roles, enabling and disabling roles to control access. Unless the application keeps some kind of mapping, end-user auditing can be difficult or impossible.

Other methods:

- **The user is reauthenticated to the database:** In this model, the user presents a credential to the application (not necessarily the same as the database credentials), and the application authenticates the user to the database. This model requires a secure method of storing user credentials in the middle tier. Using LDAP directory services is one of the few methods that can store credentials securely. Single sign-on is a secure solution for this model.
- **The user is identified to the database:** The application can identify the user with a token of some kind. This token maps the end user to a session. The end user is still unknown to the database server, but end-user auditing is possible. The application uses the `DBMS_APPLICATION_INFO.SET_CLIENT_IDENTIFIER` procedure or sets `CLIENT_IDENTIFIER` with the `DBMS_SESSION.SET_IDENTIFIER` procedure in conjunction with the application context to make this identification. The later method is covered in the lesson titled “Using Application Contexts.”
- **The user is proxied:** Oracle Database supports three forms of proxy authentication:
 - The middle-tier server authenticates itself to the database server and provides an end-user name. The end user has already authenticated to the middle-tier server. End-user identities can be maintained all the way through to the database.
 - The database user is not authenticated by the middle-tier server. The end-user identity and database password are passed through the middle-tier server to the database server for authentication. This is another form of the pass-through method.
 - The end user—in this case, a global user—is authenticated by the middle-tier server and passes one of the following through the middle tier for retrieving the end-user name:
 - Distinguished name (DN)
 - Certificate

The use of certificates for proxy authentication may not be supported in future Oracle Database releases.

Quiz

Which of the following are valid for proxy authentication?

- a. End users unknown to the database
- b. Enterprise users that are maintained in Oracle Internet Directory
- c. Database user that supplies a password
- d. Application users that are known to the application but not to the database



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b, c, d

Using Proxy Authentication for Database Users

- Authenticate the user without a database password:

```
SQL> ALTER USER ann GRANT CONNECT THROUGH hrapp;
```



- Authenticate the user with a database password:

```
SQL> ALTER USER tom GRANT CONNECT THROUGH hrapp
      AUTHENTICATION REQUIRED;
```



```
SQL> CONNECT hrapp [ann] /hrapp_pwd
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Middle-tier authentication allows one JDBC connection (session) to act as a proxy for other JDBC connections. Use the CONNECT THROUGH clause in the ALTER USER command to indicate that the user is authenticated through a middle tier.

Authenticating Without a Database Password

When the middle tier authenticates the user, you may not want to give the middle tier the user's database password. If the middle tier does not know the password, the user can be authenticated without a database password, using the following command:

```
ALTER USER ann GRANT CONNECT THROUGH hrapp;
```

The user can connect as `ann` by using the already authenticated credentials of the middle-tier `hrapp`. *This method assumes that the middle tier is trusted to perform the authentication.* The created session behaves as if `ann` has been connected normally; `ann` does not have to divulge her password to the middle tier. The proxy session accesses the schema of `ann`. This method is sometimes appropriate for application servers in a trusted region.

Authenticating With a Database Password

To authenticate the user with a password, use the following command:

```
ALTER USER tom GRANT CONNECT THROUGH APPSVR AUTHENTICATION REQUIRED;
```

The Oracle instance expects the proxy to authenticate the user. *In this method, the middle tier is not assumed to be trusted.* The middle tier may not perform any authentication. The user authenticates to the database by providing the database password. This method is appropriate to application servers that are outside a trusted region (firewall). The user provides a password that is passed through to the database.

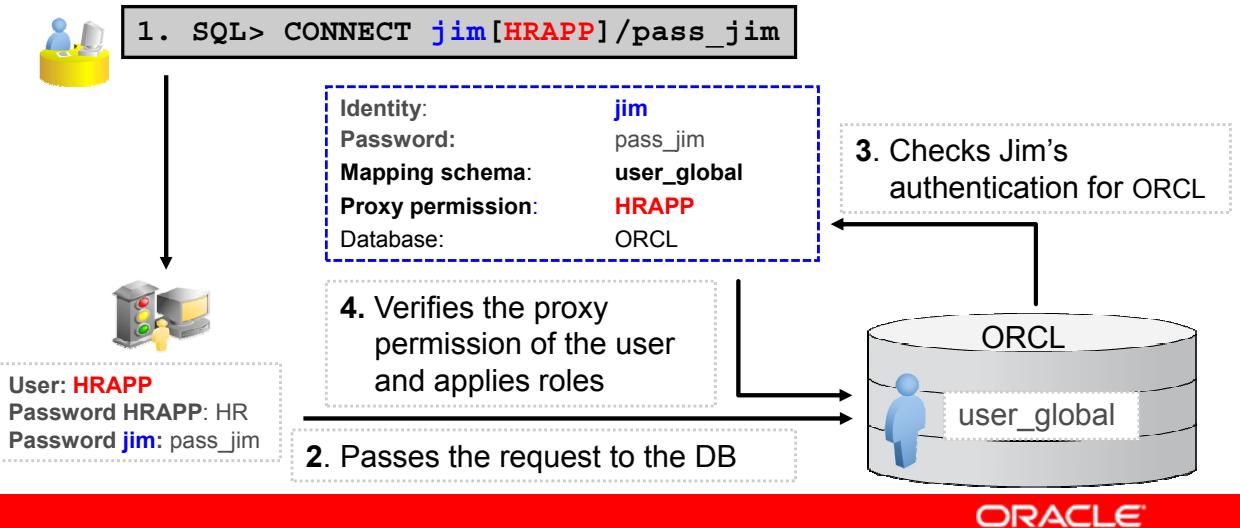
Using Proxy Authentication for Enterprise Users

Use enterprise users with current applications.

- Let the directory authenticate the users.

```
SQL> CREATE USER hrapp GRANT CONNECT THROUGH ENTERPRISE USERS;
```

- Connect as a database user.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

In Oracle Database, the enterprise user proxy is available to allow you to use enterprise user security in combination with existing applications that use the “one big-application user” model. All the users have been connecting as HRAPP, and now they have been given enterprise user credentials in the directory. They can continue to use the application with a proxy connect as shown. The users provide their Enterprise User Security (EUS) credentials and the target user, and connect to the database as the target user.

Enterprise users can be individually granted permissions to proxy as local database users.

Enterprise user proxy permissions are created and stored in Oracle Internet Directory. A permission allows one or more enterprise users or groups to proxy as a target database user. By default, domain administrators manage proxy permissions in the directory for an enterprise domain. These permissions are configured and managed using Enterprise Manager Enterprise User Security pages.

Note: In most cases, enterprise users such as JIM are unknown to the database. They are called proxy users. The mapping of a proxy user to a database user is called a proxy permission. The user making the connection, HRAPP, is called the target user, is a database user, and is not identified globally.

Revoking Proxy Authentication

- Revoke proxy authentication through a middle tier:

```
SQL> ALTER USER ann REVOKE CONNECT THROUGH hrapp;
```

- Do not use the AUTHENTICATED REQUIRED clause with REVOKE.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The following statement takes away the right of the user PHALL to connect through the proxy user APPSVR:

```
ALTER USER phall REVOKE CONNECT THROUGH APPSVR;
```

You cannot specify the AUTHENTICATED USING or AUTHENTICATION REQUIRED clause as part of a REVOKE CONNECT THROUGH <PROXY> clause.

Application-User Model

- Use OCI, thin JDBC, or thick JDBC.
- End-user identity is set by the middle tier.
- The authentication process is as follows:
 1. The middle tier authenticates to the database.
 2. The end user authenticates to the middle tier.
 3. The middle tier allocates a session to the user, identifying the user with `CLIENT_IDENTIFIER`.
 4. Optionally, the middle tier can enable roles to restrict the privileges of the user.
- Examples:
 - Certificate
 - Application username and password



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Many applications use session pooling to set up a number of sessions to be reused by multiple users. In this context, the end users who are not known to the database are authenticated to the middle tier of an application. The Oracle Database supports an application-user proxy for these types of applications.

In this model, the middle tier passes a client identifier to the database upon session establishment. (The client identifier can actually be anything that represents an end user connecting to the middle tier—for example, the end-user ID or an IP address.) The client identifier, representing the end user, is available in user-session information and can also be accessed via an application context (via the `USERENV` naming context). In this way, applications can set up and reuse sessions, while still being able to keep track of the *end user* in the session.

Applications can reset the client identifier and, thus, reuse the session for a different user, enabling high performance. For OCI-based connections, the call to change `CLIENT_IDENTIFIER` is combined with other OCI calls to further enhance performance. An application-user proxy is available in thin JDBC, thick JDBC, and OCI, and provides the benefits of connection pooling without the overhead of separate user sessions (even lightweight ones).

Note: `V$SESSION.CLIENT_IDENTIFIER` and

`V$SESSION_CONNECT_INFO.AUTHENTICATION_TYPE` can provide additional information about the end user's identity and authentication, for auditing purposes.

Authenticating Application Users

The full authentication sequence from the client to the middle tier to the database occurs as follows:

1. During startup, the middle tier authenticates itself to the database server and creates a connection pool. The method of authenticating to the database can be a password or an authentication mechanism supported by Oracle Advanced Security, such as a Kerberos ticket or an X.509 certificate (SSL).
2. The end user authenticates to the middle tier, using whatever form of authentication that the middle tier accepts. Two examples are:
 - The user can authenticate to the middle tier by using an X.509 certificate by means of SSL.
 - The user can authenticate to the middle tier by using a username and password stored in the application.
3. The middle tier uses an available connection from its connection pool to create a session for the end user, and uses JDBC or OCI calls to pass the end-user identifier to the database.
4. Depending on the information stored in the application, the middle tier can also set roles for the end user. For example, if the application has multiple roles, you can do the following:
 - a. Create database roles that match the application roles.
 - b. Assign appropriate privileges to the database roles.
 - c. Assign these roles to the application-server user, but disable the roles.
 - d. When the user starts a session, the application server enables the appropriate roles, depending on the roles assigned to the user in the application. These roles may be secure application roles, which may be enabled only through a secure package.

Using Proxy Authentication with Roles

- Specify roles that the proxy user is allowed to activate:

```
SQL> ALTER USER ann GRANT CONNECT THROUGH hrapp  
      WITH ROLE hr_clerk;
```

- Prevent any roles from being activated by the proxy user:

```
SQL> ALTER USER ann GRANT CONNECT THROUGH hrapp  
      WITH NO ROLES;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

By default, a proxy user can activate all roles that are granted to a user. If a user is allowed only certain privileges through a specific application, you can limit the roles that the proxy may activate, as shown in the slide example or this example:

```
ALTER USER ann  
GRANT CONNECT THROUGH hrapp  
WITH ALL ROLES EXCEPT hr_manager;
```

If the user does not require any privileges in a specific application, the proxy user can be prevented from activating any roles that may be granted to the user for use in other applications, as shown in this example:

```
ALTER USER ann  
GRANT CONNECT THROUGH hrapp  
WITH NO ROLES;
```

Data Dictionary Views for Proxy Authentication

- DBA_PROXIES: All proxy connections
- USER_PROXIES: Connections that the current user is allowed to proxy
- PROXY_USERS: Users who can assume the identity of other users
- V\$SESSION_CONNECT_INFO: Network connections for all current sessions
- V\$SESSION: Session-connect details:
 - The PROGRAM column contains “proxy-user...”
 - The MODULE column contains “proxy-user...”



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

- USER_PROXIES: Displays information about connections that the current user is allowed to proxy. This view does not display the PROXY column.
- PROXY_USERS: Describes users who can assume the identity of other users
- V\$SESSION_CONNECT_INFO: Displays information about network connections for all current sessions
- V\$SESSION: The PROGRAM and MODULE columns have a value of “proxy-user...” when the session is being proxied.

Data Dictionary Views: DBA_PROXIES and USER_PROXIES

```
SQL> SELECT * FROM dba_proxies;
```

PROXY	CLIENT	AUTHENTICATION	AUTHORIZATION_CONSTRAINT	PROXY_AUTHORITY
HRAPP	JIM	NO	PROXY MAY ACTIVATE ROLE	DIRECTORY
HRAPP	TOM	NO	NO CLIENT ROLES MAY BE ACTIVATED	DATABASE
HRUSER	PFAY	YES	PROXY MAY ACTIVATE ALL CLIENT DATABASE ROLES	

Indicates whether the proxy is required to supply the client's authentication credentials



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

DBA_PROXIES

DBA_PROXIES displays information about all proxy connections in the system. In the example in the slide, HRUSER may activate any or all roles granted to PFAY. HRUSER may activate only a specified role for PHALL. APPSVR may not activate any roles for PHALL.

USER_PROXIES

USER_PROXIES contains connections that the current user is allowed to proxy.

Columns

The columns for both DBA_PROXIES and USER_PROXIES are:

- PROXY: Proxy username, in DBA_PROXIES only
- CLIENT: Name of the client user on whose behalf the proxy user can act
- AUTHENTICATION: Credential passed by the proxy for the client
- AUTHORIZATION_CONSTRAINT: Authority of the proxy to exercise roles on the client's behalf. The value PROXY MAY ACTIVATE ROLES indicates that only selected roles may be activated.
- ROLE: Role referenced in authorization constraint

Data Dictionary Views: V\$SESSION_CONNECT_INFO

```
SQL> SELECT sid, authentication_type,
  2          osuser, network_service_banner
  3  FROM v$session_connect_info WHERE sid = 148;

  SID AUTHENTICA OSUSER NETWORK_SERVICE_BANNER
-----
 148 DATABASE    oracle TCP/IP NT Protocol Adapter for Linux:
                  Version 11.2.0.1.0 - Production
 148 DATABASE    oracle Oracle Advanced Security: encryption
                  service for Linux: Version 11.2.0.1.0 -
                  Production
 148 DATABASE    oracle Oracle Advanced Security:
                  crypto-checksumming service for Linux:
                  Version 11.2.0.1.0 - Production
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This view displays information about network connections for the current session. The **SID** column can be used to join this view with **V\$SESSION**. There can be multiple rows returned for each session. This is a banner for one session started by **OSUSER oracle**.

Columns

The columns and descriptions are:

- **SID:** Session identifier
- **AUTHENTICATION_TYPE:** How the user is authenticated:
DATABASE: Username and password
OS: External operating system
NETWORK: Network or Oracle Advanced Security
PROXY: OCI proxy connection
- **OSUSER:** External username for the database user
- **NETWORK_SERVICE_BANNER:** Product banners for each Oracle Net service used for this connection, with one row per banner

Auditing Actions Taken on Behalf of the Real User

- Audit SELECT from the EMPLOYEES table that HRAPP initiates for ANN as follows:

```
SQL> CREATE AUDIT POLICY pol_proxy  
      ACTIONS SELECT ON employees;
```

```
SQL> AUDIT POLICY pol_proxy BY ann;
```

- Audit SELECT from the EMPLOYEES table that HRAPP initiates for any user as follows:

```
SQL> AUDIT POLICY pol_proxy;
```

- View the result in the following columns of the UNIFIED_AUDIT_TRAIL view:
 - DBUSERNAME: ANN
 - DBPROXY_USERNAME: HRAPP
 - AUTHENTICATION_TYPE: EUS distinguished name (DN)



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can use the proxy authentication features of the database to audit the actions that the middle tier performs on behalf of a user.

For example, suppose an application server HRAPP creates multiple lightweight sessions for the ann user. You can enable auditing for SELECT from the EMPLOYEES table that HRAPP initiates for ANN, as follows:

1. Create an audit policy specifying the action to audit.
2. Enable the audit policy for the ann user.

Auditing Enterprise Users

With enterprise user proxy, the distinguished name of the enterprise user is available in the PROXY_ENTERPRISE_IDENTITY attribute of the USERENV context. The AUTHENTICATION_TYPE column of the UNIFIED_AUDIT_TRAIL view can indicate how the user has been authenticated:

- DATABASE: Authentication has been done by the password.
- NETWORK: Authentication has been done by Oracle Net Services.
- PROXY: The client has been authenticated by another user; the name of the proxy user follows the method type.
- EXTERNAL_NAME: The distinguished name is provided in addition to the other comments if the user is an enterprise user.

Summary

In this lesson, you should have learned how to:

- Describe how proxy authentication works
- Manage users authenticated by proxy authentication
- Audit users authenticated by proxy



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

Practice 8-1 covers implementing and testing database proxy authentication.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

Module

Data Access Control



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

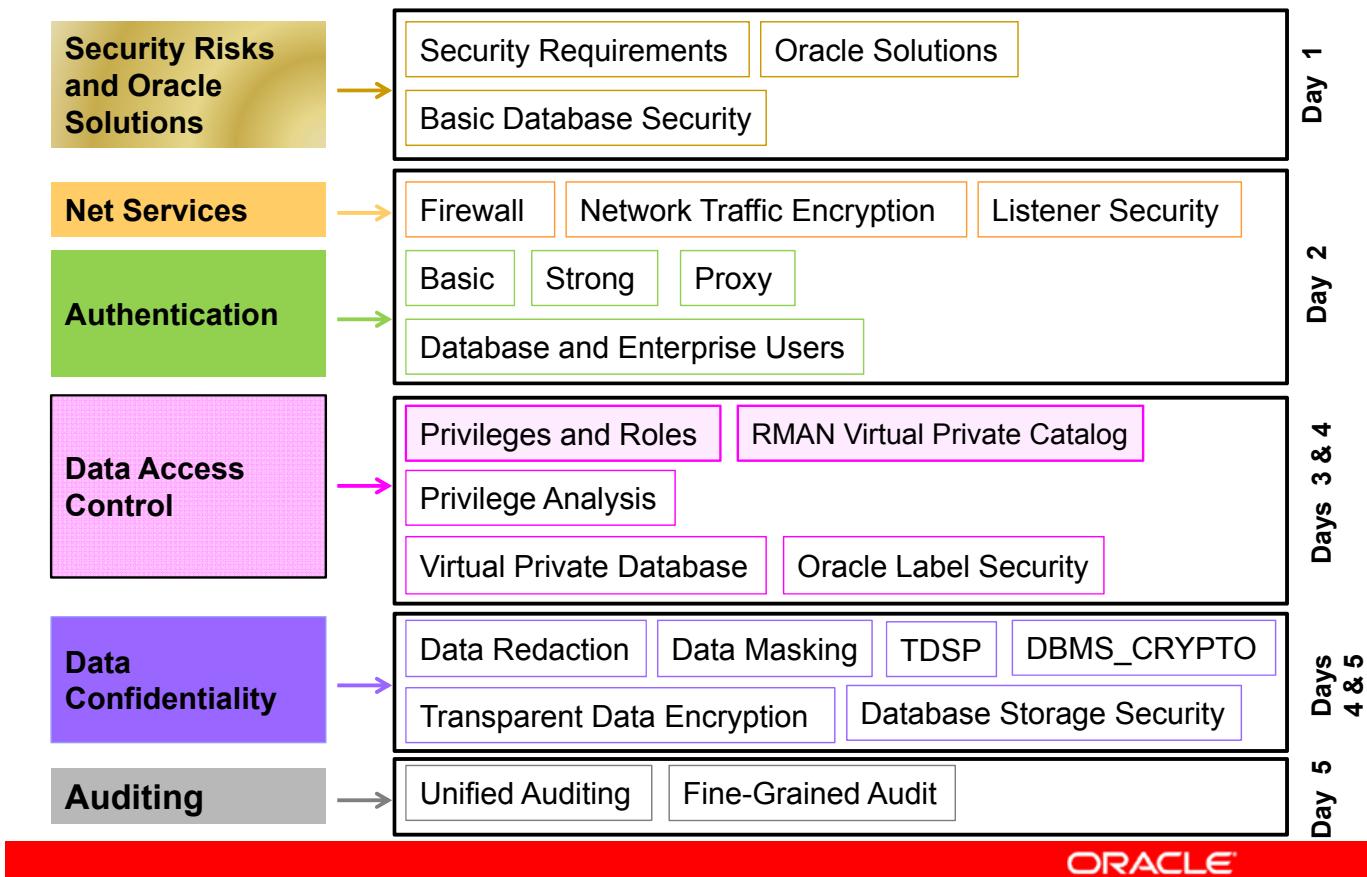
Using Privileges and Roles



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Schedule



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Use database administrative privileges to maintain separation of duties
- Describe system and object privileges
- Describe the different types of roles
- Implement database roles
- Implement common and local roles
- Use secure application roles
- Use procedures to secure objects
- Use Code-Based Access Control for applications
- Secure RMAN catalogs creating Virtual Private Catalogs



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Authorization

Authorization determines the privileges that the user has in the database.

- **Administrative** privileges granted to DBAs according to their specific tasks
- **User** privileges granted to users according to their specific tasks:
 - System privileges
 - Object privileges



User privileges can be grouped in roles allowing administration ease.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Authorization is the process that determines the privileges that the DBA or the user is allowed to exercise. In Oracle Database, authorization is determined by the administration of administrative, system, and object privileges.

A privilege is a right to execute a particular type of SQL statement or to access another user's object. Oracle Database allows very fine-grained control over what users can or cannot do within the database. User privileges are divided into two categories:

- **System privileges:** Each system privilege allows a user to perform a particular database operation or class of database operations (for example, the privilege to create tablespaces is a system privilege). System privileges can be granted by the administrator or by someone who is explicitly given permission to administer the privilege. There are over 150 distinct system privileges.
- **Object privileges:** Object privileges allow a user to perform a particular action on a specific object, such as a table, view, sequence, procedure, function, or package. Without specific permission, users can access only their own objects. Object privileges can be granted by the owner of an object, by the administrator, by someone with the GRANT ANY PRIVILEGE privilege, or by someone who has been explicitly given permission to grant privileges on the object.

A role is a named set of privileges and may be used to grant the privileges as a unit.

Administrative Privileges

Separation of duty: Administration tasks rely on SYSDBA.

- New task-specific privileges for standard administrative tasks
 - Oracle RMAN administrator: backup and recovery
 - Data Guard administrator
 - Key management administrator: TDE keystore management
- Security administrators set with Oracle Database Vault
 - DV owner
 - DV account manager
- No change in SYSDBA privilege
- Administrative privileges mandatorily audited



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

As one of the fundamental security requirements, the principle of separation of duty dictates that completion of any critical task has an associated role. In Oracle Database 11g, the administration of Oracle Database depends heavily on the SYSDBA administrative privilege.

For better separation of duty, Oracle Database now provides task-specific privileges to handle standard administration duties for Oracle Recovery Manager (Oracle RMAN), Oracle Data Guard, and transparent data encryption (TDE). The new privileges are based on the least privilege principle, in which a user is granted only the absolutely necessary privileges required to perform a specific function, and no more. This feature alleviates the need to unnecessarily grant the SYSDBA administrative privilege for many tasks. SYSDBA is necessary for critical situations, such as installation, upgrade, and urgent recovery.

The separation of duty improves database security by maintaining the privacy and the compliance requirements for data.

New Administrative Privileges

Administrative Privilege	Username	Tasks
SYSDBA, SYSOPER	SYS / PUBLIC	Same operations as in 11g
SYSASM	SYS	Specific to ASM instances only
SYSBACKUP	SYSBACKUP	Perform RMAN backup and recovery operations from RMAN or through SQL
SYSDG	SYSDG	Perform Data Guard operations with Data Guard Broker or DGMGRL
SYSKM	SYSKM	Manage transparent data encryption keystore operations

- Distinction between administrative privileges (**SYSBACKUP**) and predefined users (**SYSBACKUP**) 
- **SYSBACKUP**, **SYSDG**, **SYSKM** users cannot be dropped.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c introduces new administrative privileges that are more task-specific and least privileged to support specific administrative tasks. The SYSDBA administrative privilege is not altered for backward compatibility. However, connections with the SYSDBA administrative privilege should be limited to major operations such as installation and upgrade.

- The SYSBACKUP administrative privilege allows you to perform Oracle RMAN backup and recovery operations from Oracle RMAN or through SQL, STARTUP and SHUTDOWN, and other operations.
- The SYSDG administrative privilege allows you to perform Data Guard operations with Data Guard Broker or the DGMGRL command-line interface, and STARTUP and SHUTDOWN and other operations.
- The SYSKM administrative privilege allows you to manage transparent data encryption keystore operations.

These privileges enable you to connect to the database even if the database is not open. After connecting with one of these privileges, you are connected under a predefined user whose name is the privilege name. Each privilege is tied to a specific user.

```
SQL> connect / as SYSBACKUP
SQL> show user
USER is "SYSBACKUP"
```

New Administrative Privilege: SYSBACKUP

System / Object Privileges		
ALTER DATABASE ALTER SYSTEM CREATE SESSION ALTER SESSION ALTER TABLESPACE DROP TABLESPACE UNLIMITED TABLESPACE RESUMABLE	CREATE ANY DIRECTORY CREATE ANY TABLE CREATE ANY CLUSTER AUDIT ANY SELECT ANY DICTIONARY SELECT ANY TRANSACTION	SELECT X\$ tables, V\$ / GV\$ views EXECUTE SYS.DBMS_BACKUP_RESTORE SYS.DBMS_RCMAN SYS.DBMS_IR SYS.DBMS_TTS SYS.DBMS_TDB SYS.DBMS_PLUGTS SYS.DBMS_PLUGTSP
Statements and Roles		
CREATE PFILE CREATE SPFILE CREATE CONTROLFILE DROP DATABASE STARTUP , SHUTDOWN	CREATE / DROP RESTORE POINT (GUARANTEED restore points) FLASHBACK DATABASE SELECT_CATALOG_ROLE HS_ADMIN_SELECT_ROLE	

- Connected as SYSBACKUP predefined user
- Can view tables existence, but not application data

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The SYSBACKUP administrative privilege allows the user being granted this privilege to:

- Perform STARTUP and SHUTDOWN operations
- Oracle RMAN tasks: backup, restore, recover (or connected to SQL*Plus) including TSPTR
- Create or drop a database, create a control file
- Perform ALTER DATABASE to change ARCHIVELOG mode
- Flashback the database: create and drop guaranteed restore points
- Create an SPFILE or PFILE
- Alter the SYSAUX tablespace or drop it if the database is started in UPGRADE mode
- Audit any operation
- View the DBA_xxx, GV\$, and V\$ views but not SELECT on application tables

After connecting as AS SYSBACKUP, you are connected under the predefined SYSBACKUP user.

```
$ rman target ''/ as sysbackup'
RMAN> select user from dual;
USER
-----
SYSBACKUP
```

New Administrative Privilege: SYSDG

System / Object privileges	
CREATE SESSION ALTER SYSTEM ALTER SESSION ALTER DATABASE SELECT ANY DICTIONARY	SELECT X\$ tables, V\$ and GV\$ views DELETE / SELECT APPQOSSYS.WLM_CLASSIFIER_PLAN EXECUTE SYS.DBMS_DRS
Statements and Roles	
STARTUP SHUTDOWN	CREATE RESTORE POINT DROP RESTORE POINT (including GUARANTEED restore points) FLASHBACK DATABASE

- Connected as SYSDG predefined user
- Can view tables existence, but not application data



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The SYSDG administrative privilege allows the user being granted this privilege to:

- Perform STARTUP and SHUTDOWN operations
- Perform ALTER DATABASE to change ARCHIVELOG mode
- Perform ALTER DATABASE RECOVER, including TSPITR
- Flashback the database
- Create and drop guaranteed restore points
- Start the observer
- Run DGMGRL
- Execute DBMS_DRDS.INITIATE_FS_FAILOVER to allow an application to request the primary database to immediately invoke a fast-start failover
- Manage the primary and standby database instances
- Make trusted call-outs for LogMiner packages
- View the DBA_xxx, GV\$ and V\$ views but not SELECT on application tables

After connecting with the SYSDG privilege, you are connected under the predefined SYSDG user.

```
SQL> connect / as SYSDG
SQL> show user
USER is "SYSDG"
```

New Administrative Privilege: **SYSKM**

System / Object privileges

```
CREATE SESSION  
ADMINISTER KEY MANAGEMENT  
SELECT SYS.V$WALLET  
SELECT SYS.V$ENCRYPTION_WALLET  
SELECT SYS.V$ENCRYPTED_TABLESPACES
```

- Connected as **SYSKM** predefined user
- Manage TDE operations
 - Keystore creation, opening, closing
 - Master Key creation and changes
 - Column and tablespace keys management
 - Access to TDE information in appropriate views
- No access to application data



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When tablespace encryption is in use, TDE encrypts data in REDO logs. Therefore, during a recovery, encrypted REDO logs must be decrypted before they can be applied, and the Oracle keystore must be opened after the database is mounted and before the database is opened. Because of this requirement and because the **SYSDBA** administrative privilege is the only privilege that allows keystore operations during the mounted state, the encryption key management currently relies on the **SYSDBA** administrative privilege.

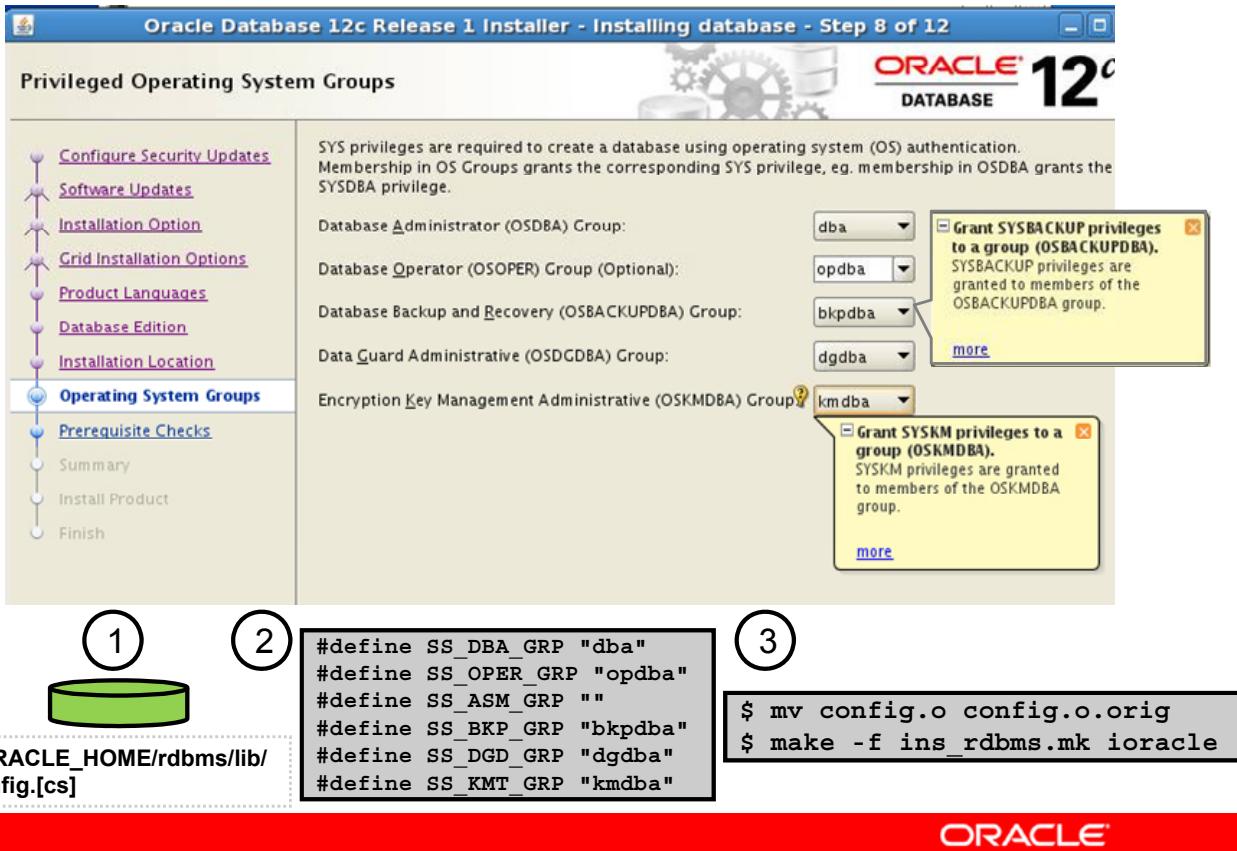
To maintain clear separation between database administration and encryption key management for compliance requirements, the **SYSKM** administrative privilege allows the user being granted this privilege to:

- Manage any TDE operation like keystore and keys management
- View the TDE-related views but no **SELECT** on application tables

After connecting with the **SYSKM** privilege, you are connected under the predefined **SYSKM** user.

```
SQL> connect / as SYSKM  
SQL> show user  
USER is "SYSKM"
```

OS Authentication and OS Groups



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In OS authentication, the groups are created and assigned specific names as part of the database installation process, provided that OS UNIX or WINDOWS groups are created and accounts are assigned to the appropriate operating-system-defined groups.

Membership in a UNIX or WINDOWS group affects your connection to the database as follows:

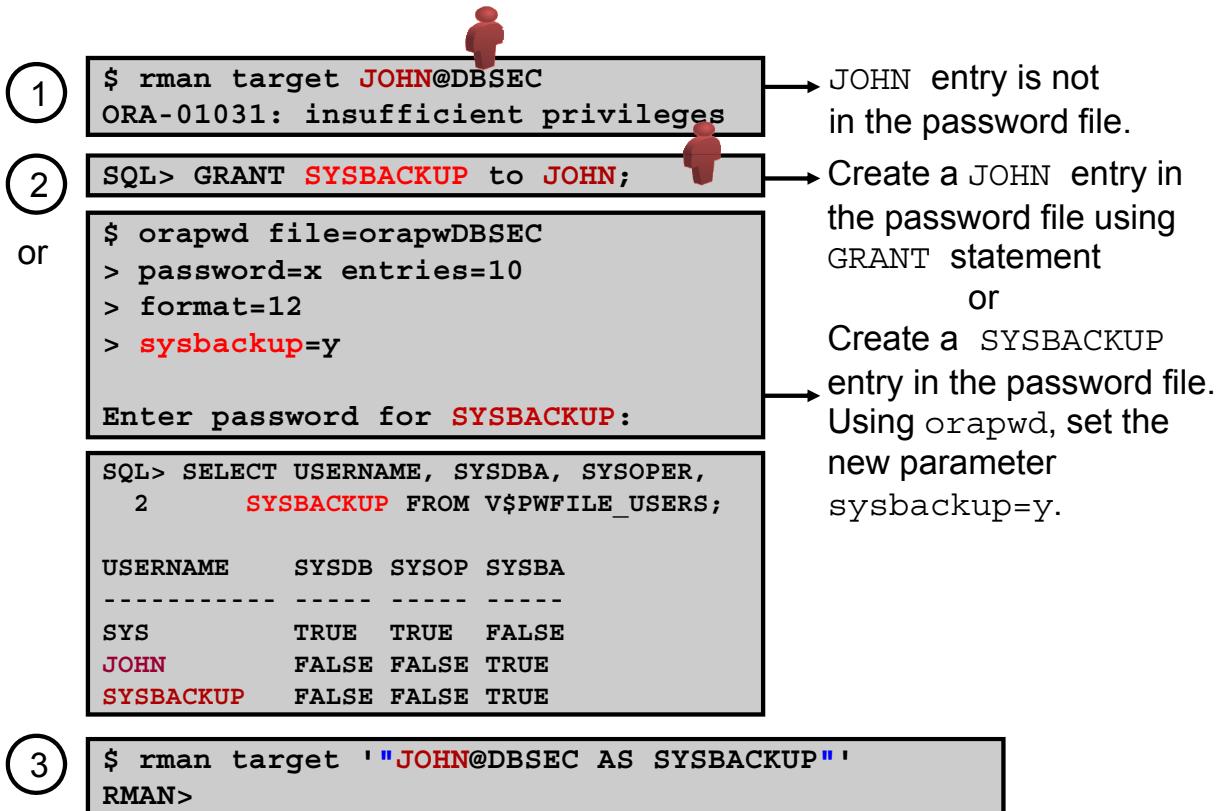
- If you are a member of the OSBACKUP group, and you specify AS SYSBACKUP when you connect to the database, you connect to the database with the SYSBACKUP administrative privilege under the SYSBACKUP user.
- If you are a member of the OSDG group, and you specify AS SYSDG when you connect to the database, you connect to the database with the SYSDG administrative privilege under the SYSDG user.
- If you are a member of the OSKM group, and you specify AS SYSKM when you connect to the database, you connect to the database with the SYSKM administrative privilege under the SYSKM user.

Without being a member of these OS groups, users will not be able to connect as administrative users by using the OS authentication. That is, “CONNECT / AS SYSDBA” fails. However, the users can still connect by using other authentication mechanisms (for example, network, password, or directory-based authentication). To change the OS group names (as shown in steps 1 and 2 in the slide), ensure that you are using the groups defined in the \$ORACLE_HOME/rdbms/lib/config.[cs] file. Next, shut down all databases, and then relink the Oracle executable as shown in step 3 in the slide.

Windows User Groups

The Windows user groups are ORA_%HOMENAME%_SYSBACKUP, ORA_%HOMENAME%_SYSDG, and ORA_%HOMENAME%_SYSKM. These user groups cannot be changed.

Password Authentication for SYSBACKUP



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Password authentication still requires the REMOTE_LOGIN_PASSWORDFILE instance parameter set to EXCLUSIVE and a password file.

New parameters during the creation of the password file define new formats and supported new administrative privileges.

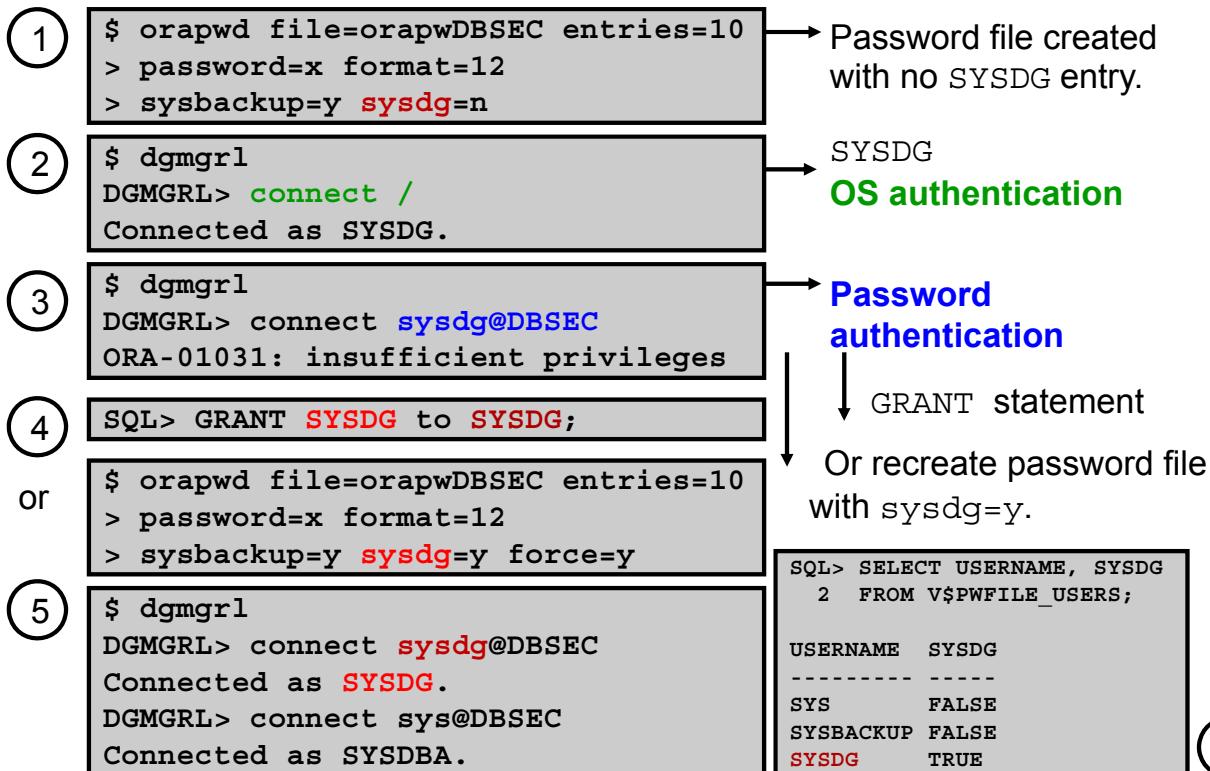
- **FORMAT:** If 12, the default, the password file is created in a version 12.x format and supports SYSBACKUP, SYSDG, and SYSKM administrative privileges. If legacy, the password file is created in legacy format, which is the format prior to Oracle Database 12c. You cannot set FORMAT to legacy when you specify the SYSBACKUP or the SYSDG argument.
- **SYSBACKUP:** If it is set to y, it creates a SYSBACKUP user entry in the password file, and you are prompted for the SYSBACKUP user password. The SYSBACKUP user password is stored in the created password file.
- **SYSDG:** If it is set to y, it creates a SYSDG entry in the password file, and you are prompted for the SYSDG password. The SYSDG password is stored in the created password file.
- **INPUT_FILE:** Name of the input password file. ORAPWD migrates the entries in the input file to a new password file. This argument can convert a password file from legacy format to 12 format.

It is recommended not to use the SYSBACKUP user to connect as SYSBACKUP. The DBAs that are responsible for backup or recovery would all share the same password. This is undesirable in terms of accountability. In addition, it would be impossible to take away this privilege from particular DBAs later (unless changing the password). It is better to create a database account designated for each DBA and grant the SYSBACKUP privilege if necessary. In this way, no password is shared, and SYSBACKUP can be revoked from any DBA without affecting other DBAs.

If you are working in a container database, each PDB of the CDB can create a local user and grant the user the SYSBACKUP privilege. The user can connect to the PDB and perform the backups for the PDB and only for the PDB he can connect to.

```
$ rman TARGET '"tim/passwor@pdb1 AS SYSBACKUP"'
```

Password Authentication for SYSDG



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Here is an example where the password file is created in the 12 format and therefore supports SYSBACKUP and SYSDG administrative privileges in the password file. However, SYSDG is set to n and so the SYSDG entry is not in the password file.

The connection with the DGMGRL utility and OS authentication is successful because the OS user is a member of the OSDG group.

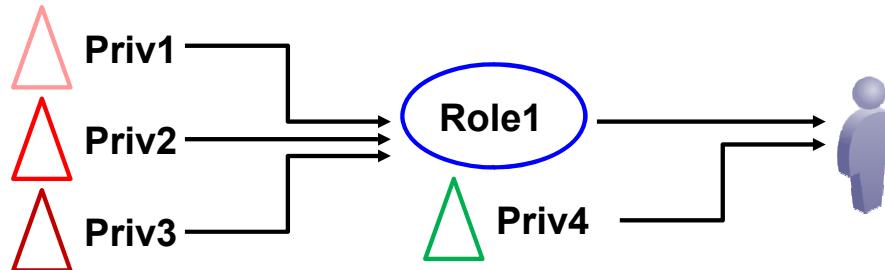
Unlike SQL*Plus, where the user specifies “AS SYSxxx” at login time, the DGMGRL command-line interface accepts only a username-password pair. To seamlessly support the SYSDG administrative privilege, the command-line interface first attempts to log in by using SYSDG. If that login attempt fails, it tries SYSDBA.

The connection with password authentication fails because the entry for the SYSDG user is not in the password file. You can create the entry in the password file by the following two methods:

- Grant the SYSDG privilege to the SYSDG user.
- Re-create the password file in the 12 format with the SYSDG argument specified as y.

Roles

- A role is a collection of privileges: system and object



- Each user can exercise granted privileges in the context of a single database: non-CDB, CDB `root` container, PDB
- Role IDENTIFIED BY *password*
- Role IDENTIFIED EXTERNALLY
- Role IDENTIFIED GLOBALLY
- Role IDENTIFIED USING *procedure*

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In most systems, it is too time-consuming to grant necessary privileges to each user individually, and doing so is too error-prone. Oracle Database provides for easy and controlled privilege management through roles. Roles are named groups of related privileges that are granted to users or to other roles. They are designed to ease the administration of privileges in the database and, therefore, improve security. The privileges can be exercised in the context of a specific database. In the past, the phrase “in the context of a single database” has been implied. When you consider that there was only one database per instance, this seems a trivial assertion, but this is key to understanding local users, local privileges, and local roles in PDBs.

Role Characteristics

- Privileges are granted to and revoked from roles in the same manner as a user.
- A role can consist of both system and object privileges.
- Roles can be granted to and revoked from users or other roles as though they were system privileges. An exception is that you cannot grant an IDENTIFIED GLOBALLY role to anything.
- A role can be enabled or disabled for each user who is granted the role.
- A role can require a password for the role to be enabled (IDENTIFIED BY *password*).
- A role can be authorized by using an external source (IDENTIFIED EXTERNALLY).
- Roles are not owned by anyone, and they are not in any schema.

Benefits of Roles

- Easier privilege management
- Dynamic privilege management
- Selective availability of privileges
- Can be granted through the operating system



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Easier Privilege Management

Use roles to simplify privilege management. Rather than granting the same set of privileges to several users, you can grant the privileges to a role, and then grant that role to each user.

Dynamic Privilege Management

If the privileges associated with a role are modified, all the users who are granted the role acquire the modified privileges automatically and immediately.

Selective Availability of Privileges

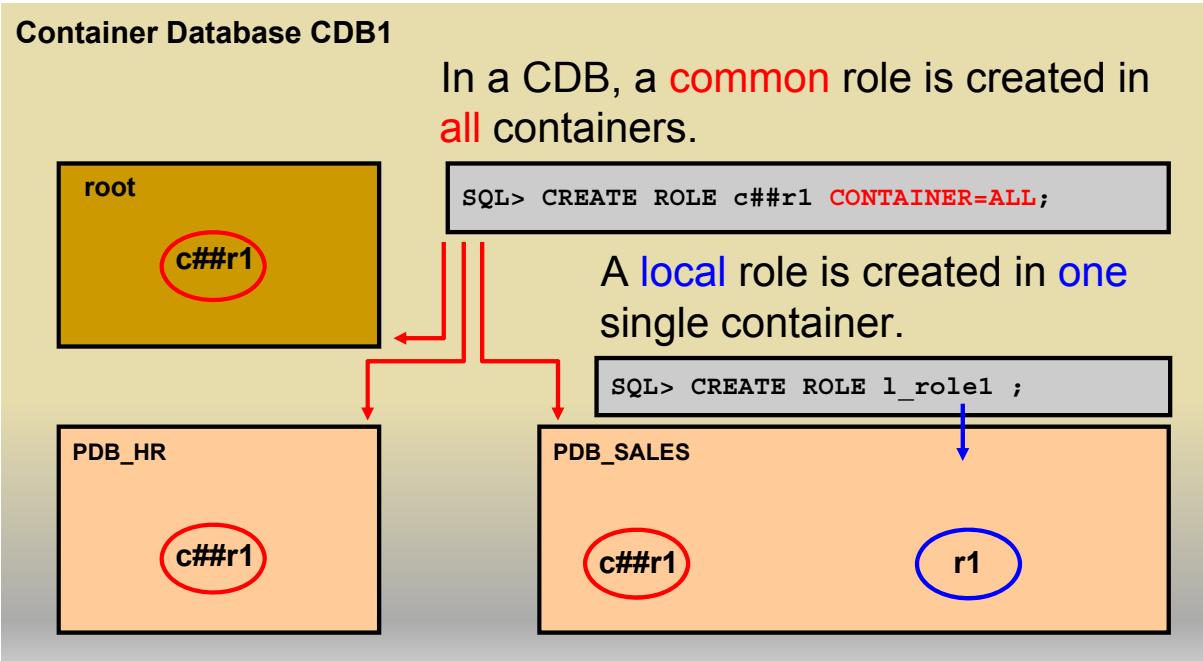
Roles can be enabled and disabled to turn privileges on and off temporarily. Enabling a role can also be used to verify that a user has been granted that role.

Granting Through the Operating System

Operating system commands or utilities can be used to assign roles to users in the database, in some operating systems.

Note: Roles are disabled in PL/SQL subprograms. The owner of a PL/SQL subprogram declared with definer's rights must have the privileges required for the subprogram granted directly and not through a role. For a subprogram with invoker's rights, roles are enabled unless the subprogram is invoked directly or indirectly from a definer's right subprogram.

Creating Common and Local Roles

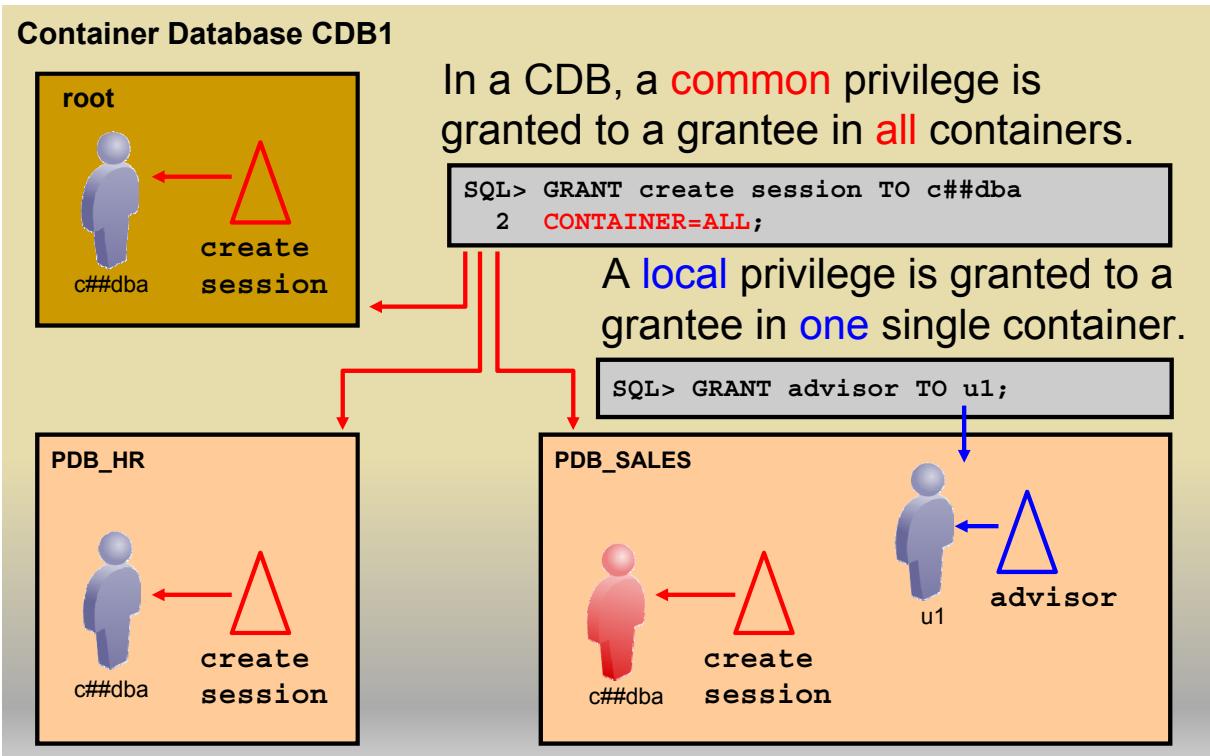


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In a CDB, a role created across all containers is a common role. In the example of the slide, the `c##r1` role is created commonly by using the `CONTAINER=ALL` clause. The create operation is replicated in all containers. Consequently, the same role `c##r1` is created in each container.

A role created in a specific PDB is local. In the example of the slide, the `l_role1` role is created locally. The create operation is not replicated in all containers. Consequently, the `l_role1` role is created in the `PDB_SALES` container only.

Granting Common and Local Privileges



ORACLE

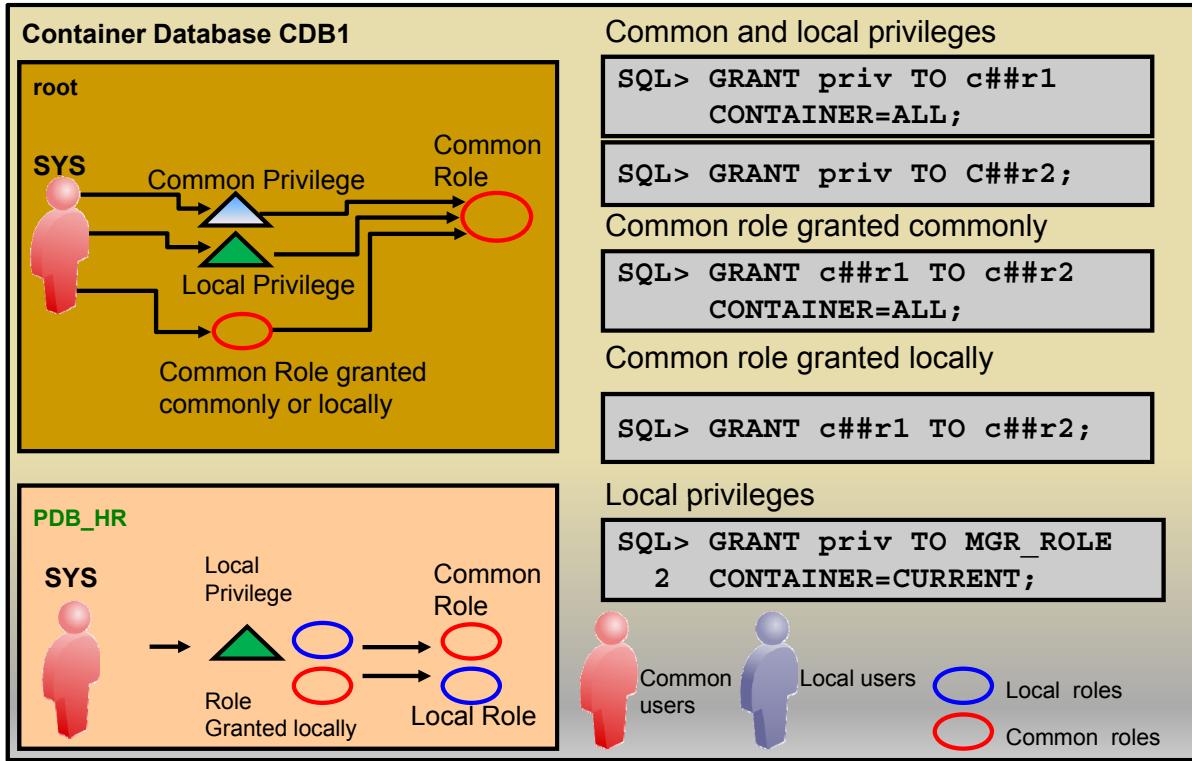
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In a CDB, a privilege granted across all containers is a common privilege. In the example of the slide, the `CREATE SESSION` privilege is granted commonly to the `c##dba` user by using the `CONTAINER=ALL` clause. The grant operation is replicated in all containers.

Consequently, the same user `c##dba` is granted the same privilege in each container.

A privilege granted in a specific PDB is local. In the example of the slide, the `ADVISOR` privilege is granted locally to the `u1` user. The grant operation is not replicated in all containers. Even if the user were a common user, the grant operation would not have been replicated. Consequently, the `u1` user is granted the privilege in the `PDB_SALES` container only.

Granting Common or Local Privileges to Roles, Common or Local Roles to Roles



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

To grant a role the grantor must have the GRANT ANY ROLE privilege or have been granted the WITH ADMIN OPTION role.

Privileges assigned to a common role are common privileges as long as CONTAINER=ALL is specified and therefore are applied in all containers. If CONTAINER=ALL is not specified, the privilege assigned to a common role is local. The first command shown in the slide shows a privilege granted commonly to a common role, and therefore usable in each of the PDBs. The second command shown in the slide shows a privilege granted locally to a common role, and therefore usable only in the PDB where the command is executed.

Common roles can be created in the root container only.

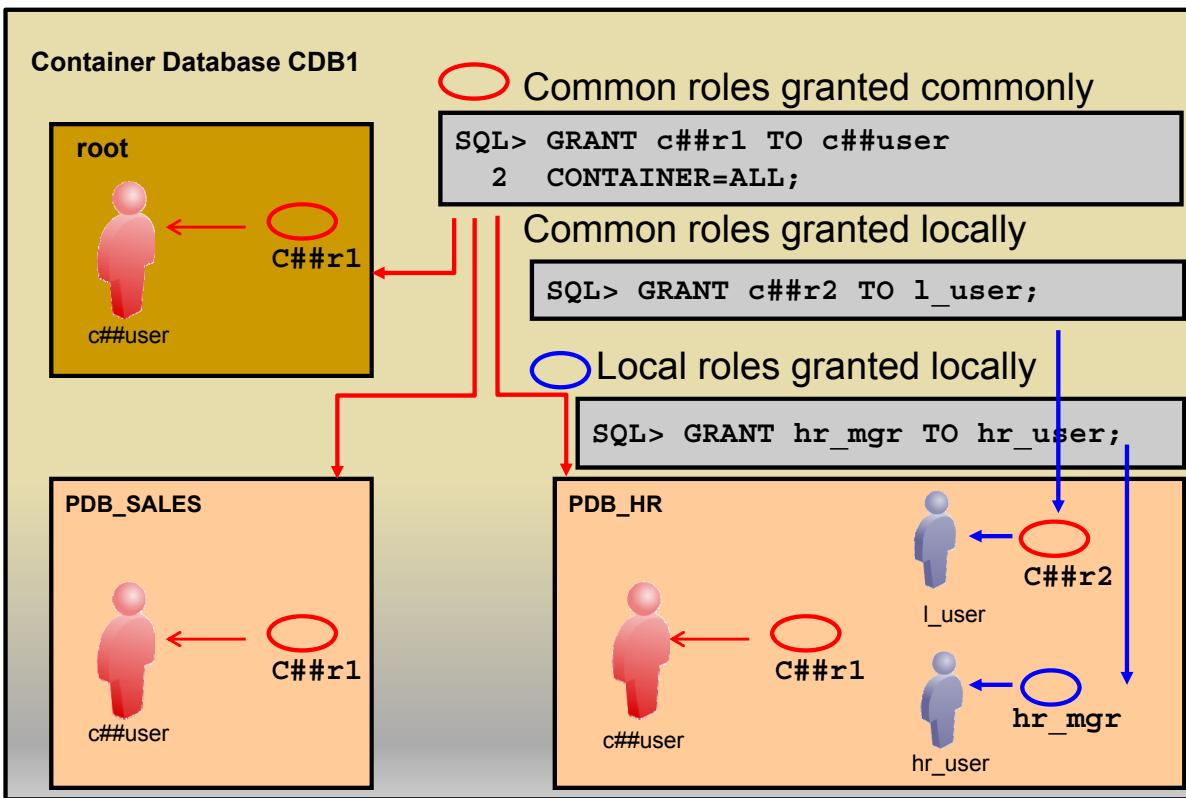
Example: A common role, C##_VIEWER, granted commonly the SELECT ANY TABLE privilege is granted to a common user C##_HR_CLERK. The SELECT ANY TABLE privilege can be used by C##_HR_CLERK while connected to any PDB.

Privileges assigned to local roles can only be local privileges. A local role can be created in a specific PDB and cannot be created in the root container. The fifth command executed in the PDB_HR container shows a privilege being granted locally to the MGR_ROLE role.

Common roles may be granted to any user or role commonly or locally in the root. Local and common roles can be granted to any user or role locally only in any PDB.

A local user can be assigned a common role, and the privileges assigned by that role can only be exercised in the PDB where the local user is defined.

Granting Common and Local Roles to Users



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Common roles may be granted to any user or role in any PDB. Local roles may be granted to any user or role that exists in the same PDB. In the first example, the common role, C##R1, is granted to the common user C##USER. This will grant the role in all the containers.

A local role can be granted to a common user and a common role to a local user. A privilege granted directly or through a role is exercised in the context of a single PDB. For example, a common role C##_VIEWER, containing the SELECT ANY TABLE privilege is granted to a common user C##_HR_CLERK. The SELECT ANY TABLE privilege can be used by C##_HR_CLERK while connected to any PDB. A local role LR_HR in the PDB_HR PDB has the INSERT INTO HR.EMPLOYEES privilege and is granted to C##_HR_CLERK. C##_HR_CLERK can use the INSERT INTO HR.EMPLOYEES only while connected to PDB_HR.

A local user can be assigned a common role, and, as before, the privileges assigned by that role can only be exercised in the PDB where the local user is defined. If the local role LR_HR is granted to the C##USER, the C##USER would be able to exercise the INSERT INTO HR.EMPLOYEES privilege only while connected to PDB_HR.

Predefined Roles

CONNECT	CREATE SESSION
RESOURCE	CREATE TABLE, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TRIGGER, CREATE TYPE, CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR
SCHEDULER_ADMIN	CREATE [ANY] JOB, CREATE EXTERNAL JOB, EXECUTE ANY CLASS, EXECUTE ANY PROGRAM, MANAGE SCHEDULER, CREATE CREDENTIAL, CREATE ANY CREDENTIAL
DBA	Most system privileges, several other roles
SELECT_CATALOG_ROLE	SELECT and EXECUTE privileges on SYS and other schema objects
AUDIT_ADMIN	AUDIT ANY, AUDIT SYSTEM
AUDIT_VIEWER	Essentially SELECT privilege on audit views



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Several roles are defined automatically for Oracle databases when you execute database creation scripts. CONNECT is granted automatically to any user who is created with Oracle Enterprise Manager. The DBA role includes nearly all privileges and should not be granted to nonadministrators.

Functional Roles

Other roles that authorize you to administer special functions are created when that functionality is installed. For example, XDBADMIN contains the privileges that are required to administer the XML database if that feature is installed. AQ_ADMINISTRATOR_ROLE provides privileges to administer Advanced Queuing. HS_ADMIN_ROLE includes the privileges needed to administer heterogeneous services. DV_OWNER includes the privileges needed to administer Database Vault components such as realms, factors, rules, and command rules. DV_ACCTMGR includes the privileges needed to manage accounts and profiles in a database protected by Database Vault. You must not alter the privileges granted to these functional roles without the assistance of Oracle Support, because you may inadvertently disable the needed functionality.

Quiz

Which of the following are characteristics of a database role?

- a. Consists of only system privileges
- b. Consists of only object privileges
- c. Is a named set of privileges
- d. Can consist of other roles



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: c, d

Secure Application Role

- The secure application role solves the problem of preventing unauthorized access to data through other client programs.
- It uses the same `SYS_CONTEXT` mechanism as Virtual Private Database (VPD).
- Enabling a role is checked through a package, and not a password.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Often, users are authorized to access certain objects only through an application that can control and check their actions.

The secure application role is enabled only through a package. The server checks the calling stack to ensure that the proper package is being used, so that it cannot be spoofed. To enable a secure application role, the associated procedure must be called, which controls whether the role is enabled. The procedure can make additional checks and get information about the user's environment by using a call to `SYS_CONTEXT('USERENV', nnn)`, where `nnn` can be, for example, `IP_ADDRESS` or `PROXY_USER`.

The `EXECUTE` privilege is granted on the package only to the application. Therefore, even if users know the details of the package, they cannot enable the role, except through the application.

Note: Refer to *Oracle Database Vault Administrator's Guide 12c Release 1 (12.1)* for information about Oracle Database Vault secure application roles.

Implementing a Secure Application Role

1. Create the IDENTIFIED USING *package* role.
2. Create the package that sets the role:
 - a. Create the package specification.
 - b. Create the package body defining when the role can be set using: dbms_session.set_role('role')
3. Grant the EXECUTE privilege on the package.
4. Write the application server code that sets the role by calling the package just created.

```
SQL> SELECT * FROM dba_application_roles
      WHERE ROLE = 'OE_SALES REP';

ROLE          SCHEMA     PACKAGE
-----        -----
OE_SALES REP   SEC         OE_ROLES
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Roles grant authorizations to the authenticated user. You may require additional validation. For example, access may be restricted by time of day or client IP address. Because a secure application role is a role implemented by a package, the package can perform the desired validation. Besides limiting the privileges of the application server, the secure application role prevents users from accessing data outside an application. They are forced to work within the framework of the application privileges that they have been granted.

To create a secure application role, perform the following steps:

1. Create the secure application role using the IDENTIFIED USING *package* clause. The CREATE ROLE command identifies the package that sets the role.
2. Create the package that sets the role. Because procedures with definer's rights always execute with the privileges of the owner, secure application roles can be enabled only inside procedures with invoker's rights. The AUTHID CURRENT_USER clause defines a procedure with invoker's rights, only applicable to packages and stand-alone procedures and functions.
3. Grant the EXECUTE privilege on the package so that the application server can set the role for the application user.
4. Write the application server code that calls a procedure in the package referenced in the CREATE ROLE command, thus setting the role for a user. The procedure is called after the application server establishes a session for the user. Do not grant the secure role to the user. If the role is granted to the user, it is enabled at login if the default role is set to all.

Securing Objects with Procedures

- Object access can be strictly controlled through procedures.
- The object owner creates procedures and functions to access the object.
- Users are granted the EXECUTE privileges on program units.
- Users do not have direct access to objects.



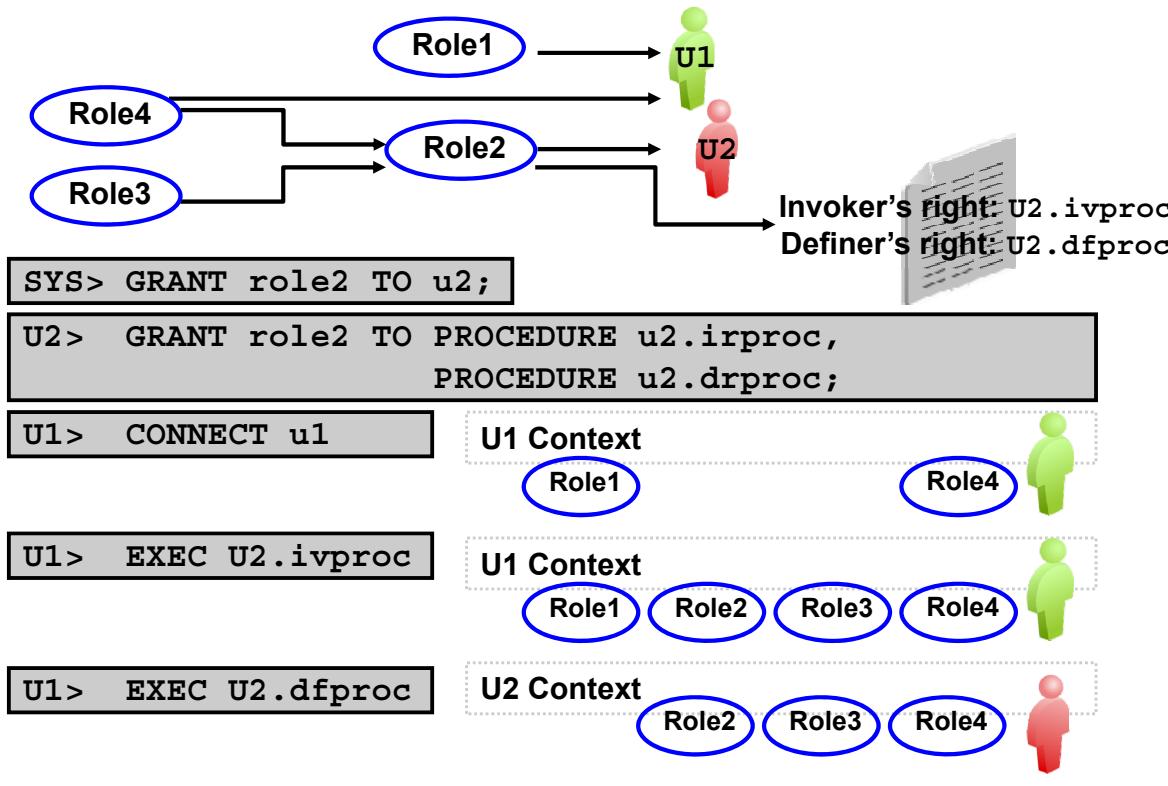
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When the built-in authorization mechanisms are not sufficient, an object-oriented approach is useful. Program units, procedures, and functions in the database, by default, run with definer's rights. This means that the program unit executes with the privileges of the owner of the unit.

Example: The HR application owner has a MEDICAL_HISTORY table with very strict access controls. The HR user creates PL/SQL procedures and functions to implement the allowed access. HR can grant the EXECUTE privilege on these program units to authorize certain users to perform only the actions that are implemented in the procedures. Users can access the table only through the procedures, because they have not been granted any other object privileges. The procedures successfully access the table because they execute with the privileges of HR.

Program units can also execute with invoker's rights. In this case, the procedure executes with only the privileges of the user that executes the procedure. This is useful for procedures that are very general and owned by SYS. An example is the DBMS_LOB package that provides the LOB access routines that may be called by any user. The invoker must have privileges to access the LOB.

Using Code-Based Access Control



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Pitfalls

- Programs traditionally make use of definer's rights to temporarily elevate the privileges of the program.
- Definer's rights-based program units run in the context of the definer's rights of the program unit, as opposed to the invoker's context.

Solutions

- Code-Based Access Control (CBAC) enables you to attach database roles to a PL/SQL function, procedure, or package. Use the syntax as follows:

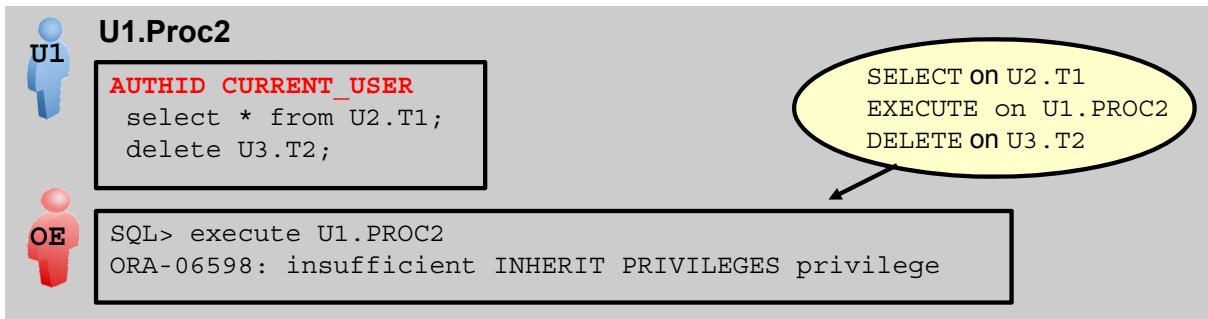
```
GRANT cb2, cb3 TO FUNCTION SCOTT.func2, PACKAGE SYS.pack2;
```
- These database roles are enabled at run time, enabling the program unit to execute with the required privileges in the calling user's environment, eliminating the need to grant these roles directly to the runtime users.

Conditions

CBAC roles can be granted to a program unit if all of the following conditions are met:

- The grantor is user `SYS` or owns the program unit.
- The roles to be granted are directly granted roles to the owner.
- The roles to be granted are standard database roles.

Privilege Checking During PL/SQL Calls



- Additional required privilege checking at run time:
 - Of a database user passing into an AUTHID CURRENT_USER PL/SQL routine
 - Of a database user passing through an AUTHID CURRENT_USER “callspec” over a C or Java routine
- INHERIT PRIVILEGES object privilege
- INHERIT ANY PRIVILEGES system privilege

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Risks

A low privilege user could own an invoker's rights procedure that could potentially perform unintended or malicious actions if it is executed by a high-privileged user.

An invoker's right procedure can perform inappropriate actions if it is invoked by another procedure that does not expect an invoker's rights procedure.

In Oracle Database 11g, the caller to a procedure had no control over who accessed the caller's privileges. Only the owner of the procedure controlled the right's inheritance.

New Privilege Checking

Privilege checking in Oracle Database 12c implements a new restriction, not a new power. Existing cases that did not require a privilege check now require one. When a user runs an invoker's rights procedure, Oracle Database checks the procedure owner's privileges before initiating or running the code. The owner must have the INHERIT PRIVILEGES object privilege on the invoking user or the INHERIT ANY PRIVILEGES privilege. If this is not the case, the runtime system raises an error.

The session is temporarily switched into an environment that treats the entered routine as the definer's rights. It then checks that it has the INHERIT PRIVILEGES object privilege on the caller's active current user or that it has the INHERIT ANY PRIVILEGES system privilege. The session then reverts to its prior environment. This treatment of the routine as definer's rights mirrors the treatment of the routine during compilation.

INHERIT (ANY) PRIVILEGES Privileges

- Invoking users can control who can access their privileges when they run an invoker's rights procedure.
- Invoking users grant the INHERIT PRIVILEGES object privilege only to **trusted users** (object is a **USER**).

```
SQL> CONNECT oe
SQL> grant INHERIT PRIVILEGES ON USER oe TO u1;
```

```
SQL> select PRIVILEGE, TYPE, TABLE_NAME, GRANTEE
  2  from DBA_TAB_PRIVS where grantee='U1';
```

PRIVILEGE	GRANTEE	TYPE	TABLE_NAME	GRANTEE
INHERIT PRIVILEGES	USER	OE		U1

- Newly created users are granted INHERIT PRIVILEGES object privilege on themselves through PUBLIC.
- INHERIT ANY PRIVILEGES means on all users.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

INHERIT PRIVILEGES Object Privilege

The benefit of this privilege is that it gives invoking users control over who can access their privileges when they run an invoker's rights procedure.

```
SQL> CONNECT invoking_user
SQL> GRANT INHERIT PRIVILEGES ON USER inv_user TO proc_owner;
```

By default, when a CREATE USER creates a new database-defined user, INHERIT PRIVILEGES privilege on that user is made PUBLIC, with the user itself listed as the grantor.

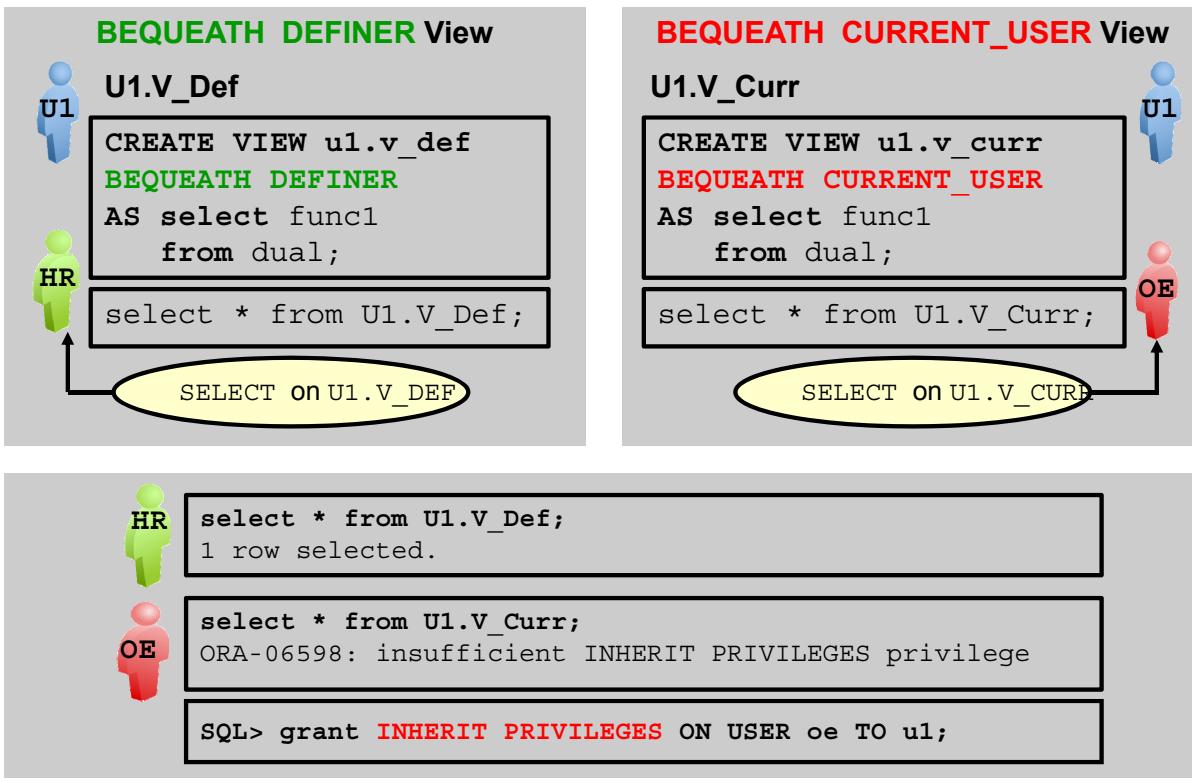
```
SQL> select PRIVILEGE, TYPE, TABLE_NAME, GRANTEE
  2  from DBA_TAB_PRIVS where type='USER' and table_name='X';

PRIVILEGE          TYPE      TABLE_NAME      GRANTEE
-----          -----      -----      -----
INHERIT PRIVILEGES    USER            X        PUBLIC
```

INHERIT ANY PRIVILEGES System Privilege

A user being granted the INHERIT ANY PRIVILEGES system privilege inherits privileges on all users.

Privilege Checking with New BEQUEATH Views



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

BEQUEATH Views

Oracle Database 12c introduces BEQUEATH CURRENT_USER views, which bring a security benefit to objects of this type similar to stored PL/SQL units. These views partially behave like invoker's rights rather than owner's rights. That is, when you call an AUTHID CURRENT_USER function or an invoker's rights PL/SQL or Java function, the current schema, current user, and currently enabled roles within the operation's execution can be inherited from the querying user's environment. BEQUEATH CURRENT_USER views are only a subset of the behavior of invoker's rights in this release.

The BEQUEATH CURRENT_USER views are in contrast to the BEQUEATH DEFINER behavior of existing views.

The BEQUEATH view type is displayed in a new BEQUEATH column in DBA_VIEWS.

INHERIT PRIVILEGES and BEQUEATH CURRENT_USER Views

The owner of a BEQUEATH CURRENT_USER view must have the INHERIT PRIVILEGES object privilege on the invoking user or the INHERIT ANY PRIVILEGES system privilege.

Quiz

Select the statement that is true when you execute an invoker's rights procedure.

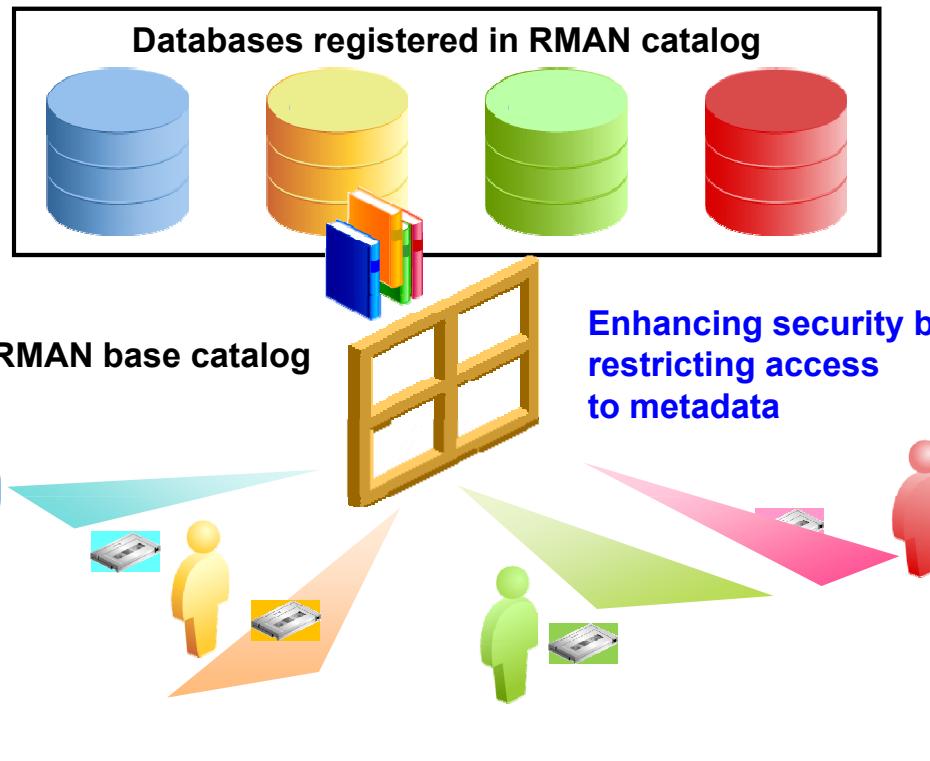
- a. The calling user must be granted the INHERIT PRIVILEGES object privilege on the user owner of the procedure.
- b. The calling user must grant the INHERIT PRIVILEGES object privilege on the user owner of the procedure.
- c. The owner of the procedure must grant the INHERIT PRIVILEGES object privilege on the calling user.
- d. The owner of the procedure must be granted the INHERIT PRIVILEGES object privilege on the calling user.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: d

Creating and Using Virtual Private Catalogs



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This feature allows a consolidation of RMAN repositories and maintains a separation of responsibilities, which is a basic security requirement.

The RMAN catalog has been enhanced to create virtual private RMAN catalogs for groups of databases and users. The catalog owner creates the base catalog and grants the RECOVERY_CATALOG_OWNER privilege to the owner of the virtual catalog. The catalog owner can either grant access to a registered database or grant the REGISTER privilege to the virtual catalog owner. The virtual catalog owner can then connect to the catalog for a particular target or register a target database. After this configuration, the VPC owner uses the virtual private catalog just like a standard base catalog.

As catalog owner, you can access all the registered database information in the catalog. You can list all databases registered with the SQL*Plus command:

```
SELECT DISTINCT db_name FROM DBINC;
```

The virtual catalog owner can see only those databases for which privileges have been granted. For most RMAN operations, you additionally need the SYSBACKUP, SYSDBA, or SYSOPER privileges on the target database.

You create virtual private RMAN catalogs for groups of databases and users.

1. The catalog owner creates the base catalog.
2. The DBA on the catalog database creates the user that will own the virtual private catalog (VPC) and grants him or her the RECOVERY_CATALOG_OWNER privilege.
3. The base catalog owner can grant access for previously registered databases to the VPC owner or grant REGISTER to the VPC owner.

The GRANT CATALOG command is GRANT CATALOG FOR DATABASE prod1, prod2 TO vpcowner;

The GRANT REGISTER command is GRANT REGISTER DATABASE TO vpcowner;

4. The virtual catalog owner can then connect to the catalog for a particular target or register a target database. After the VPC is configured, the VPC owner uses it just like a standard base catalog. Create a virtual private catalog.
 - a. If the target database is an Oracle Database 11g database and the RMAN client is an 11g client, you can use the RMAN command: CREATE VIRTUAL CATALOG;
 - b. If the target database is Oracle Database 10g Release 2 or earlier (using a compatible client), you must execute the supplied procedure from SQL*Plus: base_catalog_owner.dbms_rcvcat.create_virtual_catalog;
5. Connect to the catalog using the VPC owner login, and use it as a normal catalog.
6. The virtual catalog owner can see only those databases that have been granted. For most RMAN operations, you additionally need the SYSDBA or SYSOPER privileges on the target database.

Summary

In this lesson, you should have learned how to:

- Use database administrative privileges to maintain separation of duties
- Describe system and object privileges
- Describe the different types of roles
- Implement database roles
- Implement common and local roles
- Use secure application roles
- Use procedures to secure objects
- Use Code-Based Access Control for applications
- Secure RMAN catalogs creating Virtual Private Catalogs



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- 9-1: Exploring DBA privileges in non-CDB and in CDBs
- 9-2: Granting the SYSBACKUP administrative privilege and viewing password file contents
- 9-3: Implementing a secure application role
- 9-4: Using CBAC to enable roles at run time
- 9-5: Using the INHERIT PRIVILEGES privilege to execute an invoker's right procedure (*Optional*)
- 9-6: Using the INHERIT PRIVILEGES privilege and BEQUEATH current_user views (*Optional*)
- 9-7: Managing local and common privileges and roles in CDB/PDBs



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

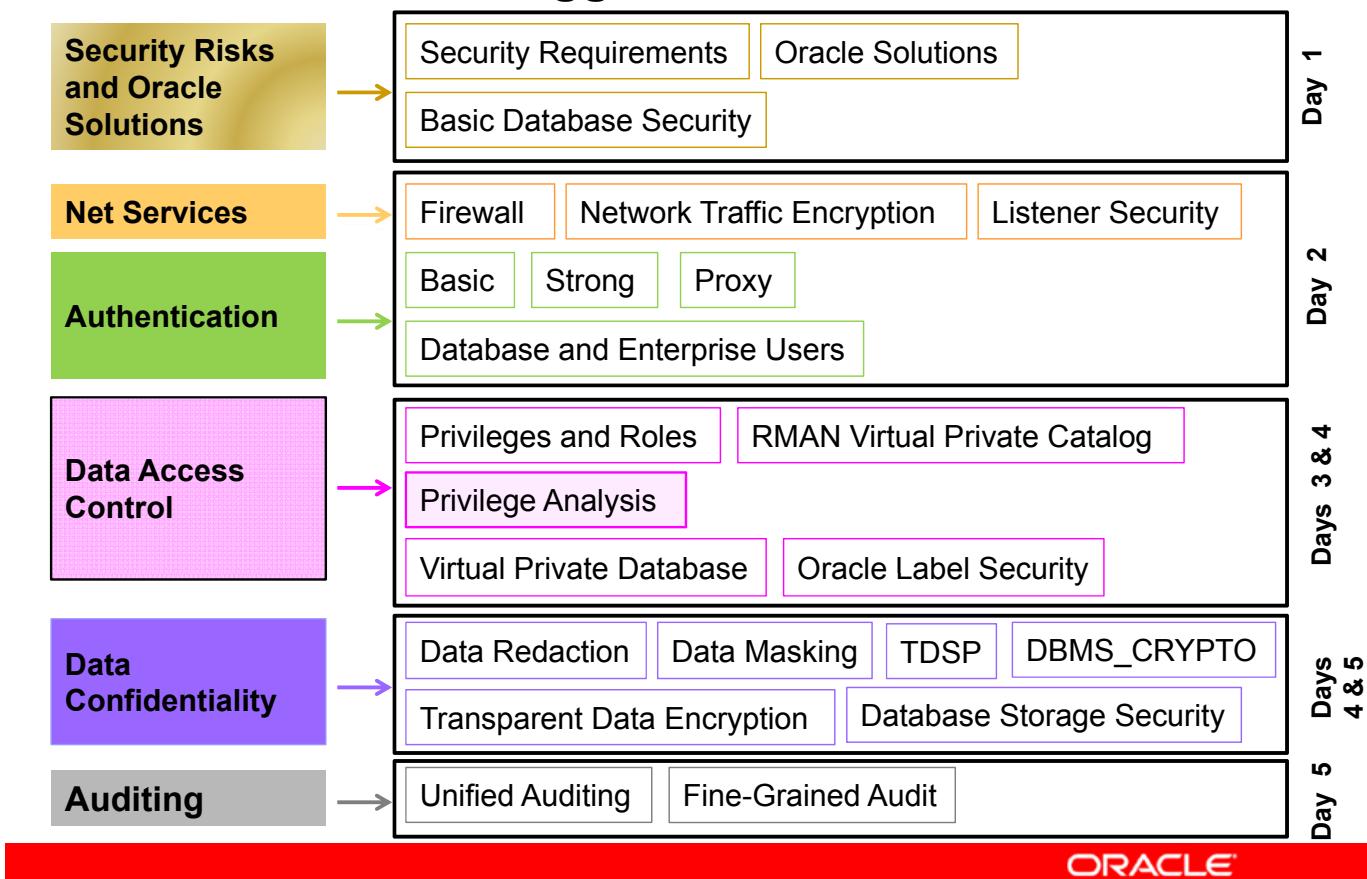
10

Using Privilege Analysis

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Schedule



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Use database privilege analysis
- Configure different types of captures
- Enable a privilege capture
- Generate results
- Analyze results
- View capture information



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

For a complete understanding of privilege analysis usage, refer to the following guide in the Oracle documentation:

- *Oracle Database Vault Administrator's Guide 12c Release 1 – “Performing Privilege Analysis to Find Privilege Use”*
- *Oracle Database PL/SQL Packages and Types Reference 12c Release 1 (12.1) – “DBMS_PRIVILEGE_CAPTURE” chapter*

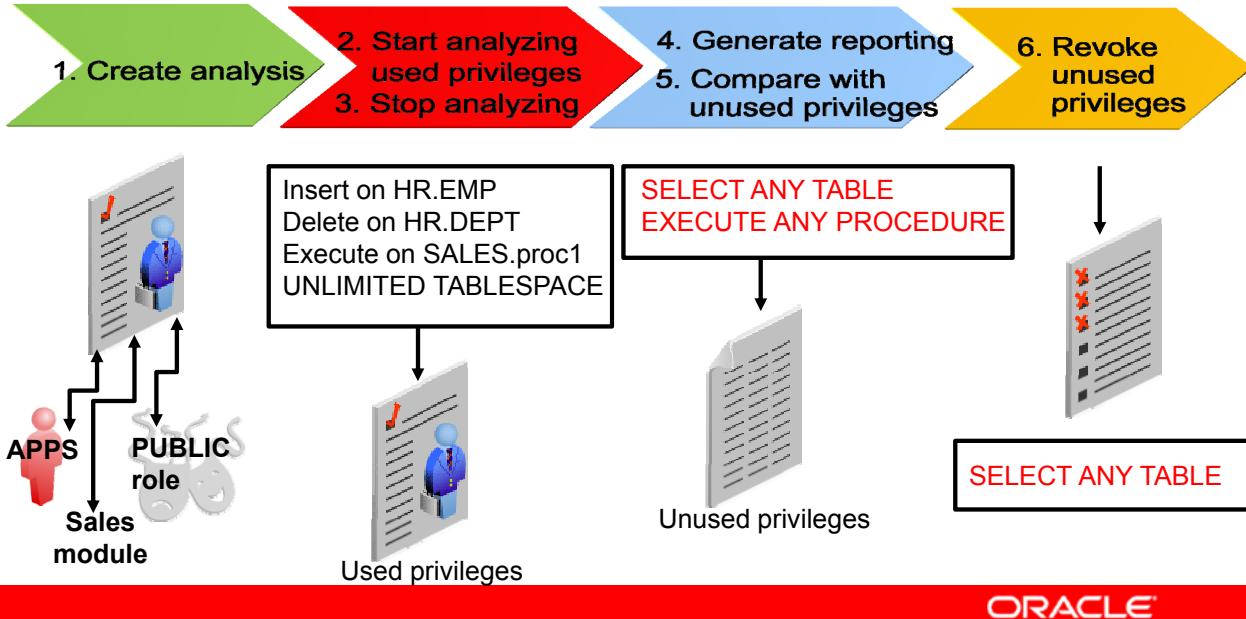
Refer to other sources of information available under Oracle Learning Library:

- *Oracle By Example (OBE)*:
 - *Determining Least Privilege Access Using Privilege Analysis*

Privilege Analysis

Increase database security: Revoke unused privileges.

- Analyze used privileges to revoke unnecessary privileges.
- Use the new DBMS_PRIVILEGE_CAPTURE package.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Excessive Privileges Challenge

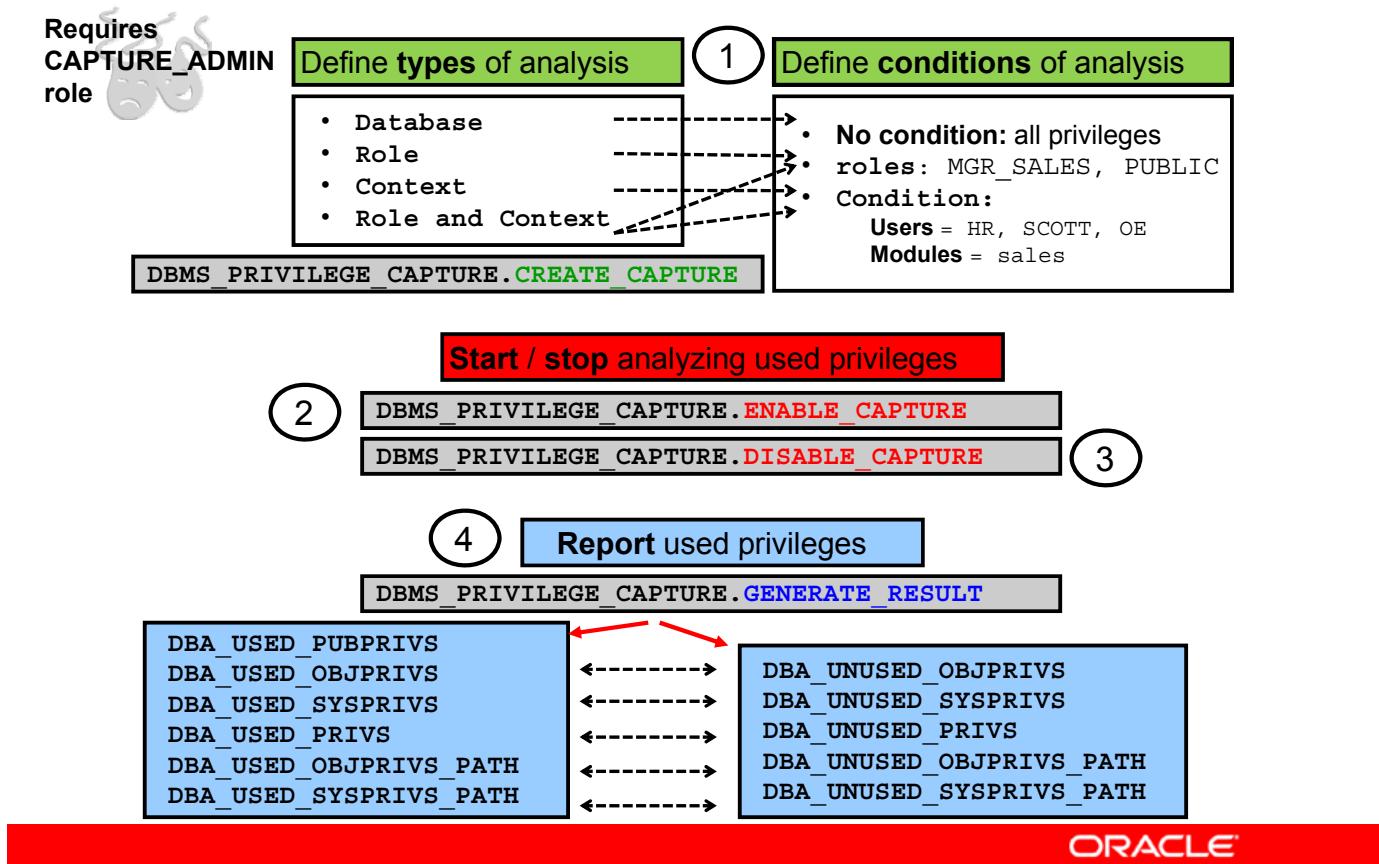
In addition to concern about the SYSDBA privilege usage, another major concern is that existing database and application users have excessive privileges. Excessive privileges violate the principle of least privilege. To achieve the least privilege principle, unused privileges need to be identified.

Privilege Analysis

Oracle Database 12c offers a new package to analyze used privileges.

- You can use a privilege analysis policy to identify object and system privileges used to run an application module or to execute certain SQL statements or privileges used by defined roles.
- You can generate reports of used and unused privileges during the analysis period.
- The report helps the security officer revoke unnecessary privileges by comparing the used and unused granted privileges lists.

Privilege Analysis Flow



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Privilege Analysis Types and Conditions

When creating an analysis policy, you first define the targeted objects to be analyzed in used privileges. You do that by setting the type of analysis:

- **Database analysis:** If no condition is given, it analyzes the used privileges (except privileges used by administrative users) within the whole database.
- **Role analysis:** If roles are defined, it analyzes the privileges exercised through any given role. For example, if you create a privilege analysis policy to analyze on PUBLIC, the privileges that are directly and indirectly granted to PUBLIC are analyzed when they are used.
- **Context-specific analysis:** If the contexts are defined, it analyzes the privileges that are used through a given application module or specified contexts.

Different conditions can be combined with “AND” and/or “OR” Boolean operators.

Because the created policy is not enabled by default, the next step is to enable the policy to start analyzing used privileges. After a certain time, you stop analyzing.

Reporting Used Privileges

The third step is to generate a report. Reporting includes two types of results:

- Used privileges visible in DBA_USED_xxx and DBA_USED_xxx_PATH views
- Unused privileges visible in DBA_UNUSED_xxx and DBA_UNUSED_xxx_PATH views

Creating Policies: Database and Role Analysis

1. Create a policy to analyze used privileges.
 - Create a database analysis policy.



Create analysis

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
  2       name          => 'All_privs', -
  3       description   => 'Captures all privilege use', -
  4       type          => dbms_privilege_capture.g_database);
```

- Create a role analysis policy.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
  2       name          => 'Public_privs_capture', -
  3       description   => 'Privileges used by PUBLIC', -
  4       type          => dbms_privilege_capture.g_role, -
  5       roles         => role_name_list('PUBLIC'))
```

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
  2       name          => 'Audit_privs_capture', -
  3       description   => 'Privileges used by audit roles', -
  4       type          => dbms_privilege_capture.g_role, -
  5       roles         => role_name_list('AUDIT_ADMIN', 'AUDIT_VIEWER'))
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The PL/SQL package used to create a privilege analysis policy is `SYS.DBMS_PRIVILEGE_CAPTURE`. To create a privilege analysis policy, use the procedure `CREATE_CAPTURE`.

- To create a database analysis policy, set the `TYPE` argument to the `dbms_privilege_capture.g_database` value.
- To create a role analysis policy, set the `TYPE` argument to `dbms_privilege_capture.g_role` and the `ROLES` argument to the list of the roles to be analyzed.

The second example in the slide shows how to set a policy to analyze all privileges that are used by `PUBLIC`.

The third example shows how to set a policy to analyze all privileges that are used through the `AUDIT_ADMIN` and `AUDIT_VIEWER` roles.

Creating Policies: Context Analysis

Create a context analysis policy.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -  
2      name    => 'Prvs_HR_OE_logged_users', -  
3      description => 'All privileges used by HR,OE', -  
4      type    => dbms_privilege_capture.g_context, -  
5      condition => -  
6          'SYS_CONTEXT(''USERENV'', ''SESSION_USER'')='''HR''' -  
7      OR -  
8          'SYS_CONTEXT(''USERENV'', ''SESSION_USER'')='''OE'''')
```

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -  
2      name    => 'Prvs_AcctPayable_capture', -  
3      description => 'All privileges used by module', -  
4      type    => dbms_privilege_capture.g_context, -  
5      condition => 'SYS_CONTEXT -  
6      (''USERENV'', ''MODULE'')='''Account Payable'''')
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To create a context analysis policy, set the TYPE argument to the dbms_privilege_capture.g_context value. In addition, set a value of the analyzed contexts in the CONDITION argument.

The first example in the slide shows how to set a policy to analyze all privileges used by the HR and OE connected users. Notice that two conditions are combined with the OR operator.

The second example shows how to set a policy to analyze all privileges that are used when you run the Account Payable module.

Creating Policies: Combined Analysis Types

Create a policy combining two analysis types.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
  2      name    => 'Prvs_context_role', -
  3      description => 'Captures Context and role', -
  4      type => dbms_privilege_capture.g_role_and_context, -
  5      roles   => role_name_list('PUBLIC')
  6      condition => 'SYS_CONTEXT -
  7 (''USERENV'', ''MODULE'')='Account Payable''')
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To create a role and context analysis type policy, set the `TYPE` argument to the `dbms_privilege_capture.g_role_and_context` value. In addition, define values for the `ROLES` argument and `CONDITION` for the context values.

The example in the slide shows how to set a policy to analyze all privileges that are used by `PUBLIC` when you run the Account Payable module.

Analyzing and Reporting



2. Start the analysis of used privileges.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE ( -  
2      name      => 'All_privs')
```

3. After some time, stop analyzing.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ( -  
2      name      => 'All_privs')
```

4. Generate the report.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT ( -  
2      name      => 'All_privs')
```



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Capturing Used Privileges

To start the analysis, use the `ENABLE_CAPTURE` procedure.

Reporting

To generate the results of the started analysis, first stop the analysis by using the `DISABLE_CAPTURE` procedure, and then use the `GENERATE_RESULT` procedure.

SYSTEM and OBJECT Used Privileges

Compare with
granted unused
privileges

- View SYSTEM privileges used during the entire analysis.

```
SQL> select USERNAME, SYS_PRIV from DBA_USED_SYSPRIVS;

USERNAME      SYS_PRIV
-----
TOM           CREATE SESSION
OE            UPDATE ANY TABLE
OE            CREATE SESSION
JIM           CREATE SESSION
```

- View OBJECT privileges used during the entire analysis.

```
SQL> select USERNAME, OBJECT_OWNER, OBJECT_NAME, OBJ_PRIV
  2  from DBA_USED_OBJPRIVS where username in ('JIM','TOM');

USERNAME      OBJECT_OWNER OBJECT_NAME          OBJ_PRIV
-----
JIM           SYS          DBMS_APPLICATION_INFO EXECUTE
JIM           HR           EMPLOYEES             DELETE
TOM           SH           SALES                 SELECT
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Reporting by Using Dictionary Views

When you generate the analysis results, SYSTEM and OBJECT used privileges populate views. To display all the following DBA_xxx views shown in the following slides requires the CAPTURE_ADMIN role, which grants SELECT on these views.

- DBA_USED_SYSPRIVS: The first example in the slide shows that TOM, JIM, and OE connected to the database by using the CREATE SESSION system privilege, and OE updated a table by using the UPDATE ANY TABLE system privilege.
- DBA_USED_OBJPRIVS: The second example shows that JIM executed the SYS.DBMS_APPLICATION_INFO procedure by using the EXECUTE object privilege, and he deleted rows from the HR.EMPLOYEES table by using the DELETE object privilege. TOM selected rows from the SH.SALES table by using the SELECT object privilege.

Used Privileges Results

- View SYSTEM and OBJECT used privileges:

```
SQL> select USERNAME, SYS_PRIV, OBJ_PRIV, OBJECT_OWNER, OBJECT_NAME
  2  from DBA_USED_PRIVS ;
```

USERNAME	SYS_PRIV	OBJ_PRIV	OBJECT_OWNER	OBJECT_NAME
JIM		SELECT	HR	EMPLOYEES
JIM		DELETE	HR	EMPLOYEES
TOM		CREATE SESSION		
TOM		SELECT	SH	SALES
OE		UPDATE	HR	DEPARTMENTS

- View the path for OBJECT used privileges:

```
SQL> select USERNAME, OBJ_PRIV, OBJECT_NAME, PATH
  2  from DBA_USED_OBJPRIVS_PATH where username in ('TOM', 'JIM')
```

USERNAME	OBJ_PRIV	OBJECT_NAME	PATH
OE	UPDATE	DEPARTMENTS	GRANT_PATH('OE')
JIM	DELETE	EMPLOYEES	GRANT_PATH('JIM', 'HR_MGR')
JIM	SELECT	EMPLOYEES	GRANT_PATH('JIM', 'HR_MGR')
TOM	SELECT	SALES	GRANT_PATH('TOM', 'SALES_CLERK')

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Reporting

You can view all SYSTEM and OBJECT used privileges in the DBA_USED_PRIVS view.

If you need to know how the privileges were granted to the users, display the PATH column from DBA_USED_OBJPRIVS_PATH and DBA_USED_SYS_PRIVS_PATH.

The second example reveals the following information:

- OE updated rows from the DEPARTMENTS table because he is directly granted the OBJECT privilege UPDATE.
- JIM deleted rows from the EMPLOYEES table because he is granted the OBJECT privilege DELETE through the HR_MGR role.
- JIM selected rows from the EMPLOYEES table because he is granted the OBJECT privilege SELECT through the HR_MGR role.
- TOM selected rows from the SALES table because he is granted the OBJECT privilege SELECT through the SALES_CLERK role.

Note: If a privilege is used during the privilege analysis process and then revoked before you run the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure, the privilege is still reported as a used privilege, but without the privilege grant path.

Compare Used and Unused Privileges

Compare with
granted unused
privileges

- View SYSTEM and OBJECT **used** privileges:

```
SQL> select USERNAME, SYS_PRIV, OBJ_PRIV, OBJECT_OWNER, OBJECT_NAME
  2  from DBA_USED_PRIVS where username='JIM';

USERNAME  SYS_PRIV   OBJ_PRIV   OBJECT_OWNER   OBJECT_NAME
-----  -----  -----
JIM          SELECT      HR        EMPLOYEES
JIM          DELETE      HR        EMPLOYEES
```

- View SYSTEM and OBJECT **unused** privileges:

```
SQL> select USERNAME, OBJ_PRIV, OBJECT_NAME, PATH
  2  from DBA_UNUSED_PRIVS where username='JIM';

USERNAME  OBJ_PRIV  OBJECT_NAME    PATH
-----  -----  -----
JIM        INSERT    EMPLOYEES    GRANT_PATH('JIM', 'HR_MGR')
JIM        UPDATE    EMPLOYEES    GRANT_PATH('JIM', 'HR_MGR')
```

- Decide whether unused privileges must be revoked.

Revoke
unused
privileges

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Comparing to Revoke Unnecessary Privileges

You can view all unused SYSTEM and OBJECT privileges granted in the DBA_UNUSED_PRIVS view.

If you compare the list of used and unused privileges, you can identify the privileges that are granted but are not used, and you can decide whether to revoke the unused privileges.

For the example in the slide, JIM used the SELECT and DELETE privileges on the HR.EMPLOYEES table, and he did not use INSERT or UPDATE on the same table. The INSERT and UPDATE privileges are granted through the HR_MGR role.

Views

- List of analysis:

SQL> select NAME, TYPE, ENABLED, ROLES, CONTEXT 2 from DBA_PRIV_CAPTURES;			
NAME	TYPE	ENA	ROLES
CONTEXT			
All_privs	DATABASE	N	
Public_privs	ROLE	N	ROLE_ID_LIST(1)
HR_SH_privs	ROLE	Y	ROLE_ID_LIST(112, 113)
→ Privils_HR_OE_logged CONTEXT		N	
SYS_CONTEXT('USERENV', 'SESSION_USER') = 'HR' OR			
SYS_CONTEXT('USERENV', 'SESSION_USER') = 'OE'			
→ HR_Sales_role ROLE_AND_CONTEXT N	ROLE_ID_LIST(113)		
SYS_CONTEXT('USERENV', 'SESSION_USER') = 'HR'			

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To view the list of created analyses, use the DBA_PRIV_CAPTURES dictionary view. The TYPE column can hold four values: DATABASE, ROLE, CONTEXT, and ROLE_AND_CONTEXT. The ENABLED column is set to Y when the analysis is analyzing used privileges. The ROLES column holds the list of roles, and the CONTEXT column holds the condition. The roles and the condition are defined in the creation of the analysis policy.

If the database is a CDB, you create, start, stop, and generate reports in the container to which you are connected.

This means that an analysis collects information from the sessions of either the root or a PDB where you created and started the analysis policy. It does not collect information for all containers during an analysis.

Dropping an Analysis Policy

1. Disable the analysis policy.

```
SQL> exec dbms_privilege_capture.DROP_CAPTURE('Capture1')
BEGIN dbms_privilege_capture.DROP_CAPTURE('Capture1'); END;

*
ERROR at line 1:
ORA-47932: Privilege capture Capture1 is still enabled.
ORA-06512: at "SYS.DBMS_PRIVILEGE_CAPTURE", line 82
ORA-06512: at line 1

SQL> exec dbms_privilege_capture.DISABLE_CAPTURE('Capture1')
PL/SQL procedure successfully completed.
```

2. Drop the analysis policy.

```
SQL> exec dbms_privilege_capture.DROP_CAPTURE('Capture1')
PL/SQL procedure successfully completed.
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To drop an analysis policy, first disable the analysis policy if it was already started. Dropping an analysis policy also drops all used and unused privilege records that are associated with this analysis policy.

Quiz

To revoke unnecessary and unused privileges granted, use the Privilege Analysis. Place the steps in the proper order.

- a. Generate the results.
- b. Stop the analysis.
- c. Set up the analysis policy type (database, role, context).
- d. Start the analysis.
- e. View the results in a DBA_COMP_PRIVS comparison view.
- f. View the results in DBA_USED_PRIVS and DBA_UNUSED_PRIVS.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: c – d – b – a – f

Summary

In this lesson, you should have learned how to:

- Use database privilege analysis
- Configure different types of captures
- Enable a privilege capture
- Generate results
- Analyze results
- View capture information



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- 10-1: Creating, starting, stopping, and analyzing privilege and cleaning up capture for all users
- 10-2: Creating, starting, stopping, and analyzing privilege and cleaning up capture for roles
- 10-3: Creating, starting, stopping, and analyzing privilege and cleaning up capture for roles and contexts



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

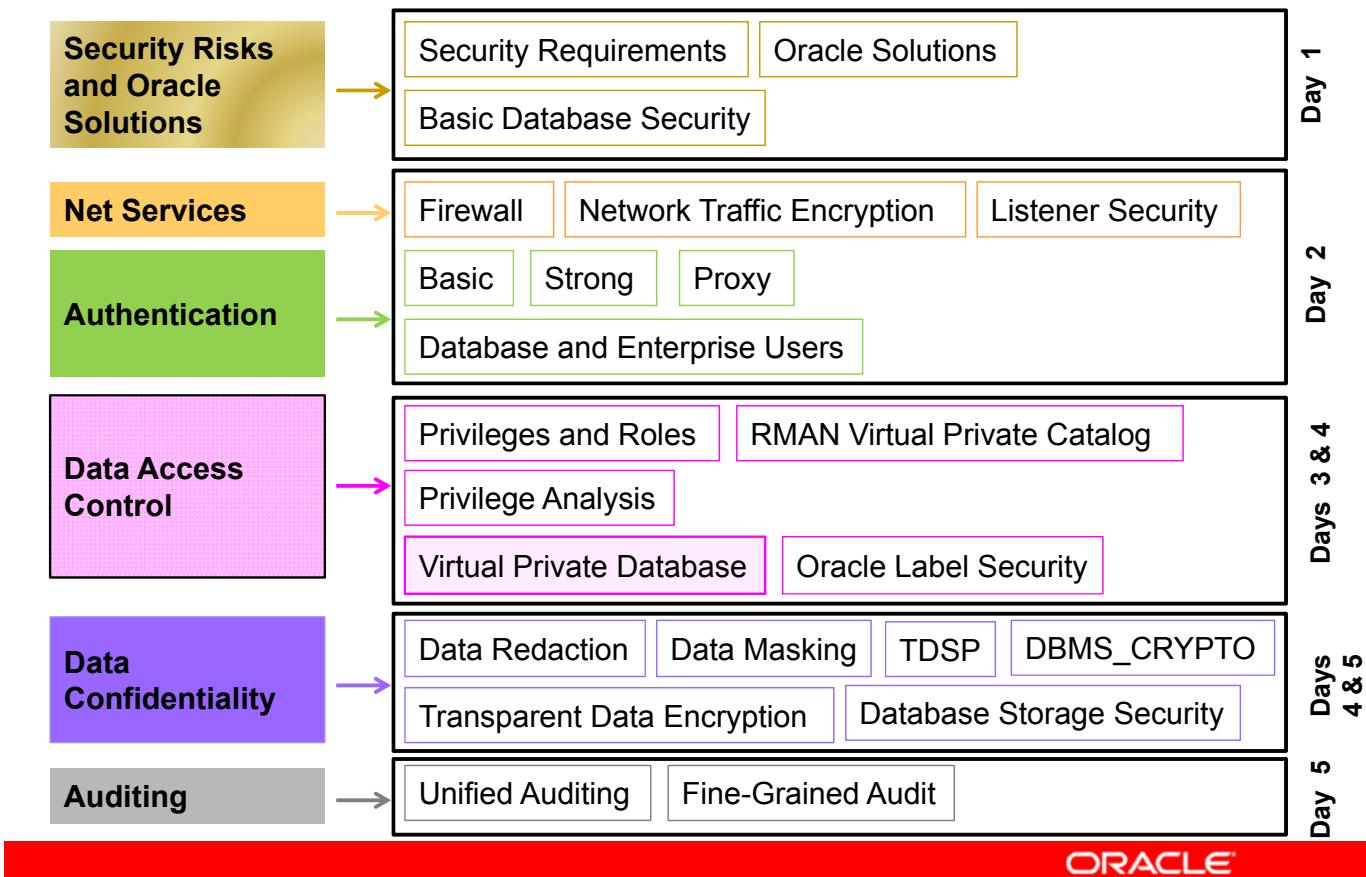
11

Using Application Contexts

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Schedule



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Use an application context
- Access the sources of application context values
- Implement an application context that is accessed globally



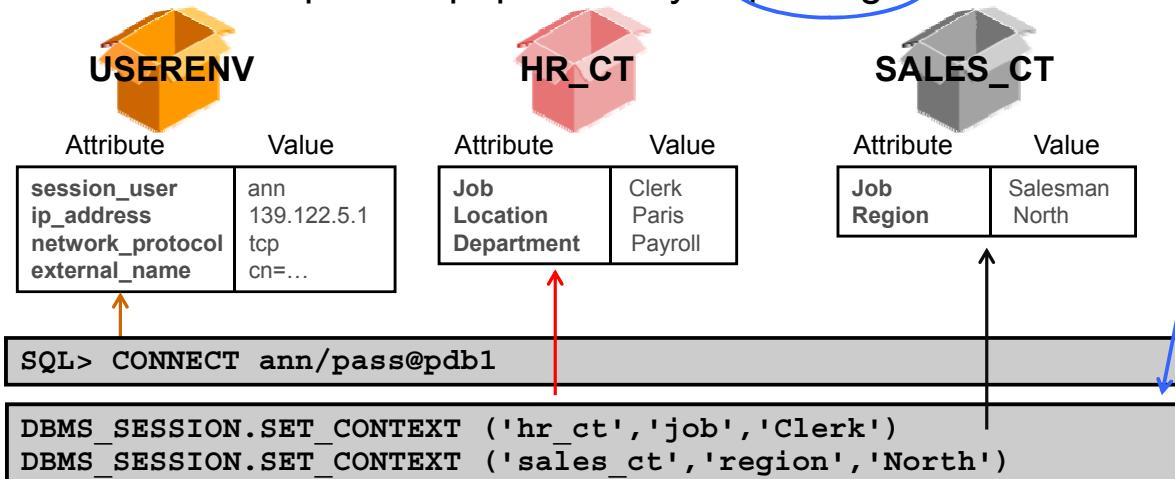
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Application Context: Description

- An application context is a memory container (namespace) with read-only attributes.

```
SQL> CREATE CONTEXT hr_ct USING pack1;
```

- Each namespace is independent of others.
- The namespace is populated by a package.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

An application context is a memory container with read-only attributes. It may be populated only by a named package. You can use these attributes explicitly or implicitly in your application. Using an application context is like writing down an often-used phone number and keeping it next to your phone, where you can find it easily, rather than looking it up every time you need it.

A context has an allocated area of memory called a *namespace*. The context or namespace has named attributes. Each attribute can hold a text string value. In some programming languages, this is known as an associative array.

The application context namespace identifies the application context. For example, you may have the HR_CT and SALES_CT namespaces that are used with the Human Resources and Sales applications, respectively. Multiple namespaces enable you to use the same attribute's name in a different namespace without interfering with other namespaces. For example, the HR_CT and SALES_CT application contexts can each have an attribute named JOB that contains a different value in each context.

A special context is the built-in USERENV context. The USERENV context is populated with values that are commonly found in the V\$SESSION and V\$PROCESS views. For a full list of the attributes of the USERENV context, see the appendix titled "USERENV Context."

Using the Application Context

An application context:

- Is read by applications
- Can be used to:
 - Authorize users
 - Limit access to data, called by a fine-grained access control policy
 - Set attributes used in the application



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Although the main benefit of using application contexts is improved performance, contexts are used with Oracle Database security features, such as Virtual Private Database (VPD) or fine-grained access control.

With application context, you can write applications that use the attributes of the application context to perform the following:

- Authorize users when they log on. For example, you can verify that the user is connecting through a specific computer by verifying the user's IP address. You would use the `USERENV` context to access the user's current IP address.
- Set context attributes that are used by fine-grained access control policies. Session properties may be used to limit the rows that the user can access. For example, in an order-entry application, customers can access only their own orders. This is accomplished by setting a context attribute with the current user's customer number. Because of the way the context is set and used, you can implement fine-grained access control without changing the application.
- Set the attributes that are used in the application. In this situation, set the attributes as part of the context, rather than accessing the attributes from a table. For example, if the user's employee number is used frequently in the application, you can create a context attribute that contains the employee number, rather than selecting the employee number from a table.

Setting and Retrieving Application Context Values

- The context attributes are set by a package, which:
 - Creates attributes in the context
 - Assigns values to the attributes of the context
 - Is usually called when a user connects (AFTER LOGON trigger)



- Each application can use one or more contexts.
- A context may be used by multiple applications.
- The context attributes values are retrieved using the SYS_CONTEXT function:

```
SELECT sys_context ('USERENV', 'ip_address') FROM dual;  
SELECT sys_context ('SYS_SESSION_ROLES', 'MGR') FROM dual;
```

```
SELECT sys_context ('HR_CT', 'job') FROM dual;
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Setting the Attributes

The application developer or security developer writes a package that assigns values to the context attributes. The attributes of the context can be set only with this package. This is the package that is associated with the context when the context is created. This package can be called by the application or a logon trigger. The package sets the attributes that are used for that session. These must be attributes that are frequently used by the application. The benefit of the context comes from caching these values in memory and not having to perform SQL or system callouts to retrieve them.

Each application can have its own application-specific context. For example, the context attributes for a human resources application can include position, organizational unit, and country, whereas context attributes for the order-entry system may be customer ID and name.

Applications can use multiple contexts, and a context may be used by multiple applications.

Retrieving the Attributes Using the sys_CONTEXT PL/SQL Function

This function returns the values of context attributes, including:

- Built-in attributes from the USERENV context, which contains session properties
- Built-in attributes from the SYS_SESSION_ROLES context, which indicates whether a specified role is currently enabled for the session (the role is case-sensitive)
- User-defined attributes from user-defined contexts

Application Context Data Sources

- A **local context** uses database objects. The developer sets these attributes.
 - Example: The `EMPLOYEE_ID` column in the `EMPLOYEES` table
- An **externalized context** can get values from an external source, such as Oracle Call Interface (OCI).
- A **global context** uses values from the directory-entry attributes.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A context can be classified by the source of the data values that it contains. The value that is assigned to the context attribute can have different sources.

Local Application Contexts

You can set attributes that are stored in a context from any database object. For example, an `EMPLOYEES` table can include cost center, title, signing authority, and other information that is useful for row level security. In addition, values returned from procedures and functions can be used. For example, use a function that returns the employee ID to assign a value to the context attribute.

Context memory accessed locally is allocated from the fixed portion of the user global area (UGA), which belongs to each server process. This memory allocation is not subject to the effects of the `PGA_AGGREGATE_TARGET` parameter.

Externalized Application Contexts

An externally initialized application context is characterized by attributes and values that are initialized through external resources, such as an OCI call, a job queue process, or a database link. It provides:

- For remote sessions, automatic propagation of context values that are in an external initialized context namespace
- For job queues, restoration of context values that are in an externally initialized context namespace
- For OCI, a mechanism to initialize context values that are in an externally initialized context namespace

Any client program can initialize this context by making OCI calls, and there are logon event triggers that can verify the values. It is up to the application to interpret and trust the values of the attributes. This type of context is created with the following command:

```
CREATE CONTEXT external USING ext_package INITIALIZED EXTERNALLY;
```

Note that with an externally initialized application context, the middle-tier server can actually initialize context values on behalf of database users. Context attributes are propagated for the remote session at initiation time, and the remote database accepts the values if the namespace is externally initialized.

Global Application Contexts

Many organizations centralize user information and user management in a Lightweight Directory Access Protocol (LDAP)-based directory, such as Oracle Internet Directory (OID). Application context attribute values can be stored in OID. This type of context is created in the database as shown in the following example:

```
CREATE CONTEXT hrgapp USING hr_g_context INITIALIZED GLOBALLY;
```

When an enterprise user connects to the database, the attributes defined in the global context of that user's OID entry are placed in the named application context. The global context named in the preceding example is HRGAPP. The attributes that are available are dependent on the attributes defined in the LDAP directory. The `SYS_CONTEXT` function can be used to access the attributes of the context as shown in the following example:

```
SYS_CONTEXT('HRGAPP', 'Title')
```

The HRGAPP context and the TITLE attribute must be added to the user's OID entry in the `OracleDBAppContext` object. For more details, see the *Oracle Database Security Guide*.

If an LDAP `inetOrgPerson` object entry exists for the user, the connection also retrieves all the attributes from `inetOrgPerson` and assigns them to the `SYS_LDAP_USER_DEFAULT` namespace as shown in the following example:

```
SYS_CONTEXT('SYS_LDAP_USER_DEFAULT', 'telephoneNumber')
```

Quiz

You must create a PL/SQL package to set an application context attribute.

- a. True
- b. False

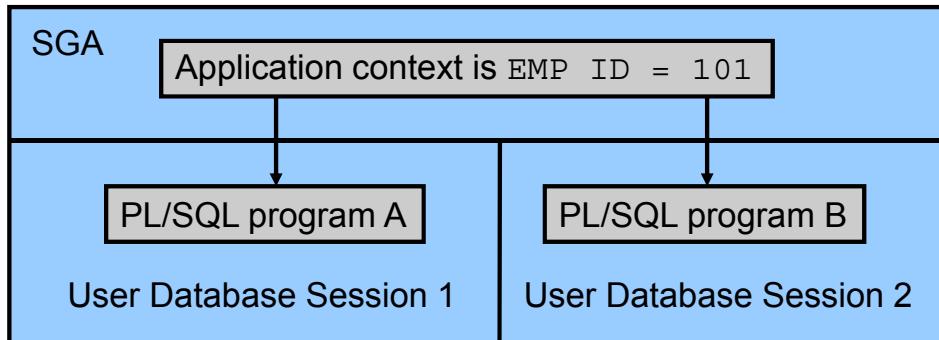


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a

Application Context Accessed Globally

- Shares a context across sessions
- Simplifies connection pooling from a middle tier
- Uses a client identifier to identify the user of a session



```
SQL> CREATE CONTEXT hr_ct_global USING hr_ct_pack  
          ACCESSED GLOBALLY;
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

By default, the memory for the application context is allocated from the session's UGA. In many application architectures, the middle-tier application is responsible for managing connection pooling for application users. Users authenticate themselves to the application. The application then uses a single identity to log in to the database and maintains all the connections. In a connection-pooling environment, usually any user can use any connection. The application decides the connection that a user request can use. In this environment, it is not possible to maintain application attributes by using session-dependent secure application context because the context is private to each session, and because of the sessionless model of the application, any user can use any session.

The application context accessed globally is a type of secure application context that can be shared among trusted sessions. It can be shared because the memory for the context is allocated from the system global area (SGA). Middle-tier applications use globally accessed application contexts to manage application attributes securely and globally. Global application contexts with connection pooling allow multiple connections to access one or more contexts, instead of setting up an application context for each user session. Globally accessed application contexts provide additional flexibility for web-based applications. They also provide enhanced performance through the reuse of common application contexts among multiple sessions, instead of setting up application contexts for each session.

Application contexts accessed globally provide performance improvements through connection reuse. These application contexts are initialized once, instead of being initialized for each session individually.

Note: The application must initialize the context.

The middle tier sets the application context for each session. The context accessed globally allows the middle tier to store the various application context definitions in a central place in the SGA and apply the context to a user session at session-creation time. This then becomes that session's driving context. This also reduces the setup time of the user session when the application is using connection pooling.

Limitation: A context accessed globally cannot be initialized from OID or an external source. Therefore, any context accessed globally must be a local context accessed globally.

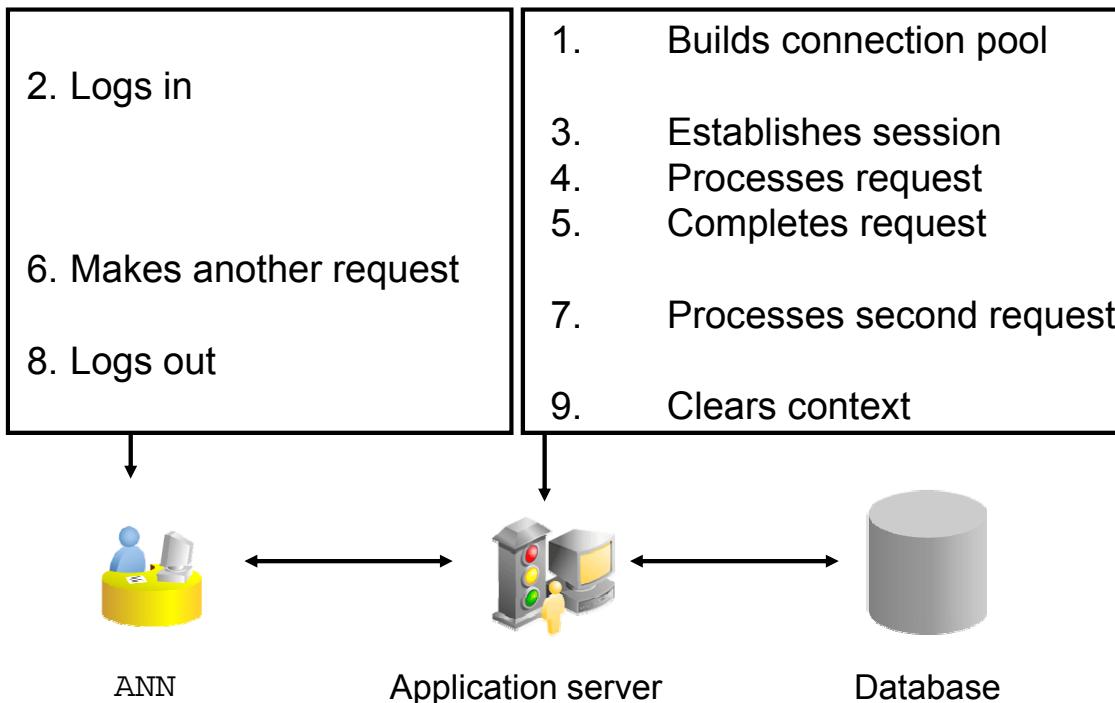
To support connection pooling managed by middle-tier applications, the `DBMS_SESSION` interface gives the application the ability to add a client identifier for each application context. The application can manage the context globally, whereas each client sees only its assigned application context.

The application must test input to prevent a malicious user from injecting a call to `DBMS_SESSION` or any SQL injection attempt. Because `DBMS_SESSION` is granted to `PUBLIC`, such an injection can allow the user unauthorized privileges.

By default, a context is not accessed globally.

Note: A context accessed globally is not available in the Real Application Clusters (RAC) environment for connections that span instances. The context accessed globally is stored in the shared pool of one instance and is not available in the other instances of the cluster.

Application Context Accessed Globally in Action



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The boxes in the slide show the actions being performed. The box on the left shows the actions taken by the user. The box on the right shows the actions taken by a program running on the application server.

1. The application server starts up and builds its connection pool by establishing multiple connections to the database as the APPSMGR user.
2. The user ANN logs in to and is authenticated by the application server.
3. The application server establishes a session for the request from ANN by:
 - Using a connection from the connection pool
 - Calling `SET_IDENTIFIER` to assign a session client identifier for the ANN username
 - Setting the application context
 - Saving the client identifier in the browser of ANN as part of a cookie, or maintaining the client identifier within the application server
4. The application server processes the request of ANN. When the application reads contexts, it does not include the client identifier because the `SET_IDENTIFIER` call in step 3 has already identified the session.
5. When the application server finishes the request of ANN, it:
 - Issues a `CLEAR_IDENTIFIER` call
 - Returns the connection to the pool

6. The ANN user makes another request.
7. The application starts differently because there is a cookie identifying the current context:
 - The session client identifier is retrieved from the cookie or the application server storage.
 - The client identifier is used in the `SET_IDENTIFIER` call after the connection is obtained from the pool.
 - The application runs as before, accessing the context from the previous call.
8. User ANN logs out from the application.
9. The application server issues `CLEAR_CONTEXT` to clear out the application context.

All `SYS_CONTEXT` calls within this database session return only application context values belonging to the client session (for example, `SYS_CONTEXT('HRAPP', 'ID')` returns ANN).

The `SET_IDENTIFIER` procedure sets an identifier that can be used to share a global context. It has the following specification:

```
PROCEDURE set_identifier ( client_id  VARCHAR2 )
```

where `CLIENT_ID` is an arbitrary identifier being set for this session. Because this identifier is often placed in the browser cookie, it should not be the information that can violate the privacy of the user. If the application code sets the client identifier with a call to `DBMS_SESSION.SET_IDENTIFIER`, `CLIENT_ID` is recorded in audit trails and can provide a way to link a user to an action, if the application maintains a `CLIENT_ID` to user mapping.

The `CLEAR_IDENTIFIER` procedure clears the current session identifier. It has no arguments.

The `SET_CONTEXT` procedure sets attributes of a global context:

```
PROCEDURE set_context (
    namespace  VARCHAR2, attribute VARCHAR2, value      VARCHAR2,
    username   VARCHAR2  DEFAULT NULL,
    client_id  VARCHAR2  DEFAULT NULL )
```

where:

`username` is the `username` attribute for the application context.

`client_id` is the client identifier that identifies a user session to set a context.

The `CLEAR_CONTEXT` procedure clears the attributes in a context. It has the following syntax:

```
PROCEDURE clear_context (namespace VARCHAR2, client_id VARCHAR2,
                        attribute  VARCHAR2 DEFAULT NULL )
```

If `ATTRIBUTE` is not included, all contexts for the client are cleared.

Implementing the Application Context Accessed Globally

1. Create the application context accessed globally.
2. Modify the program that establishes a session:
 - Set the application context.
 - Set the session client identifier.
 - Clear the client identifier when the request ends.
3. Modify the application program that handles subsequent requests in the same session:
 - Set the session client identifier from this session.
 - Clear the client identifier when the request ends.
4. Create or modify the application program that ends a session to clear the context.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To implement an application context accessed globally, perform the steps mentioned in the slide.

Because the application server sets the session client identifier when a session is established, the application programs that access the application context are coded the same, whether or not the context is accessed globally. In either case, you do not include the session client identifier when reading an application context.

These instructions apply only to application contexts that are preserved between database connections. For a stateless session, you can use an application context without accessing it globally.

The `SET_IDENTIFIER` procedure sets an arbitrary value as `CLIENT_ID`. This value is used to identify a user. A user can receive the same client identifier every time he or she connects, or it can be unique every day or every hour. When you decide on the assignment of the client identifier, consider how cookies can be intercepted and possibly used to hijack user sessions or spoof user authentications. The client identifier must never contain any private or personal information that may be used to steal a user's identity. It is essential that the database trust the identifier from the application server. A secure application role enabled by the application server together with the identifier forms a defense-in-depth solution.

Viewing Application Context Information

```
SQL> CREATE CONTEXT hr_ct USING hr_context;

Context created.

SQL> SELECT *
  2   FROM dba_context
  3  WHERE namespace = 'HRAPP';

NAMESPACE      SCHEMA PACKAGE      TYPE
-----
HR_CT          SYS     HR_CONTEXT ACCESSED LOCALLY
HR_CT_GLOBAL   SYS     HR_CT_PACK ACCESSED GLOBALLY

SQL>
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The data dictionary includes the following views that are related to application contexts:

- DBA_CONTEXT describes all context namespaces that are defined in the database, regardless of whether any attributes have been specified for them by using the DBMS_SESSION.SET_CONTEXT procedure. The example in the slide uses this view.
- DBA_GLOBAL_CONTEXT contains a description of all context information that is accessible globally. This view is a subset of the information in DBA_CONTEXT.
- SESSION_CONTEXT, V\$CONTEXT, and GV\$CONTEXT list the attributes and the attribute values for the current session.
- SESSION_CONTEXT and V\$CONTEXT list the same information. GV\$CONTEXT is used with RAC.
- V\$GLOBALCONTEXT and GV\$GLOBALCONTEXT list the attributes and the attribute values for application contexts that are accessed globally. The views include the following columns:
 - NAMESPACE is the namespace that the active attribute is in.
 - ATTRIBUTE is the name of the active attribute.
 - VALUE is the value of the active attribute.
 - USERNAME is the username for the session.
 - CLIENTIDENTIFIER is the unique session identifier.

Application Context Usage Guidelines

- Attempting to change the context outside of its package results in the following error message:
ORA-01031: insufficient privileges.
- SYS_CONTEXT works much like a bind variable.
- Versioning does not apply to contexts accessed globally.
- There are parallel query and RAC limitations.
- Context sources must be validated.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Setting the Context Outside of the Package

If you try to change the context outside of the package specified in the CREATE CONTEXT command, you receive an error message stating that you do not have the privilege required to change the context:

```
SQL> exec DBMS_SESSION.SET_CONTEXT ('hrapp', 'emp_id', 0);
BEGIN DBMS_SESSION.SET_CONTEXT ('hrapp', 'emp_id', 0); END;
*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SYS.DBMS_SESSION", line 78
ORA-06512: at line 1
```

Performance

If the `SYS_CONTEXT` arguments are constants, the `SYS_CONTEXT` function works much like a bind variable, enabling cursor sharing.

Versioning in Application Context

When you execute a statement, Oracle Database takes a snapshot of the entire application context. Within the duration of a query, the context remains the same for all fetches of the query. If you attempt to change the context within a query, the change does not take effect in the current query. Because a simple application context is allocated per session, the application context can be versioned.

Versioning is not available for the application context accessed globally. Versioning returns the `SYS_CONTEXT` values at a point in time. Because multiple client sessions may be accessing the same global application context values at any time, versioning is not possible.

Parallel Queries

If you try to execute `SYS_CONTEXT` in a parallel query environment, you receive a query error.

If `SYS_CONTEXT` is used inside a SQL function that is embedded in a parallel query, the function cannot pick up the application context. This is true because the application context exists only in the user session. To use these features in combination, you must call `SYS_CONTEXT` directly from the query.

Application Contexts Accessed Globally and RAC

Application contexts accessed globally are not available in RAC.

Validating Context Sources

When using an application context for security, the source of the values for the context attributes must be thoroughly validated. If the source of the context is user input, there is a possibility that the attribute may be altered to allow unintended access.

Summary

In this lesson, you should have learned how to:

- Use an application context
- Access the sources of application context values
- Implement an application context that is accessed globally



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

Practice 11-1 covers implementing an application context.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

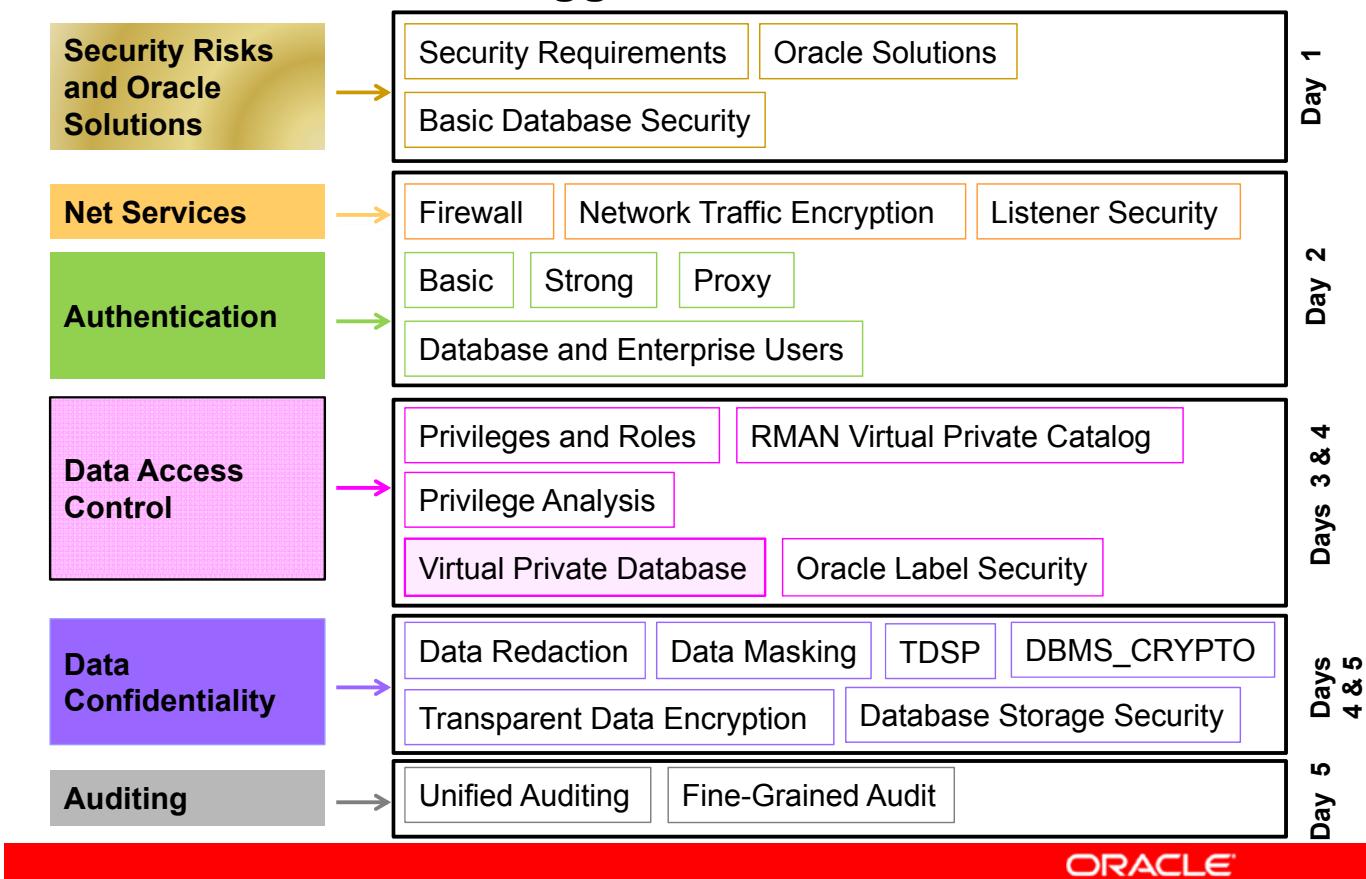
12

Implementing Virtual Private Database

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Schedule



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

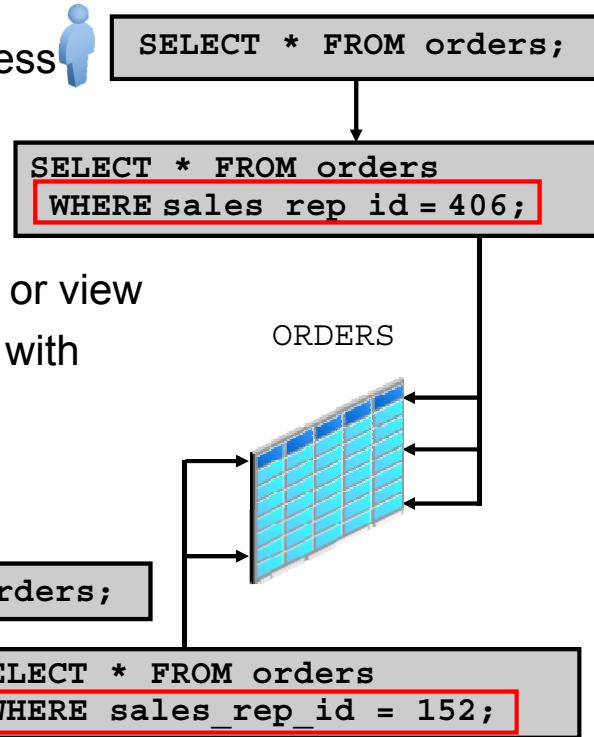
- Describe fine-grained access control (FGAC)
- Describe Virtual Private Database (VPD)
- Implement row-level VPD policies
- Implement column-level VPD policies
- Implement static VPD policies
- Implement context-sensitive VPD policies
- Review VPD guidelines



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Fine-Grained Access Control: Overview

- Limits row or column access
- Uses a predicate
- Is returned from a function
- Is associated with a table or view
- Is automatically enforced with SELECT and DML



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Fine-grained access control (FGAC) enables you to build applications that enforce security policies at a low level of granularity. You can use FGAC to restrict access by rows and columns, as in the following examples:

- Customers can see only their own accounts.
- Physicians see only the records of their own patients.
- Managers see only the records of employees who work for them.
- A fulfillment clerk sees customer records for shipping purposes, but not columns with credit card numbers.

To use FGAC, create a security policy function that returns a predicate (a WHERE condition). After you attach the policy to a table, view, or synonym, the predicate controls the rows accessed from the table or view. When a user executes a SQL statement (SELECT, INSERT, UPDATE, or DELETE) on that object, the Oracle server dynamically modifies the user's statement—transparently to the user or application—so that the user sees only the appropriate rows. The policy can be set to add the predicate only when certain columns of interest are accessed, allowing the control to be more fine-grained and reducing the overhead when those columns are not accessed.

Using FGAC, you can:

- Create different policies for SELECT, INSERT, UPDATE, and DELETE statements
- Use security policies only on the tables where you need them
- Use multiple policies for each table
- Group policies so that the policies are applied per application
- Limit the rows that the user can access, even though the application is not aware that FGAC is being used
- Limit rows that can be viewed if sensitive columns are included
- Allow all rows to be viewed and mask the sensitive columns

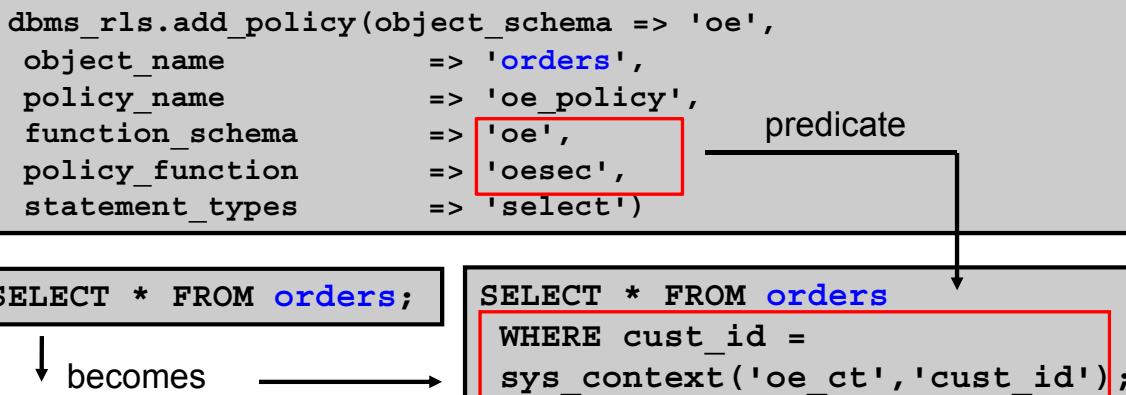
You administer the security policies by using the DBMS_RLS PL/SQL package. With this package, you can add, drop, enable, disable, and refresh the policies that you have created.

Example:

In the example in the slide, two different users enter the same SQL statement; however, the security policy is applied to limit the query to those rows where the user is the sales representative for the order. In this example, the security policy consists of the entire WHERE clause.

Understanding FGAC Policy Execution

1. The user accesses a table or view with a policy.
2. The data server calls the policy function.
3. The policy function returns a predicate.
4. The data server adds the predicate to the statement.
5. The data server executes the modified statement.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

An example of fine-grained access control policy execution is as follows:

1. The user accesses the ORDERS table object with the following statement:

```
SELECT * FROM orders;
```

The DML statement can be SELECT, UPDATE, INSERT, DELETE, or INDEX. Oracle Database does not implement FGAC during MERGE statements. You must use equivalent INSERT and UPDATE statements instead of MERGE to avoid error messages and to ensure correct access control. For SELECT, the security is enforced even when the table is accessed in a subquery.

2. The Oracle server calls the security function that implements the security policy for the ORDERS table. If you are using policy groups, the Oracle data server looks up the driving context to determine the policy group in effect and calls the security function in that group.
3. The policy function returns the predicate:

```
cust_id = sys_context ('oe_ct', 'cust_id')
```

The call to SYS_CONTEXT gets the CUST_ID attribute from the OE_CT context. Because the predicate uses an application context, this is also an example of a VPD. However, you do not need to use a context to implement FGAC.

4. The Oracle server dynamically modifies the user's statement to read:

```
SELECT * FROM orders WHERE cust_id = SYS_CONTEXT  
('oe_ct','cust_id');
```

If multiple policies are attached to a table, the data server combines and enforces all the predicates.

5. The Oracle server executes the dynamically modified statement. Upon execution, the function employs the username returned by
SYS_CONTEXT ('USERENV', 'SESSION_USER')
to look up the corresponding customer and to limit the data returned from the ORDERS table to that customer's data only.

Multiple Policies

You can establish several policies for the same table or view. Suppose that you have a base application for order entry, and each division of your company has its own special rules for data access. You can add a division-specific policy function to a table without having to rewrite the policy function of the base application.

All policies applied to a table are enforced with the AND syntax. Thus, if you have three policies applied to the CUSTOMERS table, each policy is applied to any access of the table.

In addition, you can use policy groups and a driving application context to partition fine-grained access control enforcement so that different policies apply, depending on which application is accessing the data.

Benefits of Using Fine-Grained Access Control

- Security:
 - The fine-grained access control policy is always applied.
 - The policy enforces business rules to limit row access.
- Simplicity:
 - Define the policy once.
 - The policy is independent of the application.
- Flexibility:
 - Apply different access rules to different SQL statements.
 - Group policies.
- High performance:
 - Define policies as static, context sensitive, or dynamic.
 - Active policies stored in memory.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Security

Fine-grained access control attaches security policies to tables, views, or synonyms to enforce business rules. No matter how a user accesses the data, the same security is always in force. Exceptions are users connecting as `SYS` and those with the `EXEMPT ACCESS POLICY` privilege.

Simplicity

Adding the security policy to the table or view means that you make the addition only once, rather than repeatedly adding it to each of your table- or view-based applications.

Flexibility

You can have one security policy for the `SELECT` statement, another policy for `INSERT`, and others for `UPDATE` and `DELETE` statements. For example, you may want to enable a human resources clerk to select all employee records in his or her division, but to update salaries only for those employees in his or her division whose last names begin with “A” through “F.”

High Performance

You can control the evaluation frequency of the policies by declaring them static, context-sensitive, or dynamic. Active policies are cached in the shared pool for fast evaluation.

Virtual Private Database

A Virtual Private Database (VPD) combines an application context and fine-grained access control to:

- Enforce business rules to limit row access
- Use a secure application context to provide high-performance resolution of user attributes



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A Virtual Private Database (VPD) uses FGAC to limit row and column access and an application context to provide the information that is used to set the policy predicates. FGAC is an enabling technology that is a building block for a VPD and Oracle Label Security. FGAC without the use of an application context often requires subqueries and repeated evaluation of the policy. By using a context, the policy predicate often remains the same, reducing or eliminating the need to reevaluate the policy and eliminating the cost of executing a subquery. The VPD feature is available with Oracle Database Enterprise Edition.

Examples of VPD

VPD allows multiple policies on the same table:

- Customer example:



```
SELECT * FROM orders;
```



```
SELECT * FROM orders  
WHERE customer_id = SYS_CONTEXT ('OE_CT', 'CUST_ID');
```

- Sales representative example:



```
SELECT * FROM orders;
```



```
SELECT * FROM orders  
WHERE sales_rep_id = SYS_CONTEXT ('OE_CT', 'EMP_ID');
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

VPD is based on dynamically modified statements. For example, you have implemented a policy that customers can see only their own orders. The policy causes any data manipulation language (DML) statement, including SELECT, that accesses the ORDERS table to be modified to restrict row access. This description is for row-level access control. For column-level access control, the columns of interest are checked first. If no columns of interest are accessed, no predicate is applied.

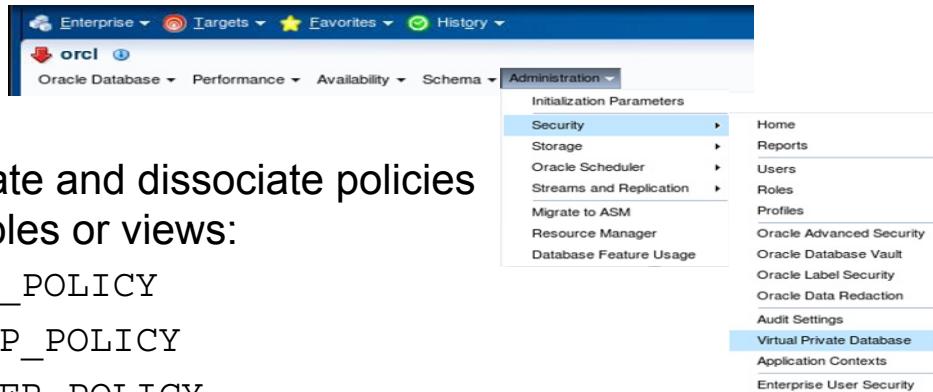
Consider an order-entry application, which enforces the following rules:

- Customers see only their own orders.
- Account managers see only orders for their customers.
- Sales representatives see only the orders that they entered.
- Sales analysts do not see credit card numbers.

These rules require different predicates: one for customers and another for employees.

For customers, the predicate uses the customer identifier to limit the rows that are accessed. Therefore, you define a context with a CUST_ID attribute. For employees, you can define an application context with an employee identifier and job-title attribute, and these attributes can be accessed within the policy function to return the correct predicate, depending on the value of the attributes. The sales analyst can view and summarize customer information without having access to sensitive information. The sales representative can access all the records for his or her customers.

Using DBMS_RLS to Manage Policies



- Associate and dissociate policies with tables or views:
 - ADD_POLICY
 - DROP_POLICY
 - ALTER_POLICY
- Enable and disable policies:
 - ENABLE_POLICY
 - DISABLE_POLICY
- Refresh policies:
 - REFRESH_POLICY

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

These procedures are used to create, drop, enable, and disable policies. Grouping policies is detailed in the appendix titled “Oracle Virtual Private Database Policy Groups.”

The DBMS_RLS procedures cause current DML transactions, if any, to commit before the operation. However, the procedures do not cause a commit first if they are inside a data definition language (DDL) event trigger. The DBMS_RLS procedures are part of DDL transactions.

Syntax for the procedures in the DBMS_RLS package is shown in the practices. The syntax for the other procedures is shown in *PL/SQL Packages and Types Reference*.

Parameters

The parameters that are used by the procedures in the DBMS_RLS package have common names. Every parameter has the same meaning regardless from which procedure the parameter is called. For example, object_schema always means the name of the schema containing the table, view, or synonym (current default schema, if NULL).

Column-Level VPD

- Statements are not always rewritten.

```
dbms_rls.add_policy(object_schema => 'hr',
object_name          => 'employees',
policy_name          => 'hr_policy',
function_schema      => 'hr',
policy_function      => 'hrsec',
statement_types      => 'select',
sec_relevant_cols    => 'salary',
sec_relevant_col_opts => dbms_rls.ALL_ROWS)
```

- Example: A policy protects the SALARY column of the EMPLOYEES table. The VPD policy is:

SQL> SELECT last_name FROM employees; Not enforced

SQL> SELECT last_name, salary FROM employees; Enforced

SQL> SELECT * FROM employees; Enforced

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A column-level VPD policy is applied and the statements are rewritten only when the security-relevant columns are accessed. This means that the combination of row-level access control and security-relevant columns implies that you can control access down to the element referenced. The column-level policies apply only to the SELECT operations.

Use the SEC_RELEVANT_COLS parameter to name the security-relevant columns when you apply the policy to the table or view. This parameter does not apply to synonyms.

You can also specify the SEC_RELEVANT_COL_OPTS constant:

- DBMS_RLS.ALL_ROWS: All rows are displayed but the columns named have NULL values for the values that the user is not permitted to access.
- NULL: Only the rows that are allowed by the policy are displayed.

Suppose that the business policy and the imposed VPD policy is that a manager can access the EMPLOYEES sensitive information only for his or her employees. The SALARY column is considered sensitive information. The Oracle Database server does not enforce the VPD policy when you select only the LAST_NAME column from the EMPLOYEES table. Therefore, all employees can access nonsensitive information in the EMPLOYEES table. However, when you issue queries that access columns considered as security relevant, VPD applies the access control policy defined by the policy function.

Note: Column-level policies must include a function that returns a predicate; if the rows are not restricted, column restrictions are not enforced.

Policy Types: Overview

POLICY_TYPE =>	How often a policy function is reevaluated
DBMS_RLS.DYNAMIC (default)	Always reexecuted on each statement parsing or execution
DBMS_RLS.STATIC DBMS_RLS.SHARED_STATIC	Executed once: <ul style="list-style-type: none"> • The same predicate is used for all statements • Cached in SGA
DBMS_RLS.CONTEXT_SENSITIVE namespace => 'hr_ctx' attribute => 'role' DBMS_RLS.SHARED_CONTEXT_SENSITIVE	For each session when: <ul style="list-style-type: none"> • The statement is first parsed • There is a related change in the local application context The predicate is cached in the user's session memory.

- Shared: Shared policies allow you to share the same policy function with different objects.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The execution of policy functions can consume a significant amount of system resources and impact your database performance. You can control when the policy function is reevaluated by specifying the POLICY_TYPE parameter.

Policies are **dynamic** by default. The Oracle Database server assumes that the predicate may be affected by any system or session attribute and, therefore, always executes the policy function on each statement parsing or execution.

The policy function of **static** policies is executed only once. VPD always enforces the same predicate for access control regardless of which user accesses the objects. The predicate is cached in the SGA. This makes static policies very fast.

Note: Each execution of the same rewritten statement can produce a different row set because the predicate may filter the data differently according to context attributes or functions (such as SYSDATE). When the predicate uses the SYS_CONTEXT function and the attribute remains the same, the function call is treated like a bind variable.

Policy predicates may be static for a particular user session, but different for other users. In some cases, policy predicates can change when certain context attributes are changed within a user session. These policies are **context-sensitive**. You can manually refresh all the cached statements associated with a context-sensitive policy by running the DBMS_RLS.REFRESH_POLICY procedure.

Shared policies allow you to specify the same policy on multiple objects.

Designing and Implementing a VPD Solution

The steps to design and implement a VPD solution are:

1. Develop a strategy to understand the security problem.
2. Analyze the data to be protected.
3. Determine user access requirements.
4. Implement VPD policies.
5. Review and document your policy decisions.



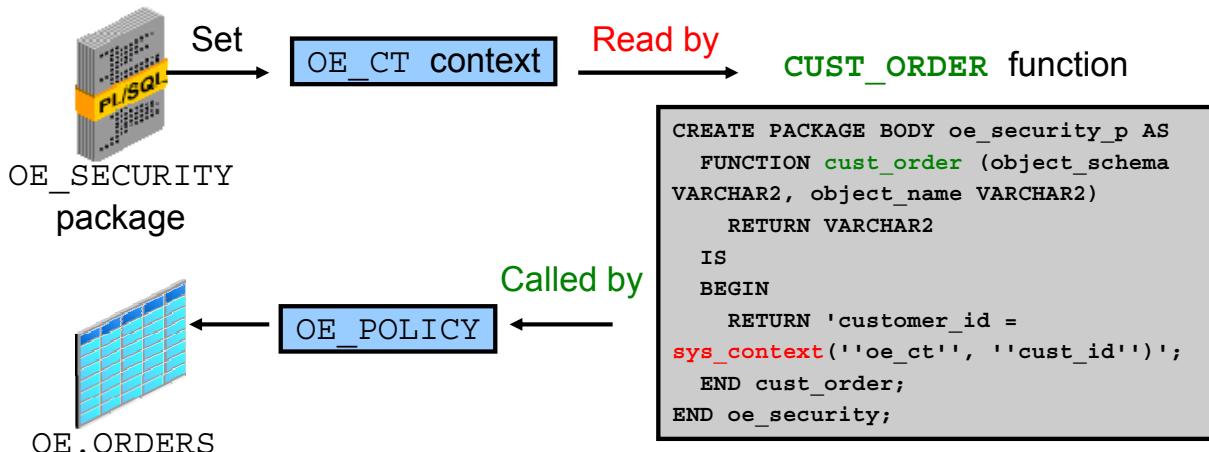
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Do not attempt to protect everything; usually, only a small number of tables contain sensitive data. The design process steps are repeated for each set of data that is to be protected.

1. **Develop a strategy:** Talk to the right people. Identify those individuals in your organization who really understand the business-security problem. Make sure that you understand the problem before adding more security to your application.
2. **Analyze the data to be protected:** Ask the following questions:
 - Where does the sensitive data reside in the application?
 - Who needs access to this data?
 - Who owns the data?
 - Who should be able to read the data?
 - Who should be able to make updates?
3. **Determine user access requirements:** This analysis includes a grouping of the user community by access needs:
 - Does that grouping follow organizational lines?
 - Does it depend on the job function?
4. **Implement VPD policies.**
5. **Review and document policy decisions:** Do the policies solve the original problem?

Implementing a VPD Policy

1. Create a PL/SQL package that sets the context.
2. Create an application context.
3. Write the function that creates a predicate.
4. Create a policy to associate the function to the table.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To implement a VPD policy, perform the steps listed in the slide.

- Steps 1 and 2 are the same as listed in the lesson titled “Using Application Contexts.”
- Step 3: Write the function that implements the security policy on the database object so that it can access the context for best performance. This function can be stand-alone or part of a package. You can easily test the function by calling it and verifying that the appropriate predicate is returned:

```

SQL> SELECT oe_security_p.cust_order('a', 'b') FROM dual;
OE_SECURITY_P.CUST_ORDER('A', 'B')
-----
customer_id = SYS_CONTEXT('oe_ct', 'cust_id')
  
```

- Step 4: Create the policy by using the `DBMS_RLS.ADD_POLICY` procedure. The policy is named `OE_POLICY` to the `ORDERS` table in the `OE` schema. The `OE_SECURITY.CUST_ORDER` function is stored in the `SEC` schema and returns the policy predicate. The policy would apply to the `SELECT` statements only.

You need not grant the `EXECUTE` privilege on the security package to application users.

Note: In a CDB, VPD policies apply only to the objects within the current PDB. This restriction applies to including shared context sensitive policies and views related to VPD policies as well. You cannot create a VPD policy for the entire CDB.

Writing a Function That Returns Different Predicates

Predicate Returned by Function	Meaning
<code>RETURN 'NULL';</code>	The owner of the table has access to all rows.
<code>RETURN 'sales_rep_id = sys_context(''oe_ct'', ''emp_id'');</code>	Sales representatives see only their orders.
<code>RETURN 'customer_id = sys_context(''oe_ct'', ''cust_id'');</code>	Customers can see only their own orders.
<code>RETURN '1=2';</code>	Other users have no access.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Exceptions to VPD Policies

Policies are not enforced for:

- DIRECT path export
- The SYS user (any user connected AS SYSDBA)
- Users granted the EXEMPT ACCESS POLICY privilege



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

FGAC policies are the basis for the VPD and Oracle Label Security. The exceptions listed here apply to both:

DIRECT Path Export

Fine-grained access control policies are not enforced during the DIRECT path export. Only SYS or a user with the EXPORT_FULL_DATABASE role enabled can perform the DIRECT path export.

SYS User (Connected AS SYSDBA)

FGAC policies cannot be applied to objects in the SYS schema. Consequently, the SYS user and users making a privileged connection to the database (for example, CONNECT / AS SYSDBA) do not have FGAC policies applied to their actions. They are always exempted from FGAC enforcement. However, the SYSDBA actions can be audited.

Users Granted EXEMPT ACCESS POLICY

Database users that are granted the EXEMPT ACCESS POLICY privilege, either directly or through a database role, are exempted from FGAC enforcements. EXEMPT ACCESS POLICY is a very powerful privilege and should be carefully managed. Do not grant this privilege and the WITH ADMIN OPTION clause.

Quiz

Identify the features that are utilized by VPD.

- a. Oracle Label Security
- b. Application context
- c. Fine-grained access control



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b, c

Guidelines for Policies and Context

- Restrict SELECT and data manipulation language (DML) commands with the same policy.
- A policy may not select from a table protected by the policy.
- If there is no context, SYS_CONTEXT returns NULL.
- Avoid recursive contexts.
- Look in the session trace file to resolve errors.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Restrict with the Same Policy

If you do not restrict all DML statements, you can get unexpected results.

For example, if you restrict only the SELECT statement, the customer issues this query to count the rows in the table:

```
SQL> SELECT COUNT(*) FROM oe.orders GROUP BY customer_id;
```

```
COUNT (*)
-----
1
```

However, if the customer updates the table, all the rows in the table are accessed:

```
SQL> UPDATE oe.orders SET sales_rep_id = 152;
107 rows updated.
```

If you apply different policies or predicates for different statement types on the same user and table, be aware of this implication.

SELECT Restriction

A policy for a table or view adds a predicate to every SQL statement executed against the table or view. A SELECT statement against that table or view cannot be executed from within the policy.

Missing Contexts

If user BOB has `SELECT` privileges on the `OE.ORDERS` table, but he does not have a `CUST_ID` set properly in the `OE.CUSTOMERS` table, his `OE.CUST_ID` context attribute is null and the predicate is evaluated as `CUSTOMER_ID = NULL`. This evaluates to `NULL` and prevents BOB from accessing any rows in the table. In addition, if BOB is not supposed to have access, be sure that an exception does not leave the predicate set for a previous user.

Avoid Recursive Contexts

If the procedure that sets the application context executes a `SELECT` statement from a table, and the policy on the table uses that application context to set the predicate, the `SELECT` statement fails. For example, if the user's employee ID is used to set the application context by selecting from the `EMPLOYEES` table when the user logs on, and a policy uses that application context to set a predicate controlling access to the `EMPLOYEES` table, the `SELECT` statement returns no rows. This is because the application context attributes are not set. To avoid this problem, access the `EMPLOYEES` table by using the privileges of a user that has the `EXEMPT ACCESS POLICY` system privilege.

Error Handling

To resolve the error message:

`ORA-28112: failed to execute policy`

look in the trace file for the session. The following trace file includes the `PLS-00306` error, which means that the wrong number of arguments was passed to the security function. Tracing of errors is automatic. There are no tracing parameters required to enable this behavior.

```
*** 2010-01-29 11:27:43.478
-----
Policy function execution error:
Logon user      : TEST
Table/View       : TEST.TAB
Policy name     : TAB_RLS_POLICY
Policy function: TEST.TAB_SECURITY
ORA-06550: line 1, column 15:
PLS-00306: wrong number or types of arguments in call to
           'TAB_SECURITY'
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored
```

References:

- [How to Know the Exact Cause of an ORA-28113 Error After Setting a FGAC Policy \[ID 250094.1\]](#)
- [Note:119335.1 How To Solve the Problem of Circular Row Level Policies](#)
- [Note:69401.1 How to resolve ORA-28110 or ORA-28112 on SELECT or DML](#)
- [Note:71408.1 OERR ORA-28113 policy predicate has error](#)
- [Note:331862.1 ORA-28113 when a policy predicate is fetched from a Context](#)

Policy Performance

For best performance:

- Consider indexing the column in the predicate.
- Do not use subqueries in the predicate.
- Do not use literals in the predicate.
- Use STATIC_POLICY=TRUE, DBMS_RLS.STATIC, or SHARED_STATIC when possible.
- Restrict the evaluation to a specific application context by including both the namespace and attribute parameters.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Index the Column in the Predicate

Because the columns used in the predicate are used to access the table, you may be able to improve query performance by indexing these columns.

Example: The predicate returned is:

```
customer_id = SYS_CONTEXT('oe', 'cust_id')
```

The optimizer could use an index on OE.CUSTOMER_ID.

Do Not Use Subqueries in the Predicate

To avoid the overhead of a subquery, do not include it in the predicate. Instead, perform the query and use the result to set a context and use the context in the predicate.

The correct code is as follows:

- Set the context value with the query:

```
SELECT customer_id INTO v_cust_id FROM oe.customers  
WHERE cust_first_name || '_' || cust_last_name =  
      SYS_CONTEXT('USERENV', 'SESSION_USER');  
DBMS_SESSION.SET_CONTEXT('oe_ct', 'cust_id', v_cust_id);
```

- From the security function, return this predicate, which uses the context:

```
customer_id = SYS_CONTEXT('oe_ct', 'cust_id')
```

Do not bypass the context and code the predicate with a subquery:

```
customer_id = ( SELECT customer_id
    FROM oe.customers
    WHERE cust_first_name || '_' || cust_last_name =
        SYS_CONTEXT('USERENV', 'SESSION_USER' )
```

This code causes the subquery to be performed every time the table is accessed.

Do Not Use Literals in the Predicate

Literal values in the predicate could make every SQL statement issued have a different predicate. The SQL cursors could not be shared, and therefore there would be an additional overhead for parsing. Because the call to `SYS_CONTEXT` is the same for all predicates, the SQL statement cursors can be shared by multiple users. The use of application context in a fine-grained access control package effectively gives you a bind variable in a parsed statement.

For example, a predicate applied to the `ORDERS` table uses the actual customer identifier, as in the following:

```
cust_id = 12345
```

Each customer that logs on gets a different predicate, and because each predicate is different, the cursor could not be shared.

Static Policy and Policy Type Parameters

Always specify the frequency that the policy needs to be evaluated either by using the `STATIC_POLICY` policy parameter or by setting a `POLICY_TYPE` when using the `ADD_POLICY` procedure. Reduce the execution overhead by specifying `STATIC_POLICY=TRUE` to indicate that the policy function always returns the same predicate. If neither is set, the default is a dynamic policy that is evaluated for every DML.

Using a Context-Sensitive Policy for Application Context Attributes That Change

If there is a change in any attribute of any application context during the user session, then by default the database re-executes the policy function to ensure that it captures all changes to the predicate since the initial parsing. This results in unnecessary re-executions of the policy function if none of the associated attributes has changed. You can restrict the evaluation to a specific application context by including both the namespace and attribute parameters:

- Ensure that you specify both namespace and attribute parameters, not just one.
- Ensure that your policy has the `policy_type` argument set to `DBMS_RLS.CONTEXT_SENSITIVE` or `SHARED_CONTEXT_SENSITIVE`. You cannot use the namespace and attribute parameters in static or dynamic policies.

If there are no attributes associated with the VPD policy function, Oracle Database evaluates the context-sensitive function for any application context changes.

```
DBMS_RLS.ADD_POLICY(
    object_schema => 'hr', object_name => 'employees',
    policy_name => 'secure_update', policy_function => 'hide_fin',
    policy_type => dbms_rls.CONTEXT_SENSITIVE,
    namespace => 'empno_ctx',
    attribute => 'emp_id')
```

Export and Import

For export and import, consider the following guidelines:

- Restoring the policies requires the EXECUTE privilege on the DBMS_RLS package.
- Exporting a table with FGAC policies enabled exports only those rows that the exporter is privileged to read.

```
Processing object type TABLE_EXPORT/TABLE/RLS_POLICY/RLS_POLICY
...
ORA-39181: Only partial table data may be exported due to fine grain
access control on "HR"."EMPLOYEES"
. . exported "HR"."EMPLOYEES"                                1 KB      10 rows
```



- Policies exported in the DIRECT path are not enforced.
- Policies are not enforced for SYS or any user with the EXEMPT ACCESS POLICY privilege.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can export tables with fine-grained access policies enabled. When doing so, consider the following:

- To restore the policies, the user who imports from an export file containing such tables must have the EXECUTE privilege on the DBMS_RLS package so that the security policies of the tables can be reinstated.
- If a user without the right privileges attempts to export a table with fine-grained access policies enabled, only those rows that the exporter is privileged to read are exported.
- Policies are not enforced for the DIRECT path export. Only SYS or a user with the EXPORT_FULL_DATABASE role enabled can perform the DIRECT path export.
- Policies are not enforced for SYS at any time, or any user who has been granted the EXEMPT ACCESS POLICY privilege.

Policy Views

- Policy views list security policies: *_POLICIES
- Dynamic performance views list active policies:

```
SQL> SELECT distinct policy, predicate, sql_text
  2      FROM v$vpd_policy p, v$sql s
  3      WHERE s.child_address = p.address;

          POLICY        PREDICATE
-----+
SQL_TEXT

-----+
OE_POLICY      1=1
select * from oe.orders

OE_POLICY      sales_rep_id = SYS_CONTEXT('hrapp', 'id')
select * from oe.orders
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Security Policy Views

- *_POLICIES: Views with information provided in the ADD_POLICY procedure

Dynamic Performance Views

- V\$VPD_POLICY: All the fine-grained security policies and predicates associated with the cursors currently in the library cache. GV\$VPD_POLICY provides the same information as V\$VPD_POLICY, except that this view is used with multiple instances with RAC.

Checking for Policies Applied to SQL Statements

V\$VPD_POLICY contains the policies that are applied to SQL statements. To find which policy corresponds to a particular SQL statement, perform the query shown in the slide.

This query is executed after a user LSMITH (a sales representative) and the OE user accessed the ORDERS table with the following statement:

```
SELECT count(*) FROM oe.orders;
```

The query shown in the slide and the V\$VPD_POLICY view are helpful when tuning statements that are being rewritten by VPD policies.

The code in the current slide can be demonstrated from the /home/oracle/labs/demos/demo_vpd_policy.sql script.

For more details, see *Oracle Database Reference*.

Summary

In this lesson, you should have learned how to:

- Describe FGAC and VPD
- Implement row-level and column-level VPD policies
- Implement static and context-sensitive VPD policies
- Review VPD guidelines



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- 12-1: Implementing a VPD policy function
- 12-2: Implementing a dynamic VPD policy
- 12-3: Troubleshooting VPD policy errors
- 12-4: Dropping VPD policies



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.