

# HOW TO

## Secure and Audit Oracle 10g and 11g



Ron Ben Natan  
Foreword by Pete Finnigan

# HOWTO

## Secure and Audit Oracle 10g and 11g

## OTHER NEW BOOKS FROM AUERBACH

**The Business Value of IT: Managing Risks, Optimizing Performance and Measuring Results**  
Michael D. S. Harris, David Herron, and Stasia Iwanicki  
ISBN: 1-4200-6474-6

**CISO Leadership: Essential Principles for Success**  
Todd Fitzgerald and Micki Krause  
ISBN: 0-8493-7943-1

**The Debugger's Handbook**  
J.F. DiMarzio  
ISBN: 0-8493-8034-0

**Effective Software Maintenance and Evolution: A Reuse-Based Approach**  
Stanislaw Jarzabek  
ISBN: 0-8493-3592-2

**The Ethical Hack: A Framework for Business Value Penetration Testing**  
James S. Tiller  
ISBN: 084931609X

**Implementing Electronic Document and Record Management Systems**  
Azad Adam  
ISBN: 0-8493-8059-6

**Implementing the IT Balanced Scorecard: Aligning IT with Corporate Strategy**  
Jessica Keyes  
ISBN: 0-8493-2621-4

**Information Security Cost Management**  
Ioana V. Bazavan and Ian Lim  
ISBN: 0-8493-9275-6

**The Insider's Guide to Outsourcing Risks and Rewards**  
Johann Rost  
ISBN: 0-8493-7017-5

**Interpreting the CMMI®: A Process Improvement Approach, Second Edition**  
Margaret K. Kulpa and Kent A. Johnson  
ISBN: 1-4200-6052-X

**Knowledge Management, Business Intelligence, and Content Management: The IT Practitioner's Guide**  
Jessica Keyes  
ISBN: 0-8493-9385-X

**Manage Software Testing**  
Peter Farrell-Vinay  
ISBN: 0-8493-9383-3

**Managing Global Development Risk**  
James M. Hussey and Steven E. Hall  
ISBN: 1-4200-5520-8

**Patterns for Performance and Operability: Building and Testing Enterprise Software**  
Chris Ford, Ido Gileadi, Sanjiv Purba, and Mike Moerman  
ISBN: 1-4200-5334-5

**A Practical Guide to Information Systems Strategic Planning, Second Edition**  
Anita Cassidy  
ISBN: 0-8493-5073-5

**Service-Oriented Architecture: SOA Strategy, Methodology, and Technology**  
James P. Lawler and H. Howell-Barber  
ISBN: 1-4200-4500-8

**Six Sigma Software Development, Second Edition**  
Christine B. Tayntor  
ISBN: 1-4200-4426-5

**Successful Packaged Software Implementation**  
Christine B. Tayntor  
ISBN: 0-8493-3410-1

## AUERBACH PUBLICATIONS

[www.auerbach-publications.com](http://www.auerbach-publications.com)

To Order Call: 1-800-272-7737 • Fax: 1-800-374-3401

E-mail: [orders@crcpress.com](mailto:orders@crcpress.com)

# HOWTO

## Secure and Audit

### Oracle 10g and 11g

Ron Ben Natan  
Foreword by Pete Finnigan



CRC Press

Taylor & Francis Group

Boca Raton London New York

---

CRC Press is an imprint of the  
Taylor & Francis Group, an **informa** business  
AN AUERBACH BOOK

Auerbach Publications  
Taylor & Francis Group  
6000 Broken Sound Parkway NW, Suite 300  
Boca Raton, FL 33487-2742

© 2009 by Taylor & Francis Group, LLC  
Auerbach is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works  
Printed in the United States of America on acid-free paper  
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-13: 978-1-4200-8412-2 (Hardcover)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access [www.copyright.com](http://www.copyright.com) (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

**Trademark Notice:** Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

---

**Library of Congress Cataloging-in-Publication Data**

---

Ben-Natan, Ron.  
How to secure and audit Oracle 10g and 11g / Ron Ben-Natan.  
p. cm.  
Includes index.  
ISBN 978-1-4200-8412-2 (hardcover : alk. paper)  
1. Oracle (Computer file) 2. Computer security. 3. Data protection. 4. Database security. I. Title.

---

QA76.9.A25B446 2009  
005.8--dc22

2009001575

---

Visit the Taylor & Francis Web site at  
<http://www.taylorandfrancis.com>

and the Auerbach Web site at  
<http://www.auerbach-publications.com>

---

# Dedication

---

To my father Danny



---

# Contents

---

<b>Foreword .....</b>	<b>xi</b>
<b>Acknowledgments .....</b>	<b>xiii</b>
<b>Author .....</b>	<b>xv</b>
<b>1    Introduction: How This Book Will Help You Be Secure and Compliant.....</b>	<b>1</b>
1.1    Why Secure the Data? .....	2
1.2    Taxonomy of Best-Practice Database Security .....	8
1.3    Using HOWTOs to Secure Oracle.....	9
<b>2    Hardening the Database .....</b>	<b>11</b>
2.1    HOWTO Choose a Hardening Guideline .....	12
2.2    HOWTO Use a Vulnerability Assessment Tool.....	15
2.3    HOWTO Create and Maintain a Secure Configuration Baseline .....	17
2.4    HOWTO Understand Critical Patch Updates.....	18
2.5    HOWTO Sanitize Data for Test .....	22
2.6    Discussion: Defense in Depth.....	26
<b>3    Securing the Listener.....</b>	<b>29</b>
3.1    HOWTO Secure Access to lsnrctl.....	31
3.2    HOWTO Limit the Ability to Change Listener Properties .....	39
3.3    HOWTO Secure EXTPROC .....	40
3.4    HOWTO Limit the Sources from Which Connections Are Accepted .....	46
3.5    HOWTO Inspect Listener Logs and Traces and HOWTO Limit Traces.....	47
3.6    HOWTO Combat TNS Protocol Attacks.....	49
3.7    Discussion: History of Listener Security Alerts.....	51
<b>4    Account Security.....</b>	<b>53</b>
4.1    HOWTO Create, Alter, Drop, and Lock User Accounts.....	53
4.2    HOWTO Understand the Standard Logon Process .....	59
4.3    HOWTO Use Password Policies. ....	61
4.4    HOWTO Enforce Password Complexity.....	63
4.5    HOWTO Check for Weak and Default Passwords .....	64
4.6    HOWTO Set Password Case.....	65
4.7    HOWTO Use Impossible Passwords.....	66
4.8    HOWTO Limit System Resources Used by Users.....	68
4.9    HOWTO View Information on Users and Profiles .....	69
4.10    A dditional Resources .....	71

<b>5</b>	<b>Cryptography, Oracle Wallets, and Oracle PKI.....</b>	<b>73</b>
5.1	HOWTO Create Wallets.....	92
5.2	HOWTO Add Certificates.....	94
5.3	HOWTO Create and Sign a Certificate Request.....	95
5.4	Discussion: Orapki Errors.....	98
<b>6</b>	<b>Authentication .....</b>	<b>99</b>
6.1	HOWTO Understand and Use O3/O5 LOGON and OS Authentication.....	99
6.2	HOWTO Use Password Files.....	105
6.3 H	OWTO Configure Clients to Use External Password Stores .....	107
6.4 H	OWTO Configure SSL-Based Authentication Using ASO .....	112
6.5 H	OWTO Configure Kerberos Authentication Using ASO .....	115
6.6 H	OWTO Configure RADIUS and Two-Factor Authentication Using ASO .....	119
6.7	Discussion: Protect Your Password Hashes .....	124
<b>7</b>	<b>Encrypting Data-in-Transit.....</b>	<b>127</b>
7.1 H	OWTO Configure Network Encryption Using ASO.....	137
7.2 H	OWTO Configure Network Encryption for JDBC Drivers .....	139
7.3 H	OWTO Configure Data Integrity Using ASO .....	140
7.4	HOWTO Use IPSEC, Tunnels, and Hardware Acceleration .....	141
7.5	Discussion: Performance Impact When Encrypting Data-in-Transit .....	149
<b>8</b>	<b>Encrypting Data-at-Rest.....</b>	<b>151</b>
8.1	Application-, Database-, and Storage-Based Encryption .....	154
8.2	HOWTO Use DBMS_CRYPTO .....	155
8.3	HOWTO Use TDE to Encrypt Columns .....	163
8.4	HOWTO Encrypt Foreign Keys and Columns Used for Indexes.....	170
8.5	HOWTO Use TDE to Encrypt Tablespaces .....	171
8.6	HOWTO Manage TDE Master Keys .....	173
8.7	HOWTO Use HSMs and TDE .....	176
8.8	HOWTO Use TDE with External Tables (Oracle Data Pump) .....	178
8.9	HOWTO Keep Data Encrypted When You Export It Using Oracle Data Pump Utilities .....	179
8.10	HOWTO Encrypt Backups with RMAN .....	181
8.11	Discussion: Why Did Oracle Pick the TDE Approach? .....	184
<b>9</b>	<b>Standard Auditing.....</b>	<b>187</b>
9.1	HOWTO Enable Standard Auditing .....	188
9.2	HOWTO Use Audit Qualifiers.....	193
9.3	HOWTO Use Statement Auditing.....	198
9.4	HOWTO Use Object Auditing.....	200
9.5	HOWTO Use Privilege Auditing .....	202
9.6	HOWTO Audit for Unexpected Errors in the Network Layer .....	203
9.7	HOWTO Read Audit Records.....	204
9.8	HOWTO View What Is Currently Being Audited.....	207
9.9	HOWTO Use NOAUDIT .....	209
9.10	Discussion—Auditing and Performance.....	211

<b>10</b>	<b>Mandatory and Administrator Auditing.....</b>	<b>213</b>
10.1	HOWTO Use Mandatory Auditing.....	213
10.2	HOWTO Enable Administrator Auditing .....	216
10.3	HOWTO Use Syslog Auditing.....	218
<b>11</b>	<b>Fine-Grained Auditing.....</b>	<b>223</b>
11.1	H OWTO Define FGA Policies .....	225
11.2	HOWTO Manage FGA Policies .....	230
11.3	HOWTO Read FGA Tables and Views.....	231
11.4	Discussion: FGA Performance .....	232
<b>12</b>	<b>Auditing Before/After Values and Monitoring Selected Data .....</b>	<b>235</b>
12.1	HOWTO Use Triggers for Capturing Before/After Values.....	235
12.2	HOWTO Use Oracle Streams for Capturing Before/After Values.....	239
12.3	HOWTO Use the SCN and Flashback Queries .....	246
12.3.1	Notification Laws.....	246
12.3.2	Using Flashback Queries: An Example.....	247
12.3.3	Getting Versions Using Flashback .....	250
12.3.4	Prerequisites for Flashback .....	251
12.4	HOWTO Use Flashback Data Archive .....	252
12.5	Discussion: Do You Really Need the Before Values?.....	253
<b>13</b>	<b>Oracle Audit Vault.....</b>	<b>255</b>
13.1	HOWTO Add, Configure, and Manage Agents.....	261
13.2	HOWTO Add, Configure, and Manage Sources .....	264
13.3	HOWTO Add, Configure, and Manage Collectors .....	266
13.4	H OWTO Configure Audit Rules.....	270
13.5	H OWTO Configure and Manage the AV Server and the Warehouse .....	273
13.6	HOWTO View Audit Data within the AV Console .....	276
13.7	H OWTO Configure Alerts .....	278
13.8	HOWTO Understand Performance and Storage Impact.....	281
13.9	Miscellaneous Discussion—Auditing AV .....	282
<b>14</b>	<b>Database Activity Monitoring.....</b>	<b>285</b>
14.1	HOWTO Protect against SQL Injection.....	292
14.2	HOWTO Categorize and Identify Misuse and Intrusions.....	297
14.3	HOWTO Understand the Compliance Landscape .....	299
14.4	HOWTO Determine Whether You Need DAM or DAMP .....	306
14.5	HOWTO Analyze Impact on Performance.....	308
14.6	HOWTO Analyze Impact on Storage .....	310
14.7	Discussion: Identifying the Real User.....	312
<b>15</b>	<b>Privileges and Authorization.....</b>	<b>315</b>
15.1	HOWTO Manage Object and Column Privileges .....	315
15.1.1	G rant Option .....	317
15.2	HOWTO Manage System Privileges.....	324
15.3	HOWTO Use Roles to Manage Privileges .....	335

15.4 HOWTO Use Secure Application Roles.....	338
15.5 HOWTO Manage the PUBLIC Role.....	342
15.6 HOWTO Use Access Control Lists (ACLs) to Limit Access to Database Network Services.....	343
15.7 HOWTO Generate Entitlement Audit Reports.....	348
15.8 Discussion—SQL92_SECURITY .....	357
<b>16 Virtual Private Database .....</b>	<b>359</b>
16.1 HOWTO Use VPD Policies to Limit Access to Rows .....	359
16.2 HOWTO Use VPD Policies to Limit Access to Sensitive Column Data .....	364
16.3 HOWTO Use VPD Policies to Hide Sensitive Column Data .....	365
16.4 HOWTO Use Policy Groups .....	367
16.5 HOWTO Choose a Policy Type for Optimal Performance .....	372
16.6 HOWTO Review and Debug VPD Policies.....	374
16.7 Discussion—Using Secure Application Roles and VPD .....	378
<b>17 Oracle Database Vault .....</b>	<b>383</b>
17.1 HOWTO Use a Realm to Secure Data Access.....	384
17.2 HOWTO Use Command Rules to Secure User Activity.....	388
17.3 HOWTO Use Rule Sets, Factors, and Secure Application Roles .....	393
17.4 HOWTO Use Reports in DV .....	401
17.5 HOWTO Enable sysdba Connections.....	403
17.6 HOWTO Disable DV and Track Whether It Is Enabled .....	405
17.7 HOWTO Better Understand DV's Impact on Performance.....	410
17.8 Miscellaneous Discussion—Is Auditing Alone Enough?.....	411
<b>Appendix A Payment Card Industry (PCI) Data Security Standard (DSS) Version 1.1: Impact on Oracle Security Implementations.....</b>	<b>413</b>
<b>Appendix B Using an “All-in-One” Solution: An Example.....</b>	<b>425</b>
B.1 Discovery.....	426
B.2 Vulnerability Assessments.....	429
B.3 Change Tracking .....	431
B.4 Auditing .....	432
B.5 Database Activity Monitoring .....	435
B.6 Data Access Protection .....	438
B.7 Compliance .....	439
<b>Index.....</b>	<b>443</b>

---

# Foreword

---

In recent years, Oracle security has assumed a whole new meaning for many people in organizations around the world; we have seen the rise of regulatory requirements and worse still a huge rise in the reporting of data loss. While not from an Oracle database, the recent loss of two CDs in the United Kingdom containing all the child benefit details of over 25 million people could not be a more graphic example for people who want their data to remain secure. Securing Oracle databases is more important today than it was many years ago when I started dedicating my business and research life purely to thinking about and providing companies and the community with assistance in securing their data held in Oracle databases.

Companies have also been more widely reporting an increase in internal rather than external attacks to their systems. This is of the highest significance for the security of data held in an Oracle database as in today's world the use of perimeter security is of little value when it comes to securing the data. Unfortunately, most companies have open network architectures and most employees have indirect or direct access to the database (whether intended or not) due to open routing policies and also standard desktop builds. As we have seen, the biggest threat in recent times comes from internal attacks; this could again be intentional or unintentional. Remember, in the world of securing data, the adversary that the Database Administrator (DBA) has to compete with may not be a hacker; he or she may be a fellow employee who has too much access that allows damage and unauthorized changes to data and the database itself to occur, or any other of the myriad of situations that allow people unauthorized access, again intended or not.

Securing an Oracle database and the data held in it should be of utmost importance to all of the people within an organization—from the managers who write the checks to the DBAs, developers, security analysts, users, and owners of the data. Securing data held in an Oracle database is not “rocket science” but it is complex because the Oracle database itself is complex and infinitely configurable and also because the applications, data needs, and use are also infinite and different for each organization. Wow! This sounds like a big problem, doesn't it? If every system, application, configuration, use, people, and so on is different we clearly need best practices to secure Oracle and the data.

I should make a clear distinction here. The task of securing the “data” should be held separately from the task of securing the “Oracle software.” Why is this? Well, simply put, following a checklist or “tip sheet” is not good enough. We must ensure that this is done as practitioners of “securing data”; yes, securing data must come first and not securing the software; we must understand the data, understand its use, understand what I call its “flow”; how does the data get into the database, how does it leave the database, and where is it at rest. Only then can we start to secure the data using the tools provided by Oracle.

I am a believer in best practices and ideas—good ideas, not just simply ticking off checklists. I believe that if you understand your data you can secure it. I have helped define best practice and helped many clients secure their data. A methodology should teach the principles and obviously discuss the features and tools that Oracle provides, but it must also teach how to understand the risks to the data. Ron’s book is clearly written and focuses on the core technology available from Oracle to secure your data, but, importantly, it discusses why you should secure your data and then provides guidelines as to how you should do it using out-of-the-box tools. This is important as it means the book is useful to any customer of Oracle since Ron uses the techniques and tools provided with the Oracle license (additional cost options are discussed, as well such as Audit vault or Virtual Private Database). This is a practical book that any layman can follow and the core concepts are well explained. Did I mention that Oracle security is complex...? ; well Ron cuts to the quick and covers all the core issues. I like to think an ideal book teaches the reader the “how” and the “why,” which they can then apply to all aspects of the security of their data. I think Ron has achieved this. I was particularly impressed with the clear discussions on privileges; this has been my main bugbear with lots of customers; I see lots of sites with excessive privileges, serious privileges, wide-ranging access for lots of people; if we can teach users of data the importance of access to only the data they should access and only at the times they should access it, we would have made massive progress toward secure data. Enjoy this book but most importantly learn from it and use it to secure your data.

Pete Finnigan  
*Managing Director,*  
*PeteFinnigan.com Limited*

---

# Acknowledgments

---

I would like to thank my wife and kids for tolerating my vanishing acts during nights, weekends, and any other time that should have been spent with them and went instead into writing this book.

I would like to thank the whole crew at Guardium for helping me deal with the complex topics of security and Governance, Risk and Compliance (GRC) in large database environments every single day.

Finally, I would like to thank the many people I have worked with over the years on Oracle security—those that I have met as Guardium customers, and others who I have interacted with in Oracle forums. The many hundreds of such interactions helped me understand what is important from a very real and pragmatic point of view, and not merely from a “tooling” perspective. The list of people is too long to mention here but if you have ever taken the time to sit down with me and explain what you need, I thank you for that.

Ron Ben-Natan



---

# Author

---

**Ron Ben-Natan** has more than 20 years of experience developing enterprise applications and security technology for blue-chip companies. Ron is currently the chief technical officer at Guardium, the leader in database security and auditing. Prior to this he has worked for companies such as Merrill Lynch, J.P. Morgan, Intel, and AT&T Bell Laboratories. He is an IBM GOLD consultant with a PhD in computer science. Ron is an expert in distributed application environments, application security, and database security. He has authored 11 technical books including *Implementing Database Security and Auditing*. He speaks frequently at database and security seminars including sessions run by Oracle University.



## *Chapter 1*

---

# Introduction: How This Book Will Help You Be Secure and Compliant

---

The word Oracle means (from Wikipedia):

An **oracle** is a person or agency considered to be a source of wise counsel or prophetic opinion; an infallible authority, usually spiritual in nature. It may also be a revealed prediction or precognition of the future, from deities, that is spoken through another object (e.g.: runemal) or life-form (e.g.: augury and auspice). In the ancient world many sites gained a reputation for the dispensing of oracular wisdom: they too became known as “oracles”, and the oracular utterances, called *khrēsmoi* in Greek, were often referred to under the same name — a name derived from the Latin verb *ōrāre*, to speak.

A pretty good name for a database management system (DBMS). But there is another important bit of history behind the use of the word Oracle to name the database engine. Much before Oracle was a company as we know it today, Larry Ellison and Bob Miner, two of the founders of Software Development Labs (which later became Oracle), were working on a consulting project for the Central Intelligence Agency (the CIA in the United States). The CIA wanted to use a new Structured Query Language (SQL) that IBM had written a white paper about. The code name for the project was Oracle (the CIA saw this as the system to give all answers to all kind of questions intelligence analysts had). Larry Ellison and Bob Miner saw the opportunity to take what they had started as part of this project and market it. So they used that project’s code name of Oracle to name their new database engine and later the company.

Today, Oracle is the number one database engine based on any metric and it dominates many geographies and many verticals/industries. Perhaps the vertical that it dominates most is that of military organizations, agencies such as the CIA, National Security Agency (NSA), Federal Bureau of Investigation (FBI), and other organizations where security is of utmost

importance. This is part of the company's legacy and it is evident in the product. Maybe this origin is the reason that Oracle has more security-related functions, products, and tools (both built by Oracle as well as available as third-party products) than any other comparable DBMS in the world.

Unfortunately, the fact that these capabilities exist does not mean that they are always well-known and used correctly. In fact most users of the Oracle database, even those who have been working with Oracle for many years, are often familiar with less than 20 percent of the security mechanisms within Oracle. This leads sometimes to insecure Oracle environments and other times to implementation which use the wrong tools—meaning a lot of unnecessary work and solutions which require too much effort to sustain over time. One of the main goals of this book is to review the methods and tools available for securing Oracle so that when you have to implement any security-related requirement (be it access control, audit trails, configuration assessment, encryption, etc.), you know how to navigate the options, which tool to use for which scenario, and how to avoid common pitfalls.

## 1.1 Why Secure the Data?

Let's start with understanding why you must invest in Oracle security. This sounds like a silly question—but you'll have to answer this question in one form or another once you start asking for budgets. It's quite obvious why you need to secure the data—right? Everything you care about sits in a database (and if you're reading this book most likely a lot of it is in an Oracle database). Whether it is financial company data, data about customers, data about employees, credit card information—it is usually stored in a database. There are many other forms that this data can take—data in documents, data in e-mails, data in IM messages, etc. These do not usually live inside the database and there are other tools and techniques (not covered in this book) for securing these endpoints. These data elements are often permutations of data that was sourced from a database. Almost all information that you and your organization consider to be valuable resides in a database in some form or at some point in time. Obviously, you need to secure these “crown jewels.”

So far so good. But now let's start asking slightly different questions. Suppose that you did your analysis and go to your management with a proposal to secure the database that will cost \$50,000 (or Euros, or Pounds, or whatever your currency may be) per database. What if your proposal required you to add 10 to your headcount? Will it still be so obvious that you need to secure the data (to that level)? Management will most likely not want to have the data “so secure” at that point. If you map your requirements and request \$1000 per database; will it then be approved? That depends on what value this investment provides at a business level and what business problem this investment solves. Security, like any other IT investment needs to be justified and being able to answer the question of “why secure the data” (or the less simplistic variants of this question) is important.

At a very high level, the reasons for securing your data fall into two broad categories—you need to avoid a data breach and you need to remain in compliance with some internal or external set of requirements. Both data breaches and noncompliance can cost an organization dearly. A data breach can damage a company's brand, can lead to loss of customers, and always costs a lot of money—money spent on remediation, compensation, investigations, audits, notification, etc. Noncompliance too can be very costly. Noncompliance leads to fines, can lead to loss of a license to operate the business, can lead to continual expensive audits for many years to come, etc.

The justification for investing in securing Oracle is simple—it is a far lower cost than the cost involved with a data breach or noncompliance.

When you build your business case for elevating the security of your data you may have to justify it with a return on investment (ROI). When you build your business justification you should first list the essential elements that must be performed to achieve an acceptable level of security and compliance. You usually do not need to justify this part with an ROI. Security is in many ways like buying insurance. There is no ROI on buying home insurance. If nothing happens during that year (and normally nothing does) then you get no return on a sizable investment. But if your house burns down—well, then you'll be very happy you bought insurance. Security is similar—if there is an attempt to hack your database and it is foiled, your investment has paid off. What is usually more important than ROI is the total cost of ownership (TCO) of the proposal.

Once you have defined the essential elements that have to be implemented comes the second part—your implementation plan. This is where ROI comes in. Given a set of requirements, there are usually many ways to achieve them. Some may involve tools and others may involve manual work done by staff. At this point, you will have to justify your decisions to implement one method over another—and this is done by showing that one method has a much better ROI than another.

## Data Breaches

There are many resources and Web sites that track data breaches. For example, the Privacy Rights Clearinghouse keeps a chronology of data breaches that have been reported that involve data that can be useful to identify thieves—including Social Security numbers, account numbers, and driver's license numbers. This list enumerates more than just database-related breaches—it lists all kinds of data-related breaches (which include also things like stolen laptops, etc.). This list only covers reported incidents—and not all incidents are reported by any stretch. The list covers only incidents reported in the United States. This includes a running total of the number of records that have been compromised. Here too, this number is understated because in many incidents the number of affected records is unknown and therefore not counted. Just to give you an idea of the magnitude of the problem—between 2005 and June 2008 there have been at least 225 million records compromised as part of the reported data breaches in the United States. Another good list is the Data Breach Blog that is maintained by SC magazine.

Here is a small sampling of some known incidents involving database breaches (not necessarily Oracle databases):

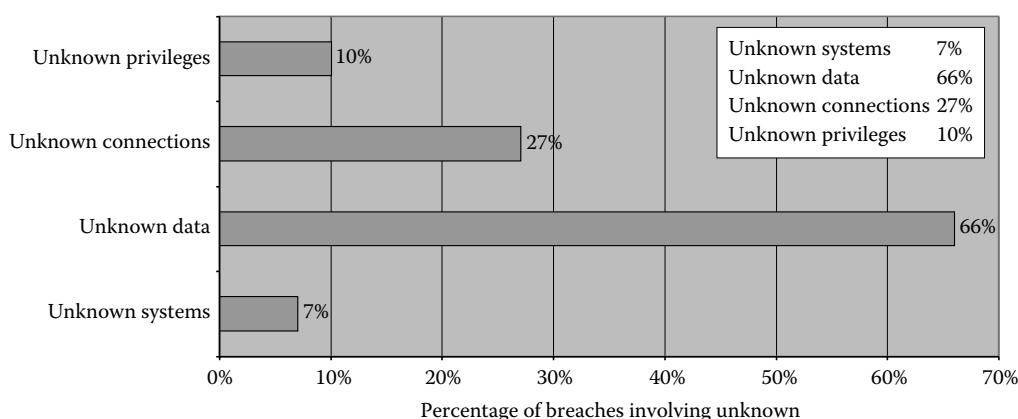
- In February 2003 the BBC reported an attack on a database that held credit card accounts where the attacker gained access to more than five million Visa and Mastercard accounts.
- In October 2004 a hacker compromised a database containing sensitive information on more than 1.4 million California residents. The breach occurred on August 1 but was not detected until the end of the month. The database in question contained the names, addresses, Social Security numbers, and dates of birth of caregivers and care recipients participating in California's In-Home Supportive Services (IHSS) program since 2001. The data was being used in a UC Berkeley study of the effect of wages on in-home care and was obtained with authorization from the California Department of Social Services. The hacker had reportedly taken advantage of a unpatched system and although officials declined to state which vendor's database was the subject of the attack they did report that it was a "commercially available product with a known vulnerability that was exploited."

- In August 2005 an Air Force spokesman reported that a hacker tapped into a U.S. military database containing Social Security numbers and other personal information for 33,000 Air Force officers and some enlisted personnel.
- In April 2006 Computerworld reported on a case in which an employee at Progressive Casualty Insurance wrongfully accessed information on foreclosure properties she was interested in buying. There was no hacking involved—just misuse of insider privileges. When the incident was discovered the company sent out letters informing people that confidential information had been accessed by this employee who was fired. The incident was discovered because a local woman complained about receiving calls from a Progressive agent inquiring about her house being under foreclosure—not because there was any database monitoring or auditing in place.
- In September 2006, the virtual reality game SecondLife reported that one of its databases containing unencrypted user information was breached.
- In October 2006, an official with the Illinois Ballot Integrity Project, a not-for-profit organization dedicated to the correction of election system deficiencies, reported that the organization hacked into the voter database for the 1.35 million voters in the city of Chicago and could have not only stolen private information but also create election problems by modifying status and data.
- In June 2007, eWeek reported that Fidelity National Information Services, an electronic payment processor, fired a database administrator (DBA) after they found that the DBA stole and sold customer data exposing as many as 2.3 million bank and credit card records. The DBA, who worked at the company's Certegy Check Services unit, sold the information to a data broker who in turn sold some of it to direct marketers. These activities led to customers receiving marketing solicitations—which was how the incident was exposed.
- In July 2007, credit card information, names, addresses, and phone numbers were hacked from a Western Union database.
- In August 2007, Monster.com reported that details of over 1.6 million job seekers and information on 146,000 subscribers to usajobs.gov residing in a resume database managed by monster.com has been stolen.
- In September 2007, TD Ameritrade reported that information on 6.3 million customers was stolen from one of its database. The stolen information included names, addresses, and e-mail addresses plus a variety of account activity information. The company reported that it discovered and eliminated unauthorized code. Although it did not provide further details, it is likely that this was done by a privileged user. TD Ameritrade said it discovered the breach after customers said they had received spam offering unsolicited investment advice.
- In January 2008, a hacker broke into a database of the Davidson Companies, a financial services firm. The hacker obtained the names and Social Security numbers of practically all of the company's clients as well as information relating to account numbers and balances.
- In March 2008, Harvard University reported that the database containing summaries of GSAS applicant data had been compromised and that about 10,000 of 2007 applicants' personal information may have been compromised. At least 6000 Social Security numbers were exposed and a compressed 125 MB file containing the stolen data is available through BitTorrent, a peer-to-peer network. The file included server backups of three databases and a note was attached which reads “maybe you don't like it but this is to demonstrate that persons like ... (admin of the server) in that they don't know how to secure ...”.

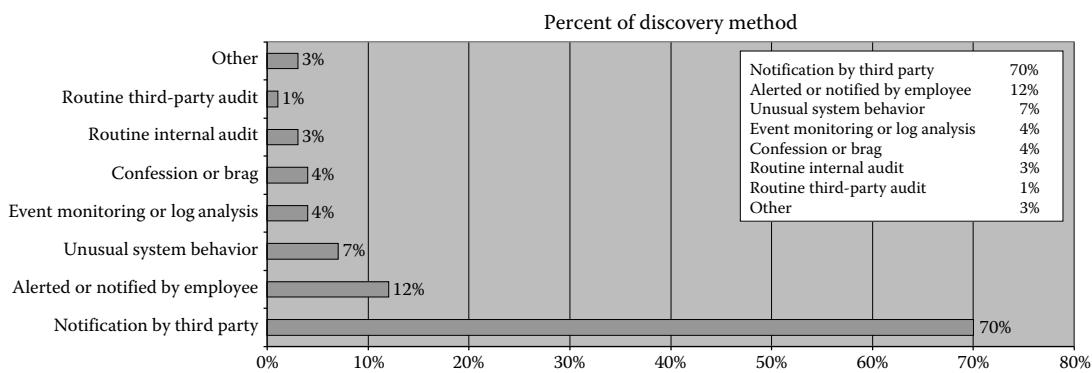
- In May 2008 Computerworld reported that a professional penetration tester (an ethical hacker) managed to hack his way through to a major FBI crime database within a mere six hours.

Data breaches have, unfortunately, become part of our daily lives. In addition to looking at long lists of incidents, it is instrumental to look at some commonalities. One of the best sources for learning about these is the 2008 Data Breach Investigations Report—a study published by the Verizon Business RISK Team. This report draws from over 500 forensic engagements handled by the Verizon Business Investigative Response team over a period of four years. The report provides statistics on how breaches occur, where they occur most (in terms of verticals), what methods were used, and more. For example, the report lists that most breaches resulted from a combination of events rather than a single event but in almost all cases some form of error contributed to a compromise, whereas less than a quarter of attacks exploited vulnerabilities. This shows how important it is to know how to use the security options correctly. Of the incidents due to vulnerabilities, 90 percent of the vulnerabilities exploited by these attacks had patches available for at least six months prior to the breach (hence, you'll learn how to read Critical Patch Updates in Chapter 2).

The report also finds that nine of ten breaches involved something that is unknown to the owner—the most common one being data that was not known to exist on the compromised system. This is shown in Figure 1.1. The report calls these recurring situations as the “Achilles heel in the data protection efforts of every organization.” This is perhaps the most important theme in data breaches—you cannot protect that which you do not know about and you cannot secure what you cannot see. The importance of visibility—visibility into where sensitive data resides, visibility into who or what is accessing sensitive data, visibility into controls—is perhaps the most important element that must exist for a database to be secure. Two other very disturbing facts that the report finds is that most breaches go undetected for a long time and are discovered more often by a third party than the breached organization and that most



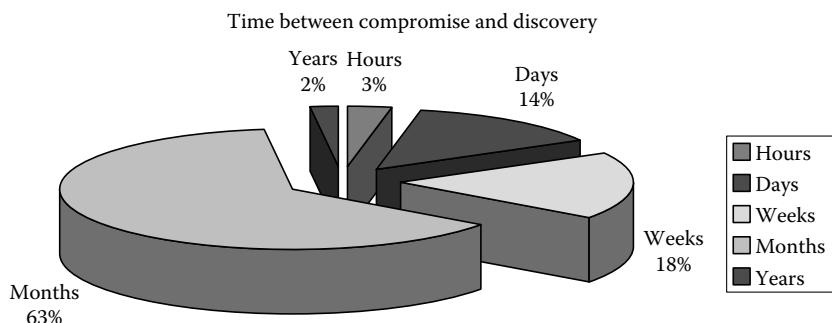
**Figure 1.1** Breaches involving unknown factors.



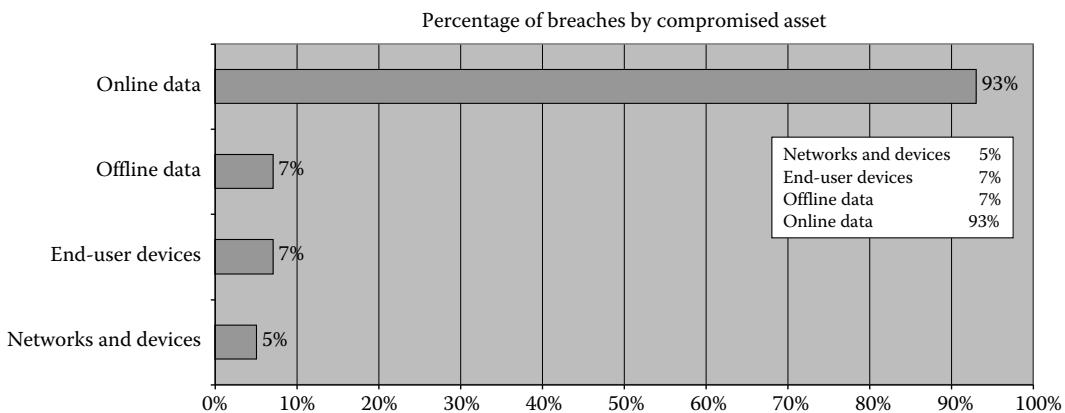
**Figure 1.2 Discovery of breaches.**

attacks are low to moderate in difficulty—i.e., the attacker does not have to work too hard. Specifically, the report finds that

- 66 percent of attacks involved data the victim did not know was on the system.
- 75 percent of the breaches were discovered by a third party and not the breached organization—as shown in Figure 1.2. The report goes on to present the data shown in Figure 1.3 regarding the time until discovery. This is a very serious finding—it shows that there is a great deficiency in monitoring and auditing. There is a huge difference between a breach that lasts for a day versus a breach that goes on for months, and between a breach that is discovered and handled versus one where the victims (the people whose data is stolen) cannot defend themselves because they don't even know they are victims.
- 83 percent of the attacks were not difficult to perform.
- 87 percent of the attacks could have been avoided through reasonable controls.



**Figure 1.3 Time until discovery.**



**Figure 1.4 Which data is most often targeted?**

Finally, if you have any doubt about the importance of database security in the battle against data breaches, the report brings some statistics on the assets that were compromised. Data sits in many places and takes many forms. Data can be in a database but can also reside on USB keys. Data that resides in a database also exists in backups and taken offline. As Figure 1.4 shows, compromises to online data repositories occurred more than five times more often than all other asset classes combined!

## Compliance

It would be wonderful if we invested in security because we were so disturbed by the many data breaches we've seen and because we always want to do the right thing—but the truth is that we usually invest in security because we're told to do so. Most of the investment in Oracle security is made because of the need to comply with a regulation or a requirement.

There are two kinds of compliance requirements—internal and external. Internal requirements are policies that are defined within the company. They include policies set by the information security or internal audit groups and they are usually derived from industry best practices, from a regulation, or from preparation for an external audit. External requirements derive from a regulator or from external auditors. There are numerous examples of regulations that affect database security—including Sarbanes Oxley (and its derivatives), the Payment Card Industry Data Security Standard (PCI DSS), various data privacy laws, and many others. Some of these regulations are industry-specific, some are national (i.e., affect only companies operating in a certain country), and others relevant to companies of a certain size. Most of these regulations do not directly set requirements related to database security—they need to be interpreted and mapped to IT terms and these interpretations determine what is required to be in compliance. Luckily, most of these regulations have been around long enough to have a standard interpretation and one that is consistent with requirements set by industry best practices.

Compliance is a very important driver—especially when it comes from an external source. If you need to comply with a certain regulation, it is very hard to shut down a project for lack of

funding or other priorities. This is the great power of compliance and the reason that so much of database security is driven by compliance requirements. When you know you have to pass an audit or face certain penalties and when you know that the auditor is an external entity, you are usually bound to make the investment and elevate the security of your data. Compliance and regulation is a very positive factor in data security and much of the improvements in the last five years can be attributed to auditors and the processes they put us through.

## 1.2 Taxonomy of Best-Practice Database Security

Before you start reading the HOWTOs, you should realize that securing an Oracle database is not a trivial task. Oracle has many capabilities. The Oracle SQL language is richer than most DBMSs. You can connect to Oracle using a great many networking libraries and authentication methods. There are many thousands of packages and procedures in any database. There is a Java virtual machine and an Hypertext Transfer Protocol (HTTP) server. You can call out from the database using external procedures or various utility packages. All these are examples of functionality that is great when you look at productivity and building applications. But from a security standpoint, the more options and capabilities a server has, the harder it is to secure and monitor it properly. The reason is simple—each such option can be used by an attacker to gain unauthorized access. Securing Oracle requires a lot of know-how and involves doing work in a variety of categories.

There is a large body of knowledge by now on what activities are required to secure Oracle and to comply with regulations and requirements. There are checklists you can follow and you will learn about them in Chapter 2. This is good—it means you can adhere to a set of best practices and achieve security and compliance by investing in the following:

- Discovery—you can't secure that which you don't know. You need to have a good mapping of your assets—both of your instances and of your sensitive data. Plus, you need to automate discovery because the state of this “asset map” six or twelve months into the future will be different from what it is today.
- Vulnerability and configuration assessment—you need to assess the configuration of your databases to ensure that they don't have security holes in them. This verification includes both the way the database is installed on the operating system and the configuration options within the database itself. You need to verify that you are not running versions of the database with known vulnerabilities.
- Hardening—the result of an assessment is often a set of recommendations. This is the first step in hardening the database. Other elements of hardening involve removing all functions and options that you do not use.
- Change auditing—once you have a hardened configuration you must continually track it to ensure that you don't digress from a secure configuration. You do this using change auditing tools which compare snapshots of the configurations (at both the operating system level and at the database level) and alert you when a change is made that may affect the security of the database.
- Database activity monitoring—although changes can and should be tracked using change auditing, you can also use database activity monitoring to alert on changes made through an SQL interface. Additionally, database activity monitoring lets you detect intrusions and misuse, detect fraud, and discover problems at real-time, limiting your exposure considerably.

- Auditing—audit trails must be generated and maintained for database activity that may have an impact on security, integrity, or on access to sensitive data.
- Authentication, access control, and entitlement management—not all data and not all users are created equal. You must authenticate users, you must ensure full accountability per user, and you must manage privileges to limit access to data. You need to enforce these privileges even for the most privileged database user. You also need to review entitlement reports periodically as part of an audit process.
- Encryption—use encryption to render sensitive data unreadable. Use encryption so that an attacker cannot gain unauthorized access from outside the database. This includes both encryption of data-in-transit so that an attacker cannot eavesdrop at the networking layer and gain access to the data when it is sent to the database client as well as encryption of data-at-rest so that an attacker cannot use the media files and extract the data there.

## 1.3 Using HOWTOs to Secure Oracle

Oracle is the largest and most-functional DBMS platform today. As such it has many aspects that need to be secured—the bigger the “surface area” of a server, the more work is required to secure it properly. Luckily, Oracle has more security features than any other database and often there is more than one way to address a problem. This book aims to help you understand what these different options how, when to use them, and how to use them. To achieve this goal the book is structured as HOWTOs—specific “recipes” that show you how to use each of these security functions in the context of Oracle 11g and Oracle 10g. Most of these HOWTOs focus on the tools that exist within the Oracle database but some extensions are provided when relevant.

The HOWTOs are organized into chapters by security category and by topic. In Chapter 2, you will learn how to harden the database—that is, remove unnecessary components and choose configuration settings that make it harder for an intruder to gain unauthorized access. You’ll learn how to choose a checklist and how to create a secure configuration baseline. You’ll learn how to read Oracle Critical Patch Updates and the importance of patching. In Chapter 3, you will focus on listener security and see what a secure listener configuration looks like. Chapter 4 deals with account security. You’ll learn about user management, password policies and complexity, and what the standard logon process looks like.

Because Chapters 6 through 8 all deal with topics that use cryptographic functions, Chapter 5 provides an overview of cryptography, including encryption, digests, and digital signatures. In this chapter you’ll also learn how to use an Oracle wallet and how to use Oracle Wallet Manager and oracleki for generating and managing certificates and other secrets. With this background you’ll learn about the various authentication options available with and without Advanced Security Options in Chapter 6, how to encrypt database communications in Chapter 7, and how to encrypt the data within the database in Chapter 8.

The next set of chapters deal with the important topic of auditing. In Chapter 9, you’ll learn how to use standard auditing, how to use and manage audit trails, and about their impact on the database. Chapter 10 extends this with a discussion on administrator auditing and Chapter 11 augments with fine-grained auditing. In Chapter 12, you’ll learn some advanced techniques for auditing before and after values and for discovering what data was extracted from the database.

Auditing is usually the first step in any Oracle security implementation. Additionally, there are some basic requirements that an audit trail must satisfy for it to be considered valid—requirements

that are architectural in nature. For example, you need to ensure separation of duties—that is, the DBAs need to be audited and cannot have control over the audit trail. For this reason, an external product—Oracle Audit Vault has been created. You’ll learn about this product and how to use it in Chapter 13. Because monitoring and auditing access to the database is such an important topic, an entire space has emerged for addressing this need called database activity monitoring. In Chapter 14, you’ll learn what these third-party tools provide and when you can use them.

Chapters 15 through 17 deal with access control and with enforcing privileges. In Chapter 15, you’ll learn about the Oracle privilege model, how to authorize access, and how to audit what privileges have been assigned. You’ll learn about roles and about secure application roles that can be used by an application to implement authorization directly within the database. In Chapter 16, you’ll learn how to use Virtual Private Database to enforce row-level access control and how to mask sensitive data. Finally, in Chapter 17 you’ll learn how to use Oracle Database Vault to limit users even if they have system privileges and how to implement an external security overlay.

The organization of the chapters follows a logical order of implementation steps, but most of the chapters can be read independently. You can read through the chapters in the order that suits the tasks you need to perform. The HOWTOs within the chapter can usually also be read independently and where possible, are self-contained. So, feel free to read the book in order or skip through it based on your needs at work.

## *Chapter 2*

---

# Hardening the Database

---

System hardening is the process by which you securely configure a system to protect it from unauthorized access. System hardening is necessary in any system that has a range of configuration options and is viable in any system that has enough security measures to make them suitable for usage in security-oriented environments. Oracle database falls into both of these categories.

The purpose of system hardening is to eliminate as many security risks as possible. This is done by removing all nonessential elements from the system and by selecting configuration options that limit access and reduce risk. As Oracle has evolved, more and more options have become available and these options offer new ways to access data—sometimes by unauthorized users if used inappropriately. The larger the footprint and capabilities of a system, the harder it is to harden and the more security risks may be present. Therefore, as the Oracle database grows in size and functionality, hardening becomes even more important. Luckily, as Oracle evolves, there are also more and more security options available that you can use to secure the data—Chapters 3 through the end of the book outline these capabilities and how you should use them. But first, you need to harden the database.

Oracle hardening covers a wide range of activities and involves many types of configuration options. The most important guideline is that if there is a feature that you do not use, remove it. The fact that you don't use something does not mean that an attacker won't. The smaller the surface area of a system, the more secure it is. Examples of this include:

- Remove or lock predefined accounts that you do not use and change the password for accounts you do use that have a predefined password (Oracle 11g already comes configured that way).
- Remove predefined roles that you do not use.
- Remove components in the database software that you do not use.
- Remove options that you do not use—for example, remove EXTPROC from your listener if you do not use external procedures.
- Remove privileges from PUBLIC that you do not require.

Because Oracle has so many capabilities and configuration options, hardening is usually an exercise that involves hundreds of activities. Coming up with a list of these required activities is a monumental task. Luckily, you don't have to come up with this list. Lists have been created and

entire books are dedicated to this topic—so all you have to do is pick a source—the topic of the first HOWTO of this chapter.

## 2.1 HOWTO Choose a Hardening Guideline

Hardening of any complex system involves many little details. The more a system can be configured, the lengthier the list of tasks you need to perform (or validate) to create a hardened configuration. The list for Oracle is long, but openly available and free.

There are two documents that provide very mature guidelines for implementing a secure Oracle configuration and that you should look at when forming your standard hardening process. One is the Database Security Technical Implementation Guide (STIG) developed by Defense Information Systems Agency (DISA) for the Department of Defense (DOD) and the second is the Center for Internet Security (CIS) Benchmark for Oracle developed by the CIS. Both are excellent and very comprehensive; use one of them (or both) rather than develop your own hardening checklist.

### Database STIG

STIGs are documents published by the DISA to assist in improvement of the security of DOD information systems. There are numerous STIG documents—all of them are accessible at <http://iase.disa.mil/stigs/stig/index.html>. The checklists can be downloaded from <http://iase.disa.mil/stigs/checklist/index.html>. The Database STIG focuses on relational databases. The Database STIG has a generic section which outlines guidelines relevant to any database management system (DBMS) and has an Oracle-specific section which adds steps relevant for Oracle only. The sections within the general document address:

1. Integrity
  - a. Software integrity
  - b. Database software development
  - c. Ad-hoc queries
  - d. Multiple services host systems
  - e. Data integrity—including file integrity, software baseline, and file backup and recovery
2. Discretionary access control
  - a. Account control
  - b. Authentication
  - c. Database accounts
  - d. Authorizations
  - e. Protection of sensitive data
  - f. Protection of stored applications
  - g. Protection of database files
3. Database auditing
  - a. Audit data requirements
  - b. Audit data backups
  - c. Audit data reviews
  - d. Audit data access
  - e. Database monitoring

4. Network access
    - a. Protection of database identification parameters
    - b. Network connections to the database
  - c. Database replication
  - d. Database links
5. Operating system (OS)
    - a. File access
    - b. Local database accounts
    - c. Administrator accounts
    - d. OS groups

The Oracle-Specific Policy and Implementation appendix specifically addresses:

6. Oracle access control
    - a. Oracle identification and authentication
    - b. Oracle connection pooling
    - c. Secure distributed computing
    - d. Oracle administrative connections
    - e. Oracle administrative OS groups
  - f. Default accounts
  - g. Default passwords
  - h. Oracle password management requirements
7. Oracle authorizations
    - a. Predefined roles
    - b. System privileges
    - c. Object privileges
    - d. Administration of privileges
8. Oracle replication
  9. Network security
    - a. Encrypting network logins
    - b. Protecting network communications
  - c. Listener security
  - d. XML DB protocol server
10. Oracle Intelligent Agent/Oracle Enterprise Manager (OEM)
  11. Oracle account protections
12. ARCHIVELOG
13. Securing SQL\*Plus
  14. Protecting stored procedures
  15. Oracle trace utility
  16. Auditing in Oracle—includes standard auditing, fine-grained auditing, mandatory auditing, and architectural discussions
  17. File and directory permissions at the OS level
  18. Critical file management—including control files, redo log files, and data files
  19. Optimal Flexible Architecture (OFA)
20. Initialization parameters
21. Miscellaneous OS requirements—including Unix, Window, and z/OS

The Database STIG is published as an unclassified document and is made available to all. DISA also publishes a set of evaluation scripts and these can help you check the security strength of your database—download these from <http://iase.disa.mil/stigs/SRR/index.html>.

## CIS Oracle Benchmark

The CIS ([www.cisecurity.org](http://www.cisecurity.org)) publishes the CIS Benchmark for Oracle as part of a set of benchmarks, scoring tools, software, data, and other services that are made public as a service to all users worldwide. You can download the benchmark from <http://www.cisecurity.org/bench-oracle.html>. The recommendations contained in the Oracle benchmark result from a consensus-building process that involves the leading Oracle security experts. The CIS benchmark takes the form of a checklist partitioned into a number of sections. Within each section is a list of items that should be validated. Each such item includes a description of the item, the action or recommended setting for parameters, comments, which Oracle version it applies to, and whether it is relevant to Unix, Windows, or both. The main sections in the CIS Oracle benchmark are

1. OS-specific settings
2. Installation and patch
3. Oracle directory and file permissions
4. Oracle parameter settings
5. Encryption-specific settings
6. Startup and shutdown
7. Backup and disaster recovery
8. Oracle user profile setup settings
9. Oracle user profile access settings
10. Enterprise Manager/Grid Control/Agents
11. Items relevant to specific subsystems
12. General policy and procedures
13. Auditing policy and procedures
14. Appendix A—additional settings

Both documents take a broad approach to hardening. They do not have a narrow interpretation that hardening only involves certain configuration settings, removing default components, locking users, etc. They provide a full checklist that also includes what activities should be audited, where separation of duties is required, what activities need to be performed, etc. Of the two—the STIG puts even more focus on the general implementation, process, roles that need to be involved in securing an Oracle environment, etc.

### Two Things to Remember about Choosing a Hardening Guideline

1. Don't build your own checklists—hardening an Oracle database is no longer an art; there are good mature guidelines for you to choose from such as the CIS Oracle benchmark or the Database STIG.
2. Use these documents not only as a hardening guide but also as the basis for putting together a complete Oracle security implementation. These documents outline configuration settings but also outline process, procedures, and what to focus on. In many ways, all the chapters in this book explain how to use Oracle tools to implement what these two documents suggest that you do.

## 2.2 HOWTO Use a Vulnerability Assessment Tool

Using hardening checklists is simple but tedious. From a pure hardening perspective, the checklists contain many checks and modifications that you need to perform and these can be automated. In fact, without automation this tasks quickly becomes unmanageable—especially if you have tens and hundreds of instances and they do not all conform to a single “gold build.” Tools that you can use to automate this process are called vulnerability assessment (VA) tools or vulnerability scanners.

For almost any type of system there are VA tools and Oracle is no exception. VA tools will scan your database instances and come back with a report showing what changes you need to perform to make your database more secure. These results are presented in the form of a security report where each problem is classified and a recommendation provided for what changes you need to make (e.g., see Figure 2.1). These checks and recommendations usually cover the items specified in the various hardening checklists—meaning that a VA tool can save you most of the tedious work involved in reviewing your databases and their alignment with the checklists.

There are many VA tools for Oracle including AppDetective, AppSentry, Guardium, IPLocks, and NGS Squirrel. Some of these tools are stand-alone VA scanners and some tools are part of a larger suite of products that address multiple aspects of Oracle security. From a pure VA perspective all these tools scan your database to recommend changes you need to make to harden your instances. The tools that are integrated within a larger suite have an advantage—in the same way that STIG and CIS take the wider interpretation to securing the database environment and define practices for auditing, practices for review, etc. (see the previous section), so do these suite-based tools. This allows you to become secure and fully compliant within a single implementation thus saving a lot of time.

VA tools perform many types of checks. These checks can be classified into three main groups:

1. Checks for software vulnerabilities
2. Checks for misconfigurations
3. Checks for misuse of the database

All of these checks are necessary to check for vulnerabilities in your database. An attacker can gain unauthorized access to your database by using a code vulnerability that exists because you have not patched your server using the latest Critical Patch Update (CPU) (an example of type 1), through the use of a default account that has not been locked and still has a default password (an example of type 2), because you have REMOTE\_OS\_AUTHENT set to true (an example of type 2), or because everyone knows the password for SYSTEM and multiple people make use of this account constantly (an example of type 3).

Checking for vulnerabilities (of all types) is done using a multipronged approach. Some things can be checked from the outside-in and other checks are done from within the database. VA tools have multiple modes in which they work to provide you with the full picture. Many checks need to be performed from the inside. For example, to check for bad configurations the VA tool needs to be able to access V\$PARAMETER. To check for bad privilege assignment (as an example—too many privileges assigned to PUBLIC can be a serious vulnerability), VA tools need certain SELECT privileges to the catalog. These tools come with scripts that grant these privileges to a user or a role (that is then assigned to a user you create for the tool to use). Review these scripts when you evaluate the tool to ensure that it does not assign itself too many privileges. The better VA tools also inspect and check files at the OS level. For example, lax file permissions or a wrong owner or group used for important Oracle files (such as data files, software files, configuration files, and log files) are a serious vulnerability. Contents of files such as sqlnet.ora can affect how your database behaves and checking a configuration must include checks on files, on registry values (on Windows), environment variables, etc. Finally, comprehensive checks will often include inspecting output from scripts and OS commands—e.g., inspecting the

Guardium: Security Assessment Results - Windows Internet Explorer

<https://192.168.3.208:8443/saResultsViewer.do?method=view&viewerType=assessmentResults&viewedTaskId=18saResult>

Showing 76 of 76 results (0 filtered)

Current filtering applied:

Servers: - Show All -  
Scores: - Show All -  
Types: - Show All -

Filter / Sort  
 Controls

Showing 76 of 76 results (0 filtered)

Assessment Test Results

Test Name	DataSource	Pif	Severity	Reason
Conf. DBA Profile FAILED_LOGIN_ATTEMPTS Are Limited	ORACLE cm_ora10_ostrich	Fail	Critical User profile [DEFAULT_MONITORING_PROFILE] setup parameter FAILED_LOGIN_ATTEMPTS found out of defined threshold value	
Conf. DBA Profile PASSWORD_LIFE_TIME Is Limited	ORACLE cm_ora10_ostrich	Fail	Recommendation: The FAILED_LOGIN_ATTEMPTS parameter is not set. A high number of failed login attempts can indicate that an unauthorized user is trying to gain unauthorized access to your data. We recommend that you set this parameter in order to limit the number of failed login attempts before locking the user's account.	
Conf. DBA Profile PASSWORD_VERIFY_FUNCTION Is Implemented	ORACLE cm_ora10_ostrich	Fail	Critical Found active profile MONITORING_PROFILE. DEFAULT with PASSWORD_VERIFY_FUNCTION not implemented	
Audt. Default Accounts Password Channed	ORACLE cm_ora10_ostrich	Fail	Recommendation: No Password Verification Routine has been implemented. We recommend that you implement a password function to prevent the use of weak passwords.	
Pri. No Authorizations To System Level Privileges	ORACLE cm_ora10_ostrich	Fail	Critical 1 active predefined users have default passwords.	
Pri. No Individual Users With Any Table Privileges	ORACLE cm_ora10_ostrich	Fail	Recommendation: Some predefined Oracle user accounts are still enabled and still have the Oracle default password. These predefined Oracle users and passwords are well-known to anyone familiar with Oracle, and represent one of the easiest entry points for attacks and data theft/damage. We recommend that you remove any predefined Oracle user accounts that are not absolutely required, and we strongly recommend that you change the passwords for any of these users who are required.	

Certificate Error

Showing 76 of 76 results (0 filtered)

Done

Figure 2.1 Sample recommendation report from an Oracle VA scanner.

output of orapatch to ensure that you have the latest CPUs installed to address known code vulnerabilities. At the end of the day, a VA tool is the only way to ensure that you have hardened your databases properly and that you remain compliant over time.

### The e Things to Remember about Using an Assessment Tool

1. Some VA tools are stand-alone scanners and others are part of a larger database security suite. If you're implementing the full set of recommendations presented by within the Database STIG or within the CIS checklist you should consider the suite products that can also support your auditing implementation, intrusion detection implementation, separation of duties, etc.
2. VA scanners check both vulnerabilities and CPUs installed (or that should be installed) as well as the configurations of your database.
3. Some of the checks that you need to perform are at the OS level. Make sure the VA tool you choose can perform checks on file ownership, file permissions, etc.

## 2.3 HOWTO Create and Maintain a Secure Configuration Baseline

Once you have finished hardening your database, you have a tight configuration, but you need to ensure that it remains tight and does not degrade over time. There are two things you can do to ensure a sustained secure configuration—(1) run assessments on a scheduled basis to find new vulnerabilities as they are created, and (2) create a baseline for the configuration once you are happy with it and track any changes from this configuration using an alert that needs to be reviewed and approved. The best practice suggests that you do both of these because they complement each other.

If you have a VA tool, then running an assessment periodically is easy. All VA tools have a scheduler that allows you to test your databases every month or every week. Most VA tools also have a “diff report” that shows you changes to the assessment results between one run and another—so once you are happy with the results of a scan you can monitor only the differences in future scans.

The second set of tools is called change tracking tools. These tools create a baseline of your configurations and alert you on any change that is made. Baselines are created by computing a digest on configuration elements and then periodically recomputing them to see if there are any changes. Digests (or hash values) can be computed on files (to ensure for example that no one modifies the software files themselves), on the output of a script (e.g., the output of a script that greps certain values from sqlnet.ora), on Structured Query Language (SQL) result sets (e.g., the output of a query that checks assignment of system privileges), etc. A change tracking tool tells you when there is a deviation from the hardened configuration.

Change tracking tools will always be part of a database security implementation. These tools are required to comply with the broad hardening checklists because this is the only way you can ensure that no one replaces your files with Trojan versions. They are also the most effective way to ensure that scripts run periodically are not used as a point-of-compromise; tracking changes made to scripts is much simpler and more effective than reviewing audit trails that show what a script did.

Because you'll have access to these tools if you're responsible for database security, use them in the context of VA change tracking tools—once you've completed your hardening process use them to

ensure that you don't deviate from this standard. If there are changes over time (and there always will be changes) you can reauthorize and update the baseline and track changes from that point on. Look for a VA tool that includes a change tracking tool to ensure sustained and continuous compliance.

### The e Things to Remember about Creating and Maintaining a Secure Configuration Baseline

1. Change tracking tools have multiple uses within an Oracle security implementation. One of these is to create and track a secure baseline following the hardening phase. VA tools that incorporate a change tracking tool offer you more options in terms of continued compliance.
2. Baselines are generated by creating digests that uniquely identify a file or a result of a script. Any change to the underlying entity will cause the digest to change and a change report to be issued. Digests act as digital fingerprints of the underlying entity.
3. Baselines include digests for files that should not change, digests for the result of an OS script, digests for the result of a query, digests for values of environment variables or registry entries, and more.

## 2.4 HOWTO Understand Critical Patch Updates

Always install patches to security issues as soon as they are available from Oracle. When code vulnerabilities are discovered (and there always will be code vulnerabilities—any large piece of software has bugs), Oracle issues patches—you must install these patches to remove these vulnerabilities from your environment.

Security patches come in the form of Oracle CPUs. An Oracle CPU is a bundle of patches that are released on a quarterly basis to fix security issues. CPUs have been around since 2005 (before that there were called Security Alerts) and they come out at 1 p.m. Pacific time on the Tuesday closest to the 15th of January, April, July, and October. The fact the CPUs are released at a known date is important—it allows you to plan ahead and define change management windows accordingly. Before CPUs were used, security alerts were issued when issues were discovered and fixed. This made installing these security patches very difficult and sometimes as people were in the processing of testing one fix, another one would be announced. Knowing when the patches are published makes it easier for you to build a process around applying them.

The Oracle CPU includes fixes for all Oracle software components. One patch is released per version of the database, application server, Enterprise Manager, etc. This has changed as of 2007 with the introduction of the n-apply process (more on that later). Patches are also released for Oracle E-Business Suite, PeopleSoft, Siebel, and other applications. Patches for the database are cumulative so that the latest CPU includes fixes for all earlier CPUs unless stated otherwise.

Each CPU includes a set of patches, an advisory, preinstallation notes, and release notes. The CPU advisory contains information which helps you evaluate the impact of the fixed vulnerabilities so that you may assess the criticality of issues and how quickly you need to install patches on production systems. The advisory includes a set of risk matrices—one per software system—in which Oracle reports on the risks of the discovered issues.

## Reading a CPU Advisory Risk Matrix

A CPU advisory contains a database risk matrix. The risk matrix helps you understand the risk of discovered issues in terms of loss of confidentiality, loss of integrity, and loss of availability—the three dimensions that can be affected by security vulnerabilities.

Figure 2.2 shows a sample from the database risk matrix of the April 2008 CPU. The matrix summarizes the list of vulnerabilities fixed within the CPU and for each one provides information about risk. Each vulnerability is given a vulnerability number composed of four characters—the first two characters represent the system and the last two are an incremental numeric code starting from 01. Database vulnerabilities are tagged as DB##. Each CPU has its own numbering scheme so the vulnerability number is unique within a CPU. Sometimes a vulnerability in another system affects users of the Oracle database—in this case that vulnerability will be included in the database risk matrix so that the risk matrix is self-contained and you do not need to read the entire CPU if you only care about the database. As an example, in Figure 2.2 EM01 is listed within the database risk matrix even though the vulnerability is in Enterprise Manager. In such a case the vulnerability number appears in italics.

The risk matrix contains information about which component the vulnerability is in, what protocol is required to exploit the vulnerability, what package or privilege is required to exploit the vulnerability and whether an attacker can exploit the vulnerability from a remote node without

Oracle Database Risk Matrix													
Vuln#	Component	Protocol	Package and/or Privilege Required	Remote Exploit without Auth.?	CVSS VERSION 2.0 RISK (see Risk Matrix Definitions)							Last Affected Patch set (per Supported Release)	Notes
					Base Score	Access Vector	Access Complexity	Authentication	Confidentiality	Integrity	Availability		
EM01	Oracle Enterprise Manager	Local	None	No	6.6	Local	Medium	Single	Complete	Complete	Complete	9.0.1.5+	
DB01	Advanced Queuing	Oracle Net	Execute on SYS.DBMS_AQ	No	5.5	Network	Low	Single	Partial	Partial	None	9.0.1.5+, 9.2.0.8, 9.2.0.80V, 10.1.0.5, 10.2.0.3	See Note 1
DB02	Change Data Capture	Oracle Net	Execute on DBMS_CDC_UTILITY	No	5.5	Network	Low	Single	Partial+	Partial+	None	10.1.0.5, 10.2.0.3, 11.1.0.6	
DB03	Core RDBMS	Oracle Net	Create Session	No	5.5	Network	Low	Single	Partial+	Partial+	None	9.0.1.5+, 9.2.0.8, 9.2.0.80V, 10.1.0.5, 10.2.0.3	See Note 1
DB04	Oracle Secure Enterprise Search or UltraSearch	Oracle Net	Execute on WKSYS.WK_QUERY or WKSYS.WK_QUERYAPI	No	5.5	Network	Low	Single	Partial+	Partial+	None	9.0.1.5+, 9.2.0.8, 9.2.0.80V, 10.1.0.5, 10.2.0.3	See Note 1
DB05	Oracle Spatial	Oracle Net	Execute on SDO_UTIL	No	5.5	Network	Low	Single	Partial	Partial	None	10.1.0.5, 10.2.0.3	
DB06	Oracle Spatial	Oracle Net	Execute on SDO_Geom	No	5.5	Network	Low	Single	Partial	Partial	None	9.0.1.5+, 9.2.0.8, 9.2.0.80V, 10.1.0.5, 10.2.0.3	See Note 1
DB07	Oracle Spatial	Oracle Net	Execute on SDO_IDX	No	5.5	Network	Low	Single	Partial	Partial	None	9.0.1.5+, 9.2.0.8, 9.2.0.80V, 10.1.0.5, 10.2.0.3, 11.1.0.6	See Note 1
DB08	Authentication	Oracle Net	None	Yes	5.0	Network	Low	None	Partial	None	None	11.1.0.6	
DB09	Oracle Net Services	Local	None	No	4.6	Local	Low	None	Partial+	Partial+	Partial+	9.2.0.8, 10.1.0.5, 10.2.0.3	See Note 2
-----													

Figure 2.2 Sample from a database risk matrix in a CPU.

first being authenticated. The next thing provided per vulnerability is a Common Vulnerability Scoring System (CVSS) score. CVSS is a standardized method for assessing security vulnerabilities in all systems. CVSS has been used by Oracle since October 2006—before then vulnerabilities were scored using a proprietary scoring system.

CVSS scores range between 0.0 and 10.0 with 10.0 being the worst possible score (implying the worst possible vulnerability, and the highest risk). Oracle currently uses CVSS version 2.0 to derive a base score and this score is included in the risk matrix. Scores are computed using a calculator available at nvd.nist.gov and shown in Figure 2.3. The score is computed after you enter a selection for each of the entries. When Oracle scores vulnerabilities they key in the answers to the base score

National Vulnerability Database CVSS Scoring - Windows Internet Explorer  
 http://nvd.nist.gov/cvss.cfm?calculator  
 nist cvss calculator

**National Vulnerability Database**  
 automating vulnerability management, security measurement, and compliance checking

Vulnerabilities Checklists Product Dictionary Impact Metrics Data Feeds Statistics  
 Home ISAP/SCAP SCAP Validated Tools SCAP Events About Contact Vendor Comments

**Common Vulnerability Scoring System Calculator**

This page provides a calculator for creating CVSS vulnerability severity scores. Please read the [CVSS standards guide](#) to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score. A [concise](#) form of this page is available to CVSS experts.

**Environmental Score Metrics**

This section addresses metrics that describe the effect of a vulnerability within an organization's environment. These metrics must be calculated separately for each organization.

Organization specific potential for loss (CollateralDamagePotential): Undefined

Percentage of vulnerable systems (TargetDistribution): Not Defined

**Base Score Metrics**

These metrics describe inherent characteristics of the vulnerability. All of these metrics must be filled in to perform any CVSS scoring.

**Exploitability Metrics**

Related exploit range (AccessVector): Undefined

Attack complexity (AccessComplexity): Undefined

Level of authentication needed (Authentication): Undefined

**Impact Metrics**

Confidentiality impact (ConfImpact): Undefined

Integrity impact (IntegImpact): Undefined

Availability impact (AvailImpact): Undefined

Impact value weighting (ImpactBias): Normal

**Temporal Score Metrics**

These metrics describe elements about the vulnerability that change over time. If all of these values are left as 'Undefined', the environmental score will be based on the base score.

Availability of exploit (Exploitability): Undefined

Type of fix available (RemediationLevel): Not Defined

Level of verification that vulnerability exists (ReportConfidence): Undefined

Figure 2.3 NIST CVSS calculator.

metrics and get a base score which then goes into the CPU risk matrix. Note that although the CVSS scores desire to be a single number through which you can tell right away whether it is critical or not, CVSS ratings depend on Oracle's interpretation of the problem. To fully understand how Oracle fills in the entries for computing CVSS scores see Metalink Note 394487.1.

It is especially important to understand how Oracle interprets the effect on confidentiality, availability, and integrity. The CVSS calculator allows you to enter one of three values—None, Partial, or Complete. Complete is defined to mean that the impact is to the “whole system.” The question is what exactly this means. One interpretation of Complete can be that the impact is to all Oracle software running on a system and the other can be that the impact is to all software running on the system—OS and all. Oracle chooses to use the latter interpretation. This means for example that if the vulnerability affects all data within the database—all tables in all schemas—the CVSS score is still going to be based on a “Partial” selection in the calculator. Having chosen the first interpretation would have created higher CVSS scores. This is not uncommon and is the way most vendors interpret CVSS levels but you should understand this when you determine what your threshold for risk is.

To distinguish between cases in which the impact can be limited versus wide (which were the terms used before adoption of CVSS), Oracle introduces another level called Partial + (see Figure 2.2). A vulnerability that affects a limited set of resources (e.g., a specific database table) will have Partial in the risk matrix. If the vulnerability affects a wide range of resources (e.g., all tables in the database) then the impact is logged as Partial +. Note that this does not impact the CVSS score—the score in both cases is computed using Partial! It is therefore important to look at the entries in the risk matrix and not only the CVSS score.

## Database n-Apply CPUs

Until 2007 CPUs included one monolithic patch for the database every quarter. It was impossible to install a subset of the fixes. This changed with the CPU of July 2007 when the n-apply patch format was introduced to CPUs. n-apply CPUs have the following benefits:

1. Customized patch conflict resolution.
2. Elimination of rollbacks and reinstallation of CPU patches that are already installed to limit downtime. CPUs are still cumulative but the installation process has been improved.
3. Ability to install only parts of the CPU fixes rather than have to install the whole CPU.

An n-apply CPU is a zip file that contains molecules and are installed using opatch. Each molecule is a group of security fixes. A molecule is an independent patch that does not conflict with any of the other molecules within the CPU.

### Five Things to Remember about CPUs

1. CPUs are released every three months at specific dates so that you can plan ahead for testing and deploying fixes.
2. CPUs include security fixes to discovered vulnerabilities. It is very important to apply security fixes because this is the best way to protect yourself from attacks that exploit such vulnerabilities. Note that once a CPU has been released it is easier for an attacker to find the vulnerability because there is some information (e.g., the component) listed per vulnerability—therefore, apply CPUs as soon as possible while going through a standard testing and change management process.

*(continued)*

**(continued)**

3. CPUs include a risk matrix that allows you to determine how critical these fixes are for your environments. Risk matrices list which components are affected and how severe the issue is—all are there to help you decide whether or not you can tolerate the risk if you can't immediately apply the CPU.
4. CPUs are cumulative—if you apply the latest CPU you have included all fixes for all previous vulnerabilities too.
5. The new n-apply CPU packaging allows you to deploy fixes to only some vulnerabilities versus the old format which was delivered as a single patch.

## 2.5 HOWTO Sanitize Data for Test

As part of the section on Database Software Development the Database STIG states:

(DG0076: CAT II) The DBA will ensure that export data from a production database used to populate a development database has all sensitive data such as payroll data or personal information, etc., removed or modified prior to import to the development database.

Production databases usually have better access controls and are monitored with higher scrutiny than development databases. This only makes sense if you assume that the sensitivity levels of data in development and test databases are lower than those of production servers. Developers have access to development and test databases but usually not to production servers (outside a “break glass,” or emergency maintenance event). Therefore, you must sanitize data before you can load it to development servers.

Sanitizing data is hard. Not only do you need to know where all your sensitive data resides, you also need to change a lot of data in a way that does not invalidate your application and your tests. You can't change data randomly. If you have foreign keys (whether they are constraints in the database or managed by the application) you need to make sure that keys are preserved and that all references stay intact. Some fields encode application logic. For example, many account numbers follow a certain scheme in which there is a checksum or some algorithm for generating numbers—not any random set of digits is a valid account number. Another example is credit card numbers. When you sanitize data you need to preserve this property or your application will break. And finally, you need to make sure that after you sanitize the data, columns used for indexes maintain a statistical distribution which is close to that of the production data. Otherwise, performance tests on the sanitized data may not be indicative of the performance you will have on the production data. All in all, sanitizing test data is a hard problem to solve—that is why many development organizations don't do it and simply use the production data as-is. This is in violation of any security guideline, and tools do exist to help you accomplish this task.

The first tool available to you is the Data Masking option in Enterprise Manager. Follow these steps once you have enabled the data masking option in Enterprise Management Grid Control:

Step 1: Log onto EM

Step 2: Click on the Targets tab and the Databases subtab.

Step 3: Select the database where you want to mask sensitive data.

The screenshot shows the 'Create Format' dialog box. At the top, there are fields for 'Name' (SSEC) and 'Description' (Random SSEC numbers). Below this is a section titled 'Format Entries' with a sub-instruction: 'Define masking format by adding one or more format entries of different types.' A table lists a single entry: 'Type' Random Digits, 'Description' Digits Length Range: 11 - 11, with 'Add', 'Edit', and 'Remove' buttons. Under 'Post Processing Function', the value is 'scott.insertdashes'. A note says 'Specify a function here (for example: scott.checksum) to process the masked data.' Below this is a 'Sample Masked Data' section showing five sample values: 64995606900, 88835798601, 99750212802, 19018244903, and 56254386503. At the bottom right are 'Cancel' and 'OK' buttons.

**Figure 2.4 Defining a masking format.**

- Step 4: Click on the Administration link. At the bottom right is a Data Masking section. The Definitions link lets you set the masking rules. The Format Library link lets you build up a library of data masking formats. A data masking format defines how you want to mask sensitive data. For example, you can use random numbers or random characters. For more advanced functions (and usually you need more advanced obfuscation techniques for real applications) you need to build PL/SQL routines. For example, if you want to test an application you probably will want data that may have indexes with the same statistical distribution as the real data—using random characters will certainly not preserve cardinalities.
- Step 5: Click on the Format Library link. This takes you to a page with a list of formats available to you. As this product matures, there will be many predefined formats for the most common identity patterns but for now click on Create.
- Step 6: Figure 2.4 shows a format for masking social security numbers. These numbers have a pattern of [0–9]{3}-[0–9]{2}-[0–9]{4}. In this case you can choose Random digits from the drop down and click on Go. Enter 1 as the start and 11 as the end to ask Oracle to create 11 random digits for you. Click on OK. Then, you'll have to call a PL/SQL procedure to put in the dashes in locations 4 and 7, so enter the name of your procedure and click OK.
- Step 7: Click on the Masking Definitions breadcrumb at the top to take you back to the masking definitions screen. Click on Mask to create a masking job. Give the job a name and select the database where the sensitive data resides.
- Step 8: Click on Add to define which column to mask and how to mask it. Put in the schema name and click on the search icon. Select the sensitive column from the list (or multiple columns). Click on Define Format and Add.
- Step 9: Click on Import From Library because you have already created the masking format. Select your format and click Import. You've now selected where the sensitive data is and how to mask it as shown in Figure 2.5. Click on Next.

The screenshot shows the 'Mask: Masking Definition' page. At the top, there is a navigation bar with four steps: Masking Definition (highlighted), Impact Report, Schedule, and Review. Below the navigation bar, the job details are listed:

- Name: MASKING\_DEF\_33
- Database: emrep
- Description: replace SSEC numbers

**Columns**

Add columns you want to mask and define masking format for each column. Foreign key columns are automatically added to maintain referential integrity.

Dependent columns are columns that do not have foreign key constraints defined, but reference a masked column due to application level constraints. You can manually add dependent columns to a masked column. Removing a column from this list will remove all foreign key and dependent columns.

Buttons: Define Format, Remove, Add, Cancel, Save, Step 1 of 4, Next.

Select	Owner	Table	Column	Data Type	Format	Foreign Key Columns	Dependent Columns
<input type="checkbox"/>	SCOTT	EMP	SSEC	VARCHAR2(11)		0	0

**Foreign Key Columns**

Owner	Table	Column	Parent Owner	Parent Table	Parent Column
No foreign key columns					

**Dependent Columns**

Owner	Table	Column	Parent Owner	Parent Table	Parent Column	Remove
No dependent columns added						

**Figure 2.5** Defining which sensitive data to mask.

Step 10: The script is generated. Review the generation information and click Next. Enter the host credentials where the script will be stored. Enter a schedule if the job should be scheduled or select Immediately. Click on Next.

Step 11: Oracle produces the script and you can review it as shown in Figure 2.6. Submit the masking job. You can then review the status (and possible errors) of masking jobs as shown in Figure 2.7.

The screenshot shows the 'Mask: Review' page. At the top, there is a navigation bar with four steps: Masking Definition, Impact Report, Schedule, and Review (highlighted). Below the navigation bar, the job details are listed:

- Database: emrep
- Logged In As: sysman
- Number of Tables: 1
- Columns: 1
- Job Name: MASKING\_JOB\_33
- Job Schedule: Run Immediately

**Script**

The script summary is a list of the database commands that will be used to mask the selected columns. The full script is a PL/SQL script that includes functions, procedures, and other commands needed during the masking operation. The full script will be created when you submit the job and will be executed by the job to perform the masking operation.

Buttons: View, Script Summary, Full Script, Save Full Script, Cancel, Back, Step 4 of 4, Submit.

```

Set echo off
set feedback off
set serveroutput on
set pagesize 0

spool C:\OracleHomes\db10g\ dbs\masking33.log

-- Script Header Section
-- =====
-- functions and procedures

CREATE OR REPLACE PROCEDURE mgmt$mask_sendMsg (msg IN VARCHAR2) IS
  msg1 VARCHAR2(1020);
  len INTEGER := length(msg);
  i INTEGER := 1;

```

Buttons: Cancel, Back, Step 4 of 4, Submit.

**Figure 2.6** Reviewing the masking script and submitting the masking job.

Database Instance: emrep > Masking Definitions >

**Masking Definition: MASKING\_DEF\_33**

---

A data masking definition specifies what columns to be masked and the format of masked data.

Name	MASKING_DEF_33	Database	emrep	Description	replace SSEC numbers
<b>Columns</b>					
Owner	Table Name	Column Name	Format		
SCOTT	EMP	SSEC	xx		
<b>Foreign Key Columns</b>					
Owner	Table Name	Column Name	Parent Owner	Parent Table	Parent Column
No foreign key columns					
<b>Dependent Columns</b>					
Owner	Table Name	Column Name	Parent Owner	Parent Table	Parent Column
No dependent columns					
<b>Masking Jobs</b>					
Job Name	Status	Ended	Elapsed Time (seconds)		
MASKING_JOB_35	Succeeded	Apr 18, 2008 10:11:39 AM (UTC-04:00)	35		
MASKING_JOB_33	Problems	Apr 18, 2008 10:05:11 AM (UTC-04:00)	32		

---

Figure 2.7 Reviewing the status of masking jobs.

As an example, if you use random digits and a PL/SQL procedure that adds the dashes, and if the data looks like:

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	SSEC
7499	ALLEN	SALESMAN	7698	20-FEB-81	100	80	30	111-11-1111
7876	ADAMS	CLERK	7788	23-MAY-87	101		20	222-22-2222
7844	TURNER	SALESMAN	7698	08-SEP-81	102	0	30	333-33-3333
7782	CLARK	MANAGER	7839	09-JUN-81	103		10	444-44-4444
7566	JONES	MANAGER	7839	02-APR-81	104		20	000-00-0000
7839	KING	PRESIDENT		17-NOV-81	105		10	888-88-8888
7900	JAMES	CLERK	7698	03-DEC-81	106		30	777-77-7777
7788	SCOTT	ANALYST	7566	19-APR-87	108		20	555-55-5555
7369	SMITH	CLERK	7902	17-DEC-80	110		20	999-99-9999
7934	MILLER	CLERK	7782	23-JAN-82	111		10	666-66-6666

Then, your masked data will look like:

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	SSEC
7844	TURNER	SALESMAN	7698	08-SEP-81	102	0	30	745-80-4600
7782	CLARK	MANAGER	7839	09-JUN-81	103		10	857-87-7301
7934	MILLER	CLERK	7782	23-JAN-82	111		10	522-91-5302
7566	JONES	MANAGER	7839	02-APR-81	104		20	658-07-6603
7839	KING	PRESIDENT		17-NOV-81	105		10	886-14-8204
7788	SCOTT	ANALYST	7566	19-APR-87	108		20	661-97-3505

7369 SMITH	CLERK	7902 17-DEC-80	110	20	995-32-6006
7900 JAMES	CLERK	7698 03-DEC-81	106	30	821-79-0807
7876 ADAMS	CLERK	7788 23-MAY-87	101	20	420-95-5508
7499 ALLEN	SALESMAN	7698 20-FEB-81	100	300	375-11-5609

The Data Masking option is a new product and therefore has only rudimentary formats. With time the masking format library will grow and you will get logic-preserving and statistically preserving formats. However, there is a large set of third-party tools that perform this function and have a mature set of operators and formats. Examples include Princeton Softech (now IBM Optim), Application, Solix, and HP/Outerbay.

### The e Things to Remember about Sanitizing Test Data

1. You must sanitize sensitive information if you generate test data by copying real data from production systems.
2. Sanitizing data is far from trivial—you cannot simply replace data with random strings or numbers. You have to preserve application logic which is often coded into data and you must preserve statistical distribution for performance tests to be valid.
3. You should use tools to sanitize data—use either the data masking pack that is now part of Enterprise Management Grid Control or use third-party tools.

## 2.6 Discussion: Defense in Depth

All modern information security is founded on a concept called defense in depth. Defense in depth involves multiple layers of defense that increase the cost of an attack and places multiple barriers between an attacker and your computing resources. Multiple techniques and systems help mitigate the impact when one component of the defense is compromised or circumvented. The deeper an attacker tries to go the harder it gets. In addition to protecting your resources better, by the mere fact that there are multiple layers, defense in depth naturally provides areas in which you can put systems that can monitor and identify intrusions. This can often buy time to detect and respond to a breach and reduce its impact.

The term “defense in depth” is derived from a military strategy called defense in depth (also known as deep defense or elastic defense). This military strategy seeks to delay the advance of an attacker rather than prevent the advance. This buys time and causes enemy casualties—so rather than defeating an attacker with a single line, defense in depth relies on the tendency of an attack to lose momentum over a period of time or when it has to cover a larger area. The defender can yield some lines and territories causing the attacker to spread. This allows the defender to identify and mount counterattacks on the attacker’s weak points.

Defense in depth is considered the only viable strategy for information systems. The reason is that there is no such thing as a perfect security layer. Anything can be cracked and every system has bugs. Moreover—the quality of both security and attacks are highly correlated with their cost. The goal of a good IT security implementation is to create an environment in which an attack will cost too much to be worth it—and do it all within a reasonable budget. Therefore, relying

on a single super-duper security system just does not work. Instead, build multiple good (but perhaps not perfect) security layers. This has been described in an excellent paper called “Defense in Depth—A practical strategy for achieving information assurance in today’s highly networked environments” published by the National Security Agency (NSA)—<http://www.nsa.gov/snac/support/defenseindepth.pdf>.

Securing Oracle environments must follow the same strategy. The hardening checklists clearly discuss multiple types of activities that, if done in concert, implement a best practice in Oracle security. Remember that the security techniques available within the database can (and should) be augmented with security systems sitting outside the database—be they network security systems, database activity monitoring systems or host security. Always think of the main thesis of defense in depth—don’t rely on one layer only.



## *Chapter 3*

---

# Securing the Listener

---

Most Oracle activity occurs over the network. Oracle clients connect to oracle servers over a communication protocol such as Transmission Control Protocol/Internet Protocol (TCP/IP). Oracle clients and servers communicate using an Oracle protocol called the Transparent Network Substrate (TNS). TNS packets travel as TCP/IP payload. Setting up connections to the database is performed by the TNS listener—or listener for short. The listener manages network connections and as such, is the entry point to most database connections. It must be secured properly. Without the listener, your database will not be serving applications. Therefore, one important aspect of securing the listener involves ensuring that no one hijacks your listener, that no one shuts down your listener, and ensuring that no one cripples your listener by making it do a lot of work that you don't want it to do (e.g., writing trace information).

The listener has a number of components:

1. The \$ORACLE\_HOME/bin/tnslsnr executable itself.
2. The Listener Control Utility (\$ORACLE\_HOME/bin/lsnrctl) used to administer the listener, configure logging and tracing, set passwords, etc.
3. The \$ORACLE\_HOME/network/admin/sqlnet.ora configuration file which defines the networking parameters used by the listener.
4. The \$ORACLE\_HOME/network/admin/listener.ora configuration file which defines the IP addresses and ports that the listener should be listening on, the instances for which it is listening using a list of services or SIDs, and additional configuration options for that listener—options such as passwords, log levels, etc. You can change these options either by directly editing the listener.ora file or by using lsnrctl (which modifies this file). In both cases remember that you may need to bounce the listener for the changes to take effect.
5. The \$ORACLE\_HOME/network/admin/tnsnames.ora configuration file used to map TNS service names to connection descriptors so that a database client can use a convenient name to connect to the listener rather than having to use IP addresses, ports, protocol types, etc.

The listener is a highly privileged process and has an important role involving the execution of new processes at the operating system (OS) level, the loading of dynamically linked libraries (DLLs), etc.

For example, if you are running Oracle on Unix in dedicated mode then for every new connection that is received, the listener spawns a new OS process. Without this, there would be no server processes to service client connections.

To understand this better let's trace what happens at the OS level when the listener fields a new TNS connection. To see which system calls the listener performs you have to know the pid for the listener process:

```
[root@rh4u2x32p ~]# ps -ef | grep tns
oracle10 3 093      1  0 Aug07 ?        00:00:01
/home/oracle10/oracle/product/10.2.0/db_1/bin/tnslsnr LISTENER -inherit
```

If you're on Linux use the strace command to see all system calls for this process (on Solaris and AIX use truss and on HPUX use tusc):

```
[root@rh4u2x32p ~]# strace -f -p 3093
```

The -f parameter tells strace to also list system calls by child processes spawned by the listener process. Now create a new TNS connection by connecting to the database over TNS (it doesn't matter whether the connection is local or over the network):

```
[oracle10@rh4u2x32p ~]$ sqlplus scott@on0rh4
SQL*Plus: Release 10.2.0.1.0 - Production on Wed Aug 8 02:46:39 2007
Copyright (c) 1982, 2005, Oracle. All rights reserved.

Enter password:

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
SQL>
```

Listing 3.1 lists the important system calls that occur while setting up this connection. The listener is process 3093. It accepts the socket connection by calling the accept system call. It then forks a new process—process 15889. This process execs the Oracle server—you can see the actual executable that is called by this process. This is what the dedicated server mode means—a new server process per connection. The listener and the server process communicate as part of the setup and eventually the listener closes all file descriptors involving this connection—the connection continues to be maintained only by the server process.

Because the listener needs to spawn server processes it normally runs as the Oracle instance owner. It can thus do a lot of damage to an instance if it “falls into the wrong hands.” For example, in November 2000, Oracle added the ADMIN\_RESTRICTIONS option as a listener security patch to resolve a potential vulnerability through which an attacker could set the trace file or the log file location. Through this capability an attacker could, as an example, overwrite the data files, overwrite log and audit trails, or even write to files such as login.sql and glogin.sql and cause Structured Query Language (SQL) to be executed by the database administrator (DBA) the next time they log in.

The different categories of attacks that you need to consider when securing the listener are:

1. Denial-of-service (DoS) attacks against the listener, the database, or even the OS, including
  - a. Stopping the listener
  - b. Creating large trace files consuming CPU and disk resources
2. Using the listener to write to the file system

```
[root@rh4u2x32p ~]# strace -f -p 3093
...
[pid 3093] accept(10, {sa_family=AF_INET, sin_port=htons(33896),
sin_addr=inet_addr("192.168.222.128")}, [16]) = 13
[pid 3093] read(13,
"1\0\0\1\0\0\0\0019\1,\f\1\10\0\177\377\177\10\0\0\1"..., 8208) = 265
[pid 3093] clone(Process..
[pid 15889] execve("/home/oracle10/oracle/product/10.2.0/db_1/bin/oracle",
["oracleon0r4231", "(LOCAL=NO)"], /* 24 vars */) = 0
[pid 15889] write(17, "NTP0 15889\n", 11 <unfinished ...>
[pid 3093] read(16, <unfinished ...>
[pid 3093] write(15, "A\0\0\0", 4 <unfinished ...>
[pid 15889] read(14, <unfinished ...>
[pid 3093] write(15, "(ADDRESS=(PROTOCOL=tcp) (DEV=13) ("..., 65
<unfinished ...>
[pid 15889] read(14, <unfinished ...>
[pid 3093] <... write resumed> = 65
[pid 15889] <... read resumed> "(ADDRESS=(PROTOCOL=tcp) (DEV=13) ("..., 65) = 65
[pid 15889] close(17 <unfinished ...>
[pid 3093] close(15 <unfinished ...>
[pid 15889] close(14 <unfinished ...>
[pid 3093] close(16 <unfinished ...>
[pid 15889] read(13, <unfinished ...>
[pid 15889] write(13, "\0 \0\0\2\0\0\0\0019\f\1\10\0\177\377\1\0\0\0\0
AA\0\0"..., 32 <unfinished ...>
[pid 3093] close(13 <unfinished ...>
```

### **Listing 3.1 Trace on the listener process on Linux running in dedicated mode.**

3. Using the listener to run code at the OS level or running external code called from within the database
4. Using the listener to extract information about the database configuration that can help in further attacking the databases

The good news is that securing the listener is a relatively simple thing to do. To fully secure a listener, follow these steps:

1. Apply all security patches and CPUs which fix any listener vulnerabilities
2. Secure access through lsnrctl using a password or Local OS Authentication
3. Limit the ability to change listener properties through lsnrctl by using admin restrictions
4. Limit the hosts from which connections to the listener are accepted
5. Limit the use of EXTPROC and access to OS executables
6. Turn on logging and monitor the log files
7. Secure the listener configuration files and executables through file permissions, ownership, and change tracking

Let's move on to the HOWTOs that describe these steps in detail.

## **3.1 HOWTO Secure Access to lsnrctl**

The first and most important aspect of securing the listener is limiting the ability to control the listener through lsnrctl. You must make sure that an attacker cannot hijack the listener using lsnrctl.

For example, if an attacker can control your listener using lsnrctl from their own host they can shut down the listener causing your database to be unavailable to all but bequeath connections. Luckily, it is very simple to limit the ability to control the listener through lsnrctl. Even better—in Oracle 10g and 11g the default installation comes with a secure set of parameters. So really, this HOWTO is more important if you still have Oracle 8i or 9i running, but it is still useful to understand some of the finer points involving lsnrctl security.

If you are running Oracle 8i or 9i then your listener may not have been assigned a password. To see this, simply connect to the listener and run the status command:

```
-bash-3.00$ lsnrctl
LSNRCTL for IBM/AIX RISC System/6000: Version 9.2.0.1.0 - Production on 03-MAR-2008
15:13:01
Copyright (c) 1991, 2002, Oracle Corporation. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> status
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=ORCL) ) )

STATUS of the LISTENER
-----
Alias L                      LISTENER
Version                     TNSLSNR for IBM/AIX RISC System/6000: Version
9.2.0.1.0 - Production
Start Date                  03-MAR-2008 15:12:51
Uptime                      0 days 0hr. 0min. 12sec
Trace Level                 off
Security                  OFF
SNMP O                      N
Listener Parameter File     /home/oracle9/ora92/network/admin/listener.ora
Listener Log File           /home/oracle9/ora92/network/log/listener.log
Listening Endpoints Summary...
  ( DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=ORCL) )
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=cardinal.guardium.com)(PORT=1521) ) )

Services Summary...
Service "on9cardi.guardium.com" has 2 instance(s).
  Instance "on9cardi", status UNKNOWN, has 1 handler(s) for this service...
  Instance "on9cardi", status READY, has 1 handler(s) for this service...

The command completed successfully
LSNRCTL>
```

If you see **Security OFF** (as shown above) you're in trouble. It means that there is no password set for the listener and anyone can connect to the listener using lsnrctl without authentication—even remotely from their own host. Once connected to the listener an attacker can (for example) stop your listener. To connect and stop this listener (running on the host cardinal) from another host (crow):

```
bash-3.00$ uname -a
AIX crow 3 5 000FFA7F4C00
bash-3.00$ lsnrctl
LSNRCTL for IBM/AIX RISC System/6000: Version 10.2.0.1.0 - Production
on 03-MAR-2008 15:30:05
Copyright (c) 1991, 2005, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.

LSNRCTL> stop on9cardi
Connecting to
```

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=cardinal)(PORT=1521))(CONNECT
_DATA=(SERVICE_NAME=on9cardi.guardium.com)) )
```

The command completed successfully

Notice that the listener on cardinal was stopped by someone running on crow without being authenticated on cardinal and without having any special privileges (in fact—all you need is an Oracle installation of your own).

Because lsnrctl controls the listener, access to lsnrctl must be secured. Depending on which version of Oracle you're running, there are different options (and different defaults) for securing the listener. If you are running Oracle 10g or 11g the listener is secure by default using an option called Local OS Authentication. If you use lsnrctl on a default installation of 10g or 11g and run the status command you will see the following, e.g., on 10g:

```
bash-2.03$ lsnrctl
LSNRCTL for Solaris: Version 10.2.0.1.0 - Production on 03-MAR-2008 16:38:23
Copyright (c) 1991, 2005, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> status
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=loki.guardium.com)(PORT=1521)) )

STATUS of the LISTENER
-----
Alias L                      LISTENER
Version                     TNSLSNR for Solaris: Version 10.2.0.1.0 - Production
Start Date                  03-MAR-2008 07:59:48
Uptime                      0 days 8 hr. 38min. 37 sec
Trace Level                 off
Security                    ON: Local OS Authentication
SNMP O                      FF

Listener Parameter File
/data/oracle10/oracle/product/10.2.0/db_1/network/admin/listener.ora
Listener Log File
/data/oracle10/oracle/product/10.2.0/db_1/network/log/listener.log
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=loki.guardium.com)(PORT=1521)) )
 ( DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=ORACLE)) )

Services Summary...
Service "on0loki0" has 1 instance(s).
  Instance "on0loki0", status UNKNOWN, has 1 handler(s) for this service...
Service "on7loki0" has 1 instance(s).
  Instance "on7loki0", status UNKNOWN, has 1 handler(s) for this service...
Service "on8loki0" has 1 instance(s).
  Instance "on8loki0", status UNKNOWN, has 1 handler(s) for this service...
Service "on9loki0" has 1 instance(s).
  Instance "on9loki0", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
```

If you try to administer these listeners remotely you will get an error:

```
bash-3.00$ lsnrctl
LSNRCTL for IBM/AIX RISC System/6000: Version 10.2.0.1.0 - Production on 03-
MAR-2008 15:53:37
```

```

Copyright (c) 1991, 2005, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> status ON1VIREO
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=vireo.guardium.com)(PORT=1521 ))(CONNE
CT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=on1vireo ) )

TNS-01189: The listener could not authenticate the user

```

Local OS Authentication means that you can only control the listener if you are logged on to the account on the host where the listener is running. This is the most secure option for the listener and you should configure all 10g and 11g instances in this way.

### Setting a Listener Password: Must for 8i and 9i, an Option for 10g

You already saw that the default with 10g and 11g is Local OS Authentication. If you have a listener that uses Local OS Authentication you can also enable a password for it using the change\_password command or by editing listener.ora. All these methods are identical for 8i, 9i, 10g, and 11g. Let's look at an example with 10g:

```

LSNRCTL> change_password
Old password:
New password:
Reenter new password:
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=ORCL) ))
Password changed for LISTENER
The command completed successfully
LSNRCTL> save_config
Connecting to (DESCRIPTION =(ADDRESS =(PROTOCOL=IPC)(KEY=ORCL) ))
Saved LISTENER configuration parameters.
Listener Parameter File
/home/oracle10/product/10.2.0/db_1/network/admin/listener.ora
Old Parameter File
/home/oracle10/product/10.2.0/db_1/network/admin/listener.bak
The command completed successfully

```

If you use change\_password and check the status of the listener security you will see that it has changed to Password or Local OS Authentication:

```

LSNRCTL> status
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=ORCL) ))

STATUS of the LISTENER
-----
Alias L                      LISTENER
Version                      TNSLSNR for IBM/AIX RISC System/6000: Version
10.2.0.1.0 - Production
Start Date                   03-MAR-2008 16:23:01
Uptime                       0 days 0hr. 2min. 21sec
Trace Level                  off
Security                     ON: Password or Local OS Authentication
SNMP O                        N

Listener Parameter File
/home/oracle10/product/10.2.0/db_1/network/admin/listener.ora
Listener Log File
/home/oracle10/product/10.2.0/db_1/network/log/listener.log

```

```

Listening Endpoints Summary...
( DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=ORCL) ) )
( DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=crow.guardium.com)(PORT=1521) ) )
Services Summary...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully

```

At this point you can control this listener either from the local OS (in which case you do not need to enter the password) or remotely with a password. Locally:

```

bash-3.00$ lsnrctl
LSNRCTL for IBM/AIX RISC System/6000: Version 10.2.0.1.0 - Production on 03-MAR-2008 16:26:14
Copyright (c) 1991, 2005, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> stop
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=ORCL) ) )
The command completed successfully

```

Remotely:

```

LSNRCTL> status ON0CROW0
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=crow)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=on0crow0) )

TNS-01169: The listener has not recognized the password
TNS-01189: The listener could not authenticate the user
LSNRCTL> set password
Password:
The command completed successfully

LSNRCTL> status ON0CROW0
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=crow)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=on0crow0) )

STATUS of the LISTENER
-----
Alias LISTENER
Version TNSLSNR for IBM/AIX RISC System/6000: Version 10.2.0.1.0 - Production
Start Date 03-MAR-2008 16:26:25
Uptime 0 days 0 hr. 4 min. 5 sec
Trace Level off
Security ON: Password or Local OS Authentication
SNMP O N

Listener Parameter File
/home/oracle10/product/10.2.0/db_1/network/admin/listener.ora
Listener Log File
/home/oracle10/product/10.2.0/db_1/network/log/listener.log
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=ORCL) ) )
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=crow.guardium.com)(PORT=1521) ) )
Services Summary...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully

```

There are some issues with these passwords over the network that you should be aware of and they differ depending on which version of Oracle you are using. In older versions of lsnrctl there was a difference between running `set password <password>` and running `set password` alone and entering the password on the prompt. Before 10g the first use of `set password` sent the password in cleartext over the wire! For example, in 9i if you issued `LSNRCTL> set password mypassword` then the password would be sent as cleartext:

```
0 x0000: 4500 010e b404 4000 4006 ffe2 c0a8 0291 E.....@.@.....
0 x0010: c0a8 0221 8017 05f1 08d6 faca 7882 780a ...!.....x.x.
0 x0020: 8018 16d0 0d6f 0000 0101 080a 0004 4ecb .....o.....N.
0 x0030: 0025 5d16 00da 0000 0100 0000 0138 012c .%].....8.,
0 x0040: 0000 0800 7fff 7f08 0000 0100 00a0 003a .....
0x0050: 0000 07f8 0c0c 0000 0000 0000 0000 0000 .....(D
0 x0060: 0d2e 0001 ef3a 0000 0000 0000 0000 2 844 .....:
0 x0070: 4553 4352 4950 5449 4f4e 3d28 434f 4e4e ESCRIPTION=(CONN
0 x0080: 4543 545f 4441 5441 3d28 4349 443d 2850 ECT _DATA=(CID=(P
0 x0090: 524f 4752 414d 3d29 2848 4f53 543d 6963 ROGRAM=)(HOST=ic
0 x00a0: 656d 616e 2928 5553 4552 3d6f 7261 636c eman)(USER=orac1
0 x00b0: 6539 2929 2843 4f4d 4d41 4e44 3d73 7461 e9 )(COMMAND=sta
0 x00c0: 7475 7329 2841 5247 554d 454e 5453 3d36 tus)(ARGUMENTS=6
0 x00d0: 3 429 2850 4153 5357 4f52 443d 6d79 7061 4)(PASSWORD=mypassword
0 x00e0: 7373 776f 7264 2928 5345 5256 4943 453d ssword(SERVICE=
0 x00f0: 6f6e 396a 756d 626f 2928 5645 5253 494f on9jumbo)(VERSIO
0 x0100: 4e3d 3135 3330 3933 3132 3029 2929 N=153093120 ) )
```

But if you set the same password as:

```
LSNRCTL> set password
Password:
The command completed successfully
```

You get a hashed password:

```
0 x0000: 4500 0114 3d87 4000 4006 765a c0a8 0291 E...=@.@.vZ.....
0 x0010: c0a8 0221 801f 05f1 39c2 bba0 a9dc f978 ...!....9.....x
0 x0020: 8018 16d0 26da 0000 0101 080a 0005 861a ....&.....
0 x0030: 0031 86aa 0e00 0000 0100 0000 0138 012c .1.....8.,
0 x0040: 0000 0800 7fff 7f08 0000 0100 00a6 003a .....
0 x0050: 0000 07f8 0c0c 0000 0000 0000 0000 0000 .....(D
0 x0060: 0d2e 0003 2689 0000 0000 0000 0000 2844 .....&.....
0 x0070: 4553 4352 4950 5449 4f4e 3d28 434f 4e4e ESCRIPTION=(CONN
0 x0080: 4543 545f 4441 5441 3d28 4349 443d 2850 ECT _DATA=(CID=(P
0 x0090: 524f 4752 414d 3d29 2848 4f53 543d 6963 ROGRAM=)(HOST=ic
0 x00a0: 656d 616e 2928 5553 4552 3d6f 7261 636c eman)(USER=orac1
0 x00b0: 6539 2929 2843 4f4d 4d41 4e44 3d73 7461 e9 )(COMMAND=sta
0 x00c0: 7475 7329 2841 5247 554d 454e 5453 3d36 tus)(ARGUMENTS=6
0 x00d0: 3429 2850 4153 5357 4f52 443d 3145 4336 4)(PASSWORD=1EC6AF8986B565F6(SE
0 x00e0: 4146 3839 3836 4235 3635 4636 2928 5345 AF8986B565F6)(SE
0 x00f0: 5256 4943 453d 6f6e 396a 756d 626f 2928 RVICE=on9jumbo)(
0 x0100: 5645 5253 494f 4e3d 3135 3330 3933 3132 VERSION=15309312
0 x0110: 3029 2929 0) ) )
```

Starting with 10g Oracle sends a hashed version in both cases. This is certainly better than cleartext but it can still be used in a replay attack. Another reason for preferring OS Local Authentication

is related to the fact that contrary to password profiles (which you will learn about in Chapter 4) you cannot protect the listener from brute force password attacks. You can't set a certain number of logon attempts after which some lockout is enabled.

As for setting the password, you saw one way to set a password for the listener—using CHANGE\_PASSWORD and then SAVE\_CONFIG. You can also set SAVE\_CONFIG\_ON\_STOP, change the password and stop the listener. In both cases the password is saved in the listener.ora configuration file as a hashed entry:

```
#----ADDED BY TNSLSNR 03-MAR-2008 16:25:16---
PASSWORDS_LISTENER = 1EC6AF8986B565F6
#-----
```

You can also edit the file directly and put in a password in cleartext but that is obviously inferior. In all cases, make sure to secure the file itself so that the hashed value does not fall into the wrong hands.

## Information Leaks

Limiting access to lsnrctl also helps you combat an attacker who tries to use lsnrctl to gather information about the database environment as a “reconnaissance stage” as part of a larger attack. Both the STATUS and the SERVICES commands provide information that is valuable and that can be considered as sensitive information—perhaps not sensitive in the way that credit holder data is, but sensitive from the point of view of a “need to know basis.” This data can be even more valuable when using the SET RAWMODE ON. For example, without RAWMODE the SERVICES command returns valuable information such as which ports are being listened to and whether or not there is an external procedure service:

```
LSNRCTL> services
Connecting to (DESCRIPTION= (ADDRESS= (PROTOCOL=IPC) (KEY=ORCL) ) )
Services Summary...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s) for this service...
    Handler(s):
      " DEDICATED" established:0 refused:0
        LOCAL SERVER
Service "on0crow0" has 1 instance(s).
  Instance "on0crow0", status READY, has 1 handler(s) for this service...
    Handler(s):
      " DEDICATED" established:0 refused:0 state:ready
        LOCAL SERVER
Service "on0crow0XDB" has 1 instance(s).
  Instance "on0crow0", status READY, has 1 handler(s) for this service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:1022 state:ready
        DISPATCHER <machine: crow, pid: 598052>
        (ADDRESS=(PROTOCOL=tcp)(HOST=crow.guardium.com)(PORT=42392) )
Service "on0crow0_XPT" has 1 instance(s).
  Instance "on0crow0", status READY, has 1 handler(s) for this service...
    Handler(s):
      " DEDICATED" established:0 refused:0 state:ready
        LOCAL SERVER
The command completed successfully
```

But if you turn RAWMODE ON you can get far more valuable information including full paths for the database environment, handler IDs, and more:

```

LSNRCTL> set rawmode on
Raw mode is ON
LSNRCTL> services

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=ORCL) ) )

Services Summary...
(SERVICE=(SERVICE_NAME=PLSExtProc) (INSTANCE=(INSTANCE_NAME=PLSExtProc) (NUM=1) (INSTANCE_STATUS=UNKNOWN) (HANDLER=(HANDLER_DISPLAY=DEDICATED SERVER) (HANDLER_INFO=LOCAL SERVER) (HANDLER_MAXLOAD=0) (HANDLER_LOAD=0) (ESTABLISHED=0) (REFUSED=0) (HANDLER_ID=47A603B9CE39-E0A6-E043- C0A80274E0A6) (PRE=any) (HANDLER_NAME=DEDICATED) (SESSION=NS) (ADDRESS=(PROTOCOL=be q) (PROGRAM=/home/oracle10/product/10.2.0/db_1/bin/extproc) (ENVS='ORACLE_HOME=/home/oracle10/product/10.2.0/db_1,ORACLE_SID=PLSExtProc') (ARGV0=extprocPLSExtPro c) (ARGS='(LOCAL=NO)' )) (NUMREL=1) )
(SERVICE=(SERVICE_NAME=on0crow0) (INSTANCE=(INSTANCE_NAME=on0crow0) (NUM=2) (HANDLER=(HANDLER_DISPLAY=DEDICATED SERVER) (STA=ready) (HANDLER_INFO=LOCAL SERVER) (HANDLER_MAXLOAD=149) (HANDLER_LOAD=17) (ESTABLISHED=0) (REFUSED=0) (HANDLER_ID=47A5E87A70A3-10E0-E043- C0A8027410E0) (PRE=any) (HANDLER_NAME=DEDICATED) (SESSION=NS) (ADDRESS=(PROTOCOL=BE Q) (PROGRAM=/home/oracle10/product/10.2.0/db_1/bin/oracle) (ARGV0=oracleon0crow0) (ARGS='(LOCAL=NO)' (ENVS='_= /home/oracle10/product/10.2.0/db_1/bin/sqlplus,LANG=en_US,LOGIN=oracle10,PATH=, OLDPWD=/home/oracle10/product/10.2.0/db_1/network/admin,ORACLE_BASE=/home/oracle10,LC_FASTMSG=true,LOGNAME=oracle10,ORACLE_SID=on0crow0,LOCPATH=/usr/lib/nls/loc,USER=oracle10,AUTHSTATE=files,SHLVL=1,DISPLAY=192.168.1.238:0.0,SHELL=/usr/bin/csh,ODMDIR=/etc/objrepos,HOME=/home/oracle10,TERM=xterm,ORACLE_HOME=/home/oracle10/product/10.2.0/db_1,PWD=/home/oracle10,TZ=C ST6CDT,ORA_NET2_DESC=8,11,ORACLE_SPAWNED_PROCESS=1,SKGP_HIDDEN_ARGS=,LD_LIBRARY_PATH=/home/oracle10/product/10.2.0/db_1/lib,NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat') (ENV_POLICY=NONE) )) (NUMREL=1) )
(SERVICE=(SERVICE_NAME=on0crow0XDB) (INSTANCE=(INSTANCE_NAME=on0crow0) (NUM=2) (HANDLER=(HANDLER_DISPLAY=DISPATCHER) (STA=ready) (HANDLER_INFO=DISPATCHER <machine:crow, pid: 598052>) (HANDLER_MAXLOAD=1022) (HANDLER_LOAD=0) (ESTABLISHED=0) (REFUSED=0) (HANDLER_ID=47A5E87A70A8-10E0-E043- C0A8027410E0) (PRE=any) (HANDLER_NAME=D000) (SESSION=NS) (ADDRESS=(PROTOCOL=tcp) (HOST=crow.guardium.com) (PORT=42392) )) (NUMREL=1) )
(SERVICE=(SERVICE_NAME=on0crow0_XPT) (INSTANCE=(INSTANCE_NAME=on0crow0) (NUM=2) (HANDLER=(HANDLER_DISPLAY=DEDICATED SERVER) (STA=ready) (HANDLER_INFO=LOCAL SERVER) (HANDLER_MAXLOAD=149) (HANDLER_LOAD=17) (ESTABLISHED=0) (REFUSED=0) (HANDLER_ID=47A5E87A70A3-10E0-E043- C0A8027410E0) (PRE=any) (HANDLER_NAME=DEDICATED) (SESSION=NS) (ADDRESS=(PROTOCOL=BE Q) (PROGRAM=/home/oracle10/product/10.2.0/db_1/bin/oracle) (ARGV0=oracleon0crow0) (ARGS='(LOCAL=NO)' (ENVS='_= /home/oracle10/product/10.2.0/db_1/bin/sqlplus,LANG=en_US,LOGIN=oracle10,PATH=,OLDPWD=/home/oracle10/product/10.2.0/db_1/network/admin,ORACLE_BASE=/home/oracle10,LC_FASTMSG=true,LOGNAME=oracle10,ORACLE_SID=on0crow0,LOCPATH=/usr/lib/nls/loc,USER=oracle10,AUTHSTATE=files,SHLVL=1,DISPLAY=192.168.1.238:0.0,SHELL=/usr/bin/csh,ODMDIR=/etc/objrepos,HOME=/home/oracle10,TERM=xterm,ORACLE_HOME=/home/oracle10/product/10.2.0/db_1,PWD=/home/oracle10,TZ=C ST6CDT,ORA_NET2_DESC=8,11,ORACLE_SPAWNED_PROCESS=1,SKGP_HIDDEN_ARGS=,LD_LIBRARY_PATH=/home/oracle10/product/10.2.0/db_1/lib,NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat') (ENV_POLICY=NONE) )) (NUMREL=1) )
The command completed successfully

```

To summarize, use passwords to prevent others from managing your listeners without authenticating and avoid managing your listeners over the network unless you encrypt the communication layer. Oracle 11g comes with an even more secure listener out of the box—you can only stop the listener as the local OS account that started the listener.

### Five Things to Remember about Securing Access to lsnrctl

1. You must secure access to lsnrctl. Lsnrctl can be used to launch multiple forms of DoS attacks (the simplest being stopping the listener), delete data or audit trails, run SQL commands within a privileged session and even to allow unauthenticated logins to the Oracle instance account (through modifications to files such as .rhosts).
2. In Oracle 8i and 9i you MUST set a password to ensure that an unauthenticated remote user does not take control of your listener. In Oracle 10g and 11g the listener is secure by default and uses Local OS Authentication.
3. Don't set or change listener passwords over the network unless you are encrypting network communications.
4. There is no account lockout for lsnrctl passwords.
5. Prior to 10g the lsnrctl would accept either a listener password or the password hash. This means that the hash (kept in listener.ora) is as valuable as the password itself and should be secured as though it holds a cleartext password.

## 3.2 HOWTO Limit the Ability to Change Listener Properties

In the previous HOWTO you saw that you can set the listener's password from lsnrctl or by modifying the listener's configuration file. The same holds true for many other settings of the listener. For example, you can set the listener's trace and log file locations and the trace levels either through the SET command within lsnrctl or by modifying listener.ora. Because there have been many cases in the past where lsnrctl was hijacked and used to launch an attack, Oracle added an option that ensures that all changes to the listener's configuration can only be done through file edits. To do that you should set ADMIN\_RESTRICTIONS to ON—this tells the listener that it should not accept any SET commands and forces you to set the listener's administration properties by changing the listener's configuration files. Edit listener.ora and add a line of the form ADMIN\_RESTRICTIONS\_<your listener name> = ON.

Once this is done SET commands for setting trc\_file, trc\_directory, log\_file, log\_directory, startup\_waitime, and more will fail and you can only change these settings when you have privileges to edit the file. For example:

```
LSNRCTL> set startup_waitime 0
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=ORCL) ))
TNS-12508: TNS:listener could not resolve the COMMAND given
```

To change values you can edit listener.ora:

```
ADMIN _ RESTRICTIONS _ LISTENER = ON
# Set the log directory
LOG _ DIRECTORY _ listener = /home/oracle10/product/10.2.0/db_1/network/log/
```

After bouncing the listener:

```
LSNRCTL> show log_directory
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=ORCL) ))
```

```
LISTENER parameter "log_directory" set to
/home/oracle10/product/10.2.0/db_1/network/log/
The command completed successfully
```

Almost all of the SET commands can be used to launch an attack. By setting trace and log levels an attacker can slow down your machine (sometimes to a crawl). By changing the location to which the log files are written an attacker can delete important files or write data to a file that controls access to the account. In all cases, ADMIN\_RESTRICTIONS ensure that only your instance owner can make these changes.

### **Two Things to Remember about Limiting the Ability to Change Listener Properties**

1. Listener properties should be changed locally by editing listener.ora and not through lsnrctl.
2. Set ADMIN\_RESTRICTIONS = ON for all listeners.

## **3.3 HOWTO Secure EXTPROC**

Oracle allows you to call out to external programs from within the database. This can be useful when you need to tap into code that already exists outside the database for reuse purposes or when you need code that runs faster. For example, if you have a computationally intensive function that you need to execute very often, you can choose to write this function in C, compile it, and use it from within the database rather than writing a PL/SQL procedure. If this is an expensive computation or is called often enough by your database application this can make a big difference in terms of performance.

External procedures and the ability to invoke an external executable are not new in Oracle and have been available since version 8. It is also not a feature that is uniquely available in Oracle databases—for example, SQL Server has an extended stored procedure called xp\_cmdshell that allows calls to any program at the Windows level from within the database. These features are valuable for developers and for enhancing productivity but they are bad from a security point of view. One of the most important principles in good security is segregation. Once the database can also make calls to programs (potentially any programs) at the OS level, the ability to limit what an attacker can do becomes much harder. Therefore, from a security perspective, the ability to make calls to external code from within the database should be avoided if possible and if you look at any hardening document for Oracle this is one of the first things you need to do. If you make use of this capability and need to continue using it you should at least take precautions to limit the exposure that this feature creates.

### **Understanding External Procedures**

To use external code you need to configure the listener to be able to load and run external programs or libraries. Because the listener already has the capability to load and run executables (it runs the Oracle server process when you use dedicated server mode), having it execute other binaries is a natural extension. To configure this you need to modify listener.ora and add an entry into your SID\_LIST.

The SID\_DESC section that allows the listener to load and run external programs is quite simple. It merely needs an SID name, what executable to run, and an environment. Typically, you

will see such entries on two occasions. The first is an SID\_DESC section that you will see very often because it is there by default. It usually has a name of PLSExtProc or ExtProc. This is the entry that allows you to build external procedures and call them from within the database. In listener.ora, as part of the SID\_LIST, this entry looks like:

```
(SID_DESC =
  ( SID_NAME = PLSExtProc)
  ( ORACLE_HOME = /export/home/oracle10/product/10.2.0/Db_1)
  ( PROGRAM = /export/home/oracle10/product/10.2.0/Db_1/bin/extproc)
)
```

In tnsnames.ora you may then also see an entry of the form:

```
EXTPROC_CONNECTION_DATA =
(D ESCRIPTION =
( ADDRESS_LIST =
  (ADDRESS = (PROTOCOL = IPC)(KEY = ORACLE) )
)
( CONNECT_DATA =
  ( SID = PLSExtProc)
)
)
```

The entry in listener.ora specifies a program that the listener is able to load—specifically, Oracle's EXTRPROC program. This Oracle executable provides the framework for allowing you to call code packaged as DLLs from within PL/SQL. The communication between the database server and this program is done over a set of pipes—hence the IPC protocol entry in the ADDRESS section.

The second occasion in which you'll see such an entry is less common and involves the ability to execute any external executable and use the listener for initiating and managing communications. For those of you familiar with Sybase this is similar to Sybase's Open Server although in Oracle this is not a documented framework. You will usually see it when you have other Oracle products. For example, Listing 3.2 shows the listener.ora file from an Oracle EBS application. You can see that the SID\_DESC sections load and run a program called FNDFS. In the same way that the PLSExtProc entry causes the listener to load the EXTPROC program, EBS utilizes the FNDFS program.

When you define a listener in listener.ora you define a static service registration section (the SID\_LIST). Each SID\_DESC in the list specifies service information for a specific database instance or a nondatabase service. You define a nondatabase service through the PROGRAM and ENV parameters. You use PROGRAM to specify the executable that gets invoked and you use ENVS to specify environment variables for that execution (Listing 3.2). For example, if you need the listener to be able to invoke your program at /export/home/ronb/externLog:

```
(SID_DESC =
  ( SID_NAME = ExternLog)
  ( ORACLE_HOME = /export/home/oracle10/product/10.2.0/Db_1)
  ( PROGRAM = /export/home/ronb/externLog)
  ( ENVS="LD_LIBRARY_PATH= /usr/lib:/export/logger/lib,LOG_DIR=/export/logger/
repository/logs")
)
```

```

# $Header: admk80ln_ux.sql 115.19 2004/04/01 21:02:29 sumishra ship $
# LISTENER.ORA For Oracle Applications
# This file is automatically generated
APPS_PROD =
(ADDRESS_LIST =
  (ADDRESS= (PROTOCOL= TCP) (Host= rondesktop) (Port= 1626))
)
SID_LIST_APPS_PROD =
(SID_LIST =
  ( SID_DESC = ( SID_NAME = FNDSM )
    ( ORACLE_HOME = /home/applmgr/prodora/8.0.6 )
    ( PROGRAM = /home/applmgr/prodappl/fnd/11.5.0/bin/FNDSM )

( envs='MYAPPSSORA=/home/applmgr/prodappl/APPPSPROD_rondesktop.env,PATH=/usr/bin:/usr/ccs/bin:/bin,FNDSM_SCRIPT=/home/applmgr/prodcomm/admin/scripts/PROD_rondeskto
op/gsmstart.sh' )
)
  ( SID_DESC = ( SID_NAME = FNDFS )
    ( ORACLE_HOME = /home/applmgr/prodora/8.0.6 )
    ( PROGRAM = /home/applmgr/prodappl/fnd/11.5.0/bin/FNDFS )

( envs='EPC_DISABLED=TRUE,NLS_LANG=AMERICAN_AMERICA.UTF8,LD_LIBRARY_PATH=/usr/dt
/lib:/usr/openwin/lib:/home/applmgr/prodora/8.0.6/lib,SHLIB_PATH=/usr/lib:/usr/dt/lib:/usr/openwin/lib:/home/applmgr/prodora/8.0.6/lib,LIBPATH=/usr/dt/lib:/usr/
openwin/lib:/home/applmgr/prodora/8.0.6/lib,APPLFSTT=PROD,APPLFSWD=/home/applmgr
/prodappl/admin;/home/applmgr/prodcomm/temp;/home/applmgr/prodcomm/html/oam/nonU
ix/launchMode/restricted' )
)
)
STARTUP_WAIT_TIME_APPS_PROD = 0
CONNECT_TIMEOUT_APPS_PROD = 10
TRACE_LEVEL_APPS_PROD = OFF
LOG_DIRECTORY_APPS_PROD = /home/applmgr/prodora/8.0.6/network/admin
LOG_FILE_APPS_PROD = APPS_PROD
TRACE_DIRECTORY_APPS_PROD = /home/applmgr/prodora/8.0.6/network/admin
TRACE_FILE_APPS_PROD = APPS_PROD
IFILE =
/home/applmgr/prodora/8.0.6/network/admin/PROD_rondesktop/PROD_rondesktop_listener_
ifile.ora

```

**Listing 3.2 Sample listener.ora file from an Oracle EBS application.**

You can put any name/value pairs in the ENVS section—it all depends on what environment the external code requires.

The EXTPROC program and the external procedure functionality in Oracle is riskier than the ability to have the listener spawn a n external executable. EXTPROC is a n infrastructure component that allows you to potentially call anything from within the database. Let's look at an example of how this works, assuming you already have the listener.ora entry for EXTPROC and the connection data:

```

PLSEXTPROC_LISTENER =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = IPC)(KEY = PLSEXTPROC) )
)
```

```

SID_LIST_LISTENER_PLSEXTPROC=
(SID_LIST =
(SID_DESC =
( SID_NAME = PLSEXTPROC)
( ORACLE_HOME = /export/home/oracle10/product/10.2.0/Db_1)
( PROGRAM = /export/home/oracle10/product/10.2.0/Db_1/bin/extproc)
)
)

```

Say you have a C function called foo that performs some computation and returns a string. For the sake of simplicity assume it has a signature of `char *foo()` and it is in `foo.c`. The first step is to compile this program and generate a shared object so that EXTPROC can load the code. For example, in Solaris:

```

$ gcc -G -c -m64 foo.c
$ ld -r -o foo.so foo.o
$ cp foo.so $ORACLE_HOME/lib

```

Within an Oracle session, create a library:

```

create library libfoo as
'/export/home/oracle10/product/10.2.0/Db_1/lib/foo.so';
/

```

Create a function within the database referencing the library. This is a wrapper for the external function:

```

create or replace function foo
  return char
  is external
  library libfoo
  name "foo"
  language C;
/

```

At this point you can call this external function from the database just as if it were a PL/SQL function:

```
select foo from dual;
```

What Oracle does under the cover is load the DLL through the EXTPROC program (if it has not already been loaded) and ask it to invoke foo. The function is running in the EXTPROC's address space—it is a separate process running the code.

## Securing External Procedures

EXTPROC is a powerful feature. At the same time, it is very dangerous—precisely because it is so powerful. The “holy grail” of attackers is the ability to run code that they have written and planted, but run it as a privileged account. EXTPROC is a dream come true for an attacker—it potentially allows them to craft code that can be placed somewhere on the OS and then through a database connection invoke this code. The beauty (as far as the attacker is concerned) is that the code is executed by the listener. Normally this means that this code (at the OS level) is running as the instance account. From a privilege perspective this code can copy or even delete data

files, change or delete log and audit files, etc. Anything that the instance account owner can do, this code can do.

Securing external procedures (and external executables in general) can be very simple—if you don't use external procedures, get rid of the EXTPROC section in listener.ora and delete the extproc executable from \$ORACLE\_HOME/bin. Use a change tracking tool to monitor listener.ora for changes and make sure that the EXTPROC section or any other section with PROGRAM does not suddenly reappear. If these sections are not in any of your listener files then the listener will not be spawning external programs or allowing database callouts. Over 95 percent of all environments have no need for such functions but unfortunately, the fact that you don't use a function does not mean that an attacker will not use this function. If you don't need it—don't have it available.

If you do use external procedures or external executables you should secure them using the principle of minimal privileges. You should limit both what can be invoked as well as what privileges this process will be running with.

To limit what can be invoked as an extended procedure, use the EXTPROC\_DLLS variable in the ENVS section of the EXTPROC SID. For example, if you are just going to be calling out to foo which is in foo.so, then you should change the SID section in listener.ora to:

```
(SID_DESC =
  (SID_NAME = PLSExtProc)
  (ORACLE_HOME = /export/home/oracle10/product/10.2.0/Db_1)
  (PROGRAM = /export/home/oracle10/product/10.2.0/Db_1/bin/extproc)
  (ENVS="EXTRPROC_DLLS=ONLY:/export/home/oracle10/product/10.2.0/Db_1/lib/foo.so")
)
```

If you have multiple DLLs which you'll be using, list them all. But always list only the DLLs you want to use—never more than that and never leave this empty. Never ever use ENVS = EXTPROC\_DLLS = ANY. This allows the EXTPROC to load and run any code in any DLL, you're asking for trouble. It's amazing that many documents that describe external procedures (including some Metalink notes) will tell you to set ENVS = EXTPROC\_DLLS = ANY. If you have a very large number of external procedures and you're not sure which libraries you're using you can run with trace level support for a short time—all calls to the EXTPROC agent will be logged with the DLL information.

Because calling an external procedure requires you to execute CREATE LIBRARY you should limit the privileges to this command and put the following additional controls in place:

- Monitor and audit any call to CREATE LIBRARY.
- Periodically check which libraries are defined. Add this to your audit cycle and check for changes in the list of library objects defined in the database.
- Use a change tracking tool to ensure that the DLLs that are being loaded do not change (except when part of a maintenance request).

Once you limit what the external procedure agent can run, limit the privileges it has by running EXTPROC as a separate listener from a less privileged user. Create a new user that has just enough privileges to run your code but that does not have the privileges that the instance account has. For example, this user should have no write privileges on the files owned by the instance owner. Take the EXTPROC entry out of your main listener.ora and create the listener.ora and tnsnames.ora files in this user's directory. Set a different set of log files for this new listener. For example,

Listener.ora:

```

PLSEXTPROC_LISTENER =
(ADDRESS_LIST =
  (ADDRESS = (PROTOCOL = IPC)(KEY = PLSEXTPROC) )
)

SID_LIST_PLSEXTPROC_LISTENER =
(SID_LIST =
(SID_DESC =
  ( SID_NAME = PLSEXTPROC)
  ( ORACLE_HOME = /export/home/oracle10/product/10.2.0/Db_1)
  ( PROGRAM = /export/home/oracle10/product/10.2.0/Db_1/bin/extproc)
  ( ENVS="EXTRPOC_DLLS=ONLY:/export/home/oracle10/product/10.2.0/Db_1/lib/foo.so")
)
)

TRACE_LEVEL_PLSEXTPROC_LISTENER = OFF
LOG_DIRECTORY_PLSEXTPROC_LISTENER = /export/home/ronb/externLog/listenerLog/
TRC_DIRECTORY_PLSEXTPROC_LISTENER = /export/home/ronb/externLog/listenerLog/

```

Tnsnames.ora:

```

EXTPROC_CONNECTION_DATA =
(D ESCRIPTION =
( ADDRESS_LIST =
  (ADDRESS = (PROTOCOL = IPC)(KEY = PLSEXTPROC) )
)
(C ONNECT_DATA =
( SID = PLSEXTPROC)
)
)

```

Start this listener from the user with limited privileges, not the Oracle instance account:

```
# su - ronb
$ lsnrctl start plsextproc_listener
```

### Four Things to Remember about Securing EXTPROC

1. EXTPROC is a dangerous extension to the database from a security point of view. If you do not use external procedures make sure there is no EXTPROC entry in any of your listeners.
2. If you do use external procedures check to make sure that you limit which DLLs can be loaded by EXTPROC using EXTRPOC\_DLLS = ONLY:..
3. If you do use external procedures, use a separate listener for EXTPROC. Delete all reference to EXTPROC from your main listeners. Use a separate listener.ora file for the EXTPROC SID. Start this listener not from the oracle instance account but from an OS account that has no privileges to change or read data from the Oracle instance account. This user should have only the privileges needed to run the external procedures.
4. Use change tracking to ensure that listener.ora files do not change and that no new listeners are added—especially entries with PROGRAM.

### 3.4 HOWTO Limit the Sources from Which Connections Are Accepted

The Oracle listener can limit the hosts it accepts connections from. If you are running a database application that needs to be accessed from an application server only or from a few hosts only, you should use a feature called valid node checking to limit the connections that the listener will accept. This is akin to a mini-firewall that is built directly into the listener.

Valid node checking allows you to define two lists of IP addresses or hosts. One list—invited nodes—defines a set of nodes from which connections will be accepted. The other list—excludes nodes—defines a list from which connections will be rejected. Normally you would use only one of these lists. If you use invited\_nodes then any node not on this list will be rejected by the listener. If you use excluded\_nodes then any node not on this list will be accepted by the listener. If you include both lists then the invited\_nodes take precedence over the excluded\_nodes. As an example, to allow connections only from the application server, a terminal server from which DBAs connect, and a server which has a database link, add the following to sqlnet.ora:

```
tcp.validnode_checking=YES
tcp.invited_nodes= (appserver,terminalserver,192.168.2.33)
```

You can use either IP addresses or host names. Bounce the listener for these checks to take effect. If you now try to connect from any node apart from these three you will get an error of the form:

```
ERROR:
ORA-12537: TNS:connection closed
```

The listener log will show the connection failure:

```
14-FEB-2008 23:19:16 * 12546
TNS-12546: TNS:permission denied
TNS-12560: TNS:protocol adapter error
TNS-00516: Permission denied
```

To trace what valid node checking is doing (especially when you just start using it and need to figure out why some connections are failing) you can set a support trace level (for a very short time obviously) in listener.ora:

```
TRACE_DIRECTORY_LISTENER = /home/oracle10/oracle/product/10.2.0/db_1/network/trace
TRACE_FILE_LISTENER = listener.trc
TRACE_LEVEL_LISTENER = support
```

Now when you try to connect and get refused, the trace file will include lines of the form:

```
[14-FEB-2008 23:31:28:422] ntvlser: valid node check on incoming node
192.168.222.1
[14-FEB-2008 23:31:28:422] ntvlser: Denied Entry: 192.168.222.1
```

Filtering is done using full IP addresses—you cannot have wildcards or define subnets—you have to manage the lists in full. You cannot use this feature if you are accessing the database from workstations which get dynamic IP addresses through Dynamic Host Control Protocol (DHCP).

Finally, an attacker can bypass this feature by spoofing IP addresses in network packets. However, because this feature is so simple to use it should always be used as an additional layer of defense in all cases where the database should be accessed by only a limited set of nodes and where the IP addresses which access the database stay fixed.

### Two Things to Remember about Valid Node Checking

1. The listener implements a mini-firewall allowing you to specify which hosts you are willing to accept connections from.
2. Setup of valid node checking is very simple and you should use it whenever possible. You can define rules either through white lists (i.e., which nodes can connect) or through black lists (i.e., which nodes are not allowed to connect).

## 3.5 HOWTO Inspect Listener Logs and Traces and HOWTO Limit Traces

The listener can write data to two types of files—log files and trace files. Both are controlled through a set of SET commands or by editing listener.ora. The relevant properties are trc\_file, trc\_directory, trc\_level, log\_file, log\_directory, and log\_status.

Listener logging is something you should be using during normal database operations. Logging helps you uncover when someone tries to launch an attack on the listener and does not generally generate too much information. Each entry in the log includes the listener command, which host it came from and which protocol was used by the connection. The log is especially important for knowing which command failed (for example due to ADMIN\_RESTRICTIONS), whether authentication failed, etc. To enable the log, edit listener.ora and set LOG\_STATUS = ON and then set the log\_directory and the log\_file properties (or use the defaults).

Tracing is a very different story and you need to be very careful when using traces. A listener trace is a troubleshooting tool that you should use selectively when you want to diagnose a connection problem. A listener trace is a dump of everything the listener did and can be very verbose. As an example, a single TNS connection that does one select statement and exits adds over 900 lines and over 50 K to the trace file. In fact, traces have been used in the past to launch a DoS attack on the database—by setting the trace level to SUPPORT the listener was forced to consume so much CPU and disk space that the database could not function. To ensure this does not happen make sure that you have TRC\_LEVEL\_<listener name> = OFF in listener.ora.

### Inspecting Listener Logs

Listener logs should be reviewed either manually or, preferably, automatically using scripts or third-party tools. What you're looking for are error conditions that may indicate an attack. Each log entry includes a time stamp, which command was issued, the result code and if Oracle returned an error, the error message. The main listener errors that you should be looking for are shown in Table 3.1. As an example, let's review the following two log entries:

```
21-SEP-2007 21:37:58 *
(CONNECT_DATA=(CID=(PROGRAM=) (HOST=) (USER=RBNatan) ) (COMMAND=status) (ARGUMENTS=6
4) (PASSWORD=xxxxxxxx) (SERVICE=on0rh4) (VERSION=169869568) ) * status * 0
21-SEP-2007 21:39:45 *
```

**Table 3.1 Important Error Messages to Extract from Listener Log**

<i>TNS Error Code</i>	<i>TNS Error Message</i>	<i>Error Cause</i>
TNS-01169	The listener has not recognized the password	The listener has the password security mechanism enabled and requires the correct password to execute any command other than VERSION. The user attempted to issue one of the privileged administrative commands, but could not be successfully authenticated with the password provided
TNS-01189	The listener could not authenticate the user	The user attempted to issue a privileged administrative command, but could not be successfully authenticated by the listener using the Local OS Authentication mechanism. For example, Local OS Authentication is being used and an attempt was made to use the listener remotely
TNS-01190	The user is not authorized to execute the requested listener command	Most of the listener administrative commands are only intended to be issued by privileged users, for example, DBAs or system administrators. If the listener password is not set, then the listener only accepts administrative requests from LSNRCTL running with the same OS credentials, or running as a local administrator (also referred to as super user)
TNS-12508	The listener could not resolve the COMMAND given	An invalid command was sent or ADMIN_RESTRICTIONS is enabled and a SET command was used
TNS-12546	Permission denied	Valid node checking denied a connection
TNS-00516	Permission denied	User has insufficient privileges to perform the requested operation
TNS-00525	Insufficient privilege for operation	OS failed to complete operation because user lacked sufficient privileges
TNS-00540 through TNS-00560	Various Secure Sockets Layer (SSL) and advanced authentication errors	
TNS-00584	Valid node checking configuration error	Valid node checking specific Oracle Net configuration is invalid
TNS-00583	Valid node checking: unable to parse configuration parameters	Valid node checking was unable to parse the configuration due to syntactical errors

```
(CONNECT_DATA=(CID=(PROGRAM=) (HOST=) (USER=RBNatan) ) (COMMAND=status) (ARGUMENTS=6
4) (PASSWORD=xxxxxxxx) (SERVICE=on0rh4)(VERSION=169869568) * status * 1189
TNS-01189: The listener could not authenticate the user
TNS-01169: The listener has not recognized the password
```

The first entry logs a status command sent at 21:37:58 which was successfully performed—hence the return code of zero. Note that if you have a password set for the listener the log file will include the password in cleartext (another security issue)—so make sure to secure the log file using file permissions. The second attempt at 21:39:45 failed because the wrong password was used.

### **The e Things to Remember about Listener Logs and Traces**

1. Traces have been used many times in the past as part of a Do S attack—they can overwhelm the server in terms of required processing and can fill up the disk. Use traces only for troubleshooting and don't forget to turn the listener traces off.
2. Traces include sensitive data—make sure to delete all traces when finished diagnosing problems.
3. Listener logs on the other hand are very useful from a security perspective. Turn on listener logs on all listeners and review the logs at least for errors.

## **3.6 HOWTO Combat TNS Protocol Attacks**

There are numerous parameters that can help you further secure the listener from attacks. The most important ones are:

- INBOUND\_CONNECTION\_TIMEOUT\_<your listener name> in listener.ora specifies the time, in seconds, that a client must complete its connection attempt after the initial network connection has been acknowledged. By default this value is not set. You should set it to the lowest value possible such that none of your client connections are timing out.
- SQLNET.INBOUND\_CONNECTION\_TIMEOUT in sqlnet.ora is similar to the previous parameter but also includes the time for the initial network connection so it should be set a bit higher than the previous value.
- SQLNET.EXPIRE\_TIME in sqlnet.ora specifies the time that the server will use to check for connections that have been terminated by the client (or a firewall for example) without an orderly shutdown. Set this to a few minutes so that resources are reclaimed in case this happens. You can also use this parameter in case you have an application server with many connections managed by a connection pool and have a firewall between the application server and the database server. In this case, especially if your application server is not heavily loaded, it is possible that some of the connections in the pool will see little usage and may remain idle for a very long time. Many firewalls will tear down such inactive sessions and this may impact the availability of your application because it will then have less available connections in the pool. By setting the EXPIRE\_TIME, Oracle will every so often send out a probe to the database client to see if it is still there. The firewall will see this activity and will not tear down the connection.

Oracle 11g has a new set of initialization parameters that can help you further harden your listener. As you see in Table 3.2 and from the discussion in the next section, the past few years have seen many attacks on the listener—a lot of them based on sending malformed packets to the listener in

**Table 3.2 Security Alerts and CPUs Addressing Listener Vulnerabilities**

<i>Alert/CPU</i>	<i>Date</i>	<i>Vulnerability</i>	<i>Oracle Doc</i>
Security alert #2	Nov-01	LOG_FILE and TRC_FILE can be spoofed and any extension can be used	Metalink Note 124742.1
Security alert #14	Jul-01	Net8 connections can used to launch a DoS attack on Windows NT systems	Metalink Note 153289.1
Security alert #15	Jul-01	15.1—DoS—Listener crashes when the offset_to_data field contains an arbitrary large value	Metalink Note 151261.1
		15.2—DoS—Listener crashes when an invalid value is sent in the client version field on Unix	Metalink Note 151290.1
		15.3—DoS—Listener crashes when the maximum transport data size is set to 0 on Solaris	Metalink Note 151291.1
		15.4—DoS attack when TNS packets are fragmented	Metalink Note 151292.1
Security alert #16	Jul-01	Buffer overflow vulnerability in the listener (8i) can be used when large amounts of command data is sent	Metalink Note 151259.1
Security alert #17	Jul-01	Listener can be corrupted or core dumps with large amounts of connect data	Metalink Note 151260.1
Security alert #34	Jun-02	DoS attack can be launched against Windows and VMS servers with a well-crafted packet	Metalink Note 198544.1
Security alert #38	Aug-02	Listener crashed when an invalid command request is sent, 9i	Patch 2467947
Security alert #40	Aug-02	Format string vulnerabilities	Patch 2467947
Security alert #42	Oct-02	DoS listener vulnerability	Patch 2540219
Security alert #54	Apr-03	DoS listener buffer overflow vulnerability	Multiple
Security alert #68	Aug-04	DoS listener vulnerability	Multiple
CPU Jan 2006	Jan-06	Multiple listener vulnerabilities (DB09, DB10, DB11)	CPU/JAN/2006
CPU Jan 2007	Jan-07	Buffer overflow vulnerability in the listener for 8 and 9 may allow an attacker to gain local privileges of the Oracle account	CPU/JAN/2007

an attempt to either crash it or exploit a bug in the code. Many of these attacks have one thing in common—they were all based on manually creating TNS packets with the wrong structure, the wrong lengths or bad data. To combat this 11g now includes parameters that can help you control what to do when such malformed packets are sent to the database. These parameters tell Oracle what to do when it observes a TNS protocol error. The important new parameters are

- SEC\_PROTOCOL\_ERROR\_TRACE\_ACTION: This parameter helps you control what to do when Oracle senses a protocol error. On the one hand, you want detailed information about such events because they may imply an attack that you should investigate. On the other hand, if you write out a lot of trace information this can be used to launch a DoS attack. The values you can use for this parameter are
  - TRACE is the default value. This means that a trace file will be written.
  - LOG: Writes a short one liner to the server trace file. This is a good balance between needing to know that this is happening and not consuming too many resources.
  - ALERT: Writes a short one liner to both the server trace file and the alert log. This is also a good, balanced option.
  - NONE: Tells the server not to generate any traces or error messages related to bad packets. Don't use this option unless you are using an external Intrusion Detection System (IDS) that supports deep packet inspection for the Oracle protocol.
- SEC\_PROTOCOL\_ERROR\_FURTHER\_ACTION: This parameter helps you control whether or not a connection that has a bad packet should be allowed to continue to talk to the server. Available values are
  - CONTINUE: The default. Oracle will continue processing the stream even if there are bad packets. The server could be subject to further attacks.
  - DELAY,  $n$ : Delays the client for  $n$  seconds before willing to accept the next request from the same client connection. This can be a good tradeoff because the attacker is slowed to a point that it may be ineffective and yet the connection is maintained and can be tracked.
  - DROP,  $n$ : Terminate the client connection after  $n$  bad packets.
- SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS =  $n$ : The number of failed logins allowed (regardless of the account used to login) on a single database connection. The default is 10. This combats brute force attacks that guess both user names and passwords.

#### Two Things to Remember about Combating Protocol Attacks

1. Oracle 11g provides new parameters for controlling what to do when malformed packets are sent from a client connection.
2. Use an Oracle-aware IDS as an additional layer of security where Oracle can be accessed over the network by a wide community.

## 3.7 Discussion: History of Listener Security Alerts

The listener is one of the simplest components to secure properly. And yet, historically, listener security has been one of the biggest pain points. Some of this has been due to bugs and security

vulnerabilities and some of this has been due to neglect in configuration. Over the years Oracle fixed over 20 vulnerabilities of the listener—often quite serious in terms of the impact and in terms of the ease by which they can be exploited. These vulnerabilities and the alert, patch or CPU in which they were fixed are listed in Table 3.2.

When you look at Table 3.2 one year stands out more than any other—2001 (although, 2002 is in close second). What happened in 2001? That's an interesting tidbit of Oracle security history. In April 2001 a security researcher calling himself jwa published a tool called tnscmd along with his research on reverse engineering the TNS protocol. In this work, which you can view at <http://www.jammed.com/~jwa/hacks/security/tnscmd>, jwa started describing how the protocol is handled and pointed out some issues and vulnerabilities. Jwa also pointed out many of the issues that occur when running with an insecure listener configuration—for example, the ability to remotely shut down the listener, the ability to turn tracing on, the ability to write to .rlogin or glogin.sql, etc. Immediately following this posting many other security researchers and hackers started playing around with the tnscmd tool or other tools which manually construct packets. They then saw many issues with the handling of malformed packets and reported these to Oracle—which issued patches and security alerts.

Currently (as of writing this book) there are no known vulnerabilities in the listener. Moreover, because historically listener security has been neglected by DBAs (perhaps because it is an independent process running outside the database), all hardening guidelines and checklists put a lot of emphasis on good listener security and Oracle 10g and 11g both come with a secure configuration as a default.

## *Chapter 4*

---

# Account Security

---

Oracle security is managed through the concept of a database user. To connect to the database you must have a valid Oracle user account. The permissions you have within the database are based on the privileges that are assigned to the user through which you are connected to Oracle. Account security is therefore one of the fundamental building blocks in ensuring a secure Oracle environment.

Oracle user accounts are created and managed within the database. In this chapter you will focus on how to create users, how to manage passwords within Oracle, how to limit accounts, and how to review which limits are assigned to accounts. This chapter does not deal with how the user is authenticated with Oracle. As you'll see in Chapter 6 there are many ways by which a user can be authenticated—some using the Oracle database and some using an external authentication mechanism. Regardless of the method by which you authenticate a user, when the connection is made it is managed as a user account within the database. When a logon to Oracle occurs, a set of credentials is presented to Oracle. Oracle (or an external authenticator that Oracle is working with) determines whether these credentials are correct and whether to allow the logon. After authentication occurs and the logon succeeds, the connection is allowed as the database user—where all of the account properties are derived from.

Although you will not learn about authentication in this chapter, you will learn quite a number of things about passwords. Weak and default passwords are considered by many to be the number one reason for security breaches in database environments. Older Oracle installations in which the change\_on\_install password was never changed exist even today and instances where the default password for the DBSNMP account has not been changed are quite common. You'll see how to combat this type of misuse as well as how to manage password profiles in general.

### **4.1 HOWTO Create, Alter, Drop, and Lock User Accounts**

Use the CREATE USER command to create an account:

```
SQL> create user ronbn identified by "0okmnhy^";  
User created.
```

Use a strong password when you create users and force your users to change their passwords (either by creating the account with an expired password or through password profiles which we'll learn

about in the next HOWTO). Once the user has been created, grant it privileges depending on what this user needs to do. If the user needs to be able to log onto the database (most any user will need that) then at a minimum you need to allow them to create a session. If you don't the user will get the following error when trying to logon:

```
SQL> connect ronbn
Enter password: *****
ERROR:
ORA-01045: user RONBN lacks CREATE SESSION privilege; logon
denied
```

Grant the privilege and then the user can logon:

```
SQL> grant create session to ronbn;
Grant succeeded.
```

If you want to ensure that the user changes their password the first time they connect, use the PASSWORD EXPIRE option:

```
SQL> create user ronbn identified by "temp8uhbgt5" password expire;
User created.
SQL> grant create session to ronbn;
Grant succeeded.
SQL> connect ronbn
Enter password: *****
ERROR:
ORA-28001: the password has expired
Changing password for ronbn
New password: *****
Retype new password: *****
Password changed
Connected.
```

Now that ronbn can connect to the database, grant other privileges as required but use the general concept of minimum privileges—i.e., grant privileges on an as-needed basis. As an example, assume that ronbn will be used within an application and needs to be able to create tables within the schema. Go ahead and grant them CREATE TABLE privileges:

```
SQL> grant create table to ronbn;
Grant succeeded.
```

From a privilege perspective, ronbn can create tables within the ronbn schema—but this is not enough. One of the important things most users will need is a quota on a tablespace. Each user is assigned to a tablespace (or multiple tablespaces). Most users will need to create database objects in which case it is not enough for them to have a tablespace, they also need to have a quota on the tablespace. As an example, if you log on as ronbn you can select data but you still cannot create a table:

```
SQL> select count(*) from user_tables;
CO UNT(*)
-----
0
1 row selected.
```

```
SQL> create table my_first_app_table(i int, j int);
create table my_first_app_table(i int, j int)

*
ERROR at line 1:
ORA-01950: no privileges on tablespace 'USERS'
```

As the error indicates, the insufficient privileges error does not stem from the inability to use the create table command (you just granted this privilege)—it is the lack of quota on the tablespace. To give the user quota on the tablespace use:

```
SQL> alter user ronbn quota 10M on users;
User altered.
```

At this point ronbn can create the table:

```
SQL> connect ronbn
Enter password: *****
Connected.
SQL> create table my_first_app_table(i int, j int);
Table created.
```

You can check your quotas when logged in as database administrator (DBA):

```
SQL> select username, tablespace_name, max_bytes
  2 from dba_ts_quotas
  3 order by username;
```

USERNAME	TABLESPACE_NAME	MAX_BYTES
DMSYS	SYSAUX	209715200
OLAPSYS	SYSAUX	-1
RONBN	USERS	10485760
SCOTT	SYSTEM	-1
SYSMAN	SYSAUX	-1
TEST1	SYSTEM	1048576
TEST1	USERS	1048576

7 rows selected.

Quotas are generally viewed as attributes related to data management but they should also be viewed as a security mechanism and the grant of a quota is actually managed as a privilege. The reason is that large quotas can be misused to create a denial-of-service (DoS) attack. For example, if an attacker manages to connect as a user that has an unlimited quota or a user with a very large quota they can start filling tables with random data until the disk runs out or until the database's quota at the operating system level is exceeded—at which point the database will stop functioning. Never assign the UNLIMITED TABLESPACE system privilege to anyone but a DBA and continuously audit which users have this (UNLIMITED) system privilege. If you want to revoke the ability to create objects in a tablespace simply alter the user:

```
SQL> alter user ronbn quota 0 on users;
User altered.
```

At which point the user will get the following error:

```
SQL> create table my_second_app_table(i int, j int);
*
ERROR at line 1:
ORA-01536: space quota exceeded for tablespace 'USERS'
```

Of course, everything you just saw is part of the CREATE USER command—you can create a user and all the associated attributes in one go:

```
SQL> create user ronbn
  2 identified by "0okmnhy^"
  3 default tablespace users
  4 quota 10M on users
  5 profile app_user;

User created.

SQL> grant create session to ronbn;
Grant succeeded.
```

Note that to run these commands you need to create the password profile before you make this call—that's the topic of the next HOWTO.

The ALTER USER format is similar:

```
SQL> alter user ronbn
  2 identified by "p455w0rd"
  3 default tablespace users
  4 quota 20M on users
  5 profile app_user;
User altered.
```

Each of the attributes themselves can be changed with the ALTER USER command. The most commonly used option is to change the password:

```
SQL> alter user ronbn identified by "0okmnhy6";
User altered.
```

Use this command if you are a DBA changing the password for another user or if you are changing your own password:

```
SQL> -- connected as system
SQL> alter user ronbn identified by "0okmnhy6";
User altered.

SQL> connect ronbn/0okmnhy6;
Connected.
SQL> alter user ronbn identified by "9ijnbgt5";
User altered.
```

HOWEVER—you should ALWAYS use the PASSWORD command rather than ALTER USER. You'll see why in Chapter 7 but for curiosity's sake, it is because ALTER USER passes the password in cleartext over the wire whereas PASSWORD passes just the hash value.

There is a difference between this process when you're changing your own password and when you're changing someone else's password (if you have the right privileges). If you're a DBA changing someone else's password you are not asked for the old password (one of the cases when you would do this would be when the user forgot their password). If you are changing your own password you need to type in the old password first so that someone doesn't just walk by your open session when you're away from your desk and change your password (regardless—don't leave sessions open and walk away). You should prefer to use `PASSWORD` than the `ALTER USER` command because the password is never visible on the screen (`ALTER USER` is useful in noninteractive password changes—e.g., within scripts):

```
SQL> connect system
Enter password: *****
Connected.
SQL> password ronbn
Changing password for ronbn
New password: *****
Retype new password: *****
Password changed
SQL> connect ronbn
Enter password: *****
Connected.
SQL> password ronbn
Changing password for ronbn
Old password: *****
New password: ****
Retype new password: ****
Password changed
```

To delete an account use the `DROP USER` command:

```
SQL> drop user ronbn cascade;
User dropped.
```

When you drop a user with the `CASCADE` option all of the user's schema objects are dropped and the schema is deleted from the data dictionary. Using `CASCADE` also ensures that all foreign keys and other dependencies on objects contained within the user schema are dropped.

A user cannot be dropped if there is an active connection using that user account. If you try to drop `ronbn` while there is a session open you will get the following error:

```
SQL> drop user ronbn cascade;
drop user ronbn cascade
*
ERROR at line 1:
ORA-01940: cannot drop a user that is currently connected
```

To drop the user you will have to find all sessions currently open for that user, kill them, and only then drop the user. Use `V$SESSION` to find all the SID-s and SERIAL#-s for that user:

```
SQL> select sid,serial#,username from v$session where username='RONBN';
```

```

SID      SERIAL#  USERNAME
-----
145      784  RONBN
156      633  RONBN

2 rows selected.

```

Next, use the KILL command to terminate these sessions:

```

SQL> alter system kill session '145,784';
System altered.
SQL> alter system kill session '156,633';
System altered.

```

Finally, drop the user:

```

SQL> drop user ronbn cascade;
User dropped.

```

If you don't want to drop the user but want to prevent them from connecting to the database you can lock their account and expire their password. These are both done using the ALTER USER command. To lock an account:

```

SQL> alter user ronbn account lock;
User altered.

```

At which point if the user tries to logon they get the following error message:

```

SQL*Plus: Release 10.2.0.1.0 - Production on Sun Jan 20 13:17:54 2008
Copyright (c) 1982, 2005, Oracle. All rights reserved.
ERROR:
ORA-28000: the account is locked

```

To unlock the account:

```

SQL> alter user ronbn account unlock;
User altered.

```

If you want to make sure that the user will need to change their password in case you later unlock their account then:

```

SQL> alter user ronbn account lock password expire;
User altered.

```

### Two Things to Remember about Actions on Accounts

1. Use PASSWORD EXPIRE to ensure that the user has to change their passwords.
2. Always change password using the PASSWORD command and not the ALTER USER command.

## 4.2 HOWTO Understand the Standard Logon Process

Passwords are the most common methods in which Oracle authenticates you when you logon to the database. Ensuring that users use strong passwords is one of the major things that you can do to protect your database. There are many things that Oracle does by default to ensure that passwords are not compromised and that the normal authentication schema is secure—but there are also things that fall within your realm of responsibility. Passwords need to be secured throughout their lifetimes. They need to be complex enough so as not to be easily guessed. They need to be secured when they travel on the network. The authentication process needs to prevent a brute-force attack in which passwords are guessed endlessly, etc.

To log you onto the database Oracle needs to get a password from you to compare it to something it has previously stored. When a client makes a connection Oracle needs to take the password that you type in when you logon and send something to the database server. The server then compares that to something already stored within the server. Passwords in Oracle are never sent in the clear and are never stored in the clear. In Oracle 11g passwords are encrypted using advanced encryption standard (AES) before they are sent over the network—irrespective of whether or not you enable network encryption. In Oracle 10g the logon scheme does not use AES but rather uses a proprietary scheme called O3LOGON. This scheme ensures that the password cannot be compromised if someone is sniffing the communication or intercepts it. The scheme works as follows:

1. The client sends the logon name to the database (a hex dump from part of a packet is shown in Listing 4.1).
2. The database sends a challenge created by encrypting a random number with the user's password hash (Listing 4.2). The random key can be thought of as session key and is different for each session.
3. The client computes the hash from the password and decrypts the random number—so now the client and the server have a common session key.
4. The client uses this random number as a key to encrypt the password and sends it to the database (Listing 4.3).

When the password is stored in USER\$, it is not stored in the clear—instead, a hash value is maintained. The hash value is based on a one-way hash function—i.e., a function that computes a value that is truly well distributed over all combinations and one which ensures that it is computationally

```
.....v.....
scott...AUTH_TERMINAL...
SERVER2003....AUTH_PROGR
AM_NM...sqlplus.exe....A
```

**Listing 4.1 Extract from a logon packet—Client sends the logon name to the database.**

```
AUTH_SESSKEY.@@0436D43AF
F246F6B1AAAAA4E9C783DC42
2F69564CD3947CC399CBF9FA
B0ECAF1....AUTH_VFR_DATA
```

**Listing 4.2 Extract from a logon packet—Database encrypts a random number with the stored password hash.**

```
.....scott.. .... AUTH_PASSWD
RD.@@C4AD8FECF2FEB66BBA3935874F6
81B868C4125985E1E62C4439CD1FB5AB
```

---

**Listing 4.3 Extract from a logon packet—Client sends the encrypted password to the server.**

not feasible to compute the original password from the hash value. In Oracle 11g the hash value is based on secure hash algorithm-1 (SHA-1)—considered one of the best one-way functions today. The algorithm in 11g has these steps:

1. Generate random bits (called salt)
2. Concatenate the password with the salt (keep the password case sensitive by default)
3. Compute a hash value using SHA-1
4. Concatenate the hash value with the randomly selected salt—this is what is stored in the database

In step 4, the salt has to be stored because when the user authenticates with the database, the same random bits must be used or the hash will come out completely different.

Prior to 11g the hash algorithm is an Oracle proprietary algorithm that works as follows:

1. Concatenate the username with the password and convert it to uppercase (which is why before Oracle 11g passwords were case insensitive).
2. Convert the string to double-byte characters and pad with zeros until the string's length is a multiple of 8 bytes.
3. Encrypt the string using triple data encryption standard with cipher block coding (3DES)/CBC using a fixed key (don't be alarmed—we're not done yet—it's ok to use a fixed key in this step even if the whole world knows what this key is).
4. Take the last 8 bytes of the encrypted text as a new key. This is why it is ok—the first encryption is used to ensure that the last 8 bytes are dependent on the entire string.
5. Encrypt the string again using 3DES/CBC and the new key generated in step 4.
6. Take the last 8 bytes of this string and convert it to printable hex—so you get a hash for 16 printable characters.

The move in Oracle 11g to standard algorithms (AES and SHA-1) is important. Although no one has broken the Oracle proprietary algorithms there is no reason to invent encryption and hashing schemes—it is more comforting to use an industry standard that has gone through VERY rigorous mathematical analysis and is trusted by even the most security-demanding organizations.

### The 6 Things to Remember about the Standard Logon Process

1. Passwords are never sent in cleartext over the wire as part of the logon process—they are always encrypted.
2. The password hash is used to encrypt random session keys. Because the password hash is known to both client and server they are used to enable the sharing of these keys.
3. Oracle 11g uses standard algorithms—AES and SHA-1.

## 4.3 HOWTO Use Password Policies

Whenever you create a user, a default password profile is assigned to the user. Password profiles define how Oracle will manage password-related issues—for example, how many failed logon attempts Oracle allows before it locks out the account, what kinds of passwords it accepts (in terms of password complexity), and how long you can keep a password before Oracle forces you to change it. It is your job to ensure that you use secure password profiles and that all users have a secure password profile. This is especially true because the default password policy does not have enough settings to protect you from password-based attacks. To create a password profile:

```
SQL> create profile sec_prof limit
  2 failed_login_attempts 5
  3 password_life_time 60
  4 password_reuse_time 60;
Profile created.
```

What you've defined is a profile by which an account will be locked if there are five continuous failed login attempts, that the password lifetime is 60 days and that the same password cannot be reused within a period of 60 days. You can change the attributes of a password profile, for example:

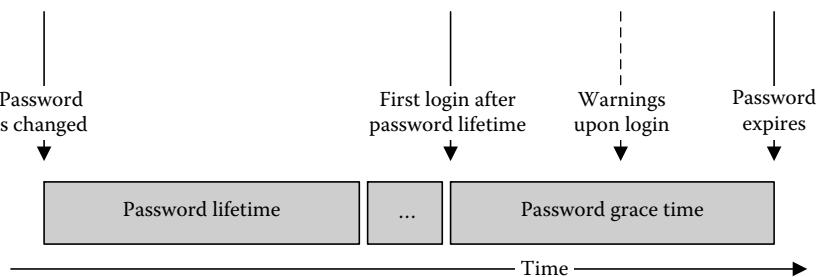
```
SQL> alter profile sec_prof limit password_reuse_max 10;
Profile altered.
```

When creating a user you assign the password profile to enforce the password protection behavior:

```
SQL> create user myappusr identified by <pwd>;
User created.
SQL> grant create session to myappusr;
Grant succeeded.
SQL> alter user myappusr profile sec_prof;
User altered.
```

The full list of attributes that you can set for a password profile is shown below. In Oracle 11g each parameter that is omitted has a default value also shown:

- FAILED\_LOGIN\_ATTEMPTS: # of failed logins before an account is locked. The default in 11g is 10.
- PASSWORD\_LIFE\_TIME: # of days after which a password must be changed before it becomes invalid. The default in 11g is 180.
- PASSWORD\_REUSE\_TIME: # of days within which the same password cannot be reused (e.g., change to another password and then change it right back). This parameter is used in concert with PASSWORD\_REUSE\_MAX—that is, you must set values for both. PASSWORD\_REUSE\_MAX is the # of times that you may reuse a password. The effect on the password profile is as follows:
  - If you specify an integer number for both then you cannot reuse a password until the password has been changed PASSWORD\_REUSE\_MAX times during PASSWORD\_REUSE\_TIME days.



**Figure 4.1 Understanding password lifetime and grace time.**

- If one of the parameters is set to UNLIMITED and the other has an integer value then you can never reuse a password.
- If one of the values is set to DEFAULT then the value is taken from the default password policy.
- If both are set to UNLIMITED then Oracle ignores these values in the password profile. This is the default behavior if you omit these parameters.
- PASSWORD\_LOCK\_TIME—# of days that must pass after an account is locked out before it is unlocked. The default in 11g is one day.
- PASSWORD\_GRACE\_TIME—When a password expires this parameter allows for an additional grace period during which a user can logon using the old password. Figure 4.1 depicts what Oracle will do based on the time values you define as the grace period and the lifetime. A warning is issued when you logon during the grace period. The default in 11g is seven days.
- PASSWORD\_VERIFY\_FUNCTION—Allows you to specify a function for validating password strength—you'll learn about this in the next HOWTO.

For all the numerical values you can use UNLIMITED—but you should prefer not to do so (except for PASSWORD\_LOCK\_TIME). This defeats the purpose of using password profiles which is to protect you from a password-based attack.

To set values for the default profile (the one used for all users unless you specifically alter a user to assign it to another profile), use:

```
SQL> alter profile default limit password_life_time 90;
Profile altered.
```

Finally, one quick mention of a new initialization parameter introduced in Oracle 11g—SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS (with a default value of 10). This parameter (set with an ALTER SYSTEM command) controls the number of authentication attempts that a connection can attempt before the server drops the connection. This is different from FAILED\_LOGIN\_ATTEMPTS in password profiles. The value in password profiles controls the attempts to logon to a user account. SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS is oblivious to which account the attempt relates to—so if an attack randomly guesses both an account name and a password then after ten attempts the server will drop the connection. Note that no account will be locked—the attacker can initiate a new connection and try ten more. However, dropping and restarting a new connection takes time—so this slows down the attack considerably and allows you to react to it.

### Two Things to Remember about Password Policies

1. A password profile is always assigned to a user—even if you don’t do so explicitly. The default password profile is used—so set the default profile to ensure all users get the values that comply with your security policy.
2. Don’t use UNLIMITED as a value within the password policy. The only exception may be functional accounts used by application connection pools.

## 4.4 HOWTO Enforce Password Complexity

Passwords should not be easy to guess. Passwords which are easy to guess allow an attacker to gain access to an account. There are many recommendations as far as creating complex passwords; one of the more common guidelines for password strength is

- Passwords should have at least 8 characters and no more than 30 characters. Passwords should never be shorter than 8 characters and 30 characters is the maximum in Oracle.
- Passwords should have at least one uppercase character, one lowercase character, at least one number, and at least one special character.
- Password should not start with a number.
- Passwords should not be a word that can be looked up in a dictionary.
- Passwords should not be constructed using a well-defined scheme.

The last directive in this list requires a little more explanation. Because passwords that adhere to complex policies are not easy to create and are even harder to remember, people have come up with some fairly clever schemes to generate passwords that will conform to these requirements and yet are not hard to remember. As an example, let’s look at an example for a scheme that many people use:

- Choose a word that you can easily remember but which is long enough.
- Choose a word that starts with a consonant and uppercase the first letter in the password.
- Start replacing some letters in the word with numbers and special characters in a way that is deterministic and easy to remember. For example, use numbers and special characters that look somewhat like the letters you are replacing—replace A with 4 and S with 5, etc. Don’t replace the first letter even if it is a vowel.

Let’s look at a few examples:

- Start with the word “malicious” and replace some letters—so your password becomes M4l1c10^5.
- Start with the word supersecret and replace some letters—so your password becomes S^p3r53cr3t.

This sounds like a reasonable scheme, doesn’t it? If you think it does, do a quick search on the Web for a program called “John the Ripper” and various variants on the original program—you’ll see that people have created password crackers based on such schemes. You should assume that any method that you come up with to generate passwords that will meet these specifications in a way

that will be easy to remember have already been devised, and that password guessers based on these method also have been created. Unfortunately, only random passwords are REALLY secure.

As discussed in Section 4.3, you enforce password strengths through the PASSWORD\_VERIFY\_FUNCTION parameter of password profiles. This parameter gets a value which is a PL/SQL function that is verified every time a user assigned the password profile tries to set a password. If the desired password is not compliant with what this function requires the change will fail.

Oracle provides a sample password verification function in the UTLPWDMG.SQL script located in \$ORACLE\_HOME/RDBMS/ADMIN. You can use this function as is or use it as a starting point to build a function that is specific to your organization's password requirements. The script file provides two different implementations of a PASSWORD\_VERIFY function—one that you can use in Oracle 11g and one that you can use in pre-11g instances.

### Two Things to Remember about Password Complexity

1. A good policy is to require passwords to have at least eight characters, at least one uppercase, one lowercase, one number, and one special character.
2. Use the sample UTLPWDMG.SQL script provided as an example for a complexity verification function.

## 4.5 HOWTO Check for Weak and Default Passwords

Default accounts that come prepackaged with the database and have default and well-known passwords have plagued Oracle (and every other database for that matter). Many of the simplest attacks have been carried out by starting with a logon to a default account with a default password. Oracle 11g has two remedies for this. First, default accounts that come with the database are either set to be locked with an expired password or have to be given a nondefault password when you install the database. This takes care of new installations of the database. But what happens with database that you upgrade from 10g? A new feature of Oracle 11g helps you with this and allows you to continuously monitor if there are any default passwords. Oracle 11g has a new view called DBA\_USERS\_WITH\_DEFPWD. If you select from this view you will get a list of all default accounts that have a default password. As an example, if you have the default SCOTT account set with a password of TIGER and you run:

```
SQL> select * from dba_users_with_defpwd;
```

You will get the result:

USERNAME
SCOTT

Oracle 11g has all the hashes for the default passwords and checks if there are any passwords in USER\$ that have the hash values for these default passwords. This makes it very easy for you to make sure that you're not exposed through default passwords. You should periodically (as part of a regular audit) select from this view and make sure that it comes back empty. You can find more resources for checking for default and weak passwords in Section 4.10.

### The ~~e~~ Things to Remember about Default Passwords

1. Many experts consider default passwords to be one of the major reasons why attacks occur.
2. In 11g all default accounts come expired and locked by default. This is not the case prior to 11g.
3. Oracle 11g has a view that shows you all accounts with default passwords—make sure to use it periodically.

## 4.6 HOWTO Set Password Case

Oracle passwords have always been case insensitive. This changed in Oracle 11g. By default, passwords are now case sensitive and therefore stronger. An initialization parameter called SEC\_CASE\_SENSITIVE\_LOGON controls whether or not passwords are case sensitive—the default being TRUE. Set this value to FALSE and restart the database if you need to remain with case insensitive passwords.

If you upgrade a database from Oracle 10g to 11g and you leave the default so that passwords are case sensitive, then the first time users change their passwords the passwords become case sensitive. A user that was defined in 10g will have an uppercase password until they change their password. When they change their password they are free to choose any case for any letter in the password. Furthermore, a new column PASSWORD\_VERSIONS in the DBA\_USERS view provides information regarding which version of Oracle the password was created in. In the example below the user RON was created in 10g but changed password after the upgrade to 11g, the user JANE was created in 11g and the user RONB was created in 10g and has not changed password since the upgrade. Therefore, RONB still has a fully uppercase password but the passwords for RON and JANE are case sensitive:

```
SQL> select username,password_versions from dba_users;
USERNAME  PASSWORD_VERSIONS
-----
RON 10G 11G
JANE 11G
RONB 10G
```

Password case sensitivity in Oracle 11g is also supported when you use password files (see Chapter 6). To make a password file support case sensitive passwords use the orapwd utility with the new parameter IGNORECASE = N. If you have a password file that was created in Oracle 10g and you upgrade your database, re-create the password file and regrant the sysdba and sysoper privileges as described in Chapter 6.

In addition to this initialization parameter and the SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS initialization parameter already mentioned in HOWTO 4.3, Oracle 11g introduces a few other initialization parameters that affect account security. These initialization parameters are used as the default password profile:

- FAILED\_LOGIN\_ATTEMPTS—default is 10.
- PASSWORD\_GRACE\_TIME—default is 7.
- PASSWORD\_LOCK\_TIME—default is 1.

- `PASSWORD_LIFE_TIME`—default is 180.
- `PASSWORD_REUSE_MAX`—default is UNLIMITED.
- `PASSWORD_REUSE_TIME`—default is UNLIMITED.

When you start using case-sensitive passwords you need to pay special attention to database links. For credential-based links you need to consider links that go between pre-11g to 11g databases. If you are linking from a pre-11g database the passwords you use for the link user must be all uppercase—because the pre-11g database will uppercase the password when it tries to connect. For links that go from 11g to 11g and for links that go from 11g to pre-11g there is no issue—the first because both sides are case sensitive and the second because the password will be converted to uppercase by the authenticating pre-11g database.

#### Two Things to Remember about Password Case

1. Passwords in 11g are case sensitive by default.
2. Links defined from pre-11g databases to 11g databases must use an account with an all-uppercase password.

## 4.7 HOWTO Use Impossible Passwords

You've already seen how password hashes are computed in both Oracle 11g and pre-11g databases. In 11g the password hash is generated by using SHA-1 and in pre-11g a complex scheme that includes encryption. Recall that the last step of this proprietary scheme involves taking the last 8 bytes which form the hash—and converting it into 16 printable hex characters. Saving hashes as printable hex is very natural because a byte has 256 possible values which are represented by two hex digits and using printable hex means that the values can be read by humans.

One implication that follows from schemes that use printable hex characters is that password hashes cannot include certain characters—at least not if they were generated by the algorithm. As an example, a password hash can never include an “S” or a “T” or a “P”—or for that matter any letter apart from A, B, C, D, E, or F. You can use this to your benefit in generating what is called an impossible password. You can do this because when you create a user you can either provide a password or you can directly provide a password hash.

Let's look at an example. The standard way you create a user is to provide a password. In this case Oracle computes the password hash and stores it in `USER$`:

```
SQL> create user usr1 identified by password;
User created.

SQL> select name,password from sys.user$ where name like 'USR%';
```

NAME	PASSWORD
USR1	46A3F68B0AAC78BC

1 row selected.

Instead, Oracle allows you to create a user and give a password hash using IDENTIFIED BY VALUES instead of IDENTIFIED in the CREATE USER command. In this case the value that you provide is saved within the USER\$ table as is:

```
SQL> create user usr2 identified by values 'password';
User created.

SQL> select name, password from sys.user$ where name like 'USR%';
```

NAME	PASSWORD
USR1	46A3F68B0AAC78BC
USR2	password

2 rows selected.

At this point USR2 will never be able to logon to Oracle. Why? Because there is no password on earth that the user can enter that, after applying the SHA-1 or the proprietary algorithm will produce a hash with a value of “password”—there are letters in this value that are not hex.

Why might you want to use impossible passwords? Why would you not just lock an account or drop the user? Usually you would—locking an account is the normal facility that you should normally use. However, there is one subtle difference in terms of the information Oracle provides when someone tries to logon using that user. Lock the USR1 account and try to logon using both USR1 and USR2 to see what happens:

```
SQL> alter user usr1 password expire account lock;
User altered.
SQL> connect usr1/password
ERROR:
ORA-28000: the account is locked
Warning: You are no longer connected to ORACLE.
SQL> connect usr2/password
ERROR:
ORA-01017: invalid username/password; logon denied
```

The subtle difference is the error message produced when the logon is unsuccessful and the information you provide to a potential attacker. When you lock out the account the message produced tells the attacker exactly what is going on. This helps the attacker—they know what the status is and can direct their efforts appropriately. In the second case, when using impossible passwords, you provide absolutely no information—the attacker cannot know if the account exists, if the password is wrong, etc. Less information is better—so use it if you have very special requirements for certain accounts. Incidentally, the ANONYMOUS user in a default installation has an impossible password:

```
SQL> select username,password from dba_users where username='ANONYMOUS';

USERNAME          PASSWORD
-----
ANONYMOUS        anonymous

1 row selected.
```

### Two Things to Remember about Impossible Passwords

1. Password hashes can only include hex characters.
2. You can set a password hash directly. If you set the hash to have any letter above F no one will ever be able to logon to that account because no password will map to the hash. The error message will not allow an attacker to learn anything about this account.

## 4.8 HOWTO Limit System Resources Used by Users

In HOWTO 4.1 you saw that when you create a user you can set their quota on the tablespace. From a security point of view, you do this to contain any DoS attack that may be launched from this account. There are many more system resource limits that you can set for an account—and from a security perspective they too should be used to contain a possible DoS attack.

Limiting system resources is done through profiles in the same way that you use password profiles (see HOWTO 4.3). For example, to limit the time a user session is allowed to remain connected and the time a session can be idle you create a profile and assign it to a user as follows:

```
SQL> create profile conn_prof limit
  2 connect_time 240
  3 idle_time 30;
Profile created.
SQL> alter user ronb profile conn_prof;
User altered.
```

Profiles can create both password policy-related parameters as well as resource limits. There is no difference as far as Oracle is concerned—both define boundaries for connections made with a certain user account. Therefore, you can mix these parameters into a single profile as follows:

```
SQL> create profile sec_prof limit
  2 connect_time 240
  3 idle_time 30
  4 failed_login_attempts 5
  5 password_life_time 60
  6 password_reuse_time 60;
Profile created.
```

The list of system resources that you can limit using profiles is shown below. There are three types of resources—connection resources, session resources, and call resources. For connection resources (CONNECT\_TIME and IDLE\_TIME) if your session exceeds these times the database rolls back the current transaction and ends your session. What you will see is an error when you issue the next command. If you violate a session resource limit Oracle will abort the operation, roll back the current statement and return an error. You can commit or roll back the current transaction but then you need to exit the session. If you violate a call limit Oracle aborts the operation, rolls back the current statement, and returns an error. Each of these values can be set to UNLIMITED or to DEFAULT in which case the value of the default profile is used.

1. CONNECT\_TIME—The maximal time in minutes for a session.
2. IDLE\_TIME—The maximal time in minutes that a session is allowed to be inactive. Both the CONNECT\_TIME and the IDLE\_TIME are not continuously checked because this is

a waste of resources. A check is performed every few minutes so the termination of a session can be up to five minutes later than the value you define in the profile.

3. SESSIONS\_PER\_USER—The maximal number of concurrent sessions allowed per user who receives this profile.
4. CPU\_PER\_SESSION—The CPU time limit in hundredths of a second that is allowed per session.
5. CPU\_PER\_CALL—The CPU time in hundredths of a second that is allowed per call.
6. LOGICAL\_READS\_PER\_CALL—The number of data blocks read from memory or disk that is allowed for processing a single SQL statement.
7. LOGICAL\_READS\_PER\_SESSION—The number of data blocks read from memory or disk that is allowed per session.
8. COMPOSITE\_LIMIT—The total resource cost in service units that is allowed per session. The COMPOSITE\_LIMIT is a weighted sum of CPU\_PER\_SESSION, CONNECT\_TIME, LOGICAL\_READS\_PER\_SESSION, and PRIVATE\_SGA and gives you a normalized measure of how many resources the session is consuming.
9. PRIVATE\_SGA—The maximal size in the shared pool of the SGA that is allowed per session.

### **Two Things to Remember about Limiting Resources**

1. Use profiles to limit system resources used by user sessions.
2. Limit system resources to combat DoS attacks.

## **4.9 HOWTO View Information on Users and Profiles**

Oracle gives you a number of views for inspecting information that you need to manage users and profiles. The views that you would use most often are

1. DBA\_PROFILES—Shows all the profiles and their defined limits. For example:

```
SQL> select * from dba_profiles order by profile;
```

PROFILE	RESOURCE_NAME	RESOURCE	LIMIT
APP_USER2	COMPOSITE_LIMIT	KERNEL	DEFAULT
...			
APP_USER2	FAILED_LOGIN_ATTEMPTS	PASSWORD	5
APP_USER2	PASSWORD_LIFE_TIME	PASSWORD	30
APP_USER2	PASSWORD_REUSE_TIME	PASSWORD	60
APP_USER2	PASSWORD_REUSE_MAX	PASSWORD	5
APP_USER2	PASSWORD_VERIFY_FUNCTION	PASSWORD	DEFAULT
APP_USER2	PASSWORD_LOCK_TIME	PASSWORD	10
APP_USER2	PASSWORD_GRACE_TIME	PASSWORD	10
...			
CONN_PROF	IDLE_TIME	KERNEL	30
CONN_PROF	CONNECT_TIME	KERNEL	240
CONN_PROF	PRIVATE_SGA	KERNEL	DEFAULT
CONN_PROF	FAILED_LOGIN_ATTEMPTS	PASSWORD	DEFAULT
...			
DEFAULT	COMPOSITE_LIMIT	KERNEL	UNLIMITED
DEFAULT	SESSIONS_PER_USER	KERNEL	UNLIMITED
...			

2. DBA\_TS\_QUOTAS (and USER\_TS\_QUOTAS for a specific user)—Shows all tablespace quotas currently defined. For example:

```
SQL> select * from dba_ts_quotas;
```

TABLESPACE_NAME	USERNAME	BYTES	MAX_BYTES	BLOCKS	MAX_BLOCKS	DRO
SYSAUX	DMSYS	262144	209715200	32		25600 NO
SYSAUX	SYSMAN	51838976	-1	6328		-1 NO
USERS	TEST1	65536	1048576	8		128 NO
SYSAUX	OLAPSYS	16318464	-1	1992		-1 NO
SYSTEM	SCOTT	0	-1	0		-1 NO
SYSTEM	TEST1	0	1048576	0		128 NO

6 rows selected.

3. DBA\_USERS shows all users in the database (ALL\_USERS for what the logged on user can see). For example:

```
SQL> select username, profile, account_status from dba_users;
```

USERNAME	PROFILE	ACCOUNT_STATUS
SECURITY_ROLES	DEFAULT	OPEN
JONES	DEFAULT	OPEN
MYAPP	APP_USER2	OPEN
HR	DEFAULT	OPEN
JANE	DEFAULT	OPEN
RONB	CONN_PROF	EXPIRED
TSMSYS	DEFAULT	EXPIRED & LOCKED
BI	DEFAULT	EXPIRED & LOCKED
PM	DEFAULT	EXPIRED & LOCKED
DBSNMP	MONITORING_PROFILE	OPEN
...		

You can also use this view to show the password hashes:

```
SQL> select username,password from dba_users;
```

USERNAME	PASSWORD
SECURITY_ROLES	952EDE259E4D2BB4
JONES	5215700C08CDCF93
MYAPP	C050EF9228761727
HR	4C6D73C3E8B0F0DA
JANE	4968BF82C7B085C6
RONB	35E71E94C51B984C
TSMSYS	3DF26A8B17D0F29F
BI	FA1D2B85B70213F3
PM	72E382A52E89575A
DBSNMP	A0070F5A6F1A9AF8
...	

4. RESOURCE\_COST—Shows the cost per resource in terms of CPU, read time, connection time, and SGA size per session.
5. USER\_PASSWORD\_LIMITS—Shows the password parameters assigned to the user. For example:

```
SQL> connect myapp
Enter password: *****
Connected.
```

```
SQL> select * from user_password_limits;
```

RESOURCE_NAME	LIMIT
FAILED_LOGIN_ATTEMPTS	5
PASSWORD_LIFE_TIME	30
PASSWORD_REUSE_TIME	60
PASSWORD_REUSE_MAX	5
PASSWORD_VERIFY_FUNCTION	NULL
PASSWORD_LOCK_TIME	10
PASSWORD_GRACE_TIME	10

7 rows selected.

6. USER\_RESOURCE\_LIMITS—Shows the resource limits for the user. For example:

```
SQL> select * from user_resource_limits;
```

RESOURCE_NAME	LIMIT
COMPOSITE_LIMIT	UNLIMITED
SESSIONS_PER_USER	UNLIMITED
CPU_PER_SESSION	UNLIMITED
CPU_PER_CALL	UNLIMITED
LOGICAL_READS_PER_SESSION	UNLIMITED
LOGICAL_READS_PER_CALL	UNLIMITED
IDLE_TIME	30
CONNECT_TIME	240
PRIVATE_SGA	UNLIMITED

9 rows selected.

7. V\$STATNAME—Shows user session statistics.

## 4.10 Additional Resources

There are many resources on the Web for helping you check for default accounts and default passwords, and many resources that can help check for weak passwords. In Oracle 11g many of that is built into the database but for pre-11g versions these resources can come in very handy:

**Lists of default accounts:**

- List of default accounts in a normal Oracle 11g installation—See Table 3-1 and Table 3-2 in Oracle 11g Release 1 Security Guide (Part Number B28337-03 Chapter 3).
- Pete Finnigan's list of default passwords (includes many options, modules, and applications above and beyond the base database): [www.petefinnigan.com/default/default\\_password\\_list.htm](http://www.petefinnigan.com/default/default_password_list.htm)

Password checkers/guessers:

- [www.cquare.net](http://www.cquare.net)
- [www.red-database-security.com/software/checkpwd.html](http://www.red-database-security.com/software/checkpwd.html)
- Password checker based on Pete Finnigan's list of default accounts/passwords: [www.petefinnigan.com/default/default\\_password\\_checker.htm](http://www.petefinnigan.com/default/default_password_checker.htm)

**Examples for writing password verifier functions and scripts for checking password complexity:**

- Scripts for checking password complexity: UTLPWDMG.SQL in \$ORACLE\_HOME/RDBMS/ADMIN. There are two functions—one checking 11g databases and one checking pre-11g databases. Among other things the script checks that
  - Passwords contain at least 8 characters and no more than 30.
  - Passwords are not the same as the user name, the user name spelled backwards, or the user name with numeric characters appended.
  - Passwords are not the same as the server name or the server name with numeric characters added.
  - Passwords are not change\_on\_install.
  - Passwords are not simple words in a dictionary or dictionary words with numeric characters added.
  - Passwords include at least one numeric and one alphabetic character.
- Examples for verifier function:
  - <http://orafaq.com/node/58>
  - <http://aspalliance.com/746>
  - <http://www.pentest.co.uk/cgi-bin/viewcat.cgi?cat=downloads>

**Password management tools (help you remember/store your passwords securely):**

- Password safe—<http://sourceforge.net/projects/passwordsafe>
- Keepass—<http://keepass.sourceforge.net>

## *Chapter 5*

---

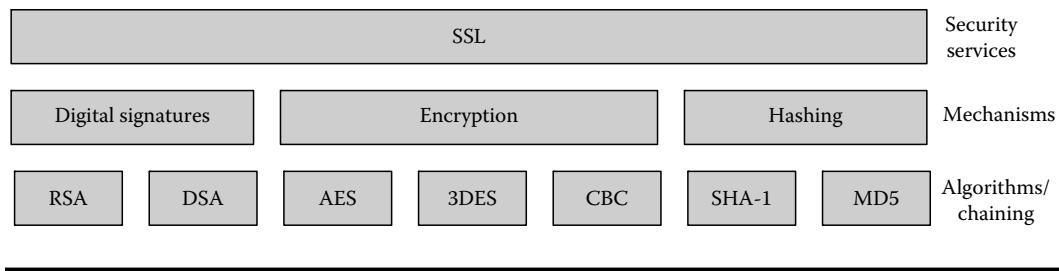
# Cryptography, Oracle Wallets, and Oracle PKI

---

In this chapter you'll learn about two things—cryptography and the Oracle wallet. The first part of this chapter is a general introduction to cryptographic techniques, encryption algorithms, message digests, certificates, etc. This introduction will be short and you won't get into too much detail—but because there are many chapters later in the book that deal with encryption and authentication using cryptographic techniques it is important to understand the terms and what they mean. As an example, throughout many chapters in the book you'll have to decide whether to use Advanced Encryption Standard (AES) or triple Data Encryption Standard (3DES), whether to encrypt traffic using Secure Sockets Layer (SSL) or Diffie–Hellman key distribution, etc. The second part of the chapter includes HOWTOs related to using the Oracle wallet and the Oracle Wallet Manager (OWM). The Oracle wallet is a password-protected container that is used to store credentials used for authentication and signing including private keys, certificates, and trusted certificates. OWM is an application and orapki is a command line utility—both used for managing keys and certificates within the wallet. It is Oracle's implementation of Public Key Infrastructure (PKI) and you will use it when implementing secure communications, secure authentication, data encryption, backup encryption, and more.

Cryptography is a broad term encompassing the use of mathematical techniques in the context of information security. The word comes from the Greek words *krypto* which means “hidden” and *grafo* which means “to write.” Most people equate cryptography with encryption—the process of converting plaintext into ciphertext (unreadable encoding of the original message). Although encryption is certainly one of the most common uses of cryptography, it is not the only thing you use keys and algorithms for. You also use cryptography to protect your data through message digests for ensuring message integrity, through digital signatures, in authentication, and more. The main elements of cryptography which you'll learn about in this chapter are shown in Figure 5.1 and include

1. Block ciphers—e.g., Cipher Block Chaining (CBC)
2. Symmetric-key algorithms—e.g., AES and 3DES
3. Public-key algorithms—e.g., rural service area (RSA)



**Figure 5.1** Cryptographic elements.

4. Certificates—e.g., X.509 certificates
5. Message digests—e.g., secure hash algorithm-1 (SHA-1) and message digests 5 (MD5)
6. Digital signatures—e.g., RSA digital signatures
7. Security services—e.g., SSL

These elements are then used for the following purposes (all of which are relevant in Oracle security):

- Confidentiality—how can you make sure that an eavesdropper or a thief that takes your data file cannot read your data? Protecting data from an eavesdropper is often called protecting data-in-transit and protecting from media-theft is often called protecting data-at-rest.
- Message integrity—how can you know that a message has not been tampered with and modified before it reaches the database?
- Authentication—how can you know that a message or an action comes from a certain person or party?
- Nonrepudiation—how can you be sure that an interaction you carry out with another party (a “contract”) will not be denied later on the pretense that “it wasn’t them?”

All of these purposes are things we take for granted these days, but our life would be quite different if we did not have mathematical techniques that allowed us to have privacy, integrity, etc.

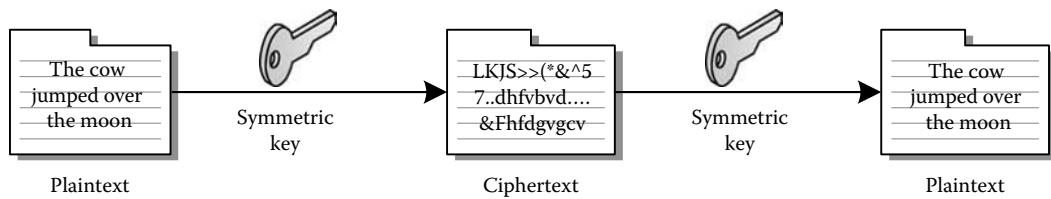
### Encryption, Cryptographic Algorithms, and Ciphers

Encryption is the process of transforming plaintext using an algorithm to make it unreadable to anyone except those possessing special knowledge. That special knowledge is the key that we mention when discussing cryptography. The algorithm is called the cipher.

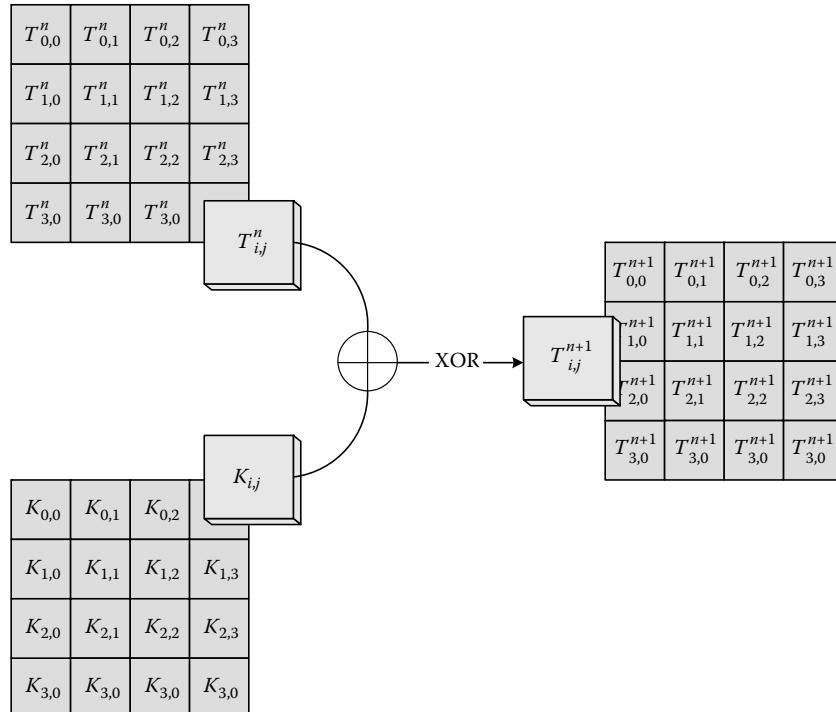
There are two types of cryptography or encryption algorithms—symmetric-key cryptography and public-key cryptography (which we'll discuss in the next subsection). In symmetric-key cryptography the two parties communicating share the same key (or, they may be different but they can be easily computed one from the other). Almost all encryption/decryption in the world is based on symmetric-key cryptography—including all the methods that Oracle uses to encrypt data-in-transit and data-at-rest. This does not mean you don't also use public-key encryption—this will become clearer in the next subsection.

There are many symmetric-key algorithms (or ciphers). All of these ciphers work the same way in terms of the fact that an encryption key is used to encrypt plaintext to form ciphertext and

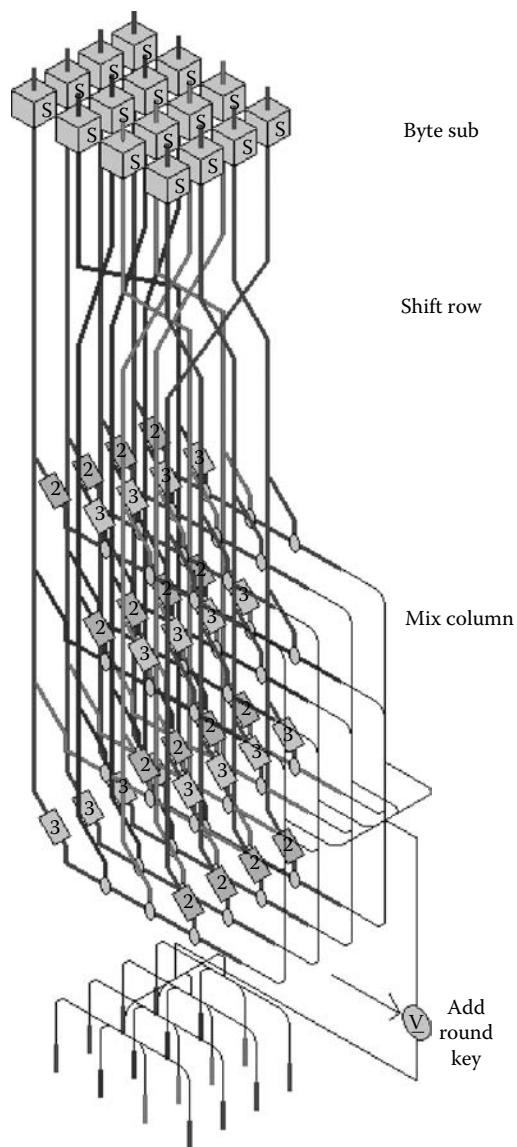
then the same key (or closely related to it) is used to decrypt ciphertext to plaintext. Symmetric-key encryption is depicted in Figure 5.2. The key is typically used to XOR data—e.g., plaintext is XOR-ed with the key as shown in Figure 5.3 where each bit of the  $(n + 1)$  step text is a bitwise XOR of the relevant bit of the  $n$  step text with the relevant bit from the key. The result of this XOR is not the ciphertext—it is just one step in an algorithm that usually involves a very large number of XORs, permutations, transformations, etc. All cryptographic algorithms are highly complex. Although they usually involve simple atomic operations that can be done very quickly on a computer (such as XOR), there is a large number of steps and iterations in any such algorithm. Cryptographic algorithms are CPU-intensive and affect the overall performance of a database.



**Figure 5.2** Symmetric-key encryption.



**Figure 5.3** Third step (AddRoundKey step) of the AES algorithm—XOR with a subkey.



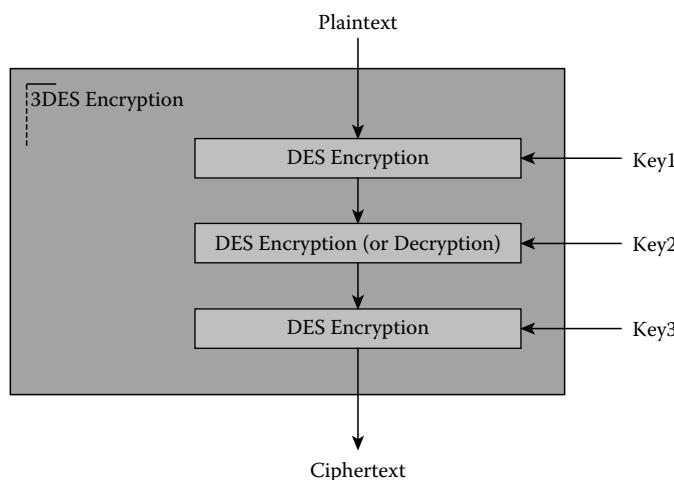

---

**Figure 5.4 One round of AES.**

server—all depending on how much data you are encrypting and decrypting. Figure 5.4 shows one round of the AES algorithm. As you can see, each round is a complex set of operations and a full AES encryption run has 9, 11, or 13 rounds. If you want to learn more, there's an excellent animation of the AES algorithm at [http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael\\_ingles2004.swf](http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf).

There are many ciphers to choose from. When you use encryption within Oracle you will be choosing from one of these ciphers:

- DES—DES is an algorithm you should never ever use. DES was selected as the official U.S. standard in 1976 when computers were much weaker in terms of processing power. The algorithm was always controversial, has a short key length of 56 bits, and because parts of it are classified it was always thought that the U.S. National Security Agency (NSA) maintained a backdoor to the algorithm. With today's computers you can break DES within 24 hours—never use it.
- 3DES—Triple DES is an algorithm which has been in use extensively and still is—especially in the financial sector. It is considered to be a strong algorithm, but there are better algorithms for you to choose from today (see AES later in this list). Triple DES, as its name implies, derives from DES by using it three times as shown in Figure 5.5. DES is too weak due to its short 56 bit key—so 3DES evolved as a simple way to strengthen the encryption against brute-force attack. There are two variants. In one variant, called EEE, three encryption steps are made. In the second variant, called EDE, the first and third step are encryption steps but the second step is a decryption step (obviously, not with the same key as the first step). 3DES has either a 168 bit key length (three times 56 bits used in DES) or 112 bits where the first and third DES keys are the same. If you are going to use 3DES, use the version with 168 bit key length.
- Ron's Code number 4 (RC4)—RC4 was designed by Ron Rivest of RSA. You should not use this as your chosen cipher unless you have no choice. As you'll see in Chapter 7 when discussing network encryption of type-4 Java Database Connectivity (JDBC) drivers, you sometimes have to use this cipher. RC4 is used in SSL and in Wired Equivalent Privacy (WEP) but is known to have vulnerabilities. With careful inspection around what happens at the beginning of the session and how random the key is, RC4 can be secure enough for practical use. Don't use RC4 unless you must and if you must, choose a 256 bit length.



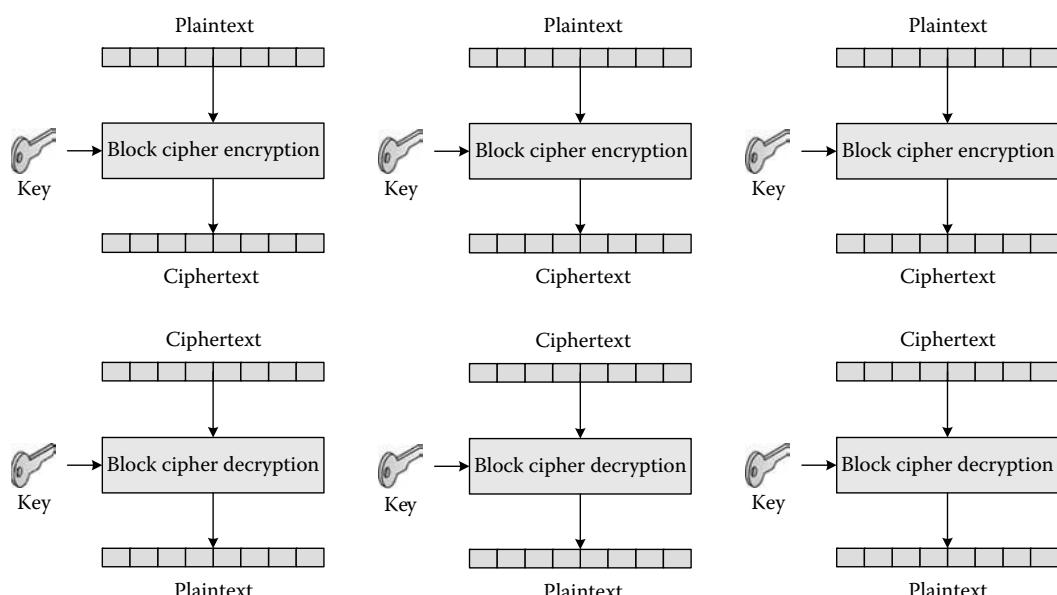
**Figure 5.5 3DES encryption.**

- AES—AES is the algorithm that you should always use if you have a choice. AES (also known as Rijndael) was adopted as the encryption standard by the U.S. government in 2002 after a five-year process of analysis, comparison, and selection. It has been analyzed extensively and is widely used worldwide. Key sizes are either 128, 192, or 256 bits—which is what AES128, AES192, and AES256 mean when they appear as a selection in Oracle.

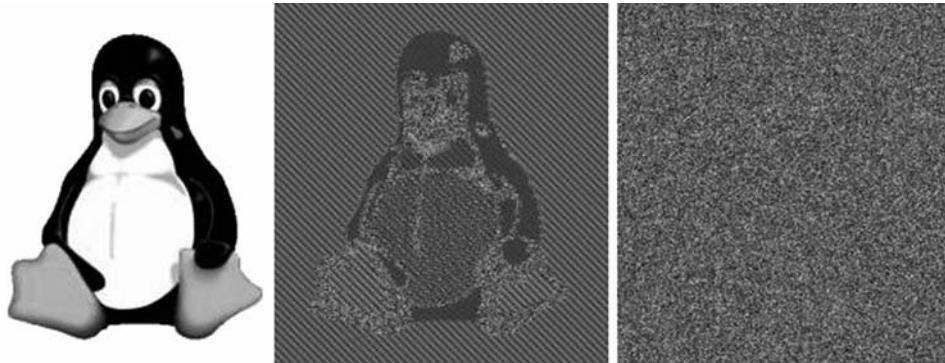
AES, 3DES, and DES are all called block ciphers (RC4 is a stream cipher). They're called block ciphers because they always work on a block of data. When you want to encrypt 1K worth of data using (as an example) AES, you don't just pass in 1024 bytes to the encryption process. The block size that a block cipher works on depends on the key length. If, for example, you use AES192 then your block length is 192 bits. To encrypt your 1024 the full encryption algorithm breaks your input into  $1024 \times 8 / 192 = 42.66$ , or 43 blocks—each one which will undergo encryption. The same is true when you want to encrypt only a bit of data. If you want to encrypt 4 bytes (32 bits) you need to pad it to a block size of 24 bytes (192 bits) to apply the algorithm.

The method for breaking the data into blocks that are fed into the encryption is of utmost importance and affects the strength of your total encryption scheme. The algorithm that controls how to create and encrypt each block is called the chaining mode, or the block cipher mode of operations. There are four chaining modes that you can choose from when using Oracle encryption:

- Electronic Codebook (ECB) is the simplest mode and is shown in Figure 5.6. In ECB, the plaintext is divided into blocks and each block is encrypted independently. You should not use ECB. It does not hide patterns, it does not provide a secure scheme and is susceptible to substitution attacks. A great example of why you should never use ECB is shown in Figure 5.7. The original is shown on the left. The middle image shows the original encrypted using ECB chaining and the right image shows the original encrypted using CBC.



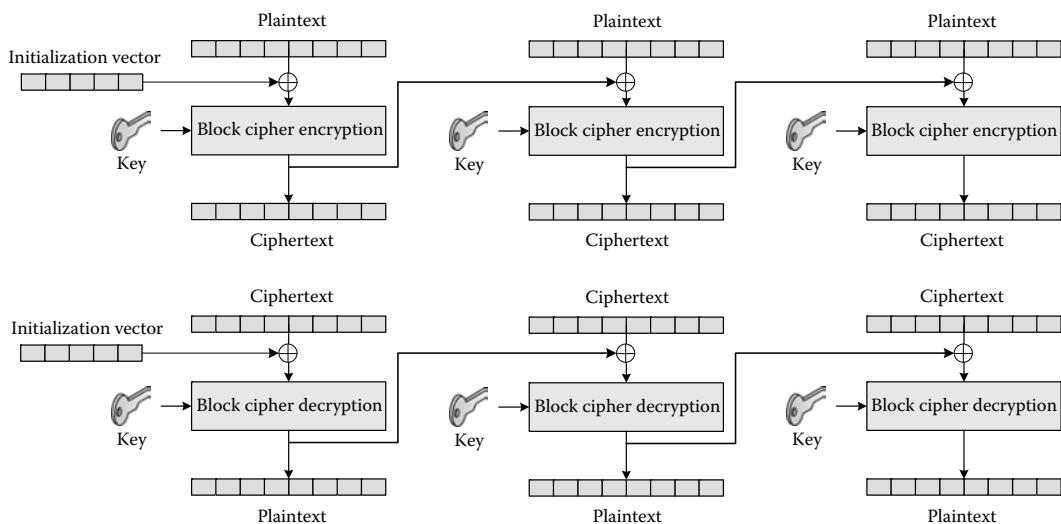
**Figure 5.6 ECB chaining.**



**Figure 5.7 Example of weak ECB encryption.**

- CBC is the mode that most schemes use and the one you should use as a default. It is explained by Figure 5.8.
- Cipher Feedback (CFB) is a variant on CBC where at each stage the previous ciphertext is encrypted and only then XOR-ed with the plaintext to form the next ciphertext.
- Output Feedback (OFB) is another variant where at each stage the ciphertext is a XOR of the plaintext with the output of the cipher algorithm but the input to the cipher algorithm is the output of the cipher algorithm of the previous stage and not the plaintext.

Both CFB and OFB turn a block cipher into a self-synchronizing stream cipher. In almost all cases you should stick with CBC and if you need something more advanced such as OFB or CFB do a little more reading on the subject.



**Figure 5.8 Cipher block chaining.**

## Key Distribution

Symmetric-key cryptography is great. It is relatively fast and it is proven. And yet, we have this other type of cryptography also—public-key cryptography. Why do we need another kind if symmetric-key cryptography is so great? One of the problems that public-key cryptography solves is key distribution. Symmetric-key cryptography is very useful but it requires a “bootstrapping” phase—the client and the server need to have access to a shared symmetric key that only they possess. The client and the server need to have a shared secret. How do you get to that state? How do you distribute and manage these keys? Do you generate keys and start SCP-ing them back and forth? The other problem is that of scalability. If you have  $N$  nodes and every two nodes need to be able to exchange data securely then you need  $N^2$  symmetric keys. This is not scalable when  $N$  becomes large; it would be better if we have a scheme by which we only needed  $N$  keys. For example, if you have 100 database servers and potentially 1,000 database clients you would have to prepare 100,000 keys, SCP them back and forth for the initial setup, manage them in case you have to replace one, etc. An even more extreme example of the problem is that of Internet E-commerce. How do millions of people shop on Amazon.com? All these interactions go over an encrypted channel and yet surely none of these people have ever been to Amazon’s headquarters to negotiate a shared secret.

Fortunately, the problem of key distribution has been solved, making symmetric-key encryption possible. Let’s look at the main method used for distributing symmetric keys called the Diffie–Hellman protocol. Then we’ll delve a little deeper into public-key cryptography.

## Diffie–Hellman Key Exchange

Diffie–Hellman (sometimes called Diffie–Hellman–Merkle) is a protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. The protocol relies on the existence of one-way hash functions  $H$  with the property that given a value  $v$  it is easy to compute  $H(v)$  but given the value  $H(v)$  it is computationally very hard (i.e., not feasible with current computational power) to compute  $v$  even when you fully know  $H$ .

Figure 5.9 describes the scheme conceptually and Figure 5.10 describes the math behind the protocol. Figure 5.9 explains the schemes with textures. The client and the server need to negotiate some shared secret without ever meeting each other and only through exchanges that are visible to the attacker. In the first step each of the parties generates a secret that only they know (and that even by the end of the scheme only they know—the scheme is not based on each party acquiring the other party’s secret). One of the parties picks a one-way hash function and communicates it to the other party—everyone can know what this hash function is. Each party then applies the hash function on their secret. In Figure 5.9 this is shown as each party “mixing” the textures. What is important to remember (that is not obvious when speaking about textures) is that knowing the result of the mixture and the texture of the hash does not mean you can compute the original texture. Each party then sends the result after mixing in the hash to the other party. This exchange is done in the clear and the attacker sees these two mixtures. Again—this is not enough for the attacker to compute the original secrets—that’s the essence of the protocol. Each party then mixes in their original secret. The order in which the functions are applied (the mixing of patterns in Figure 5.9) does not affect the end result. Therefore, each party ends up with the same exact value—the shared key that they can use now for the symmetric-key algorithm. The attacker has no way of eavesdropping on the rest of the encrypted session because at no point in time did the

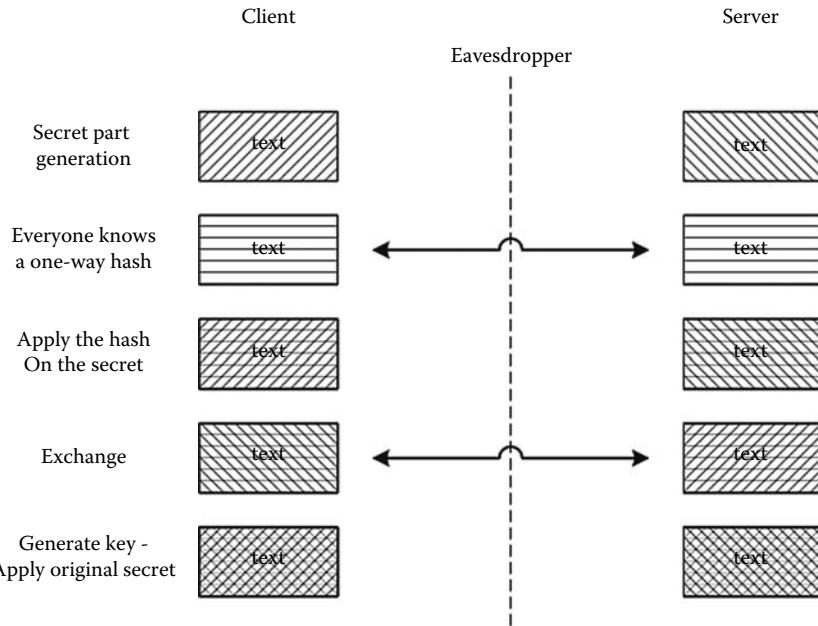


Figure 5.9 How Diffie–Hellman key exchange works.

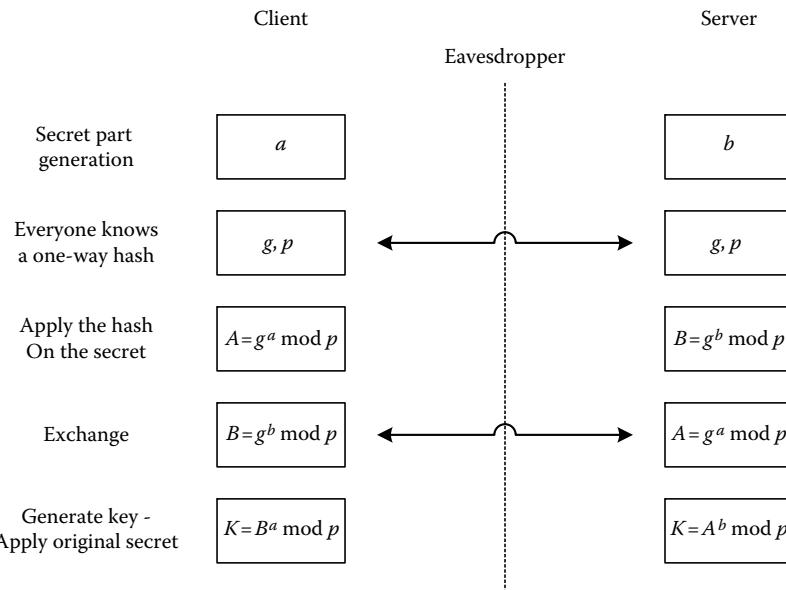


Figure 5.10 Math behind Diffie–Hellman key exchange.

attacker have one of the secret parts or anything that they can use to compute the secret parts or the shared key. The attacker has no way of computing the shared key used for the symmetric-key encryption.

Figure 5.10 shows the exchange as it happens in the real world. Without getting too deep into the math behind the protocol, the protocol is based on choosing a cyclic group  $G$  and a generating element  $g$  for the group. The group  $G$  is a group of order  $p$  where  $p$  is a large prime number. In reality subgroups are sometimes used with a Sophie Germain prime. The important properties of these selections are shown in Figure 5.10:

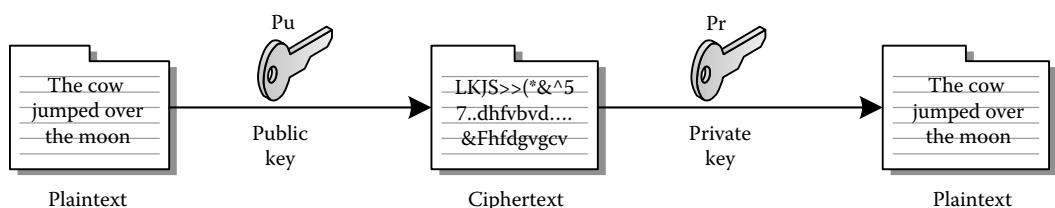
1. Within the group,  $g^{ab} = g^{ba}$ . In Figure 5.10 this means that  $(g^a \text{ mod } p)^b \text{ mod } p = (g^b \text{ mod } p)^a \text{ mod } p$ .
2. If you know  $g$ ,  $p$ , and  $A$ , you cannot effectively compute  $a$ . If you know  $g$ ,  $p$ , and  $B$ , you cannot effectively compute  $b$ .

You don't need to fully understand the math—the important thing is to know that you can negotiate a shared secret over an insecure channel. If you would like to get more details check out RFC 2631 at <http://tools.ietf.org/html/rfc2631>.

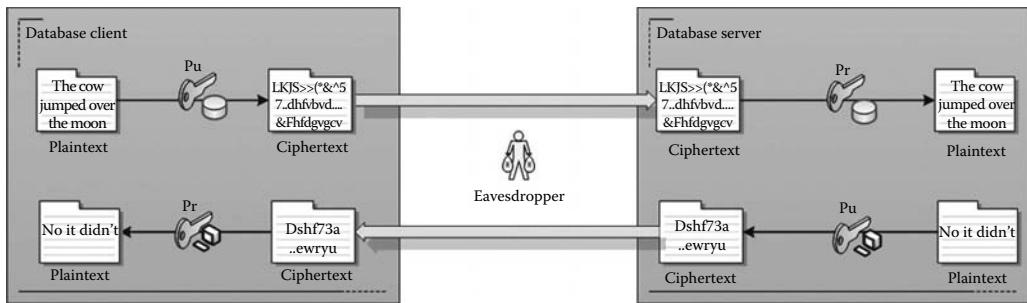
### Public-Key Cryptography

In symmetric-key encryption, the two keys used for the encryption and decryption are the same (or closely related). This is why the key distribution issue comes up—you need to make sure both parties have the same key. Public-key cryptography does away with this basic premise. In public-key cryptography, the encrypting key is completely different from the decrypting key and one cannot be computed from the other. When you use public-key cryptography you have two keys—a public key and a private key. When you want to send a message to another party you use the other party's public key to encrypt the message and the party that receives the message uses their private key to decrypt the message as shown in Figure 5.11. If you use the same key used for encryption to decrypt the message you will get gibberish. If you try to use any other key in the world apart from the specific private key, you will get gibberish. It is only the private key that matches the public key that will get you the plaintext back from the ciphertext. There is a very important mathematical property that public-key cryptography is based on that we won't prove here—the existence of pairs of keys that are complementary to each other, that can easily be generated by an algorithm (together), that there is no other key that can be the inverse to each other, and that you cannot compute one from the other.

Public-key cryptography doesn't have the key-distribution problems that symmetric-key cryptography has. If you want the client and the server to communicate over an insecure channel all you have to do is generate two public/private key pairs—one for the client and one for the server as shown in Figure 5.12. When the client generates the key pair it keeps the private key to itself and



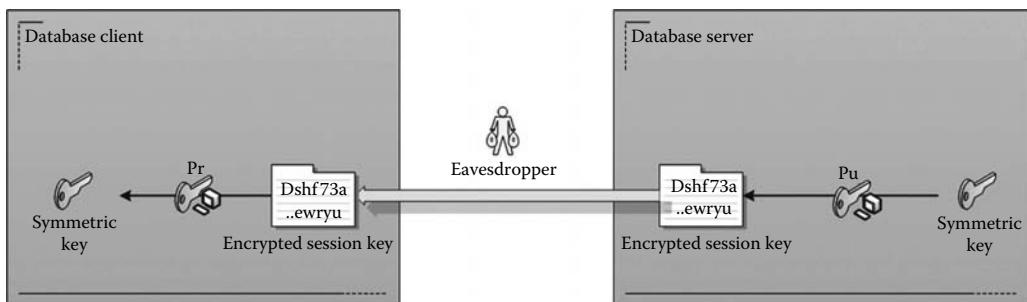
**Figure 5.11 Basic use of public-key encryption.**



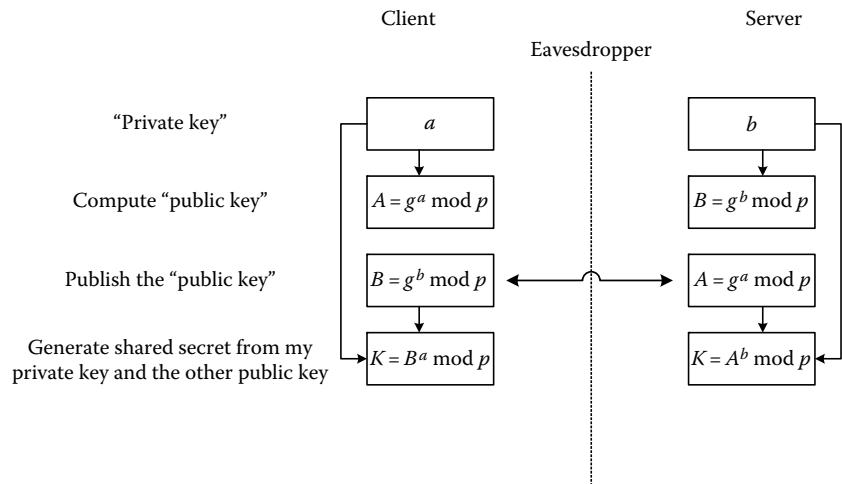
**Figure 5.12 Two parties communicating using public-key encryption.**

publishes the public key (hence the names public and private). Everyone in the world can know the public key—and in fact, that's how you communicate with the client. The server does the same thing—generates a key pair, keeps the private key secret, and publishes the public key. When the client wants to send an encrypted message to the server it encrypts the plaintext with the server's public key (which—like everyone else in the world—it knows). This ciphertext is then sent over the insecure channel. The attacker cannot read the message because it doesn't have the private key—only the server has the private key. The public key cannot be used to decrypt the message—only the private key can do that and the private key cannot be computed from the public key. The server gets the message and decrypts it using its private key. Then, when it wants to send a response to the client it takes the message and encrypts it using the client's public key (which it knows—as does everyone else). This ciphertext is sent over the insecure channel and the client decrypts it using its own private key. The client and the server manage to communicate without ever sharing a secret!

So why do we need symmetric-key cryptography? The reason is that although public-key cryptography does indeed solve the problem of key distribution, public-key encryption is around 1000 times slower than symmetric-key encryption. That's why we still use symmetric-key encryption always. What we normally do is use public-key cryptography to negotiate a shared symmetric key at the beginning of a session and then use symmetric-key encryption for the encryption/decryption that takes place throughout the session. In Figure 5.13 the client connects to the server



**Figure 5.13 Using public-key cryptography to set up symmetric keys.**



**Figure 5.14** Diffie–Hellman in the context of public/private keys.

requesting a connection. The server picks a random symmetric session key and encrypts it with the client’s public key. The client decrypts this message using its private key. Now the server and the client share a secret that no one in the world has and they can communicate over the insecure channel using symmetric-key algorithms that are relatively fast. In fact, when you look back to Figure 5.10 you’ll see that the Diffie–Hellman protocol does something similar. Think of “a” as the client’s private key and of “b” as the server’s private key. In this case, A is the client’s public key and B is the server’s public key. Everyone knows what A and B are—they are sent over the insecure channel. You can neither compute “a” from A nor “b” from B. The shared secret in Diffie–Hellman is directly computed by each party is shown in Figure 5.14.

### Certificates and Digital Signatures

Now that you understand how public keys are used to set up encrypted communications between parties that have never met, it’s time to take one more step and understand certificates. A certificate binds a public key with an identity using a digital signature. To understand why you need certificates let’s look back at the example of any Internet E-commerce site such as Amazon.

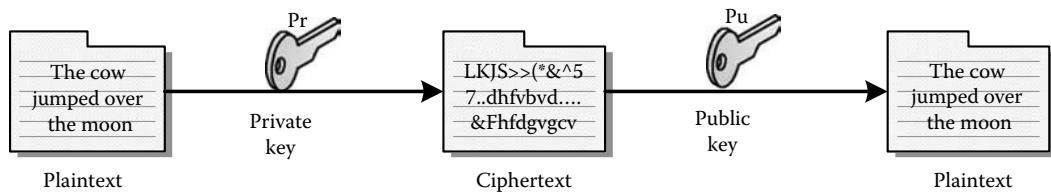
When you buy something from Amazon you give Amazon your credit card information. You know that you can do this without fear that someone can eavesdrop on your session because the communication is encrypted. You get Amazon’s public key and then you use it to communicate with Amazon. But where do you get the public key from? From Amazon’s Web site—right? Yes. But what you get is not the public key; you get a certificate which has the public key within it. The reason is that anyone can put up a Web site and advertise themselves as Amazon. An attacker may be able to hijack a session so that when you go to [www.amazon.com](http://www.amazon.com) you really will get to the attacker’s Web site. The attacker will build a Web site that looks exactly like Amazon’s and this Web site will have its own public key. When you ask for a public key to initiate the encrypted session you will get the attacker’s public key. You will then set up an encrypted session with the attacker and proceed to hand over your credit card number. True—no one else can eavesdrop on this communication, but you’ve already given you credit card number to the wrong person.

This is the problem that certificates solve—the problem of how you can trust the public key that you’re about to start using.

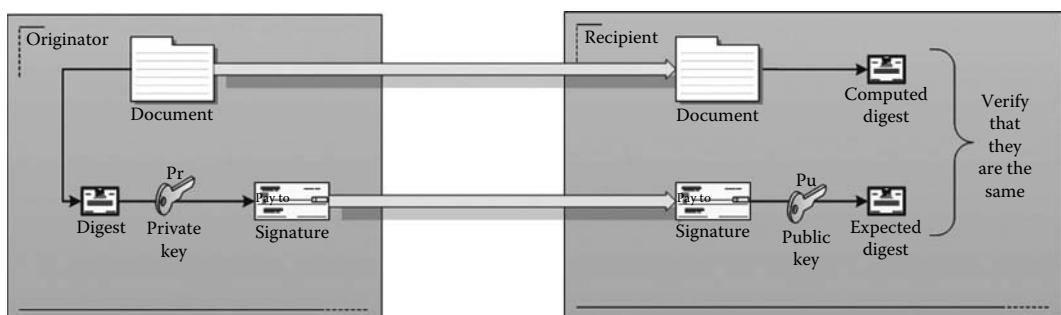
To understand certificates you first need to understand digital signatures. A digital signature uses public-key cryptography to mimic signatures we put on paper. A signature on paper is used to uniquely identify an individual and serves as proof that this was indeed signed by that particular individual. Digital signatures are similarly used to allow you to verify that a message came from a certain party and allows you to authenticate a message. The digital signature allows you to authenticate a message, but the message can be pretty much anything. As an example, it can be used to authenticate a public key as in the case of certificates.

Digital signatures use public-key cryptography in an “inverse mode” as compared with encryption. The same set of properties that make encryption possible, make digital signatures possible. The main attribute that is critical here is that the private key and the public key operate as inverses. You already saw that decrypting with the private key after encrypting with the public key gives you back the plaintext (Figure 5.11). What is important for digital certificates is that the inverse is also true (Figure 5.15)—if you encrypt plaintext with the private key and then decrypt the result with the public key you will get the original plaintext back. This attribute is not useful for encryption because everyone has access to the public key but it is very useful for proving identity because no one except one party has the private key.

Figure 5.16 shows how RSA digital signatures work. On the left you have the originator that wants to sign a document. The document will be sent to the recipient and we want to make sure that the recipient has a way to verify that the document that it received is the right one. The digital



**Figure 5.15** Using the private and public keys in reverse.



**Figure 5.16** RSA digital signatures.

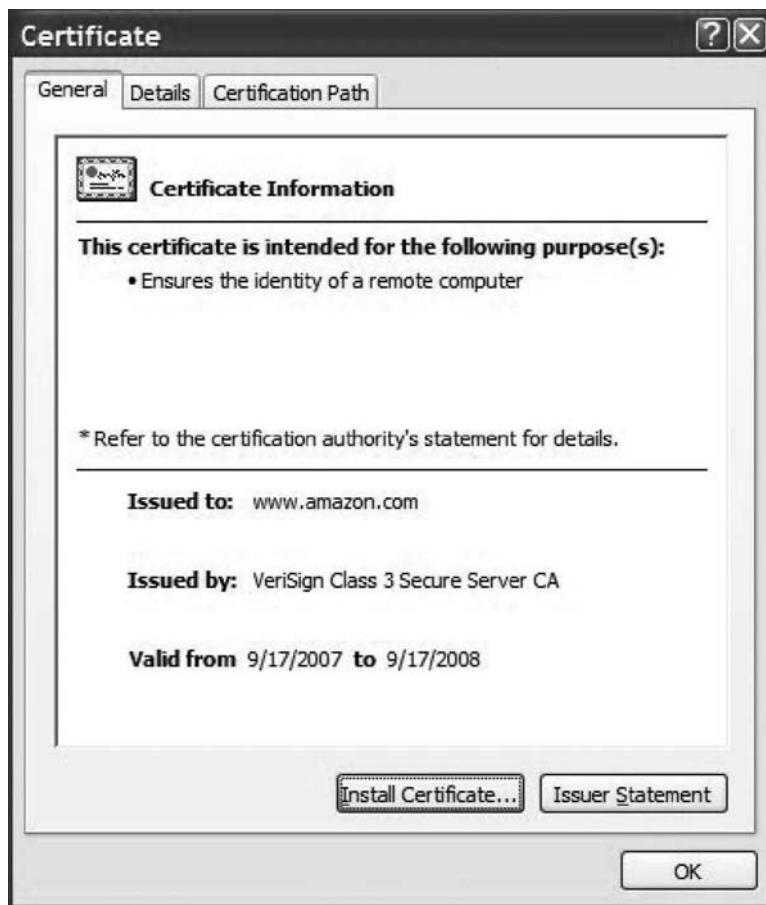
signature will do that. First, the originator will compute a “fingerprint” of the document. This is a relatively short number (a digest) that fully represents the document. Even if one bit has changed in the document, the digest will be different. You can use either the SHA-1 algorithm (better) or the MD5 algorithm to generate such digests. The originator then takes the digest and encrypts it with their private key (which only they have). This is the signature and it is sent along with the document to the recipient. The recipient takes the document and also computes the message digest. It then takes the signature and decrypts it using the originator’s public key (which everyone has). If the computed digest and the decrypted digests match then the recipient knows that the document is authentic.

Digital signatures are useful for many things, but let’s go back and see how they are used within certificates and what benefit this provides. Remember that the problem we’re trying to solve is that of interception—when you want to talk to some party and it offers you a public key with which to set up secure communications, how do you know the public key really belongs to that party? How can you trust the public key to set up the secure communications? The issue here is one of trust—how do you trust anything? Trust in real life is a relative thing. You trust something when someone trustworthy endorses it. When you apply for a job you’re asked for references. When you apply for a passport or passport renewal you might be asked for a picture endorsed by a doctor or judge. Generally—life is full of trust relationships and trust hierarchies where something is trusted to the same extent that the person endorsing or signing it is trusted.

This is how certificates work as well. A certificate is an electronic document that binds a public key with an identity through the use of a digital signature. The digital signature is not that of the entity to which the public key belongs to. The digital signature is that of a party that you agree to trust. What you’re saying is that if the party who signed this certificate (the issuer) trusts the identity and public key of the subject of the certificate, then you will also trust it and accept the public key to start secure communications. The digital signature is used so that you know for sure who really signed and issued the certificate.

Let’s look at an example. Use your browser to navigate to the Amazon Web site and add something to the shopping cart. Click on checkout (without being signed in so you don’t accidentally buy the item). At that point you’ll see that your browser has already initiated secure communication. Click on the secure icon and on View Certificate (in Internet Explorer (IE) for example). This will open up the certificate information tab as shown in Figure 5.17. This certificate was issued to [www.amazon.com](http://www.amazon.com)—this is the identity it is bound to. It can only be used for that address—if anyone tried to use this certificate (and public key) within another Web site they would get error messages when people tried to validate the certificate. You can also see that the certificate was issued by Verisign and that it is valid until September 17, 2008. If you click on the Details tab (Figure 5.18) you’ll see all the gory details of the certificate. A certificate includes a lot of information including a version, a serial number, an algorithm ID, an issuer, a validity period, a subject, the subject’s public key, the certificate’s digital signature, etc.

Trust hierarchies are the basis for all PKIs. The issuer of the certificate is called a Certificate Authority (CA). In the example shown before (and in Figure 5.19 which shows the certification path for [www.amazon.com](http://www.amazon.com)’s certificate), the CA is Verisign. A CA is a company that you pay to issue certificates. They check you out and validate that you are a legitimate business, have no history of fraud, etc. Then they issue the certificate with their signature—they endorse you. You’ll see later when using orapki and the OWM that you can issue certificates yourself! You don’t have to spend money on a certificate signed externally if your users (or your database clients) are all within the enterprise. You can easily create a self-signed certificate that can be respected within your realm. Don’t be surprised—certificates are all about trust hierarchies—anyone who trusts you (for example, within your organization) will be happy to use such a certificate.




---

**Figure 5.17 General certificate information.**

There's one more question to answer before we can move on to the next topic. In the Amazon example the public certificate is signed using Verisign's private key. But how can you know that indeed Verisign signed it? If the whole problem certificates solve is that of impersonation and interception (where an attacker gives you their public key to set up secure communications), what will happen if an attacker signs a certificate using their private key—how will you know not to trust that certificate? The answer is again one of a trust hierarchy. If you look at the browser that you use, you will see that it has some built in CAs which it trusts. For example, in IE click on Tools > Internet Options and navigate to the Content tab. Click on Certificates and you'll get a dialog shown in Figure 5.20. As you can see, your browser has many built in CAs which it trusts. Any certificate that this browser sees signed with a private key of a CA that has their public key embedded within the browser, will be trusted by the browser. If you access a site that has a public key which was not certified by a CA that the browser already has within its trust hierarchy then you'll get a message of the form shown in Figure 5.21—the browser is telling you that it can set up secure communications but it has no way to know with whom you're going to talk to and whether you can trust them. This happens when the certificate is not signed by someone that the browser can trust, when the certificate's subject is not the host on which it is installed, when the certificate has

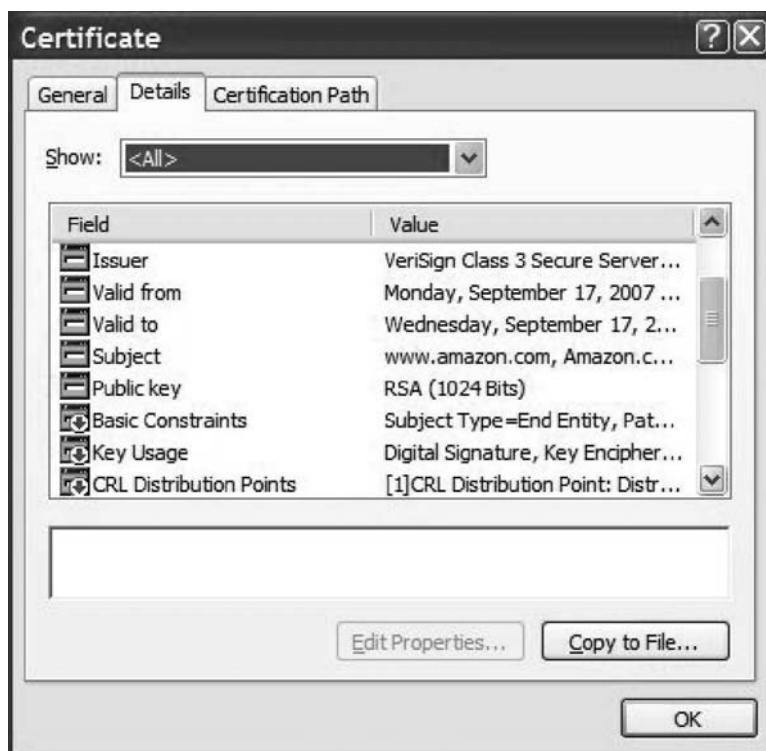


Figure 5.18 Detailed certificate information.

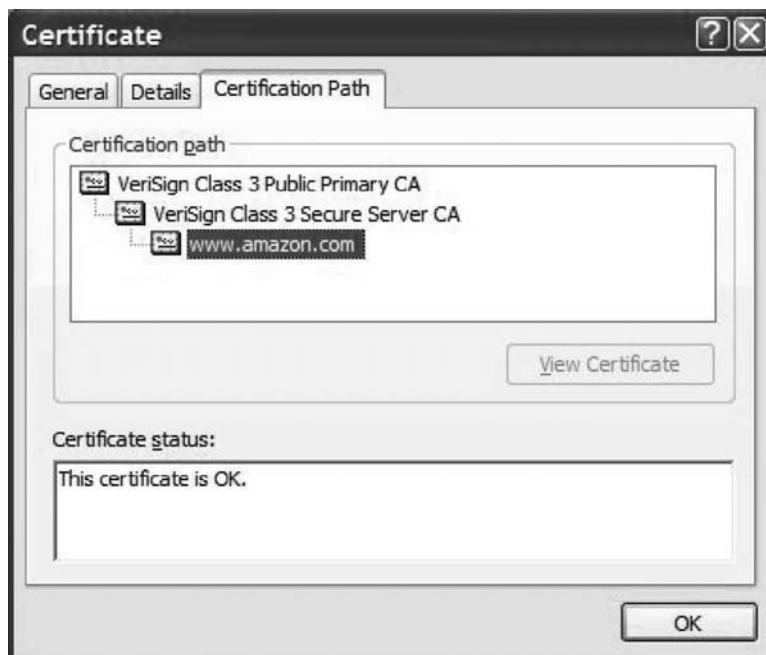
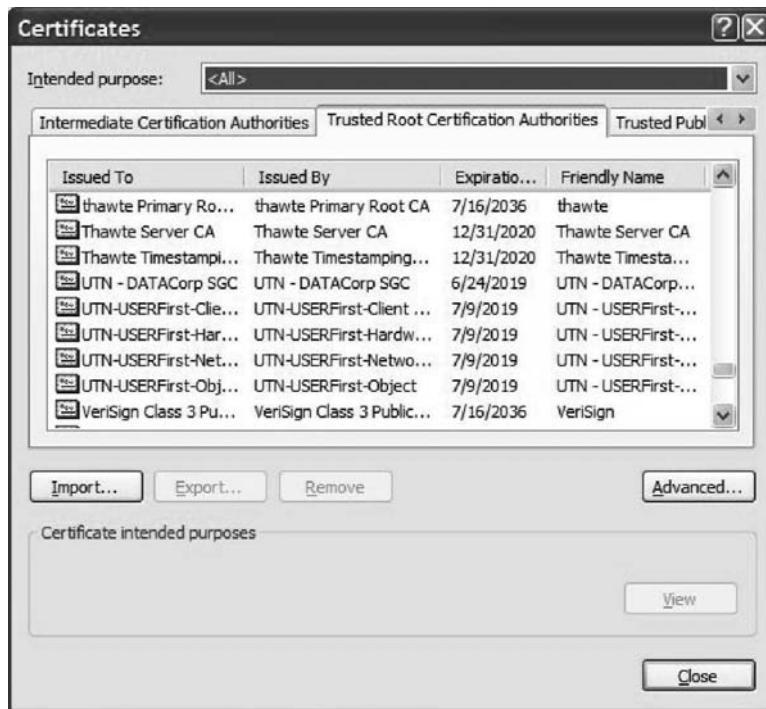


Figure 5.19 Certification path for www.amazon.com.



**Figure 5.20** Built in certificates in IE.



**Figure 5.21** Message showing a problem with a security certificate.

expired, or in any number of other such scenarios—all mean that the browser cannot guarantee the identity of the party for which you are about to accept the public key from.

### Some Important Standards

Here are a few more terms and standards that you should be familiar with and that will come up occasionally as we discuss Oracle encryption in various chapters:

- The Federal Information Processing Standard-140 (FIPS-140)—FIPS-140 are U.S. government computer security standards that specify requirements for cryptography modules. The current standard is called FIPS 140-2 and was published in 2001. FIPS-140 does not guarantee that modules that conform to the specification are secure or that a system built using such modules are secure but it does offer a level of validation for algorithms, methods, etc. FIPS-140 imposes requirements in 11 areas:
  - What of the cryptographic module must be documented?
  - What information flows in and out of the cryptographic module and how is that information segregated?
  - Who can do what with the module and how it is checked?
  - Documentation of the states the module can be in and what transitions can occur.
  - Tamper evidence and tamper resistance of the module (physical security).
  - What sort of operating system the module uses and what operating systems can use the module—i.e., what is the operational environment?
  - Key management.
  - Self-tests.
  - What documentation must be provided to show that the module has a good design and has been tested?
  - Mitigation of other attacks.
  - Susceptibility to electromagnetic interference.

FIPS-140 defines four levels of security—based on what is tested when validation occurs:

- Level 1, the lowest level, means that all components must be “production grade.”
- Level 2 adds requirements for physical tamper evidence and role-based authentication.
- Level 3 adds requirements for physical tamper resistance and identity-based authentication. It also requires a separation between the interfaces that control the module and other interfaces (i.e., the way in which you modify parameters or view them is separate from the way in which, for example, you feed in data that is to be encrypted/decrypted).
- Level 4 adds multiple requirements for physical security and resilience in the face of environmental attacks.

You can view a list of all vendors who have achieved some level of FIPS-140 certification at <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401vend.htm>. You should note that certification is a technical process that validates a particular configuration. This means that only the specific configuration that was validated is certified. When you look at <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2000.htm#90> you may be surprised to see that it mentions validation for Solaris 2.6 and some of the algorithms. This does not mean that the

modules in Oracle 9, 10, and 11 are insecure—but not all of them have been formally certified and it may be that Oracle considers it too expensive or impractical to certify all versions on all operating systems with all settings. What Oracle goes on to say (in the context of the SSL libraries within Oracle Advanced Security for example) is that the libraries are designed to meet FIPS-140 Level 2 certification, even if they have not been formally certified. You even have a file called fips.ora where you can set a parameter SSLFIPS\_140 = TRUE to configure the adapter to run in FIPS-140 mode.

FIPS-140 certification is a perfect example of where adherence to formal certification requirements does not blindly imply good security. To satisfy the FIPS 140-1 criteria (the older spec) only DES or DES40 can be used. These are algorithms that you should never use and they don't conform to FIPS 140-2. Setting SQLNET.FIPS\_140 = TRUE means that you must set the encryption type to something that you know is insecure and does not conform to 140-2. Therefore, you either use Oracle encryption without it being formally certified or you have to resort to an external tool that gives you the same algorithms but that is certified with FIPS 140-2. Unfortunately this usually means a more complex implementation, dragging with it the possibility of weaker security.

### **Additional Terms and Standards:**

- **Hardware Security Module (HSM)**—HSM is a hardware device that manages, secures, generates, and uses cryptographic keys. Keys are the most important part of any cryptographic operation. If it is possible to access your keys then you should not be encrypting anything—there is no point. You'll see shortly that keys usually sit within an Oracle wallet. The wallet is a file, so much of the security of the keys depends on the host operating system. If you need higher levels of protection you can use an HSM integrated with Oracle. We will talk more about HSMs in various chapters.
- **PKCS** refers to a group of Public-Key Cryptography Standards created and published by RSA Security.
- **PKCS-11** defines a platform-independent application programming interface (API) to HSMs. Many vendors create HSM devices. Some are packaged as cryptographic appliances, some as Peripheral Component Interconnect (PCI) cards, some as USB tokens, etc. Software such as Oracle may need to interface with many HSMs produced by various vendors. The PKCS-11 standard was created to allow this. Oracle uses the API calls defined in PKCS-11 so any HSM that conforms to this standard is usable from within Oracle.
- **PKCS-12** defines a standard file format for storing keys, certificates, etc. The Oracle wallet conforms to this standard. This standard is the reason that when you create a wallet your file will always have a .p12 extension—e.g., wallet.p12.
- **SSL and Transport Layer Security (TLS)**—SSL and TLS(which replaces SSL 3) secure communications over an insecure channel. You use TLS (SSL) daily when you send sensitive information on an Internet connection or use an application which has a browser interface and wants to secure the communication between the browser and the server. The Oracle database can also be configured to use SSL for securing the communications between a database client and a database server. But SSL is not merely a way to encrypt the communications. SSL also supports data integrity checks to ensure no one tampers with the data and can support endpoint authentication using certificates. For key exchange SSL supports RSA and Digital Signature Algorithm (DSA) public-key cryptography, Diffie–Hellman and a few other methods. The encryption itself uses symmetric-key encryption using 3DES, AES, RC4, or Camellia. For hashing you can use SHA-1 or MD5.

### Oracle Wallet, Oracle Wallet Manager, and Orapki

The Oracle wallet is a protected container for keys, certificates, and other secrets that you need to secure over time. There are two classes of keys or secrets—there are short-term keys such as session keys and long-term keys and secrets such as private/public keys and passwords. Short-term keys are usually negotiated per session and do not need to be stored anywhere. Public and private keys that are used for authentication or for setting up a random session key need to be stored—but they need to be stored securely. The Oracle wallet is a secure store for such secrets that is used by the Oracle database and every other Oracle product; it is a part of Oracle's implementation of a PKI.

The OWM is an application through which you manage public key security credentials for database clients and servers. You use OWM to create wallets, generate certificate requests, import and export certificates, and more. Similarly, the orapki utility allows you to perform pretty much the same activities on a command line interface rather than a graphical user interface (GUI) application. When you use OWM you usually manage an Oracle wallet—i.e., the operations you perform usually persist in a wallet. However, OWM works using the PCKS-11 API and therefore can be used to manage secrets within another store such as an HSM.

## 5.1 HOWTO Create Wallets

The first step is to create a wallet. You can do this using orapki or OWM. If you use orapki you will run an orapki wallet command. All the orapki commands name a module as the first parameter—in this case wallet is the module name. The command is create. The full command to create a wallet is orapki wallet create -wallet <path to wallet> :

```
[oracle11@o11g db_1]$ orapki wallet create -wallet $ORACLE_HOME/wallets
Enter password:
Invalid Password....
PASSWORD_POLICY: Passwords must have a minimum length of eight characters and
contain alphabetic characters combined with numbers or special characters.
Enter password:
Enter password again:
```

As you can see, you need to enter a strong password for the wallet. At this point the wallet will be created for you:

```
[oracle11@o11g db_1]$ ls -l $ORACLE_HOME/wallets
total 12
-rw----- 1 oracle11 dba 7312 Nov 17 01:37 ewallet.p12
```

To display the contents of a wallet:

```
[oracle11@o11g db_1]$ orapki wallet display -wallet $ORACLE_HOME/wallets
Enter wallet password:
Requested Certificates:
User Certificates:
Trusted Certificates:
```

```

Subject: O      U=Class 1 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
Subject: O      U=Class 3 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
Subject: C      N=GTE CyberTrust Global Root,OU=GTE CyberTrust Solutions\, Inc.,O=GTE
Corporation,C=US
Subject: C      N=Entrust.net Secure Server Certification Authority,OU=(c) 2000 Entrust.
net Limited,OU=www.entrust.net/SSL_CPS incorp. by ref. (limits liab.),O=Entrust.net
Subject: C      N=Entrust.net Certification Authority (2048),OU=(c) 1999 Entrust.net
Limited,OU=www.entrust.net/CPS_2048 incorp. by ref. (limits liab.),O=Entrust.net
Subject: O      U=Secure Server Certification Authority,O=RSA Data Security\, Inc.,C=US
Subject: O      U=Class 2 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
Subject: C      N=Entrust.net Secure Server Certification Authority, OU=(c) 1999
Entrust.net Limited, OU=www.entrust.net/CPS incorp. by ref. (limits
liab.),O=Entrust.net,C=US

```

Notice that you have many trusted certificates within any wallet, even one that was just created. When you buy a certificate from a commercial CA (such as Verisign or Entrust) your certificate will be signed by the CA's private key and will be validated through the use of these trusted certificates.

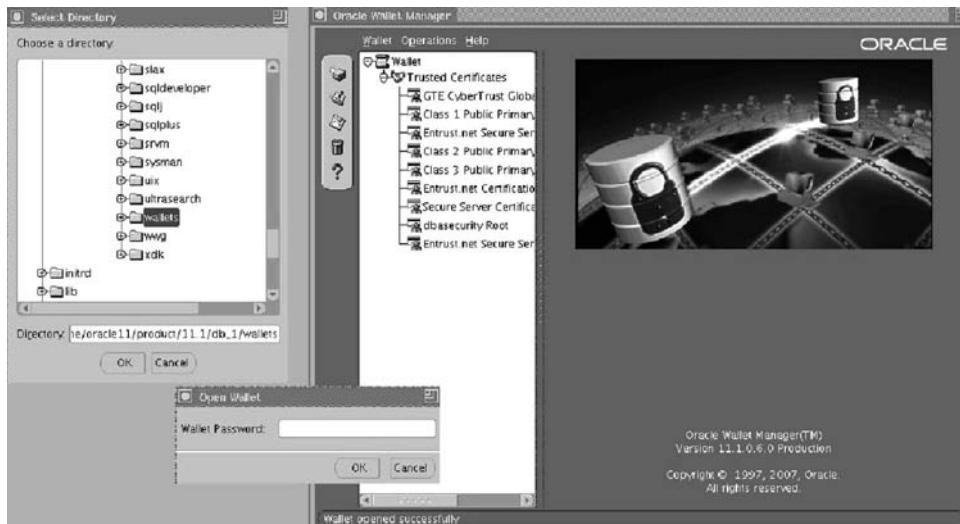
To add a new trusted certificate to the wallet, get your certificate from the external CA and use:

```

orapki wallet add
-      wallet $ORACLE_HOME/wallets
-      trusted_cert
-cert <path to cert>

```

You can perform all of these operations using OWM as well. For example, to open your wallet start OWM, and click Wallet- > Open. Enter your password and you're in the wallet (Figure 5.22).



**Figure 5.22** Opening the wallet using OWM.

### Two Things to Remember About Creating Wallets

1. You can use orapki or the OWM to create wallets—they are equivalent.
2. If you plan on getting an external entity to generate your certificates then the simplest thing is to choose a vendor that is already listed as a trusted certificate in the default Oracle wallet.

## 5.2 HOWTO Add Certificates

If you are only going to use certificates within your realm of responsibility there is no point in paying someone for certificates—use orapki or OWM to create your own certificates. You will be acting as the CA and you will be generating the trust hierarchy yourself.

To create a new self-signed root certificate use the wallet add command:

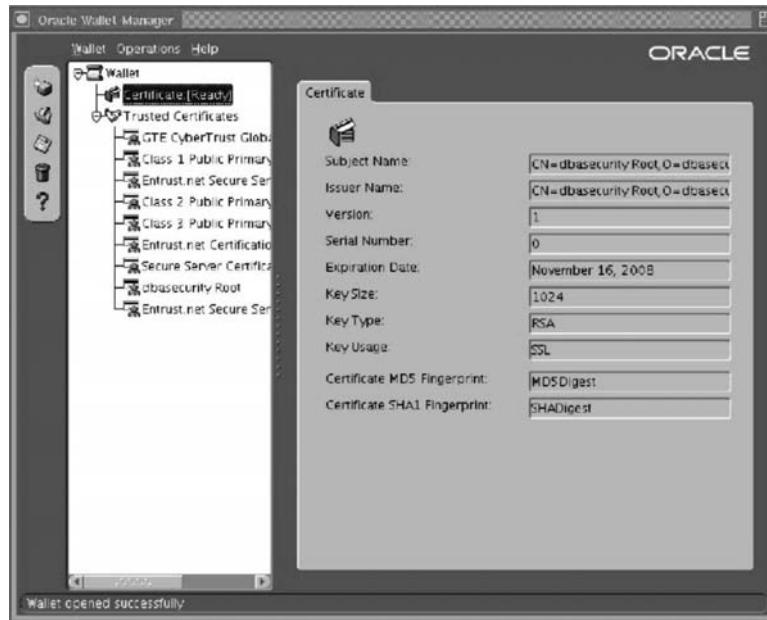
```
[oracle11@o11g db_1]$ orapki wallet add
      -           wallet $ORACLE_HOME/wallets
      -           -dn "CN=dbasecurity Root,O=dbasecurity,C=US"
      -           self_signed
      -           validity 365
      -k           eysize 1024
```

When you add a certificate you must add all the trusted certificates that make up the certificate chain. For example, the syntax for adding a user certificate is  
 orapki wallet add –wallet \$ORACLE\_HOME/wallets –user\_cert –cert < path to cert >

When you add such a certificate to your wallet, the wallet will check whether the entire trust chain is there. But when you use self\_signed your certificate is created immediately because you are acting as the CA. The certificate you created before therefore appears immediately in the wallet:

```
[oracle11@o11g db_1]$ orapki wallet display -wallet $ORACLE_HOME/wallets
Enter wallet password:

Requested Certificates:
User Certificates:
Subject: CN=dbasecurity Root,O=dbasecurity,C=US
Trusted Certificates:
Subject: C   N=GTE CyberTrust Global Root,OU=GTE CyberTrust Solutions\, Inc.,O=GTE Corporation,C=US
Subject: O   U=Class 1 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
Subject: C   N=Entrust.net Secure Server Certification Authority,OU=(c) 2000 Entrust.net Limited,OU=www.entrust.net/SSL _ CPS incorp. by ref. (limits liab.),O=Entrust.net
Subject: O   U=Class 2 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
Subject: O   U=Class 3 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
Subject: C   N=Entrust.net Cer tification Authority (2048),OU=(c) 1999 E ntrust.net Limited,OU=www.entrust.net/CPS_2048 incorp. by ref. (limits liab.),O=Entrust.net
Subject: O   U=Secure Server Certification Authority,O=RSA Data Security\, Inc.,C=US
Subject: C   N=dbasecurity Root,O=dbasecurity,C=US
Subject: C   N=Entrust.net Secure Server Certification Authority,OU=(c) 1999 Entrust.net Limited,OU=www.entrust.net/CPS incorp. by ref. (limits liab.),O=Entrust.net,C=US
```



**Figure 5.23 Self-signed certificate in OWM.**

When you open your wallet in OWM you will see your new certificate as shown in Figure 5.23.

**Two Things to Remember About Adding Certificates:**

1. Use orapki and the wallet to generate your own certificates which you can use internally or for testing purposes.
2. In this case generating self-signed certificates is the simplest process.

### 5.3 HOWTO Create and Sign a Certificate Request

Now let's go through a process of creating a request and signing it. This is similar to a process you may do with a real CA, but for internal use or testing purposes, signing it yourself may be enough. In this sequence you create a certificate request and then sign it yourself to create a certificate. In the next HOWTO you'll export this certificate and import it into another wallet.

To create the certificate request:

```
[oracle11@o11g ~]$ orapki wallet add -wallet $ORACLE_HOME/wallets -dn
"CN=ronb,O=dbasecurity,C=US" -keysize 1024
Enter wallet password:
```

If you look at the contents of the wallet now you'll see the new certificate request:

```
[oracle11@o11g ~]$ orapki wallet display -wallet $ORACLE_HOME/wallets
Enter wallet password:
```

**Requested Certificates:**

```
Subject: CN=ronb,O=dbasecurity,C=US
```

User Certificates:

```
Subject: C N=dbasecurity Root,O=dbasecurity,C=US
```

Trusted Certificates:

```
Subject: OU=Class 2 Public Primary Certification Authority,O=VeriSign\,
Inc.,C=US
```

...

The signing process can occur on another machine or on the same machine. Export the certificate request so that you can sign it anywhere:

```
[oracle11@o11g ~]$ orapki wallet export -wallet $ORACLE_HOME/wallets -dn
"CN=ronb,O=dbasecurity,C=US" -request./ronb.req
Enter wallet password:
[oracle11@o11g ~]$ ls -l./ronb.req
-rw----- 1 oracle11 dba 584 Nov 17 02:04./ronb.req
```

And now create a signed certificate from the request:

```
[oracle11@o11g ~]$ orapki cert create -wallet $ORACLE_HOME/wallets -
request./ronb.req -cert./ronb.cert -validity 365
Enter wallet password:g
[oracle11@o11g ~]$ ls -l./ronb.cert
-rw----- 1 oracle11 dba 706 Nov 17 02:05./ronb.cert
```

This time the module used in the orapki command is cert because you are acting on a certificate. At this point the certificate file is created but it is not in any wallet. For example, if you display the wallet you just used in the create process you will not see the certificate there:

```
[oracle11@o11g ~]$ orapki wallet display -wallet $ORACLE_HOME/wallets
Enter wallet password:g a
Requested Certificates:
Subject: C N=ronb,O=dbasecurity,C=US
User Certificates:
Subject: C N=dbasecurity Root,O=dbasecurity,C=US
Trusted Certificates:
Subject: OU=Class 2 Public Primary Certification Authority,O=VeriSign\,
Inc.,C=US
...
```

You need to add the certificate into a wallet. Both the certificate request and the certificates are text files which you can copy over to any other server:

```
[oracle11@o11g ~]$ cat ronb.cert
--BEGIN CERTIFICATE--
MIIB3zCCAUGCAQAwDQYJKoZIhvvcNAQEEBQAwPjELMAkGA1UEBhMCVVMxFDASBgNVBAoTC2RiYXNl
Y3VyaXR5MRkwFwYDVQQDExBkYmFzZWN1cm10eSBsb290MB4XDTA3MTEXnZA3MDU1MFoXDTA4MTE
NjA3MDU1MFowMjELMAkGA1UEBhMCVVMxFDASBgNVBAoTC2RiYXNlY3VyaXR5MQ0wCwYDVQQDEwRy
b25iMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCJOLXwjf412tbt2Hkfod/9EzdMatmc
dSbVtB1I2zmVaCwcwJgulhnrkJ1xM8NfH3u/Balyf6rxhNUZm6dIxnbyy4zH1JQIXcx5wvissccM1V
kxmlTUEmu3IE/HJK9tJnie9v/UjB3sX7QvxmVeC5iZXavXkgQtUzPtu+TkWqpqmwlDAQABMA0GCSqG
```

```

S1b3DQEBAUAA4GBAH0eCHsb7i/vMjfx979R9ZPXPWKod9eo3gkjhSf5smB8bygFFU + JFKOBAM8e
5N6gwV8T4JsluJFnQ3Tv+p14tv40m0W3/hWFcL9yvDVUI0VJ8Sm59HuzW+SD14MM87KZBuSx/zG0
vCV5DFB0FU1xLrkLpvZ3pC1xMI1vaKLXyABL
-----END CERTIFICATE-----
[oracle11@o11g ~]$ cat ronb.req
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBcTCB2wIBADAYMQswCQYDVQQGEwJVUzEUMBIGA1UEChMLZGJhc2VjdXJpdHkxDTALBgNVBAMT
BHJvbmbIwgZ8wDQYJKoZIhvCNQEBBQADgY0AMIGJAoGBAIk4tfCN/jXa21vYeR+h3/0TN0xq2Zx1
JtW0HUjbOZVoLBzAmC6WGeuQnXEzw18fe78FqXJ/qvGE1Rmbp0jGdvLLjMfUlAhdzHnC+yxwIzV
WTGaVNQSa7cgT8ckr20meJ72/9SMHexftC/GZV4LmJldq9eSBC1TM+2750RaqmqbAgMBAAGgADAN
BgkqhkiG9w0BAQFAAOBgQB98oOEHihrEwrfiFUDkXEc+Nxwk9568j5XE0c7rVkOKBO7vFELjxbe
3jwFvlSmss6/IT67jDZFZBzGfsf94AVYp9+L63M4z98asnGAxNjeSDkmdqdsUce6OM2pax0vA80j
jXuRAF1Sd1STnSPfx91EnQV5iI8AJApAQOjxxNW6A==
-----END NEW CERTIFICATE REQUEST-----

```

You can add the certificate either as a user certificate or as a trusted certificate. For example, to add the certificate as a user certificate:

```
[oracle11@o11g ~]$ orapki wallet add -wallet $ORACLE_HOME/wallets -user_cert -
cert./ronb.cert
Enter wallet password:
```

The certificate will show up in the user section:

```
[oracle11@o11g ~]$ orapki wallet display -wallet $ORACLE_HOME/wallets
Enter wallet password:
Requested Certificates:
User Certificates:
Subject: CN=ronb,O=dbasecurity,C=US
Subject: C      N=dbasecurity Root,O=dbasecurity,C=US
Trusted Certificates:
Subject:      CN=GTE CyberTrust Global Root,OU=GTE CyberTrust Solutions\, Inc.,O=GTE
Corporation,C=US
```

Notice that the subject is no longer listed as a certificate request because it has now been signed and added as a certificate. If you want this certificate to be part of a trust hierarchy you can add it as a trusted certificate into the wallet:

```
[oracle11@o11g ~]$ orapki wallet add -wallet $ORACLE_HOME/wallets -trusted_cert -
cert./ronb.cert
Enter wallet password:
```

The certificate will now show up in the trusted certificate section:

```
[oracle11@o11g ~]$ orapki wallet display -wallet $ORACLE_HOME/wallets
Enter wallet password:g rd
Requested Certificates:
User Certificates:
Subject: C      N=dbasecurity Root,O=dbasecurity,C=US
Subject: C      N=ronb,O=dbasecurity,C=US
Trusted Certificates:
...
```

```

Subject: OU=Class 1 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
Subject: OU=Class 3 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
Subject: CN=dbasecurity Root,O=dbasecurity,C=US
Subject: OU=Class 2 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US

```

## 5.4 Discussion: Orapki Errors

Orapki is a very simple utility that is easy to use. There are many options in the utility but they all follow a concise and well-defined pattern. It will be much easier to use if you really understand what certificates are, what requests are, what the trust hierarchy is, etc. Hopefully, this chapter has given you a good enough introduction to make all this a little less confusing.

The main reason it is so important to understand the process and terms is that although simple to use when you understand what you're doing, orapki error messages can be frustrating if you're working in trial-and-error mode. You should avoid trying to use orapki this way. This is especially true in 10g as some of these problems were fixed in 11g. As an example, if you put in a wrong password you will get a good error message in 11g but in 10g the error message may lead you to start searching for the default location of the wallet. In 11g:

```
[oracle11@o11g ~]$ orapki cert create -wallet $ORACLE_HOME/wallets -
request./ronb.req -cert./ronb.cert -validity 365
Enter wallet password: <wrong>
Could not open wallet. Check password
```

In 10g:

```
saturn:oracle10% orapki cert create -wallet $ORACLE_HOME/wallet/ -request
ronb.req -cert ronb.cert -validity 365 -summary
Enter wallet password: < wrong >
Unable to load wallet at /home/oracle10/product/10.2.0/db_1/wallet/
```

In some cases even the 11g error messages are not helpful. For example if you use the wrong syntax the error message will keep you guessing. In both 10g and 11g:

```
[oracle11@o11g ~]$ orapki wallet add -wallet $ORACLE_HOME/wallets -dn ronb -
keysize 1024
Enter wallet password:
Unknown error occurred:
```

# *Chapter 6*

---

# Authentication

---

Authentication comes from the Greek word *αὐθεντικός*, which means real or genuine and “from the author” (authentes). Authentication is the process in which Oracle attempts to identify your identity when you logon to the database and decides whether to allow it. Authentication is the first step in any access to the database—it is the front gate to your castle. If you have no way to identify a party making the connection or there is a way to attack the authentication scheme and fake an identity, then nothing else matters.

In this chapter you’ll review the various methods Oracle employs to authenticate users. Starting with the basic/standard authentication method using usernames and passwords, through password stores and password files, and all the way to advanced authentication options, you’ll learn that you have many options for authenticating your users. You’ll learn about methods in which the authenticating entity is the database as well as schemes in which the database delegates authentication to an external server. All of these options allow you to choose the most appropriate authentication scheme for your database environments or for different users.

## **6.1 HOWTO Understand and Use O3/O5 LOGON and OS Authentication**

There are two main forms of basic authentication in Oracle—one in which authentication is done by the database and one in which Oracle trusts the operating system (OS) to perform the authentication of a user. In Chapter 4 you saw the basic working of the O3LOGON authentication scheme; instead of repeating it here you’ll see what the equivalent method for Oracle 11g looks like. The authentication scheme in Oracle 11g is called O5LOGON.

### **O5LOGON**

The changes in O5LOGON include the use of secure hash algorithm-1 (SHA-1), the use of salt and a longer session key. O5LOGON looks like this (these are segments from the network packets sent between the client and the server in a O5LOGON authentication process):

- Logon to the database using one of the service names defined within your tnsnames.ora:

```
[o    racle11@o11g ~]$ sqlplus scott/tiger@o11g
SQL*Plus: Release 11.1.0.6.0 - Production on Fri Nov 16 16:25:55 2007
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Co    nnected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining,
Oracle Database Vault, and Real Application Testing options
SQ    L>
```

- The client communicates with the server and sends it the username (scott in this case):

0	x0000:	4500	0128	95cf	4000	4006	6684	c0a8	de15	E..(..@.f.....
0	x0010:	c0a8	de15	802a	05f1	8082	8a04	806c	8dad	....*.....l.....
0	x0020:	8018	0100	3e97	0000	0101	080a	003d	1399	.... > ..... =.....
0	x0030:	003d	1394	00f4	0000	0600	0000	0000	0376	.= .....v.....
0	x0040:	02fe	fffff	ff05	0000	0001	0000	00fe	fffff	.....
0	x0050:	ff05	0000	00fe	ffff	fffe	ffff	ff05	73 63	.....sc.....
0	x0060:	6f74	740d	0000	000d	4155	5448	5f54	4552	ott.....AUTH_TER
0	x0070:	4d49	4e41	4c05	0000	0005	7074	732f	3100	MINAL.....pts/1.
0	x0080:	0000	000f	0000	000f	4155	5448	5f50	524f	.....AUTH_PRO
0	x0090:	4752	414d	5f4e	4d25	0000	0025	7371	6c70	GRAM_NM%.....%sqlp
0	x00a0:	6c75	7340	6f31	3167	2e67	7561	7264	6975	lus@o11g.dbasecu
0	x00b0:	6d2e	636f	6d20	2854	4e53	2056	312d	5633	r.com.(TNS.V1-V3...
0	x00c0:	2900	0000	000c	0000	000c	4155	5448	5f4d	).....AUTH_M
0	x00d0:	4143	4849	4e45	1100	0000	116f	3131	672e	ACHINE.....o11g.
0	x00e0:	6775	6172	6469	756d	2e63	6f6d	0000	0000	dbasecur.com.....
0	x00f0:	0800	0000	0841	5554	485f	5049	4404	0000	.....AUTH_PID..
0	x0100:	0004	3430	3432	0000	0000	0800	0000	0841	..4042.....A
0	x0110:	5554	485f	5349	4408	0000	0008	6f72	6163	UTH_SID.....orac
0	x0120:	6c65	3131	0000	0000					le11.....

- The server responds with the AUTH\_SESSKEY and the AUTH\_VFR\_DATA. The AUTH\_SESSKEY has been extended to 48 bytes in O5LOGON. The AUTH\_VFR\_DATA is the password hash salt value of length 10 bytes:

0	x0000:	4500	0192	9ea7	4000	4006	5d42	c0a8	de15	E.....@.].B.....
0	x0010:	c0a8	de15	05f1	802a	806c	8dad	8082	8af8	.....*..l.....
0	x0020:	8018	0100	3f01	0000	0101	080a	003d	13a6	...?.....=.....
0	x0030:	003d	1399	015e	0000	0600	0000	0000	0803	.= .....^ .....
0	x0040:	000 c	0000	000c	4155	5448	5f53	4553	534b	.....AUTH_SESSK
0	x0050:	4559	6000	0000	6039	3443	3439	3735	3938	EY'.....'94C497598
0	x0060:	3445	4537	3441	3345	3739	4339	4538	3546	..4EE74A3E79C9E85F
0	x0070:	4446	3038	3542	3430	4236	3444	3234	4333	..DF085B40B64D24C3
0	x0080:	4138	4234	4239	4533	3741	4138	4546	3434	..A8B49E37AA8EF44
0	x0090:	3832	4544	4139	3636	4234	3446	3136	3843	..82EDA966B44F168C
0	x00a0:	4338	4534	4534	3133	4242	4138	4337	3438	..C8E4E413BBA8C748
0	x00b0:	3932	3130	3233	3600	0000	000d	0000	000d	92 10236.....
0	x00c0:	4 155	5448	5f56	4652	5f44	4154	4114	0000	AUTH_VFR_DATA.....
0	x00d0:	0014	3242	3233	3445	3434	3437	3832	3541	.....2B234E4447825A
0	x00e0:	3537	3046	3144	251b	0000	1a00	0000	1a41	570F1D%.....A
0	x00f0:	5554	485f	474c	4f42	414c	4c59	5f55	4e49	UTH_GLOBALLY_UNI
0	x0100:	5155	455f	4442	4944	0020	0000	0020	3434	QUE_DBID.....44
0	x0110:	3 336	3339	3935	3432	4142	3630	3539	4645	..36399542AB6059FE
0	x0120:	3134	3432	3832	3743	3834	3144	4433	0000	1442827C841DD3.....

0x0130:	0000	0	401	0000	0002	0001	0000	0000	0000	.....
0x0140:	0000	0000	0000	0000	0000	0000	0000	0000	0000	.....
0x0150:	0000	0000	0000	0000	0000	0000	0000	0000	0000	.....
0x0160:	0000	000	2	0000	0000	0000	3601	0000	0000	.....6.....
0x0170:	0000	005	5	550f	0000	0000	0000	0000	0000	.....UU.....
0x0180:	0000	0000	0000	0000	0000	0000	0000	0000	0000	.....
0x0190:	0000									..

0

The 10 byte salt value is sent to the client because in Oracle 11g the password hashing scheme is based on SHA-1 with a salt component added (some random bits used to strengthen the password in the face of brute-force attack). The client will need to use the same salt value that the server uses in order for the match to occur. In Oracle 11g the password hash is created by concatenating the salt component to the password. SHA-1 is then applied to the salted password. The first 20 bytes are converted to printable hex—and this is the password hash that is stored in SPARE4 of SYS.USER\$. The salt is also stored in SPARE4:

```
SQL> select name,spare4 from user$ where name='SCOTT';
NAME
-----
SPARE4
-----
-----
SCOTT
S:D520818940AE5CA8F426B1C3EC408AB446C203AA2B234E4447825A570F1D
```

As you can see, there are 30 bytes stored in SPARE4 just after “S:”. They are stored as printable hex so there are 60 hex characters. The first 40 characters are the 20 bytes from the SHA-1 result and the last 20 characters come from the ten salt bytes. The salt (2B234E4447825A570F1D) is sent in the packet shown above as the AUTH\_VFR\_DATA.

4. The client takes the salt that was sent by the server and computes the hash for the salted password using SHA-1. It then uses this value to decrypt the session key. The session key was encrypted by the server using AES192 using the hash—so the client performs the inverse operation. The decryption key is derived by taking the first 20 bytes of the SHA-1 (i.e., 160 bits) and adding four zeros (to make 192 bits). The client then creates its own session key. The two AUTH\_SESSKEY values are combined by the client to form the key that is then used for password encryption. Then, the client sends both its encrypted session key (AUTH\_SESSKEY below) and the encrypted password (AUTH\_PASSWORD) to the server:

0x0000:	4	500	03bf	95d1	4000	4006	63eb	c0a8	de15	E.....@.c.....
0x0010:	c0a8	de15	802a	05f1	8082	8af8	806c	8f0b	....*	.....1.....
0x0020:	8018	0100	412e	0000	0101	080a	003d	13ad	....A.....	=.....
0x0030:	003d	13a6	038b	0000	0600	0000	0000	0373	..=	.....S
0x0040:	03fe	ffff	ff05	0000	0001	0100	00fe	ffff	ffff	.....
0x0050:	ff12	0000	00fe	ffff	ffffe	ffff	ff05	7363	.....	sc
0x0060:	6f74	740c	0000	000c	4155	5448	5f53	4553	ott.....	AUTH SES
0x0070:	534b	4559	6000	0000	fe40	4231	3338	4639	SKEY`.....	@B138F9
0x0080:	4246	4635	4632	3333	4437	3534	4632	3130	BFF5F233D754F210	
0x0090:	3230	3143	4631	3637	3645	3433	4238	3435	201CF1676E43B845	
0x00a0:	4139	4146	4242	4445	3646	3845	4245	4545	A9AFBBDE6F8EBEEE	
0x00b0:	4643	3645	3842	4544	4443	2041	3841	4234	FC6E8BEDDC.A8AB4	
0x00c0:	3746	3145	3132	3230	4141	3333	3335	3946	7F1E1220AA33359F	
0x00d0:	3032	3844	4645	3633	3345	3400	0100	0000	028DFE633E4.....	
0x00e0:	0d00	0000	0d41	5554	485f	5041	5353	574f	.....	AUTH_PASSWO

0	x00f0:	5244	4000	0000	4046	3043	3845	4543	3039	RD@.....@F0C8EEC09
0	x0100:	3843	3644	3335	3836	3946	3830	4346	3738	8C6D35869F80CF78
0	x0110:	4432	3144	3246	3736	3146	3244	3446	3743	D21D2F761F2D4F7C
0	x0120:	4543	3838	3144	4632	4446	4131	3033	4544	EC881DF2DFA103ED
0	x0130:	3430	3534	4536	4300	0000	0008	0000	0008	4054E6C.....

5. The server receives both values. It decrypts the session key sent from the client using AES192 and the password hash in the same way that the client decrypted the server's session key. Now the server has both the server session key and the client session key so it can go ahead and decrypt the password and check whether or not to authenticate the user.

O5LOGON improves on O3LOGON but it also uses a username and a password, password hashes, etc. In all these authentication methods, the usernames and passwords are kept within the data dictionary and it is the Oracle server that does the authentication. Although these are by far the most common ways that you might be authenticated to an Oracle server, you can also configure Oracle to trust the OS to authenticate you on its behalf.

## Operating System Authentication

In OS authentication, the user authenticates at the OS level and then connects to the database. When connecting to the database the user does not have to supply a password and Oracle maps the database user to the OS user based on its name. Privilege management is still done by the database—only the authentication is delegated to the OS.

Oracle distinguishes between two cases—when the user is logged onto the local OS and when the user is connecting over Oracle\*Net. When a user is already logged onto the local OS, Oracle only needs to know how to map the OS credentials to its credentials and it will trust the OS. Trusting a remote OS is one of the worst things you can do in terms of security and in fact, this has been deprecated in Oracle 11g. The risk is that someone can just put up a node on the network, configure a user that maps to a database administrator (DBA), and gain entry with no authentication being done on the server.

To enable remote OS authentication on versions prior to 11g you need to set the initialization parameter REMOTE\_OS\_AUTHENT:

```
SQL> show parameter remote_os_authent;
```

NAME	TY	PE	VALUE
remote_os_authent	b	oolean	FALSE
SQL> alter system set remote_os_authent=true scope=spfile;			
System altered.			
SQL> shutdown			
Database closed.			
Database dismounted.			
ORACLE instance shut down.			
SQL> startup			
ORACLE instance started.			
Total System Global Area			418484224 bytes
Fixed Size			1300324 bytes
Variable Size			268437660 bytes
Database Buffers			142606336 bytes
Redo Buffers			6139904 bytes
Database mounted.			
Database opened.			

```
SQL>show parameter remote_os_authent;
```

NAME	TYPE	PE	VALUE
remote_os_authent	b	boolean	TRUE

At this point, Oracle is able to accept also remote users authenticated by the client OS. You do not need to set that to have users logged onto the local OS to gain access—for that you just need to define a user as IDENTIFIED EXTERNALLY.

This does not mean you can't authenticate to Oracle using a username and a password. For users not defined as IDENTIFIED EXTERNALLY, Oracle still performs the authentication:

```
SQL> connect scott
Enter password:
Connected.
```

When using OS authentication you need to decide how to map OS usernames to Oracle usernames. The initialization parameter OS\_AUTHENT\_PREFIX controls that mapping. The database username that will be used within the database is the concatenation of this value along with the OS username. Before Oracle 11g the default for this parameter was ops\$. In 11g it is set to be null and this is a more secure setting which you should adopt for pre-11g servers. It is also the value that is set once you install Oracle Database Vault as part of general hardening that Database Vault performs. If this value is set to ops\$ and you are logged onto the OS as jane then your logon to the database will be as user ops\$jane. If this value is set to null then you will log onto the database as user jane.

If you enable remote OS authentication and have (as an example) the prefix set to ops\$, you can allow access from OS users by creating a user IDENTIFIED EXTERNALLY:

```
SQL> create user ops$jane identified externally;
User created.
SQL> grant create session to ops$jane;
Grant succeeded.
```

Now logon to the OS as jane:

```
% su - jane
```

You don't need to specify a username or a password to logon to Oracle:

```
saturn:oracle10% sqlplus /
SQL*Plus: Release 10.2.0.1.0 - Production on Fri Jan 25 11:09:05 2008
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, and Data Mining options
SQL> show user
USER is "OPS$JANE"
```

Why did the default for this parameter change from ops\$ to a null value? The reason has to do with confusion that can happen when you have a nontrivial mapping. The ops\$ default was originally created so it would be clear which users were authenticated by Oracle and which users were authenticated by

the OS. However, it allowed a situation in which you have both a user called OPS\$JANE within the database and a user called JANE within the database. Both are valid and different—but you can see that it can be very confusing. Once you change the parameter value to none, you avoid this confusion—but also lose the indication within the username of the authentication. If you are going to use ops\$ (or any non-null mapping value) you should adopt a practice by which you create a user with an impossible password (see Section 4.7) for each user with the ops\$ prefix—e.g.,

```
SQL> create user jane identified by impossible;
User created.
```

### **Administrator Authentication**

One question that you may be asking is whether you already use OS authentication when you connect “/ as sysdba”? The answer is that this is indeed a form of Oracle trusting a user that has been authenticated with the OS but this mode is called administrator authentication and is quite different. You have special handling for administrator authentication to solve the chicken-and-egg problem of starting a database. If the normal authentication method is O3/O5 LOGON and the database does the authentication using usernames and passwords stored in the database, how do you authenticate the person who needs to start the database? If the database is down, how is it going to authenticate this administrator and check their password? To solve this, Oracle authenticates administrators (users who have the sysdba or sysoper system privileges) using the OS separately from IDENTIFIED EXTERNALLY.

In almost all installations of Oracle, if you are logged onto the OS as the Oracle instance account you can connect as the SYS user using sysdba or sysoper privileges without supplying a password:

```
oracle10% sqlplus "/ as sysdba"
SQL*Plus: Release 10.2.0.1.0 - Production on Fri Jan 25 11:59:41 2008
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, and Data Mining options
SQL> show user
USER is "SYS"
```

To understand how this works, let’s look at Oracle on Unix. When you install Oracle on Unix there are two OS groups, OSDBA and OSOPER, that map into two Unix user groups—dba and oper. When an OS user tries to log onto Oracle as /, Oracle checks the Unix user group for which the user belongs to. If the user belongs to the Unix user group dba and the connection is being done as sysdba then Oracle will allow the connection and log the user on as SYS with the sysdba privileges. If the user belongs to the Unix user group oper and the connection is being done as sysoper, then Oracle will allow the connection and log the user on as PUBLIC with the sysoper privileges:

```
oracle10% sqlplus "/ as sysoper"
SQL*Plus: Release 10.2.0.1.0 - Production on Fri Jan 25 12:04:39 2008
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, and Data Mining options
SQL> show user
USER is "PUBLIC"
```

In both these cases there is no explicit username or password. This is a form of OS authentication—but it requires you to be logged in locally to the host where the database is installed and it relies on Unix user groups. It is not a general-purpose authentication scheme that you can use for any user. Also, it cannot be used for connecting as sysdba or sysoper over the network—for that you need to set up password files.

### Two Things to Remember about OS Authentication

1. Prefer to use a null REMOTE\_AUTHENT\_PREFIX.
2. If you are using OPS\$ (due to historical reasons) make sure that for every externally identified account by the name OPS\$ < USER > you create a regular account < USER > with an impossible password.

## 6.2 HOWTO Use Password Files

As you saw in the previous HOWTO, one way to logon using sysdba and sysoper privileges is by signing onto the OS as the instance owner in which case you can logon with no password. But what happens if you want to sign on with sysdba or sysoper privileges over the network or from another account that perhaps is not the one that belongs to the special Unix user groups? Then—you can use password files.

Password files allow you to set passwords that are stored outside the database and that are used for authenticating administrators. These passwords are stored in an external file that is encrypted by Oracle. They can be used even if the database is down—so even if you logon and need to start the database you can do so and have Oracle authenticate you with a password.

To create a password file use the ORAPWD utility:

```
$ orapwd file=./11g_pwd entries=100 ignorecase=n
Enter password for SYS:
```

When you run orapwd Oracle asks you for the SYS password. It creates a file based on the file name you provide for the parameter. Provide a full path name or a relative path to your current working directory. You can define the maximal number of entries that the file may contain. You can also specify whether you want passwords in the password file to be case sensitive or case insensitive (this feature is new in 11g).

Creating the password file is not enough. You also have to set the REMOTE\_LOGIN\_PASSWORDFILE initialization parameter to an appropriate value (by default, the value allows you to use password files). Available values for this parameter are

- NONE—causing Oracle to behave as though a password file does not exist.
- EXCLUSIVE—which is the default, causes Oracle to use the password files if connections with sysdba or sysoper are attempted other than from the local server. EXCLUSIVE means that the password file is being used only by your database and that you can modify it from within the database (whenever, e.g., you grant such privileges to an administrator).
- SHARED—allows you to use a single password file for multiple databases, but none of them can update the password file. If you need to update the password file then you need to switch this parameter to EXCLUSIVE in one of the databases, change the password file and then change it back to be used as SHARED.

Now you can connect with sysdba or sysoper privileges even beyond the security of the instance account. For example, you can be logged onto another machine and connect as sysdba:

```
$ sqlplus sys@o11g as sysdba
SQL*Plus: Release 10.2.0.1.0 - Production on Fri Nov 16 02:52:10 2007
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Enter password:
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining,
Oracle Database Vault, and Real Application Testing options
SQL>
```

Passwords are added as needed to the password file. A password needs to be added whenever you grant such a privilege to a user. For example, if you GRANT SYDBA to a user, Oracle must write the new password to the file so that this user can logon using the password file. As an example, the password file will be updated when you perform the following grant:

```
SQL> grant sysdba to ronb;
Grant succeeded.
```

If you have to re-create your password file you have to write all passwords for users with these privileges out to the password file. For each user it is enough to regrant the privilege. To know to which users you need to regrant the privilege, use the V\$PFILE\_USERS view. As an example, if you see the following in the view:

```
SQL> select * from v$pwfile_users;

USERNAME          SYSDB  SYSOP  SYSAS
-----
SYS              TRUE   TR UE  F  ALSE
JANE T           RUE    FALSE  FALSE
RONB T           RUE    FALSE  FALSE
```

then you can re-create the passwords in the password file by performing

```
SQL> grant sysdba to sys;
Grant succeeded.
SQL> grant sysdba to jane;
Grant succeeded.
SQL> grant sysdba to ronb;
Grant succeeded.
```

### Two Things to Remember about Password Files

1. Prefer password files for sysdba and sysoper connections.
2. Use shared password files only after you review the process you will need to perform to change a password and ensuring that you can perform this without bringing any of your production databases down.

## 6.3 HOWTO Configure Clients to Use External Password Stores

In Chapter 5 you learned about the Oracle wallet. In addition to storing keys, certificates, and certificate requests, the wallet can be used as an external password store for allowing you to authenticate with an Oracle server without supplying a password at logon time. The wallet serves as a “central container of secrets” for Oracle environments and as such it is a natural place to put passwords. When you put a password in a wallet you can be authenticated to an Oracle database without manually supplying a password—the password is taken from the wallet where it is stored. This is a very convenient and secure option that allows you, among other things, to have batches and scripts connect to the database without having to embed passwords inside these batches, a dangerous security hole.

To enable connections through an external password store (in a wallet) follow these steps:

Step 1: Create an auto-login wallet:

```
$ mkstore -wrl./oracle/product/10.2.0/db_1/wallets/ -create
Enter password:
Enter password again:
```

You need to enter a password that protects the wallet but the wallet you create is an auto-login wallet so that you can use it without embedding a password in the script. The password is needed to create, change, or delete credentials but is not needed to make the connection. The wallet used is the .sso wallet (an auto-login wallet):

```
$ ls -l./oracle/product/10.2.0/db_1/wallets/
total 24
-rw----- 1 oracle10 dba 7940 Jan 25 03:52 cwallet.sso
-rw----- 1 oracle10 dba 7912 Jan 25 03:52 ewallet.p12
```

Step 2: Once you have the sso wallet in place, you can create a secure credential within the wallet. You need to give mkstore the username you will be using to log on and the service name that you use. For example, suppose you want to connect to the database on0rh4 that appears in your tnsnames.ora as follows:

```
on0rh4 =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.222.128)(PORT = 1521))
  (CONNECT_DATA =
    ( SERVER = DEDICATED)
    ( SERVICE_NAME = on0r4231.guardium.com)
  )
)
```

If you want to be able to connect as the user system in this database without supplying a password, insert the password into the wallet by running:

```
$ mkstore -wrl./oracle/product/10.2.0/db_1/wallets -createCredential '
on0rh4' 'system' <your password here>
Enter password:
Create credential oracle.security.client.connect_string1
```

You must see the message Create credential oracle.security.client.connect\_string1 in the output—that means that your credentials were created. The password that you enter is the

wallet password—not the account password to the database you want to connect to. The account password is passed on the command line (which is a bit of a problem from a security perspective). If you don't supply the password on the command line the credentials will not be created and you will get an error message:

```
$ mkstore -wrl ./oracle/product/10.2.0/db_1/wallets/ -createCredential
on0rh4 system
Enter password:
Argument needed for command: -createCredential
```

You can check the size of your files to make sure new credentials were created:

```
$ ls -l ./oracle/product/10.2.0/db_1/wallets/
total 32
-rw----- 1 oracle10 dba 8308 Jan 25 19:46 cwallet.sso
-rw----- 1 oracle10 dba 8280 Jan 25 19:46 ewallet.p12
```

You can also list the credentials to make sure they are within the wallet:

```
$ mkst\ore -wrl ./oracle/product/10.2.0/db_1/wallets -
listCredential
Enter password:

List credential (index: connect_string username)
1: o11g system
2: on0rh4 system
```

Step 3: Edit sqlnet.ora to tell SQL\*NET that it should use the wallet as an external password store, and that it should not be using Secure Sockets Layer (SSL) authentication (which we'll cover later). You need to tell Oracle where to look for the wallet from which credentials should be taken:

```
SQLNET.WALLET_OVERRIDE = TRUE
SSL_CLIENT_AUTHENTICATION = FALSE

WALLET_LOCATION =
( SOURCE =
  ( METHOD = FILE)
  ( METHOD_DATA =
    ( DIRECTORY = /home/oracle10/oracle/product/10.2.0/db_1/wallets/ )
  )
)
```

The first line tells SQL\*NET that any connection of the form /@connection\_string should cause a lookup for the credentials in the wallet.

Step 4: Bounce the listener:

```
$ lsnrctl stop
LSNRCTL for Linux: Version 10.2.0.1.0 - Production on 25-JAN-2008 20:13:49
Copyright (c) 1991, 2005, Oracle. All rights reserved.
Connecting to (DESCRIPTION= (ADDRESS= (PROTOCOL=IPC) (KEY=ORCL) ) )
```

```
The command completed successfully
$ lsnrctl start
LSNRCTL for Linux: Version 10.2.0.1.0 - Production on 25-JAN-2008 20:13:59
Copyright (c) 1991, 2005, Oracle. All rights reserved.
Starting /home/oracle10/oracle/product/10.2.0/db_1/bin/tnslsnr: please wait...
...
The command completed successfully
```

At this point you can connect to the remote database without supplying a password:

```
$ sqlplus /@on0rh4
SQL*Plus: Release 10.2.0.1.0 - Production on Fri Jan 25 20:20:31 2008
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP, and Data Mining options
SQL>
```

If you need to change the password stored within the wallet:

```
$ mkstore -wrl ./oracle/product/10.2.0/db_1/wallets -
modifyCredential o11g system <your password here>
Enter password:
Modify credential
Modify 1
```

If you need to delete a credential from the wallet:

```
$ mkstore -wrl ./oracle/product/10.2.0/db_1/wallets -deleteCredential o11g
Enter password:
Delete credential
Delete 1
$ mkstore -wrl ./oracle/product/10.2.0/db_1/wallets -listCredential
Enter password:
List credential (index: connect_string username)
2: on0rh4 system
```

As Listing 6.1 shows, using an external store only affects the fact that you do not supply a password on the connection—the authentication scheme is unchanged (in this case still using O3LOGON) and the logon is not encrypted. In the next HOWTO, you'll see another scheme that makes a more fundamental change.

### Two Things to Remember about External Password Stores

1. Wallets are used to store all secrets—not only certificates. An example is the use of a wallet to store a password.
2. Use an auto-login wallet to store passwords so that your scripts don't have to include database passwords in cleartext. Embedding passwords is a serious risk because attackers will often search through scripts.

0x0000:	4500 013d 2eab 4000 4006 ccbd c0a8 de80	E...=@.@@.....
0x0010:	c0a8 de80 8de5 05f1 4937 ee2e 49a2 176d	.....I7..I..m
0x0020:	8018 1000 3f82 0000 0101 080a 01b9 c08d	....?.....
0x0030:	01b9 c08c 0109 0000 0100 0000 0139 012c	.....9.,
0x0040:	0c01 0800 7fff 7f08 0000 0100 00cf 003a	.....
0x0050:	0000 0200 4141 0000 0000 0000 0000 0000	....AA.....
0x0060:	0000 0000 0000 0000 0000 0000 0000 2844	.....(D
0x0070:	4553 4352 4950 5449 4f4e 3d28 4144 4452	ESCRIPRION=(ADDR
0x0080:	4553 533d 2850 524f 544f 434f 4c3d 5443	ESS=(PROTOCOL=TC
0x0090:	5029 2848 4f53 543d 3139 322e 3136 382e	P) (HOST=192.168.
0x00a0:	3232 322e 3132 3829 2850 4f52 543d 3135	222.128) (PORT=15
0x00b0:	3231 2929 2843 4f4e 4e45 4354 5f44 4154	21) )(CONNECT_DAT
0x00c0:	413d 2853 4552 5645 523d 4445 4449 4341	A=(SERVER=DEDICA
0x00d0:	5445 4429 2853 4552 5649 4345 5f4e 414d	TED) (SERVICE_NAM
0x00e0:	453d 6f6e 3072 3432 3331 2e67 7561 7264	E=on0r4231.dbase
0x00f0:	6975 6d2e 636f 6d29 2843 4944 3d28 5052	cur.com) (CID=(PR
0x0100:	4f47 5241 4d3d 7371 6c70 6c75 7329 2848	OGRAM=sqlplus) (H
0x0110:	4f53 543d 7268 3475 3278 3332 702e 6775	OST=rh4u2x32p.db
0x0120:	6172 6469 756d 2e63 6f6d 2928 5553 4552	asecur.com) (USER
0x0130:	3d6f 7261 636c 6531 3029 2929 29	=oracle10) )) )
0x0000:	4500 0134 2eb7 4000 4006 ccba c0a8 de80	E..4..@.@@.....
0x0010:	c0a8 de80 8de5 05f1 4937 f150 49a2 18e7	.....I7.PI...
0x0020:	8018 1000 3f79 0000 0101 080a 01b9 c117	....?Y.....
0x0030:	01b9 c115 0100 0000 0600 0000 0000 0376	.....v
0x0040:	02d0 a108 0806 0000 0001 0000 00d8 bcff	.....
0x0050:	bf05 0000 0080 b9ff bfa8 beff bf06 7379	.....sy
0x0060:	7374 656d 0d00 0000 0d41 5554 485f 5445	stem....AUTH_TE
0x0070:	524d 494e 414c 0500 0000 0570 7473 2f32	RMINAL....pts/2
0x0080:	0000 0000 0f00 0000 0f41 5554 485f 5052	.....AUTH_PR
0x0090:	4f47 5241 4d5f 4e4d 2a00 0000 2a73 716c	OGRAM_NM*...*sql
0x00a0:	706c 7573 4072 6834 7532 7833 3270 2e67	plus@rh4u2x32p.d
0x00b0:	7561 7264 6975 6d2e 636f 6d20 2854 4e53	dbasecu.com.(TNS
0x00c0:	2056 312d 5633 2900 0000 000c 0000 000c	.V1-V3) .....
0x00d0:	4155 5448 5f4d 4143 4849 4e45 1600 0000	AUTH_MACHINE....
0x00e0:	1672 6834 7532 7833 3270 2e67 7561 7264	.rh4u2x32p.dbase
0x00f0:	6975 6d2e 636f 6d00 0000 0008 0000 0008	cur.com.....
0x0100:	4155 5448 5f50 4944 0500 0000 0531 3735	AUTH_PID.....175
0x0110:	3233 0000 0000 0800 0000 0841 5554 485f	23.....AUTH_
0x0120:	5349 4408 0000 0008 6f72 6163 6c65 3130	SID.....oracle10
0x0130:	0000 0000	....
0x0000:	4500 0115 d7ef 4000 4006 23a1 c0a8 de80	E.....@.#@.#.....
0x0010:	c0a8 de80 05f1 8de5 49a2 18e7 4937 f250	.....I...I7.P
0x0020:	8018 1000 3f5a 0000 0101 080a 01b9 c120	....?Z.....
0x0030:	01b9 c117 00e1 0000 0600 0000 0000 0802	.....
0x0040:	000c 0000 000c 4155 5448 5f53 4553 534b	.....AUTH_SESSK
0x0050:	4559 4000 0000 4030 3234 3245 3030 4431	EY@...@0242E00D1
0x0060:	4645 4432 4641 3332 3139 3039 3941 3938	FED2FA3219099A98
0x0070:	3443 3641 3842 4446 3233 4234 3142 3941	4C6A8BDF23B41B9A
0x0080:	3531 4130 3934 3433 4146 3646 3038 3442	51A09443AF6F084B
0x0090:	3233 3444 3043 4200 0000 000d 0000 000d	234D0CB.....
0x00a0:	4155 5448 5f56 4652 5f44 4154 4100 0000	AUTH_VFR_DATA...
0x00b0:	0039 0900 0004 0100 0000 0200 0100 0000	.9.....
0x00c0:	0000 0000 0000 0000 0000 0000 0000 0000	.....
0x00d0:	0000 0000 0000 0000 0000 0000 0000 0000	.....

Listing 6.1 Authentication sequence does not change when using an external password store.

0x00e0:	0000 0000 0000 0200 0000 0000 0036 0100	.....6..
0x00f0:	0000 0000 0020 13bd 0c00 0000 0000 0000	.....
0x0100:	0000 0000 0000 0000 0000 0000 0000 0000	.....
0x0110:	0000 0000 00	....
0x0000:	4500 02e4 2eb9 4000 4006 cb08 c0a8 de80	E.....@. ....
0x0010:	c0a8 de80 8de5 05f1 4937 f250 49a2 19c8	.....I7.PI...
0x0020:	8018 1000 4129 0000 0101 080a 01b9 c123	....A) .....#
0x0030:	01b9 c120 02b0 0000 0600 0000 0000 0373	.....s
0x0040:	03d0 a108 0806 0000 0001 0100 00fc e8ff	.....
0x0050:	bf0d 0000 00a4 e5ff bf90 f5ff bf06 7379	.....sy
0x0060:	7374 656d 0c00 0000 0c41 5554 485f 5345	<u>stem</u> .....AUTH_SE
0x0070:	5353 4b45 5940 0000 0040 4642 3931 4334	SSKEY@...@FB91C4
0x0080:	3239 4346 3639 3742 3242 3739 4532 3142	29CF697B2B79E21B
0x0090:	3243 3841 3643 3733 3038 4543 4537 3442	2C8A6C7308ECE74B
0x00a0:	3146 4539 3546 4637 4438 4130 4545 3734	1FE95FF7D8A0EE74
0x00b0:	4236 3330 3436 3743 3043 0100 0000 0d00	B630467C0C.....
0x00c0:	0000 0d41 5554 485f 5041 5353 574f 5244	....AUTH_PASSWORD
0x00d0:	4000 0000 4039 4539 3345 4135 4430 4644	@...@9E93EA5D0FD
0x00e0:	4531 4435 4443 4637 4539 4137 4338 3938	E1D5DCF7E9A7C898
0x00f0:	4446 3933 3444 4130 4342 3137 3139 3035	DF934DA0CB171905
0x0100:	4338 3438 4642 4632 4446 3746 3941 3736	C848FBF2DF7F9A76
0x0110:	3744 3744 4600 0000 0008 0000 0008 4155	7D7DF.....AU
0x0120:	5448 5f52 5454 0500 0000 0531 3639 3630	TH_RTT.....16960
0x0130:	0000 0000 0d00 0000 0d41 5554 485f 434c	.....AUTH_CL
0x0140:	4e54 5f4d 454d 0400 0000 0434 3039 3600	NT_MEM.....4096.
0x0150:	0000 000d 0000 000d 4155 5448 5f54 4552	.....AUTH_TER
0x0160:	4d49 4e41 4c05 0000 0005 7074 732f 3200	MINAL.....pts/2.
0x0170:	0000 000f 0000 000f 4155 5448 5f50 524f	.....AUTH_PRO
0x0180:	4752 414d 5f4e 4d2a 0000 002a 7371 6c70	GRAM_NM*...*sqlp
0x0190:	6c75 7340 7268 3475 3278 3332 702e 6775	lus@rh4u2x32p.db
0x01a0:	6172 6469 756d 2e63 6f6d 2028 544e 5320	asecur.com.(TNS.
0x01b0:	5631 2d56 3329 0000 0000 0c00 0000 0c41	V1-V3).....A
0x01c0:	5554 485f 4d41 4348 494e 4516 0000 0016	UTH_MACHINE.....
0x01d0:	7268 3475 3278 3332 702e 6775 6172 6469	rh4u2x32p.dbsec
0x01e0:	756d 2e63 6f6d 0000 0000 0800 0000 0841	ur.com.....A
0x01f0:	5554 485f 5049 4405 0000 0005 3137 3532	UTH_PID.....1752
0x0200:	3300 0000 0008 0000 0008 4155 5448 5f53	3.....AUTH_S
0x0210:	4944 0800 0000 086f 7261 636c 6531 3000	ID.....oracle10.
0x0220:	0000 0008 0000 0008 4155 5448 5f41 434c	.....AUTH_ACL
0x0230:	0400 0000 0434 3430 3000 0000 0012 0000	....4400.....
0x0240:	0012 4155 5448 5f41 4c54 4552 5f53 4553	..AUTH_ALTER_SES
0x0250:	5349 4f4e 2500 0000 2541 4c54 4552 2053	SION%...%ALTER.S
0x0260:	4553 5349 4f4e 2053 4554 2054 494d 455f	ESSION.SET.TIME_
0x0270:	5a4f 4e45 3d27 2d30 353a 3030 2700 0100	ZONE='-05:00'...
0x0280:	0000 1700 0000 1741 5554 485f 4c4f 4749	.....AUTH_LOGI
0x0290:	4341 4c5f 5345 5353 494f 4e5f 4944 2000	CAL_SESSION_ID..
0x02a0:	0000 2034 3439 3545 3741 3835 3637 3744	...4495E7A85677D
0x02b0:	3935 4445 3034 3041 3843 3038 3044 4534	95DE040A8C080DE4
0x02c0:	3437 3300 0000 0010 0000 0010 4155 5448	473.....AUTH
0x02d0:	5f46 4149 4c4f 5645 525f 4944 0000 0000	_FAILOVER_ID....
0x02e0:	0000 0000	....

**Listing 6.1 (continued)**

## 6.4 HOWTO Configure SSL-Based Authentication Using ASO

Another authentication method supported by Oracle that uses the wallet is SSL authentication. This authentication method requires you to have Oracle Advanced Security Options (ASO) installed. This authentication method is more complex to set up than external password stores but is far more comprehensive and functional. SSL authentication makes use of certificates rather than passwords. The certificates are stored within the wallet and used when you make the connection. Like in the external password store HOWTO, this allows you to connect without supplying a password. However, unlike the previous HOWTO, SSL authentication buys you much more—authentication can happen from anywhere on the network, the communications is encrypted, and certificates are used for mutual authentication. This can come in handy when you want to ensure that only your application server makes a connection to the database—in which case you will place the certificate in the application server's wallet and thus guarantee that connections can only be made locally on the database server or from the application server.

Figure 6.1 shows what the handshake between the client and the server looks like when you use SSL authentication:

1. The client initiates an SSL connection using a TCPS entry in tnsnames. It does not need to logon with a password.
2. The client and the server negotiate a cipher suite, which they will use during the communications.
3. The server gets its certificate from the wallet and sends it to the client.
4. The client checks whether it can trust the server's certificate. It checks within the wallet whether this certificate is a trusted certificate or is signed by a trusted certificate authority (CA) (see Chapter 5).
5. If client authentication is required then the client gets its certificate from its wallet and sends it to the server.
6. The server goes through the same process to check whether it can trust the client's certificate.
7. The client and server use the public keys within the certificates to exchange random session keys that will be used as a symmetric key for network encryption.
8. At this point communication can begin—each side has authenticated and approved the other party and they have an encryption key and a cipher suite with which they can set up encrypted communication.

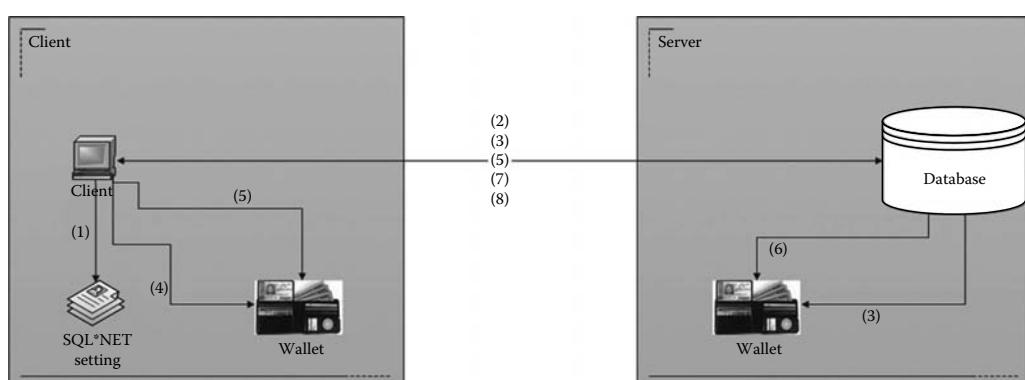


Figure 6.1 Initiating a connection using certificates for authentication.

Next, let's go through a complete setup process and see how to set up SSL authentication. This sequence includes the minimum setup requirements. In addition to this sequence you can control many things such as the available cipher suites, the prioritization of the cipher suites, etc. The basic steps are:

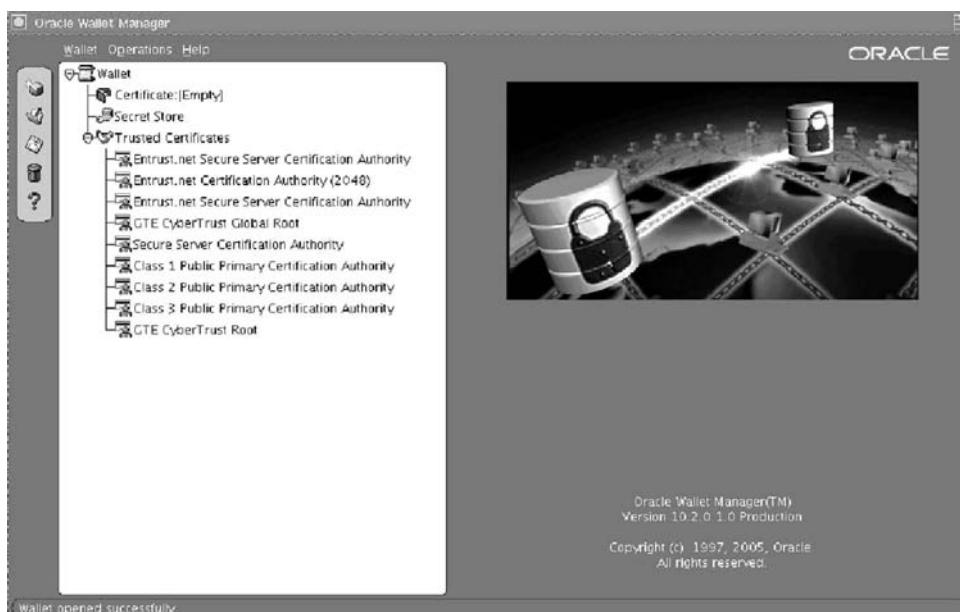
Step 1: First you need to create an auto-login wallet to store your certificates and in order for Oracle to be able to check whether certificates can be trusted. In Chapter 5 you learned how to use orapki to set up wallets and certificates; let's set things up using the Oracle Wallet Manager (OWM) this time. On the server, you already created an auto-login wallet in the previous HOWTO—so we'll use that wallet on the server. Start up the OWM:

```
$ owm
```

Click on Wallet → Open and browse your wallet location. Enter the password to the wallet to open it. At this point the wallet is empty—there are no certificates that you can use as shown in Figure 6.2. Verify that the wallet is indeed an auto-login wallet by clicking on the Wallet menu option—the second to last entry should have Auto Login checked. If it is not, check it.

Step 2: Edit sqlnet.ora and listener.ora on the server and add an entry to include the wallet location. The entry should look like the following and include a full path name:

```
WALLET_LOCATION =
( SOURCE =
( METHOD = FILE)
( METHOD_DATA =
(DIRECTORY = /home/oracle10/oracle/product/10.2.0/db_1/wallets/)
)
)
```



**Figure 6.2 Viewing the wallet using OWM.**

Step 3: Edit sqlnet.ora to define that the server should also authenticate the client (i.e., the default). You want the server to use the client certificate rather than require you to logon with a password:

```
SSL_CLIENT_AUTHENTICATION = TRUE
```

Step 4: Edit sqlnet.ora to add the SSL authentication service (TCPS) as one of the possible options the server can use:

```
SQLNET.AUTHENTICATION_SERVICES = (TCPS)
```

Step 5: Edit listener.ora to create a default TCPS listening port and set the wallet location. The default port number is 2484:

```
LISTENER =
(DESCRIPTION_LIST =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = IPC)(KEY = ORCL))
    (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.222.128)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCPS)(HOST = 192.168.222.128)(PORT = 2484))
  )
)
WALLET_LOCATION =
( SOURCE =
  (METHOD = FILE)
  (METHOD_DATA =
    (DIRECTORY = /home/oracle10/oracle/product/10.2.0/db_1/wallets/)
  )
)
```

You've completed the setup for the server—it's time to move on to the client.

Step 6: Create an auto-login wallet on the client. Open owm and select Wallet→ New. Enter a password. At this point the wallet is created and is empty. You will create the certificates later. By default the wallet is not created as an auto-login wallet so click on the Wallet, pull down, and click the Auto Login check box. Save the wallet.

Step 7: In listener.ora on the client add the TCPS protocol:

```
LISTENER =
(DESCRIPTION_LIST =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = IPC)(KEY = ORCL))
    (ADDRESS = (PROTOCOL = TCP)(HOST = o11g.dbasecur.com)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCPS)(HOST = o11g. dbasecur.com)(PORT = 2484))
    (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
  )
)
```

Step 8: Edit sqlnet.ora on the client and set the wallet location, TCPS authentication, and the SSL\_CLIENT\_AUTHENTICATION setting:

```
SQLNET.AUTHENTICATION_SERVICES = (TCPS)
SSL_CLIENT_AUTHENTICATION = TRUE
WALLET_LOCATION =
```

```
( SOURCE =
  ( METHOD = FILE)
  ( METHOD_DATA =
(DIRECTORY = /home/oracle11/product/11.1/db_1/wallets/
)
)
```

Step 9: Edit the server's entry in tnsnames.ora on the client to add a security clause for SSL\_SERVER\_CERT\_DN. This entry tells Oracle to match the server's DN with its host name and makes sure that the server certificate matches the host definitions. This client-side validation helps you ensure that no one is impersonating the server. Edit tnsnames.ora and create an entry that specifies the security property and the TCPS protocol:

```
on0rh4 =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCPS)(HOST = 192.168.222.128)(PORT = 2484))
  (SECURITY = (SSL_SERVER_CERT_DN="cn=rh4u2x32p,o=dbasecur"))
  (CONNECT_DATA =
    ( SERVER = DEDICATED)
    ( SERVICE_NAME = on0r4231.guardium.com)
  )
)
```

Now generate the certificates with which the server and client can authenticate each other (and trust each other) using OWM or using orapki as reviewed in Chapter 5. Create the relevant Oracle users that you will connect as identified externally. At that point you can logon to the database by using

```
SQL> connect /@ on0rh4
```

Note that if you chose SSL\_CLIENT\_AUTHENTICATION = FALSE then you will still be connecting over an SSL session in terms of network encryption but authentication will have to be done using the password so you should connect using

```
SQL> connect system@on0rh4
```

Oracle will require you to enter a password.

### **Two Things to Remember about SSL Authentication using ASO**

1. SSL authentication allows you to connect to the database using a certificate rather than a password. It is easier to control certificates and keys than it is to control passwords—so this method is more secure.
2. Setting up SSL also means that the communications will be encrypted—so you get both stronger authentication and network encryption in one configuration.

## **6.5 HOWTO Configure Kerberos Authentication Using ASO**

Kerberos is perhaps the world's most popular authentication method. It was created by the Massachusetts Institute of Technology (MIT) and you can freely download it from MIT at <http://web.mit.edu/kerberos/dist/index.html>. The protocol was named after the Greek mythological

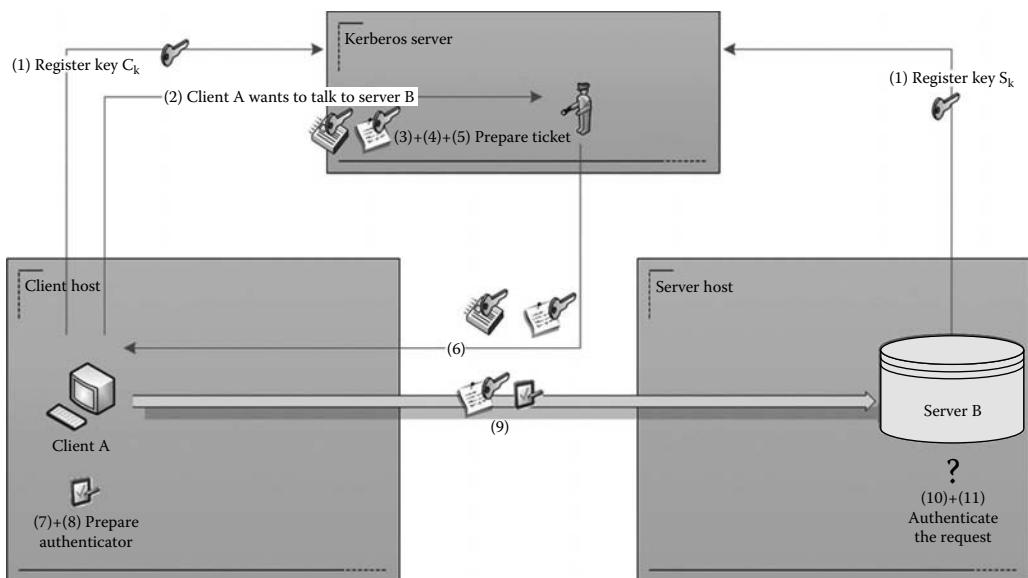


Figure 6.3 Kerberos authentication scheme.

character Cerberus—the three-headed guard dog of Hades. What made Kerberos so popular is, surprisingly, Microsoft. Microsoft fully embraced Kerberos rather than invent a complex authentication method and on any modern Windows environment all authentication occurs using Kerberos as a default.

Figure 6.3 shows how the Kerberos authentication process works (in reality it is more complex and has more steps but for the purpose of this HOWTO this understanding is enough).

1. Kerberos requires an initial setup phase where the parties are registered with the Kerberos server. The Kerberos server is called the Key Distribution Center (KDC) and consists of an Authentication Server and a Ticket Granting Server (TGS). Kerberos uses symmetric keys and does not rely on certificates or PKI. It assumes that there is a trusted third party—the Kerberos server. The KDC maintains a database of secret keys—each entity on the network shares a secret key known only to itself and the KDC. Therefore, in the first step the server and the client need to register with the Kerberos server. At this point, a key for the client  $C_k$  and a key for the server  $S_k$  are generated. For the client, the secret key is generated by applying a one-way hash function on the client's password.
2. After the client has authenticated itself to the Kerberos server it wants to connect to the database. At this point, the client sends a message to the Kerberos server of the form “client A wants to talk to Server B.” This is passed in cleartext to the Kerberos server.
3. The Kerberos server picks a random session key,  $K$ .
4. Kerberos encrypts the session key and the string “Server B” with the client key—creating  $C_k(K, \text{Server B})$ .
5. Kerberos encrypts the session key and the string “Client A” with the server key—creating  $S_k(K, \text{Client A})$ . This is called the ticket.
6. Both  $C_k(K, \text{Server B})$  and  $S_k(K, \text{Client A})$  are sent to the client as the response to the request in phase 2.

7. The client knows what the client key is and therefore can decrypt  $C_k(K, \text{Server B})$ . It extracts the session key  $K$ . It also validates that it got the string “Server B”—the server it asked to talk to in the first place.
8. The client uses the session key  $K$  as an encryption key to generate  $K(\text{time}, S_k(K, \text{Client A}))$ ; it takes the ticket and the current time and encrypts them with the session key. This is called the authenticator.
9. The client sends the ticket and the authenticator to the server.
10. The server has the server key so it can decrypt the ticket—i.e., it can decrypt  $S_k(K, \text{Client A})$  and extract the session key  $K$ .
11. The server uses the session key to decrypt the authenticator  $K(\text{time}, S_k(K, \text{Client A}))$ . It checks that the ticket within the authenticator is the same as the ticket it received from the client and that the time is correct (i.e., there is no large shift in time that might indicate a replay attack). At this point it knows the client is authenticated and it knows who the client is—Client A was in the ticket itself.

Let's go ahead and see how to set up Kerberos authentication for an Oracle server. Kerberos authentication is an advanced option that is part of Oracle ASO so the first step is to install ASO on your clients and servers. In the sequence below there are three machines—the KDC, the database server, and a client machine. In the example, the name of the machine which hosts the KDC is *goose*. The database machine runs on *saturn* and the client connects from a machine called *trex*. Before you can configure Kerberos authentication for Oracle you have to download and install Kerberos and make sure that ASO is installed for your database. Then, follow these steps to configure Kerberos authentication:

Step 1: Configure a Service Principal (SP) for the Oracle database server.

The format for a SP is `service/instance@REALM`. You can choose to use any name for your database but a good convention is to map the service name or SID to the service and the hostname to instance—e.g., `orcl/saturn.guardium.com@GUARDIUM.COM`. It is very important that the realm be uppercase—if it is not, nothing will work but the errors will not be helpful.

To create the SP start the `kadmin` utility:

```
[root@goose sbin]# cd /usr/krb5/sbin
[root@goose sbin]# ./kadmin.local
```

Once in `kadmin` (you'll see `kadmin.local:` as the prompt), add the SP:

```
kadmin.local: addprinc -randkey orcl/saturn.guardium.com@GUARDIUM.COM
```

To verify that the SP has been added:

```
kadmin.local: getprincs
kadmin/admin@GUARDIUM.COM
kadmin/changepw@GUARDIUM.COM
kadmin/history@GUARDIUM.COM
orcl/saturn.guardium.com@GUARDIUM.COM
```

Exit `kadmin`:

```
kadmin.local: exit
```

If you want to see the activity in the log you can:

```
tail -f /var/krb5/log/krb5kdc.log
```

Step 2: Extract the Service Table from Kerberos. Extract the file from the KDC and then copy the file to the database server (e.g., to /tmp/keytab). On the database server add it using kadmin:

```
kadmin.local: ktadd -k /tmp/keytab orcl/saturn.guardium.com
Entry for principal orcl/saturn.guardium.com with kvno 2, encryption
DES-CBC-CRC added to the keytab WRFILE: '/tmp/keytab'
kadmin.local: exit
```

Run the oklist utility:

```
oklist -k -t /tmp/keytab
```

Step 3: Configure Kerberos authentication. Edit sqlnet.ora and add the following entries:

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

On the server set the following two initialization parameters:

```
REMOTE_OS_AUTHENT=FALSE
OS_AUTHENT_PREFIX=""
```

You don't have to set the OS\_AUTHENT\_PREFIX to null—but you should. It is generally a better security practice than leaving OPS\$, and more importantly Kerberos names tend to be very long (because of the qualified name and the realm) and you need to remain within the limit of a maximum of 30 characters—no point in wasting three of them on the prefix.

Step 4: Create a Kerberos user. On the KDC run:

```
[root@goose sbin]# cd /usr/krb5/sbin
[root@goose sbin]# ./kadmin.local
kadmin.local: addprinc SCOTTY
Enter password for principal: "SCOTTY@GUARDIUM.COM":
Re-enter password for principal: "SCOTTY@GUARDIUM.COM":
kadmin.local: exit
```

Step 5: Create an externally authenticated Oracle database user:

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

Step 6: Get a ticket for the user and then connect with it to the database. Ask the KDC for an initial ticket:

```
trex:oracle10% okinit -f SCOTTY
Password for SCOTTY@GUARDIUM.COM:password
```

Now you can connect to the database without supplying a username and a password. The client will use the Kerberos ticket and the database will authenticate you based on the ticket:

```
trex:oracle10% sqlplus /@orcl
```

If this is the first time you are setting this up, make sure to check that you cannot connect without the ticket. You need to make sure you did not open up the database completely. Destroy the ticket cache using the following command and then try to connect (hopefully unsuccessfully):

```
trex:oracle10% okdstry  
trex:oracle10% sqlplus /@orcl
```

Kerberos is one of the most secure authentication schemes. Part of its security stems from the fact that what passes on the network is not enough for an attacker to be able to discover passwords. This is in contrast to the examples with the O5LOGON scheme that you saw in HOWTO 6.1. In O5LOGON, the password hash stored in SPARE4 of the SYS.USER\$ table is used. If, for example, someone can access to this table (through some faulty privilege definitions or role assignment) then, with the hashes in hand and through sniffing of the authentication packets, an attacker can get all of the cleartext passwords! This is because all the AUTH\_SESSKEY values sent over the wire are encrypted using the hash values. So if you have the password hashes you can decrypt the session keys and then decrypt the password that is eventually passed from the client to the server. To protect O3LOGON or O5LOGON authentication from such an attack you should encrypt the network traffic (see Chapter 7). For example, if you encrypt the communications using ASO network encryption then as soon as the session is established, the encryption begins and the sending of the AUTH\_SESSKEY is already encrypted as you can see from Listing 6.2.

#### **Two Things to Remember about Oracle Kerberos Authentication**

1. If you have Unix systems, use the MIT Kerberos package. If you have a Windows-only environment, then use the Microsoft domain controller as the KDC.
2. Remember that the realm must be all uppercase.

## **6.6 HOWTO Configure RADIUS and Two-Factor Authentication Using ASO**

The Remote Authentication Dial-In User Service (RADIUS) is also one of the most widely used authentication protocols. It is ubiquitously used to control access to network resources through dialup, Digital Subscriber Line (DSL) providers, or other Internet Service Providers (ISPs). It has also been adopted for its AAA capabilities (authentication, authorization, and accounting) by many networking groups within the enterprise. When you use RADIUS, the client machine requests access to network resources via a Network Access Server (NAS). The NAS issues a RADIUS access request message to the RADIUS server requesting authorization to grant access. This request includes credentials that are provided by the user. The RADIUS protocol does not transmit passwords in cleartext between the NAS and the RADIUS server. A shared secret is used along with the message digests

Without Encryption:	With Encryption:
<pre> 0x0000: E..\$..@..f..... 0x0010: .....*.....l.- 0x0020: ....&gt;.....=.. 0x0030: .=.....:.,. 0x0040: .A.....:. 0x0050: ....AA..... 0x0060: .....(D 0x0070: ESCRIPTION=(ADDR 0x0080: ESS=(PROTOCOL=TC 0x0090: P)(HOST=o11g.dba 0x00a0: secur.com)(PORT= 0x00b0: 1521))(CONNECT_D 0x00c0: ATA=(SERVER=DEDI 0x00d0: CATED)(SID=on1r4 0x00e0: 531)(CID=(PROGRA 0x00f0: M=sqlplus)(HOST= 0x0100: o11g.dbasecur.co 0x0110: m)(USER=oracle11 0x0120: ))))  0x0000: E.....@..]B.... 0x0010: .....*..1..... 0x0020: ....?.....=.. 0x0030: .=....^..... 0x0040: .....AUTH_SESSK 0x0050: EY`...`94C497598 0x0060: 4EE74A3E79C9E85F 0x0070: DF085B40B64D24C3 0x0080: A8B4B9E37AA8EF44 0x0090: 82EDA966B44F168C 0x00a0: C8E4E413BB8C748 0x00b0: 9210236..... 0x00c0: AUTH_VFR_DATA... 0x00d0: ..2B234E4447825A 0x00e0: 570F1D%.....A 0x00f0: UTH_GLOBALLY_UNI 0x0100: QUE_DBID.....44 0x0110: 36399542AB6059FE 0x0120: 1442827C841DD3.. 0x0130: ..... 0x0140: ..... 0x0150: ..... 0x0160: .....6..... 0x0170: ...UU..... 0x0180: ..... 0x0190: .. </pre>	<pre> 0x0000: E..+..@..0#..... 0x0010: .....:.....fy[ 0x0020: .....7b% 0x0030: .7b%.....:., 0x0040: .A.....:. 0x0050: ....AA..... 0x0060: .....(D 0x0070: ESCRIPTION=(ADDR 0x0080: ESS=(PROTOCOL=TC 0x0090: P)(HOST=abcd32.d 0x00a0: basecur.com)(POR 0x00b0: T=1522))(CONNECT 0x00c0: _DATA=(SERVER=DE 0x00d0: DICATED)(SERVICE 0x00e0: _NAME=on1abcd3)( 0x00f0: CID=(PROGRAM=sql 0x0100: plus@abcd32)(HOS 0x0110: T=abcd32)(USER=o 0x0120: racle11)) )  0x0000: E.....@..]. 0x0010: .....:..f{..... 0x0020: .....}.....7b0 0x0030: .7b0.d.....N 0x0040: (&amp;.G..0...B...Y. 0x0050: #...a.5.....P&gt;.. 0x0060: ..L.....q.....9 0x0070: ?Z!S...5...t!9&lt;- 0x0080: ..)2...&lt;}#....=. 0x0090: -v..b..\$.C.&gt;...h 0x00a0: ...L...U..A....W 0x00b0: n..ZW...F.!..\$.. 0x00c0: ..C.t..7.'t.~/..} 0x00d0: .hu."....HFbnb.. 0x00e0: .....'. 0x00f0: .-^.)."..Gr..G.. 0x0100: u.....t6/...^r 0x0110: aj.....1#.....A 0x0120: ...Ev8...osyfG/_N 0x0130: .cG.....Zm..rE. 0x0140: L.....C..N5y..... 0x0150: D....`....e....`T.. 0x0160: .p.....t....~ 0x0170: ._1.....].mw..g. 0x0180: [....u..q{.0..a. ... </pre>

---

**Listing 6.2 First packets from O5LOGON authentication process—with and without network encryption.**

5 (MD5) hashing algorithm. Additionally, the traffic is usually encrypted using IPSEC or some tunneling method to add protection. RADIUS is often used with an Lightweight Directory Access Protocol (LDAP) directory (e.g., Active Directory) where RADIUS performs the authentication but information is stored within the LDAP store.

Oracle databases live within the data center. They may be accessible through a virtual private network (VPN) connection, but this is usually transparent to Oracle—the user logs onto the network over a VPN and then can connect to various resources on the LAN. It is not common to have an Oracle database as a network resource available as a network resource managed by RADIUS. If that is so, why bother learning about RADIUS authentication for Oracle? The reason is that because RADIUS is so pervasive and because it is a fairly simple protocol to implement, it is used by many vendors who offer two-factor authentication as an API through which they can be accessed. If you want to harden your authentication process using two-factor authentication, it is most likely that you will be using RADIUS authentication for your databases.

Let's understand what two-factor authentication means. The most common authentication scheme is the one which is based on a password—i.e., based on “something you know.” This is considered by many to be a weak scheme—passwords can be guessed, can be seen when you type them in, and can be shared by multiple users. Two-factor authentication makes the logon dependent on two things—usually on “something you know” and “something you have.” Almost all two-factor authentication schemes still use passwords but they augment that with one more thing—usually some physical device that you must have to logon. This physical device can be a USB key, can be a card that you carry with you, etc.—the important thing is that it is something that you have, and only you can have. There is only one device in the world that could be used to logon to your account.

Many companies make two-factor authentication devices and there are many forms that it can take. Figure 6.4 shows three such examples. On the left is an RSA SecureID token. You carry this little token with you (e.g., on your keychain). As you can see the token presents a number with eight digits. This number changes every 60 s. Each such token has a unique internal key so every




---

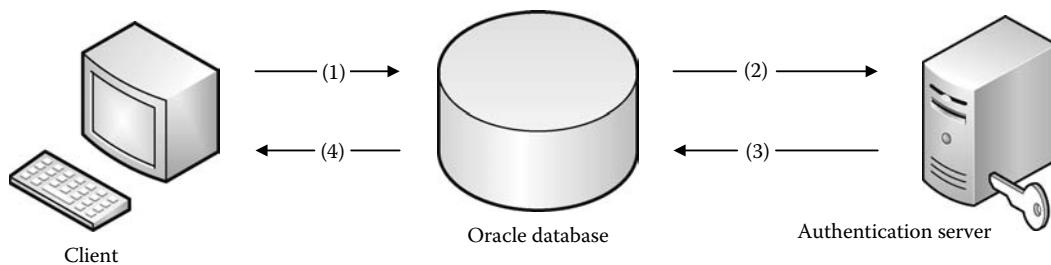
**Figure 6.4 Example of two-factor authentication devices.**

such token will switch numbers in a different sequence that cannot be guessed. Therefore, if you have 30,000 employees in your organization and everyone has such a token, the number sequences are all unrelated. However, they are NOT random. In fact, a centralized RSA ACE authentication server knows at any point of time what number each token is showing. When you log onto a system that delegates authentication to the ACE/Server, you provide the number on your token in addition to your password. Because the ACE/Server can compute the number sequences on each device it can verify that you not only know the password but that you indeed have the physical device.

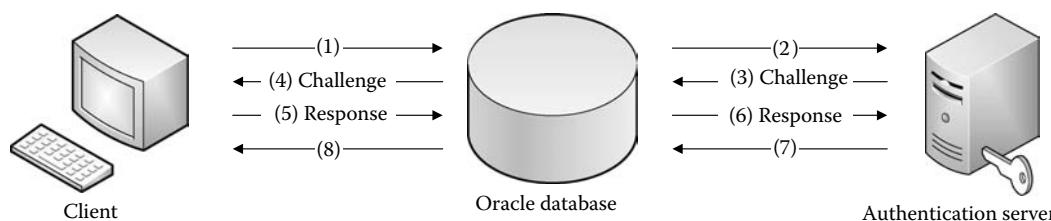
The two tokens on the right-hand side of Figure 6.4 use a different scheme. In this scheme when you want to be authenticated, the server will give you a random number. You then key in that random number on the keypad. The device performs some mathematical transformation on this number using a key embedded into the device and shows you the output on the liquid-crystal display (LCD). You then enter this number as part of the authentication process. Because the authentication server knows the keys per device, it too can compute the transformation and verify that you indeed have the device.

There are two scenarios that can occur when you use two-factor authentication. Figure 6.5 shows the basic scenario. Let's stick with the RSA SecureID token as an example. In this case, when you log onto Oracle you provide your username, your password, and the number you see on the token (step 1 in Figure 6.5). Oracle forwards this information to the authentication server using a RADIUS protocol (step 2). The authentication server responds to the Oracle server (step 3) and Oracle responds to the client (step 4); you are now logged on to Oracle.

As mentioned, the number on the SecureID device changes every 60 s. What happens if you type in the value you see on the device 2 seconds before it changes and then try to logon? By the time you key it in and press enter, 2 seconds may pass and when the authentication server performs its check, it will have a different number—in which case the numbers will not match. In this scenario, shown in Figure 6.6, the authentication server will initiate a challenge/response sequence.



**Figure 6.5 Synchronous two-factor authentication sequence.**



**Figure 6.6 Challenge/Responses in two-factor authentication sequence.**

In this scenario the authentication server will not immediately reject the logon. Instead, it sends a challenge as shown in steps 3 and 4. You will see the challenge because your logon screen will have some form of message that tells you to wait a full cycle until the number changes once more and then logon using the new number. This time you have 60 seconds until the next change and therefore the response (steps 5 and 6) will likely succeed. To try to avoid this complexity most devices such as SecureID show you how long you have left until the next change. In Figure 6.4 the bars on the left of the number show you how much time is left until the next change. When the number changes (and you have 60 seconds until the next switch) there are six bars. One bar is dropped every 10 seconds so you can always tell when the number will switch next. If you see that you have only one bar left, you will usually wait until the number switches again before attempting a logon. With the devices with the keypad shown on the right of Figure 6.4, there is always a challenge/response sequence—the authentication server issues a random number that you type in using the keypad (the challenge) and the resulting number is your response.

Let's move on to setting up RADIUS authentication using ASO. Setting up RADIUS authentication for Oracle looks the same regardless of which system you are using for authentication. Sometimes the challenge/response phase requires custom handling—but for the most part, the fact that the RADIUS protocol is a standard supported by ASO allows you to interface Oracle with many systems. In this specific example you'll follow the procedure for integrating Oracle with the RSA ACE/Server using RADIUS. The sequence for this setup is

Step 1: On your ACE/Server start the RSA Authentication Manager application. Click on Agent Host→ Add Agent Host. You need to register your database server. Put in the host name, address, and select Net OS Agent as the Agent type as shown in Figure 6.7. Uncheck Requires Name Lock and all the other default options as shown in Figure 6.7.

Step 2: Click on Secondary Nodes and enter all the hostnames/IP addresses for Oracle.

Step 3: Click on Create Node Secret File to generate the node secret file.

Step 4: Using the import utility, convert the .rec file to a .key file. Save this key on the Oracle server in \$ORACLE\_HOME/network/security/radius.key and ensure that the file is protected through file permissions.

Step 5: On the database server and the clients, edit sqlnet.ora and add the RADIUS authentication method:

```
SQLNET.AUTHENTICATION_SERVICES= (BEQ, RADIUS)
```

Step 6: Start up netmgr on the database server. Select Oracle Advanced Security and go to the Other Params tab. Select RADIUS from the pull down and enter the RADIUS server properties as shown in Figure 6.8.

Step 7: Set the following two initialization parameters:

```
REMOTE_OS_AUTHENT=FALSE
OS_AUTHENT_PREFIX=""
```

Step 8: Create a database user identified externally:

```
SQL> create user ronb identified externally;
SQL> grant create session to user ronb;
```



**Figure 6.7 Adding the database as a RADIUS client.**

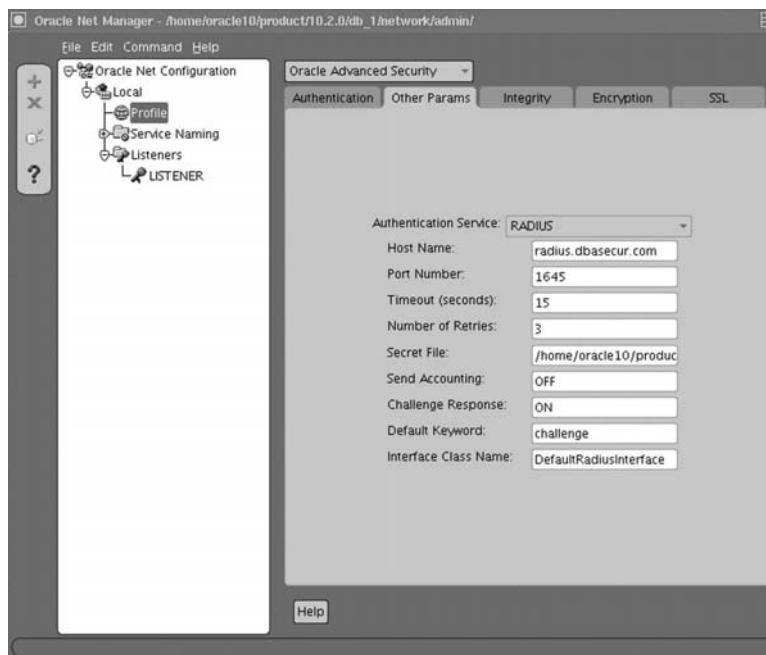
When you connect using this user the database will authenticate it through the RSA/ACE server using a RADIUS protocol.

#### Two Things to Remember about Oracle RADIUS Authentication

1. Use RADIUS authentication when you want to add two-factor authentication to the database.
2. Make sure to use a package that has a well-documented setup procedure for the challenge-response sequence.

## 6.7 Discussion: Protect Your Password Hashes

As you saw at the end of HOWTO 6.5 you must protect your password hashes in USER\$ through privileges and through auditing. With the password hashes, an attacker can get access to the plain-text passwords if they can sniff the network and you don't encrypt the network traffic. This type



**Figure 6.8** Setting RADIUS settings using netmgr.

of attack relies on the fact that the session keys are encrypted using the password hashes. With the password hashes, an attacker can know the session keys and thus the plaintext passwords. Password hashes can also be used in a password cracking attack. Because the password hashing algorithm is well known, an attacker can run a program that guesses passwords, computes the hash value, and compare it with your values. They can do this on their own database or even in a C program optimized for fast iterations. If you choose your passwords randomly, this may take a very long time but usually passwords are not random and a brute force attack can be effective. For example, one of the better programs, orabf, calculates over 1 million hashes per second. For passwords of length eight this can take a few days to complete—that's pretty scary when you think about it.

These are only two examples in which password hashes can be used to take over a database—and there are many more. The bottom line is that you need to think of password hashes as highly sensitive data that must be secured well. Alternatively, you may decide that this is an area that you don't want to own and you would prefer to delegate this—i.e., not use the database for authentication, not keep passwords within the database, and rely on an external authentication system as described in this chapter.

When you use database authentication you must secure your password hashes as though they were the passwords themselves. At a minimum you must do the following:

1. Use access control to ensure that access to the hashes (PASSWORD in SYS.USER\$ and DBA\_USERS prior to 11g and SPARE4 in SYS.USER\$ in 11g) is very limited.
2. If you use an advanced solution such as Oracle Database Vault then you can place better controls on this data (see Chapter 17).
3. Periodic audits to review privileges through which users may access these columns.
4. Continuous auditing and real-time alerting on any access to this data by any user.



## *Chapter 7*

---

# Encrypting Data-in-Transit

---

Most connections to an Oracle server are network connections which use Transmission Control Protocol/Internet Protocol (TCP/IP) and Ethernet as the underlying protocols. These connections are made up of network packets—TCP/IP packets packaged within Ethernet frames. The packets that run over the network can be categorized into three main groups:

1. Packets that form the initiation of the session and include the logon packets.
2. Packets that form the requests from the client to the server—including the Structured Query Language (SQL) statements/queries.
3. Packets that form the responses from the server to the client—including the result sets/replies to the queries.

When you finish a default installation of an Oracle database and an Oracle client, these communications run in cleartext over the network infrastructure. If you have sensitive data that is communicated in cleartext then you have a potential point of vulnerability as far as confidentiality of the data is concerned. You already saw in the Chapter 6 that passwords passed between the client and the server during the session initiation process (category 1 above) are not sent in the clear even when you don't encrypt at the network layer. The main point of vulnerability is therefore packets belonging to categories 2 and 3 above—i.e., your data-in-transit.

Let's look at two such examples. Both examples are based on sniffing network packets between a client and the server (we'll discuss network sniffers in the next subsection). In the first example the user SCOTT logs onto the database and changes their password:

```
% sqlplus scott@orclsol
SQL*Plus: Release 10.2.0.1.0 - Production on Wed Jan 30 08:59:06 2008
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - 64bit Production
With the Partitioning, OLAP and Data Mining options
SQL> alter user scott identified by NEWPASSWORD;
User altered.
```

```

0x0000: 4500 00fa efb7 4000 3c06 c82c c0a8 0274 E.....@.<...,...t
0x0010: c0a8 0255 8817 05f1 7bc8 a7fe 9e62 5e85 ...U....{....b^.
0x0020: 5018 4470 1f0d 0000 00d2 0000 0100 0000 P.Dp.....
0x0030: 0139 012c 0c01 0800 7fff 7f08 0000 0001 .9.,.....
0x0040: 0098 003a 0000 0200 4141 0000 0000 0000 .....AA.....
0x0050: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x0060: 0000 2844 4553 4352 4950 5449 4f4e 3d28 ..(DESCRIPTION=(
0x0070: 4144 4452 4553 533d 2850 524f 544f 434f ADDRESS=(PROToco
0x0080: 4c3d 5443 5029 2848 4f53 543d 7472 6578 L=TCP) (HOST=trex
0x0090: 2928 504f 5254 3d31 3532 3129 2928 434f ) (PORT=1521)) (CO
0x00a0: 4e4e 4543 545f 4441 5441 3d28 5345 5256 NNNECT_DATA=(SERV
0x00b0: 4943 455f 4e41 4d45 3d6f 6e30 7472 6578 ICE_NAME=on0trex
0x00c0: 3029 2843 4944 3d28 5052 4f47 5241 4d3d 0) (CID=(PROGRAM=
0x00d0: 7371 6c70 6c75 7340 6372 6f77 2928 484f sqlplus@crow) (HO
0x00e0: 5354 3d63 726f 7729 2855 5345 523d 6f72 ST=crow) (USER=or
0x00f0: 6163 6c65 3130 2929 2929 acle10))) )
...
0x0000: 4500 0114 efb0 4000 3c06 c80c c0a8 0274 E.....@.<.....t
0x0010: c0a8 0255 8817 05f1 7bc8 b293 9e62 6842 ...U....{....bhB
0x0020: 5018 4470 46f4 0000 00ec 0000 0600 0000 P.DpF.....
0x0030: 0000 0376 0200 0000 0110 25d9 1800 0000 ...v.....%....
0x0040: 0500 0000 010f ffff ffff ffaf 0800 0000 .....
0x0050: 050f ffff ffff ffaf 880f ffff ffff ffaf .....
0x0060: 7005 7363 6f74 7400 0000 0dd0 4155 5448 p.scott....AUTH
0x0070: 5f54 4552 4d49 4e41 4c00 0000 0505 7074 _TERMINAL....pt
0x0080: 732f 3000 0000 0000 0000 0f0f 4155 5448 s/0....AUTH
0x0090: 5f50 524f 4752 414d 5f4e 4d00 0000 1818 _PROGRAM_NM.....
0x00a0: 7371 6c70 6c75 7340 6372 6f77 2028 544e sqlplus@crow.(TN
0x00b0: 5320 5631 2d56 3329 0000 0000 0000 000c S.V1-V3) .....
0x00c0: 0c41 5554 485f 4d41 4348 494e 4500 0000 .AUTH_MACHINE...
0x00d0: 0404 6372 6f77 0000 0000 0000 0008 0841 ..crow.....A
0x00e0: 5554 485f 5049 4400 0000 0606 3434 3235 UTH_PID.....4425
0x00f0: 3838 0000 0000 0000 0008 0841 5554 485f 88.....AUTH_
0x0100: 5349 4400 0000 0808 6f72 6163 6c65 3130 SID.....oracle10
0x0110: 0000 0000 .....
...

```

---

**Listing 7.1 Selected packets from a session where a password is changed.**


---

Listing 7.1 shows a few packets from this session. The listing is not a full packet dump—the Oracle\*Net protocol is quite verbose and for the sake of understanding the problem you don't need to see all packets. The first packet shown in Listing 7.1 is the client initiating the connection. The client sends all sorts of information that may in itself be sensitive—the service name, the OS user from which the client is connecting from etc. But—none of this is too sensitive. As part of the session initiation and logon, usernames, and session keys are sent (the second and third packets in Listing 7.1). The passwords are not sent in the clear—so this is OK. The problem is in the fourth packet in Listing 7.1. Scott changed his password using an ALTER USER command and this SQL statement is sent to the server as is—password and all. The attacker sniffing the network can now logon to the SCOTT account with the new password.

In Chapter 4 when you learned about account security, you saw that there are two ways to change the password—using ALTER USER or using the PASSWORD command. From a functional perspective there are minor differences but from a security perspective it is far better to use

```

0x0000: 4500 02c4 efbf 4000 3c06 c65a c0a8 0274 E.....@.<...Z...
0x0010: c0a8 0255 8817 05f1 7bc8 b37f 9e62 6905 ...U....{....bi.
0x0020: 5018 4470 6080 0000 029c 0000 0600 0000 P.Dp`.....
0x0030: 0000 0373 0300 0000 0110 25d9 1800 0000 ...s.....%....
0x0040: 0500 0001 010f ffff ffff ffb7 4800 0000 .....H....
0x0050: 0d0f ffff ffff ffc1 580f ffff ffff ffad .....X.....
0x0060: f005 7363 6f74 7400 0000 0c0c 4155 5448 ..scott....AUTH
0x0070: 5f53 4553 534b 4559 0000 0040 4044 3838 _SESSKEY...@@D88
0x0080: 4438 3339 4436 4545 3436 3042 4230 3134 D839D6EE460BB014
0x0090: 3930 3341 4135 3445 3439 3233 3935 4232 903AA54E492395B2
0x00a0: 4241 4442 3243 4638 3145 3734 3742 4331 BADB2CF81E747BC1
0x00b0: 4434 4334 4432 4634 3830 4543 3200 0000 D4C4D2F480EC2...
0x00c0: 0100 0000 0d0d 4155 5448 5f50 4153 5357 .....AUTH_PASSW
0x00d0: 4f52 4400 0000 4040 4536 4244 4435 4646 ORD...@@E6EDD5FF
0x00e0: 3941 3836 4235 3630 3237 3837 4236 3946 9A86B5602787B69F
0x00f0: 3737 3132 4632 3145 4430 3641 3336 4344 7712F21ED06A36CD
0x0100: 3442 3041 3231 4343 4344 4243 4544 4545 4B0A21CCCDDBCEDEE
0x0110: 3330 3839 3434 3738 0000 0000 0000 0008 30894478.....
...
0x0000: 4500 0124 f013 4000 3c06 c7a6 c0a8 0274 E..$..@.<.....t
0x0010: c0a8 0255 8817 05f1 7bc8 bede 9e62 76c6 ...U....{....bv.
0x0020: 5018 4470 d6c9 0000 00fc 0000 0600 0000 P.Dp.....
0x0030: 0000 035e 1b00 0080 2100 0000 0000 0000 ...^....!.....
0x0040: 0110 1f24 6000 0000 2a00 0000 0110 1cdc ...$`....*.....
0x0050: 1c00 0000 0d00 0000 0000 0000 0000 0000 .....P.....
0x0060: 0110 1cdc 5000 0000 0000 0000 0100 0000 .....P.....
0x0070: 0000 0000 0000 0000 0000 0000 0000 0000 .....P.....
0x0080: 0000 0000 0000 0000 0000 0000 0000 0000 .....P.....
0x0090: 0000 0000 0000 0000 0000 0000 0000 0000 .....P.....
0x00a0: 0110 1cdc 5200 0000 0110 1efb 1800 0000 ....R.....
0x00b0: 0000 0000 0000 0000 0000 0000 0000 0000 .....R.....
0x00c0: 0110 1cdc 682a 616c 7465 7220 7573 6572 ....h*alter.user
0x00d0: 2073 636f 7474 2069 6465 6e74 6966 6965 .scott.identifie
0x00e0: 6420 6279 204e 4557 5041 5353 574f 5244 d.by.NEWPASSWORD
0x00f0: 0000 0001 0000 0001 0000 0000 0000 0000 .....P.....
0x0100: 0000 0000 0000 0000 0000 0000 0000 0007 .....P.....
0x0110: 0000 0000 0000 0000 0000 0000 0000 0000 .....P.....
0x0120: 0000 0000 .....P.....

```

---

**Listing 7.1 (continued)**

PASSWORD. The reason is—again—related to network sniffing. You see in Listing 7.1 that when using ALTER USER the password is sent in the clear; when you use PASSWORD—it is not! For example, if you run:

```

SQL> password
Changing password for SCOTT
Old password:
New password:
Retype new password:
Password changed

```

The packet for the password change only contains the hashed password—not the cleartext password:

0	x0000:	4500	0221	f14b	4000	3c06	c571	c0a8	0274	E...!.K@.<....q....t
0	x0010:	c0a8	0255	8818	05f1	92fd	49d9	c8dc	c63f	...U.....I.....?
0	x0020:	5018	4470	722c	0000	01f9	0000	0600	0000	P.Dpr,.....
0	x0030:	0000	116b	1d00	0001	1500	0002	9400	0000	..k.....
0	x0040:	0103	731e	0000	0001	102c	5510	0000	0005	..s.....,U.....
0	x0050:	0000	0112	0fff	ffff	ffff	c480	0000	0008	.....
0	x0060:	0fff	ffff	ffff	cc90	0fff	ffff	ffff	bdf0	.....
0	x0070:	0553	434f	5454	0000	000d	0d41	5554	485f	.SCOTT.....AUTH_
0	x0080:	5041	5353	574f	5244	0000	0040	4034	3435	PASSWORD...@@445
0	x0090:	4344	3742	4344	4239	3137	3245	3636	4442	CD7BCDB9172E66DB
0	x00a0:	3441	4343	3039	3344	3342	4233	4644	3834	4ACC093D3BB3FD84
0	x00b0:	4330	4439	3537	3530	3037	3341	3133	3238	C0D95750073A1328
0	x00c0:	3239	3545	3841	4436	4537	4339	3300	0000	295E8AD6E7C93....
0	x00d0:	0000	0000	1010	4155	5448	5f4e	4557	5041	.....AUTH_NEWPA
0	x00e0:	5353	574f	5244	0000	0040	4031	3436	3843	SSWORD....@@1468C
0	x00f0:	3344	3944	3944	3045	4246	3532	4335	3746	3D9D9D0EBF52C57F
0	x0100:	3745	3038	3535	3231	3731	3136	4531	3632	7E0855217116E162
0	x0110:	3939	3732	4234	4131	4145	3532	3846	3933	9972B4A1AE528F93
0	x0120:	3444	3534	3043	3242	3937	4300	0000	0000	4D540C2B97C.....
0	x0130:	0000	0d0d	4155	5448	5f54	4552	4d49	4e41	....AUTH_TERMINA
0	x0140:	4c00	0000	0505	7074	732f	3000	0000	0000	L.....pts/0.....
0	x0150:	0000	0f0f	4155	5448	5f50	524f	4752	414d	....AUTH_PROGRAM
0	x0160:	5f4e	4d00	0000	1818	7371	6c70	6c75	7340	_NM.....sqlplus@
0	x0170:	6372	6f77	2028	544e	5320	5631	2d56	3329	crow.(TNS.V1-V3)
0	x0180:	0000	0000	000c	0c41	5554	485f	4d41	.....AUTH_MA	
0	x0190:	4348	494e	4500	0000	0404	6372	6f77	0000	CHINE.....crow..
0	x01a0:	0000	0008	0841	5554	485f	5049	4400	.....AUTH_PID.	
0	x01b0:	0000	0606	3434	3635	3230	0000	0000	0000	....446520.....
0	x01c0:	0008	0841	5554	485f	5349	4400	0000	0808	....AUTH_SID.....
0	x01d0:	6f72	6163	6c65	3130	0000	0000	0000	0012	oracle10.....
0	x01e0:	1241	5554	485f	414c	5445	525f	5345	5353	.AUTH_ALTER_SESS
0	x01f0:	494f	4e00	0000	2525	414c	5445	5220	5345	ION...%ALTER.SE
0	x0200:	5353	494f	4e20	5345	5420	5449	4d45	5f5a	SSION.SET.TIME_Z
0	x0210:	4f4e	453d	272d	3036	3a30	3027	0000	0000	ONE='06:00'.....
	0x0220:	01								

This is certainly far more secure and you should always use PASSWORD for changing your password rather than ALTER USER. Many companies have a policy by which passwords can only be changed when logged onto the host directly (e.g., through ssh which encrypts everything) and where the database connection is local. This is a good policy; if you don't do this already you should consider adopting it.

In the second example let's assume that the SCOTT schema has a table called CREDIT\_CARD that has card holder information:

```
SQL> select * from CREDIT_CARD;
CC_NUMBER F      IRST_NAME      LAST_NAME      ADDRESS      EXPIR      SEC_C
-----  -----  -----  -----  -----  -----  -----
1111111111111111 r    on        ben       usa   01/08  555
2222222222222222 r    on        ben       usa   01/09  444
```

If the attacker is sniffing or intercepting the packets when SCOTT runs this select statement they will be getting the card holder information. Listing 7.2 shows the sample packets for the query and the response where you can clearly see credit card information.

The risk in both of these examples is that an attacker can get your sensitive data without even having access to the database. The attacker does not need a user in the database and never even tries

```

0x0000: 4500 0113 27b0 4000 4006 8c1b c0a8 0255 E...!..@.@@.....U
0x0010: c0a8 0274 bf42 05f1 e82c 90fd dddb a6e0 ...t.B...,.....
0x0020: 5018 c141 b862 0000 00eb 0000 0600 0000 P..A.b.....
0x0030: 0000 035e 2b00 0080 6100 0000 0000 0000 ...^+...a.....
0x0040: 0100 1fc9 0000 0000 1900 0000 0100 1db6 .....
0x0050: 3400 0000 0d00 0000 0000 0000 0000 0000 4.....
0x0060: 0100 1db6 6800 0000 0000 0000 0100 0000 ....h.....
0x0070: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x0080: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x0090: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x00a0: 0100 1db6 6a00 0000 0100 1fa2 d000 0000 ....j.....
0x00b0: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x00c0: 0100 1db6 8019 7365 6c65 6374 202a 2066 ..... select.*.f
0x00d0: 726f 6d20 4352 4544 4954 5f43 4152 4400 rom.CREDIT CARD.
0x00e0: 0000 0100 0000 0000 0000 0000 0000 0000 ..... .
0x00f0: 0000 0000 0000 0000 0000 0000 0000 0100 ..... .
0x0100: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x0110: 0000 00 ..... .
...
0x0000: 4500 025f f8e4 4000 3c06 bd9a c0a8 0274 E.._.@.<.....t
0x0010: c0a8 0255 05f1 bf42 dddb a6e0 e82c 91e8 ...U...B....,..
0x0020: 5018 4470 a000 0000 0237 0000 0600 0000 P.Dp....7.....
0x0030: 0000 1017 0251 a527 793b d309 5577 e10b ....Q.'y;;.Uw..
0x0040: b2b7 2f94 786c 011e 0f30 1100 0000 7e00 .../.x1...0....~.
0x0050: 0000 064d 0180 0000 0000 0010 0000 0000 ...M.....
0x0060: 0000 0000 0000 0000 0000 001f 0100 0000 ..... .
0x0070: 1001 0900 0000 0909 4343 5f4e 554d 4245 .....CC_NUMBE
0x0080: 5200 0000 0000 0000 0000 0001 8000 0000 R.....
0x0090: 0000 1400 0000 0000 0000 0000 0000 0000 ..... .
0x00a0: 0000 1f01 0000 0014 010a 0000 000a 0a46 .....F
0x00b0: 4952 5354 5f4e 414d 4500 0000 0000 0000 IRST_NAME.....
0x00c0: 0000 0101 8000 0000 0000 2800 0000 0000 .....(.....
0x00d0: 0000 0000 0000 0000 0000 1f01 0000 0028 .....(.....
0x00e0: 0109 0000 0009 094c 4153 545f 4e41 4d45 .....LAST_NAME
0x00f0: 0000 0000 0000 0000 0002 0180 0000 0000 ..... .
0x0100: 0028 0000 0000 0000 0000 0000 0000 0000 ..(.....
0x0110: 001f 0100 0000 2801 0700 0000 0707 4144 .....(.....AD
0x0120: 4452 4553 5300 0000 0000 0000 0000 0301 DRESS.....
0x0130: 8000 0000 0000 0500 0000 0000 0000 0000 ..... .
0x0140: 0000 0000 0000 1f01 0000 0005 010a 0000 ..... .
0x0150: 000a 0a45 5850 4952 4154 494f 4e00 0000 ...EXPIRATION...
0x0160: 0000 0000 0000 0401 8000 0000 0000 0500 ..... .
0x0170: 0000 0000 0000 0000 0000 0000 0000 1f01 ..... .
0x0180: 0000 0005 0108 0000 0008 0853 4543 5f43 .....SEC_C
0x0190: 4f44 4500 0000 0000 0000 0000 0500 0000 ODE.....
0x01a0: 0707 786c 011e 0f35 2900 0000 0100 001f ..x1...5).....
0x01b0: e800 0000 0a00 0000 0a06 0a00 0600 0000 ..... .
0x01c0: 0000 0000 0100 0000 0000 0000 0000 0007 ..... .
0x01d0: 292c 0106 0f32 3232 3232 3232 3232 3232 )...222222222222
0x01e0: 3232 3232 0372 6f6e 0362 656e 0375 7361 2222.ron.ben.usa
0x01f0: 0530 312f 3039 0334 3434 0800 0608 2921 .01/09.444....)! ..... .
0x0200: f300 0000 0000 0000 0100 0000 0100 0000 ..... .

```

**Listing 7.2 Sensitive data in the clear.**

(continued)

```

0x0210: 0000 0000 0000 0000 0000 0000 0004 0000 ..... .
0x0220: 0002 0029 0000 0001 0000 0005 0000 0001 ...) ..
0x0230: 0000 0300 2000 0000 0008 d178 0004 0000 ..... x....
0x0240: 0012 3600 0100 0000 0000 2b00 0000 0000 ..6.....+.....
0x0250: 0100 0000 0000 0000 0000 0000 0000 00 ..... .

0x0000: 4500 00cf f8e5 4000 3c06 bf29 c0a8 0274 E.....@.<...)...t
0x0010: c0a8 0255 05f1 bf42 dddb a917 e82c 91fd ...U...B.....,..
0x0020: 5018 4470 cef0 0000 00a7 0000 0600 0000 P.Dp..... .
0x0030: 0000 060a 0006 0000 0000 0000 000f 0000 ..... .
0x0040: 0000 0000 0000 0000 0729 2c01 060f 3131 .....),...11
0x0050: 3131 3131 3131 3131 3131 3131 3103 726f 111111111111.ro
0x0060: 6e03 6265 6e03 7573 6105 3031 2f30 3803 n.ben.usa.01/08.
0x0070: 3535 3504 0000 0002 002a 0000 0002 057b 555.....*.....{
0x0080: 0005 0000 0001 0000 0300 2000 0000 0008 ..... .
0x0090: d178 0004 0000 0012 3600 0100 0000 0000 ..x.....6..... .
0x00a0: 2c00 0000 0000 0100 0000 0000 0000 0000 ,..... .
0x00b0: 0000 0000 0019 4f52 412d 3031 3430 333a .....ORA-01403:
0x00c0: 206e 6f20 6461 7461 2066 6f75 6e64 0a .no.data.found.

```

**Listing 7.2 (continued)**

to logon to the database. They use network techniques to intercept or sniff these packets. Before moving on to discuss how to secure data-in-transit, let's spend a bit of time to understand what techniques are used to intercept or sniff data at the network level and whether this is easily done or is it a farfetched scenario.

### Network Sniffing

You sniff networks by using a network sniffer. A network sniffer is a system that can read packets and show you their contents. Network sniffers can come in the form of a software package or a hardware/software combination. Network sniffers are used by network administrators to troubleshoot and resolve networking issues, but they can be used by an attacker to gain access to data.

There are many network sniffers which are readily available and most are even free of charge. In terms of software packages, a few very common ones include Ethereal, Wireshark, Windump, tcpdump, etc. Ethereal is the world's most popular network protocol analyzer (people can't really use the word sniffer in a product name because it is a trademark). You can download it from [www.ethereal.com](http://www.ethereal.com) and it is open source. It is a great product that allows you to inspect network packets very easily. Ethereal needs a packet capture library to work—it does not do the actual "sniffing" itself. On Unix it uses the pcap library and on Windows it installs Winpcap. Winpcap is distributed by [www.winpcap.org](http://www.winpcap.org) where you can download the library itself or an application called Windump which allows you to dump packets and view them on Windows. On Unix, the pcap library comes with almost every operating system and is used by the tcpdump utility that gives you access to the packet data. For example, to see packet data in all its glory between host db1 and host cl1 do:

```
# tcpdump -i eth1 -s0 -X host cl1 and host db1
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
16:59:37.475496 arp who-has db1.dbasecur.com (Broadcast) tell cl1.dbasecur.com
```

```

0  x0000: 0001 0800 0604 0001 0800 20b8 a801 c0a8 ..... .
0  x0010: 0255 ffff ffff c0a8 0274 5555 5555 .U.....tUUUUU
0  x0020: 5555 5555 5555 5555 5555 5555 UUUUUUUUUUUUUUUU
16:59:37.475620 arp reply db1.dbasecur.com is-at 00:06:29:04:7b:ba (oui Unknown)
0  x0000: 0001 0800 0604 0002 0006 2904 7bba c0a8 .....{...
0  x0010: 0274 0800 20b8 a801 c0a8 0255 5555 5555 .t.....UUUUU
0  x0020: 5555 5555 5555 5555 5555 5555 UUUUUUUUUUUUUUUU
...

```

Different operating systems have different such programs—e.g., snoop on Solaris and netstat on HPUX—but they all do pretty much the same thing and operate in pretty much the same way.

Which brings us to the very important question of where you would need to sniff to see database packets? When you use Ethereal or tcpdump or any of these programs, the first thing you need to do is attach to a network interface from which you read the packets. In the tcpdump command above the first parameter is `-i eth1`. This tells pcap to attach to the eth1 interface to get the packets. The same goes for any packet capture program. Figure 7.1 shows an example in which you pick which interface to sniff using Ethereal. The important piece of understanding you need to take away from this is that the packets that you want to sniff must be on the network interface from which you will be running the packet capture program. If it is not—you will not see these packets.

So the question you have to answer when determining whether or not you are at risk from an attacker sniffing your packets is whether an attacker has access to a network interface where these packets exist. Not every packet ends up on every network interface—if it did, we would have very limited bandwidth. Every interface has a unique identifier (e.g., every Ethernet card has a message authentication code (MAC) address) and most interfaces have an IP address. When you send a packet to a certain IP address then only the interface with that IP address will get it (well—not exactly—we'll talk more about that). So which interfaces can packets arrive at when you send it from IP1 to IP2?

Two places are obvious—the machine which sends the packet and the machine which gets the packet. In our world that simply means that anyone who has access to the host on which the database runs, and has privileges that allows them to connect to pcap and run tcpdump (or some other such program), can get all the packets sent to the database server. The second place is on the machine from which the database client connects from. If an attacker could only sniff packets when logged on to one of these two hosts then life would be fairly simple. But obviously there are additional places from which you can capture packets. After all, network administrators troubleshoot networking issues by attaching to network gear—they don't have to go to the hosts

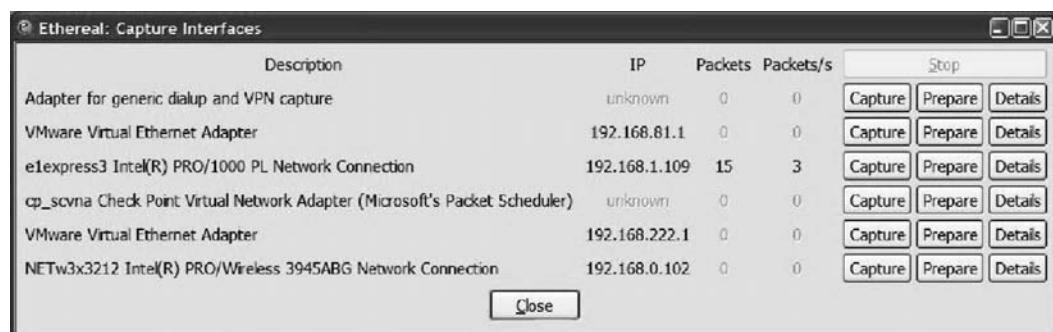


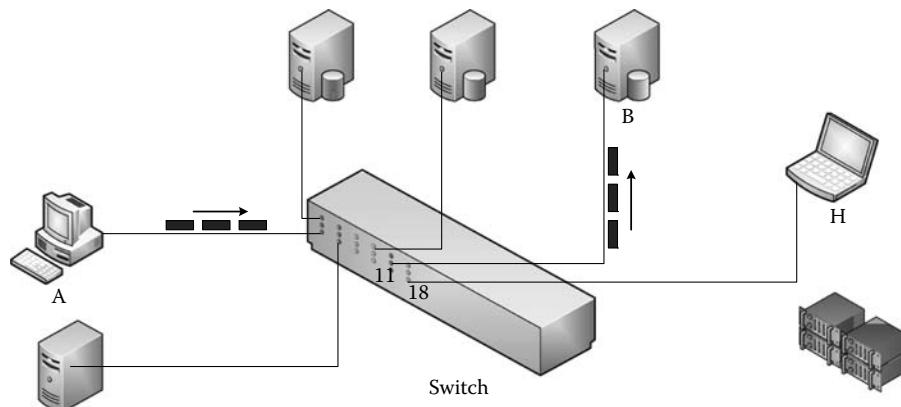
Figure 7.1 Picking an interface from which to capture packets in Ethereal.

all the time. To understand how to capture packets without attaching to the interfaces on the hosts themselves we have to talk a bit about hubs and switches.

### Sniffing and Intercepting in Switched Networks

Way back when Ethernet and TCP/IP were making their first steps we had coaxial cables and we connected multiple workstations to a single cable. This single network segment served all workstations at the same time using Ethernet Carrier Sense Multiple Access with Collision Detection (CSMA/CD). When one workstation wants to send data on a coaxial cable there is no problem. But if two workstations try to send frames at the same time a collision occurs and data is rendered useless. To lessen the problem, CSMA/CD forces workstations to “listen” to the Ethernet before sending to ensure that no other workstation is sending at the same time. This does not limit collisions completely. You still can have two workstations listening at the same time, seeing that the Ethernet is free, and then sending at the same time. But it does limit collisions substantially. Moreover, collisions are directly correlated to the number of workstations on that segment. When hubs came into being—they made connectivity much simpler and cheaper and we did away with coaxial cables. But the basic networking pattern stayed the same—a hub is a single networking segment where all workstations can hear all communications on the segment. This means that if your database is connected to a hub then all an attacker needs to do is connect their machine to the same hub and then they can sniff all your database traffic (within a single network segment, you can think of each frame as being broadcast on that segment).

Luckily, we don’t use hubs anymore in the data center. Hubs simply don’t work for anything but the smallest environments and very low bandwidth requirements. The collisions are the problem. Within a single segment only one machine can be transmitting at any point in time—this simply does not scale. Instead, we use switches today. Even when you have a small network at home you most probably use a switch—you can buy a simple switch to put at home for around \$30. The reason you use switches is that they do away with the single segment model. Instead, they allow each workstation connected to the switch to live on its own virtual segment. This means that when an Ethernet frame needs to be delivered from workstation A to workstation B (Figure 7.2)



**Figure 7.2** Switched network environment.

the frame will go from workstation A to the port where workstation A is connected to the switch, the switch will examine the frame and determine which workstation it needs to go to, it will copy the frame to the port where workstation B is connected and the frame will go to workstation B. At no point will this Ethernet frame ever appear on any of the other ports! So an attacker who is connected to the network will not get frames and packets unless they are intended for the attacker's host and the attacker will not be able to sniff the traffic. There is one exception to this rule. Switches have special ports called SPAN ports or port-mirrors which are used for administrative and troubleshooting purposes. You can configure a switch to mirror all traffic that comes through a certain port to the SPAN port (e.g., all traffic coming and going to your database server). An administrator can then connect a network sniffer to the SPAN port and see all database traffic. Like system administrators that can run tcpdump on the database server, these privileged users are not necessarily your problem but you should be aware of this option and ensure you have the right controls and procedures around the use of SPAN ports and port mirrors.

So it seems like you're safe—right? You should know better by now. The pitfall is the sentence “will not get frames unless they are intended for the attacker's host.” Indeed—switches provide more “privacy” in terms of network communications—but only if everyone plays fair. Switches do their job well—but they can be fooled. For example, if a hacker can fool the switch into believing that their machine (H in Figure 7.2) is really the database server, the switch will unknowingly send the packets intended for the database server B to H. The hacker will then be able to inspect the packets, but must also forward the packets on their way to the real database server (or the communication will not go on). And so while this is an interception attack and not a sniffing attack the result is the same—your data ends up in the hands of the attacker.

There are many methods an attacker has to get packets even in a switched environment. The main methods that are relatively simple are

- Address Resolution Protocol (ARP) Spoofing—To understand ARP spoofing you need to first understand how the mapping between IP addresses and MAC addresses happen. When host A wants to send data to host B it knows what its IP address is. But Ethernet communications through a switch does not use IP addresses—it uses MAC addresses. Host A's network cards put Ethernet frames on the wire—and they need the MAC address for host B to do that. The translation is part of the ARP. When host A wants to send something to host B it sends an ARP request of the form “Who has B?” Because this is a discovery question, the switch will broadcast this request on ALL ports. Normally, host B would reply with “I am B—send it to me—here is my MAC address.” Host A will then record this and send frames to this MAC address. Here's an example of this interchange:

```

0 9:10:10.011623 arp who-has dbl1.dbasecur.com (Broadcast) tell cl1.dbasecur.com
0 x0000: 0001 0800 0604 0001 0800 20b8 a801 c0a8 ..... .
0 x0010: 0255 ffff ffff c0a8 0274 5555 5555 .U.....tUUUUU
0 x0020: 5555 5555 5555 5555 5555 5555 5555 UUUUUUUUUUUUUUUU
09:10:10.011721 arp reply dbl1.dbasecur.com is-at 00:06:29:04:7b:ba (oui Unknown)
0 x0000: 0001 0800 0604 0002 0006 2904 7bba c0a8 ..... ).{...
0 x0010: 0274 0800 20b8 a801 c0a8 0255 5555 5555 .t.....UUUUU
0 x0020: 5555 5555 5555 5555 5555 5555 5555 UUUUUUUUUUUUUUUU

```

However, if host H (the attacker) is fast enough and can answer before B does, it can reply to the ARP request with its own MAC address. Host A records this mapping and from

now on all communications that go to the database server go to host H. Host H in turn will inspect the data and forward the frames to the real MAC address of the database server. Examples of tools that perform such interception and sniffing are parasite, dsniff, switchsnarf, bigboss etc.

- **MAC Flooding**—Switches are responsible for setting up virtual circuits from one host to another. Switches keep a translation table that tracks which MAC address is on which physical port. When host B replies and says “here is my MAC address” the switch puts this in a table so that when a frame comes in it can very quickly know which port to copy the frame to. The amount of memory for this translation table is limited. On some switches, it is possible to bombard the switch with a very large number of bogus MAC addresses. If the table fills up the switch will prefer to copy too many frames than risk frames that never reach their destinations. It will revert to a hub and will start broadcasting all frames to all ports. At this point, a simple network sniffer will work.
- **MAC Duplication**—MAC duplication is somewhat related to ARP spoofing but the attack is not based on fooling the sending host but rather fooling the switch. On various operating systems you can change the MAC address of a network card. At this point, even without intercepting the traffic going to the database, the switch will think that it has two ports with the same MAC address. When it gets a frame that goes to this MAC address it will put it on both ports—and the attacker will see your data.

Given that all these techniques exist, how do you protect yourself from these kinds of attacks? Some of the methods to protect against such attacks are functions which exist within switches. For example, most switches have port security features that, when enabled, protect you against MAC flooding and MAC spoofing attacks. This feature prevents the switch from recognizing more than one MAC address on a physical port. Switches also have IP filtering capabilities—so your network administrator can ensure that traffic going to the database server’s IP can only go out of a specific physical port. These are effective methods but they require a lot of setup and maintenance. Other methods involve using an Intrusion Detection System (IDS) to detect an attack. But the simplest approach is to encrypt the network packets as they leave the client and decrypt them when they arrive at the server (and vice versa). Any attacker that gets these packets (through sniffing or interception) will find that they are useless because the attacker does not have access to the encryption keys—the data is simply not readable.

In this chapter you’ll learn about methods for encrypting data at the network layer to ensure that sensitive data does not fall into the wrong hands through the use of network techniques. One quick note before we delve into the first HOWTO. Network attacks can be very sophisticated and the techniques used are almost always unknown to people who manage databases. However, these techniques are very well known to people within the networking groups and within the security groups. More importantly, an attack at the network level on a database server is often very similar to a network attack on an application server or a Web server and there are many security systems that look for such attacks and help combat them. For example, Network Intrusion Detection Systems (NIDS) and Instruction Prevention Systems (IPS) specifically address attacks such as ARP cache poisoning and others that were mentioned previously. As mentioned in the first chapter of this book, good security comes through defense-in-depth and you need multiple security layers for effective data security. This is a perfect example of this principle—you may not have to solve everything by database configuration and there are other security systems and security groups that will help you limit the type of attacks you have to deal with.

## 7.1 HOWTO Configure Network Encryption Using ASO

If you want to encrypt network traffic between the Oracle clients and the Oracle server you can use the data encryption feature within Oracle Advanced Security Options (ASO). There are two offerings within ASO that encrypt the data-in-transit—one offering that we'll call Network Data Encryption (NDE) and one using Secure Socket Layer (SSL). You already saw how to configure network encryption using SSL in Section 6.4—just enable connections using the TCPS protocol and your data will be encrypted using SSL. SSL provides you more than just encryption—it allows you to do certificate-based authentication.

SSL is seldom used when you want encryption for data-in-transit. The reason is that SSL requires quite a bit of setup whereas NDE requires almost no setup whatsoever. NDE uses the Diffie-Hellman scheme to negotiate a shared symmetric key for the session (see Chapter 5). The encryption strengths provided by NDE are not weaker (if anything—they are stronger). Therefore, it is better, stronger, faster—so if you just want to encrypt the network communications, use NDE.

When you setup NDE you need to tell Oracle very few things. You need to tell it what algorithms to use and how to behave when other parties want to encrypt or not encrypt the communications with you. There is no wallet that needs to be created, no certificates, no new listener entries, etc. The sequence to follow to enable NDE is

Step 1: Install ASO on both the server and the client.

Step 2: On both the client and the server edit sqlnet.ora and list encryption algorithms that you want to use. As an example, in each of the server's and clients's file add the appropriate line:

```
SQLNET.ENCRYPTION_TYPES_SERVER= (AES192,3DES168,3DES112)
SQLNET.ENCRYPTION_TYPES_CLIENT= (3DES168)
```

The list of algorithms tells Oracle which encryption method you are willing to use for the communications. For the communications to succeed the server and the client must agree on an algorithm. Therefore, there has to be a nonempty intersection between the list the client can talk with and the list that the server can talk with. It is the server that determines which algorithm is used and the selection is made by going through the algorithms from left to right and finding the first algorithm that the client can also use. In the example above, encrypted communications between this client and this server will use Triple Data Encryption Standard (3DES) with a key length of 168 bits. Table 7.1 lists the algorithms supported by NDE and the type code.

Step 3: Edit sqlnet.ora and set the parameter that affects the negotiation. This parameter is defined on both the server and the client and affects whether the session will be running encrypted or unencrypted. As an example

```
SQLNET.ENCRYPTION_SERVER = requested
SQLNET.ENCRYPTION_CLIENT = requested
```

There are four possible values for this parameter with the following meaning:

1. REQUIRED—Meaning that encryption is required and Oracle should not allow unencrypted communications. If the other side does not allow encrypted sessions don't set up the connection.
2. REQUESTED—Meaning that encryption should be used unless the other side does not allow it.

**Table 7.1 ENCRYPTION\_TYPES\_\* Code per Supported Algorithm**

Algorithm	ENCRYPTION_TYPES_* Code	Available for JDBC Type-4 Driver		
		10g R1 and Prior	10g R2	11g
RC4 256-bit key	RC4_256	Yes	Yes	Yes
RC4 128-bit key	RC4_128	Yes	Yes	Yes
RC4 56-bit key	RC4_56	Yes	Yes	Yes
RC4 40-bit key	RC4_40	Yes	Yes	Yes
AES 256-bit key	AES256			Yes
AES 192-bit key	AES192			Yes
AES 128-bit key	AES128			Yes
3-key 3DES	3DES168		Yes	Yes
2-key 3DES	3DES112		Yes	Yes
DES 56-bit key	DES	Yes	Yes	Yes
DES 40-bit key	DES40	Yes	Yes	Yes

3. ACCEPTED—Meaning that encryption will be used only if the other side prefers it, but if the other side is oblivious to encryption then communications will not be encrypted.
4. REJECTED—Meaning that Oracle should not allow encrypted communication even if the other side wants it encrypted. If the other side demands it then the connection will fail.

If you set REQUIRED on the server it means that you will never talk in cleartext—either the connection with the client will be encrypted or the connection will fail. If you set REQUESTED you are saying that you want the traffic to be encrypted. If a client tries to connect to you but it absolutely does not want encryption turned on (i.e., it has REJECTED as its setting) then you will agree to talk to it without encrypting the traffic, but otherwise all client communication is encrypted. If you set ACCEPTED you are saying that you’re willing to talk encrypted if the other side requests or requires it, but you’re not the one that is causing the encryption. And finally if you set REJECTED you’re saying that you do not want to encrypt ever. Table 7.2 shows all the combinations and the resulting encryption state of the connection.

#### Two Things to Remember about Configuring Network Encryption

1. NDE is much simpler than SSL—always prefer to use it unless you also want certificate-based authentication.
2. Prefer to use AES algorithms.

**Table 7.2 Connection State Based on SQLNET.ENCRYPTION\_SERVER and SQLNET.ENCRYPTION\_CLIENT**

<i>SQLNET.ENCRYPTION_CLIENT</i>	<i>SQLNET.ENCRYPTION_SERVER</i>	<i>Connection State</i>
REJECTED	REJECTED	Not Encrypted
ACCEPTED	REJECTED	Not Encrypted
REQUESTED	REJECTED	Not Encrypted
REQUIRED	REJECTED	Connection fails
REJECTED	ACCEPTED	Not Encrypted
ACCEPTED	ACCEPTED	Not Encrypted
REQUESTED	ACCEPTED	Encrypted
REQUIRED	ACCEPTED	Encrypted
REJECTED	REQUESTED	Not Encrypted
ACCEPTED	REQUESTED	Encrypted
REQUESTED	REQUESTED	Encrypted
REQUIRED	REQUESTED	Encrypted
REJECTED	REQUIRED	Connection fails
ACCEPTED	REQUIRED	Encrypted
REQUESTED	REQUIRED	Encrypted
REQUIRED	REQUIRED	Encrypted

## 7.2 HOWTO Configure Network Encryption for JDBC Drivers

There are two types of Java Database Connectivity (JDBC) drivers that you can use to connect to an Oracle server. The Type-2 JDBC driver is a Java wrapper for the Oracle Call Interface (OCI) libraries using Java Native Interface (JNI). As such, all communications use OCI which uses the sqlnet.ora settings. Therefore, if you use the Type-2 driver, setup for network encryption is identical to the previous HOWTO.

The Type-4 JDBC driver writes Oracle\*Net packets directly to the network—all within Java. You do not configure encryption for such encryption though sqlnet.ora. Instead, you use Java Properties objects to set attributes that affect the communications. As an example, Listing 7.3 shows a code snippet for a Type-4 JDBC connection where the client requires encryption using RC4 with 256 bits as the encryption algorithm. The parameters that you set up are the same—the difference is only that you set them in the Java code and not in sqlnet.ora. Also, depending on which version of Oracle you are using, you may have fewer encryption algorithms to choose from; only the algorithms with Yes in the last columns in Table 7.1 can be used for Type-4 drivers. You should never use RC4 or DES with 40 or 56 bit key length – so if you’re running a very old version of Oracle you’re left with RC4

```

Properties prop = new Properties();
prop.setProperty("user", "scott");
prop.setProperty("password", "tiger");
prop.setProperty("oracle.net.encryption_client", "REQUIRED");
prop.setProperty("oracle.net.encryption_types_client", "(RC4_256)");
Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@dbsrv:1521:orcl_aso", prop);

```

---

**Listing 7.3 Enabling encryption in a Type-4 JDBC driver connection.**

with 128 or 256 bit key length to choose from. Make sure you use the latest JDBC drivers – some of the older 10g JDBC drivers will have communications problems when using 3DES168.

Note: If you're setting up ASO NDE on a server that may need to accept connections from older JDBC clients make sure to include RC4\_256 and/or RC4\_128 to SQLNET.ENCRYPTION\_TYPES\_SERVER. Without this, Type-4 clients may never be able to set up encrypted communications with the server.

## 7.3 HOWTO Configure Data Integrity Using ASO

ASO data integrity guarantees that the contents of a request or response were not altered while in transit. It prevents attacks that may not require the attacker to fully understand the packets even when these are encrypted but still have an impact in terms of the result in the database. The use of cryptographic methods and one-way hash functions for ensuring data integrity in-transit was described in Chapter 5. Briefly, data integrity algorithms help you withstand two types of attacks:

- Data modification attacks in which an attacker alters data within the packets (e.g., using ARP spoofing) and sends the altered message to the database. As an example, if the SQL request had an update statement for crediting \$10 to an attacker's account, the packet may be modified to credit \$10,000 instead. As another example the WHERE clause specifying which account should be credited can be changed to transfer funds to an attacker's account.
- Replay attacks in which an attacker captures a sequence that credits their account and then replays that sequence over and over again.

Setting up data integrity checking in ASO is very simple and follows the pattern described in HOWTO 7.1. You just need to edit sqlnet.ora and set up which hashing algorithm to use and how the negotiation takes place:

```

SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = (SHA1)
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT = (SHA1)
SQLNET.CRYPTO_CHECKSUM_SERVER = requested
SQLNET.CRYPTO_CHECKSUM_CLIENT = requested

```

The available checksum types are SHA1 or MD5.

Similarly, to set up data integrity in a Type-4 JDBC driver use the following properties:

```

prop.setProperty("oracle.net.crypto_checksum_client", "REQUIRED");
prop.setProperty("oracle.net.crypto_checksum_types_client", "( MD5 )");

```

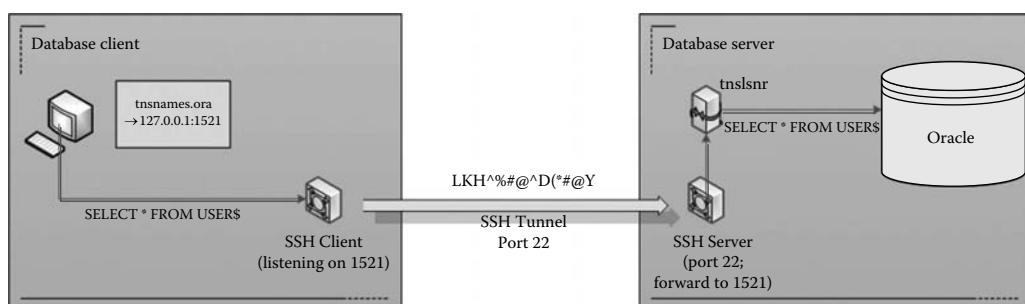
## 7.4 HOWTO Use IPSEC, Tunnels, and Hardware Acceleration

This chapter is mostly about network encryption. So why should you be involved—you’re not a network administrator. There is some merit to this argument and in this HOWTO you’ll see that you can ensure that your database communications are encrypted without doing this within the database. You will see three methods for encrypting networked communications—encrypted tunnels, Internet Protocol Security (IPSEC), and hardware-based network encryption. What is common to all these methods is that the encryption is not done by the Oracle software and as far as Oracle is concerned the traffic is not encrypted—the client sends unencrypted traffic and the server gets it after it has already been decrypted. The responsibility for setup in all these cases is not the database administrator (DBA) group—it is either the responsibility of the system administrators or the network administrators.

### Tunnels

A tunneling protocol is a network protocol in which the payload encapsulates another protocol. It allows you to use attributes of the tunneling protocol within the payload protocol. In the context of this chapter, the payload protocol is the Oracle\*Net communications which run in cleartext but because it is tunneled through another protocol that knows how to encrypt its payload, the end result is encrypted Oracle traffic. As shown in Figure 7.3, the Oracle traffic is only encrypted within the tunnel. The traffic itself is not encrypted and if you look at the traffic between the database client and the tunnel endpoint or between the tunnel endpoint and the database server then the data-in-transit is not encrypted. But when you look at the packets between the two hosts the traffic is encrypted. It is encrypted because the communications between the two hosts is done by the tunneling protocol which is encrypted. Anything that is carried by this protocol—database traffic or otherwise—is encrypted.

There are many tunneling solutions that you can use for encrypting cleartext Oracle sessions between hosts. Some are open source and others are commercial packages. Some use SSL, others use 3DES or AES—but most have similar attributes. Perhaps the most common one in use is Secure Shell (SSH) tunneling which is so common because SSH is installed on practically every Unix system (and some Windows systems) and is used ubiquitously for encrypted remote connections. In fact, SSH tunnels are so common that they are often referred to as a “poor man’s VPN.”



**Figure 7.3** Setup for an SSH tunnel.

Figure 7.3 shows how to use a tunnel. An SSH session is set up on the database client's host and connects to the SSH server on the database server's host. Anything done within this SSH session is sent encrypted to the server's SSH session—similar to what happens when you open a remote shell under SSH. The difference is that the SSH tunnel is listening on port 1521 and is told that any packet it gets on port 1521 should be forwarded to the server (over the encrypted SSH session) and delivered on the other side to port 1521. The database client makes a connection locally on port 1521 and this is propagated by the SSH tunnel to the real listener who is sitting on the database server listening on port 1521. The main disadvantage in this scheme is that you need to setup the tunnel for every database client.

Setting up the tunnel is simple. There are two kinds of tunnels—one in which the SSH session is created using a username and a password and one in which you use keys. To set up a tunnel that you can use between the database client `cl1` and the database server `db1`, run the following on `cl1`:

```
$ ssh -f -N -L 1521:127.0.0.1:1521 oracle10@db1
The authenticity of host 'db1 (192.168.2.144)' can't be established.
RSA key fingerprint is bd:02:4a:f8:d7:db:45:59:87:64:3f:7b:e9:df:e6:6e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'db1,192.168.2.144' (RSA) to the list of known
hosts.
oracle10@db1's password:
$
```

You must make sure to allow port forwarding. In `ssh_config` add the following line and restart the ssh daemon:

```
AllowTcpForwarding yes
```

This setup is less than optimal because every time the tunnel needs to be started you have to be there to supply a password. Instead, what you can do is setup the login to be based on a public/private key pair. In this scheme you generate a key pair on the client. You leave the private key on the client and import the client's public key to the server's authorized keys. Now when you set up the tunnel you need not give a password because the server can authenticate the client using the keys. The sequence follows:

Step 1: Generate a key pair on the client:

```
% ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/oracle10/.ssh/id_dsa):
/home/oracle10/.ssh/id_dsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/oracle10/.ssh/id_dsa.
Your public key has been saved in /home/oracle10/.ssh/id_dsa.pub.
The key fingerprint is
69:1e:32:97:80:d2:51:b1:21:fa:05:f1:ab:ad:8d:d3 oracle10@cl1
```

This creates two `id_dsa` files in `~/.ssh/`:

```
% ls ~/.ssh/id*
id_dsa id_dsa.pub
```

The first file is the private key—never touch that and never copy it anywhere:

```
% cat id_dsa
---BEGIN DSA PRIVATE KEY---
MIIBVQIBAAKBgQDjHmkqlU8AOyhoZznPzF9/SMOem0z00gQybKJgR7wQWB+6x/FN
tu9cKW6ExOB5ZoVOyM3K7yvREg/D/otbgahII41fKoa6fgpP4reDrRjfNJuNCik
ZulMHTxEyX+8qTM26IiY43l5e6UeLyioFF1ZNLF1lL3/T2YTi6PJ7C5HuaQIVAPwP
9qQ76zHnYY425MvUGUntXGvdAoGBALmXqYwa+SZL8rBaQMHWxzCGLfbw1RM+Bk
VBxMgipBBQqpTPGjgZdmduTcCCWBAazsdy3aer9OT7zR8LZhP/Q8Xqr823uENeMq
ean2VjnBhEzg1ZSbIiUrcza92WyE7VFNYwDOexd7SOFL1vGQ/1U421+CGgE48WUn
VWu4T0vwAoGBAKyJsD6BsSeUAEEHnTJNmK2MYYq2jn5ReNdX7LatX0EVrqQC7ERn
tZHW84HNvJrCgwhGp++ghR89/YNVS8fiw5JZd9kd/jk0jDczYkNaTVFVBpO3Lpou
RbZ8zeDI4+Vgg7SbInETuqXNz2WY6Pl0nNc9FqXVoCy6njLURf1OWM7yAhUA+7IY
7EnbqRwSD3irdKPLsiccQ5A=
---END DSA PRIVATE KEY---
```

The second file is the one you care about—that's the public key that needs to be copied to the server:

```
% cat id_dsa.pub
ssh-dss
AAAAB3NzaC1kc3MAAACBAOMeaSqVTwA7KGhnOc/MX39Iw56bTM7SBDJsmoBHvBBYH7rH8U2271wpboT
E4Hlmh7IzcrvK9ESD8P+i1uBqG0gjjV8o5rp+Ck/it40tGN80m40KKRm6UwdMTJf7ypMzboiJjeX1
7pR4vKKgUWVk0sXWUvf9PZhOlo8nsLkdRpAAAAFQD8D/ak0+sx52GONuTL1B1J7Vxr3QAAIEAuZepj
Br5JkvysFpAwczBfMIYt9vLCVEz4GRUHEyCKkEFCqlM8aOB12Z25NwIjYEBrOx3Ldp6v05PvNHwtmGn
9Dxeqvzbbe4Q14yp5qfZwoCGETODV1JsiJStxlr3ZbITtUU1jAM57F3tI4UvW8ZD/VTjaX4IaATjxZsd
Va7hPTC8AACBAKYJsD6BsSeUAEEHnTJNmK2MYYq2jn5ReNdX7LatX0EVrqQC7ERntZHW84HNvJrCg
hGp++ghR89/YNVS8fiw5JZd9kd/jk0jDczYkNaTVFVBpO3LpouRbZ8zeDI4+Vgg7SbInETuqXNz2WY6
Pl0nNc9FqXVoCy6njLURf1OWM7y oracle10@cl1
```

Step 2: On the server edit the file.ssh/authorized\_keys under the home directory for the account that will accept the SSH session. Paste the public key into this file:

```
[.ssh]$ cat authorized_keys
ssh-dss
AAAAB3NzaC1kc3MAAACBAOMeaSqVTwA7KGhnOc/MX39Iw56bTM7SBDJsmoBHvBBYH7rH8U2271wpboT
E4Hlmh7IzcrvK9ESD8P+i1uBqG0gjjV8o5rp+Ck/it40tGN80m40KKRm6UwdMTJf7ypMzboiJjeX1
7pR4vKKgUWVk0sXWUvf9PZhOlo8nsLkdRpAAAAFQD8D/ak0+sx52GONuTL1B1J7Vxr3QAAIEAuZepj
Br5JkvysFpAwczBfMIYt9vLCVEz4GRUHEyCKkEFCqlM8aOB12Z25NwIjYEBrOx3Ldp6v05PvNHwtmGn
9Dxeqvzbbe4Q14yp5qfZwoCGETODV1JsiJStxlr3ZbITtUU1jAM57F3tI4UvW8ZD/VTjaX4IaATjxZsd
Va7hPTC8AACBAKYJsD6BsSeUAEEHnTJNmK2MYYq2jn5ReNdX7LatX0EVrqQC7ERntZHW84HNvJrCg
hGp++ghR89/YNVS8fiw5JZd9kd/jk0jDczYkNaTVFVBpO3LpouRbZ8zeDI4+Vgg7SbInETuqXNz2WY6
Pl0nNc9FqXVoCy6njLURf1OWM7y oracle10@cl1
```

Step 3: Setup the tunnel—now you can connect with no password:

```
% ssh -f -N -L 1521:127.0.0.1:1521 oracle10@db1
```

## IPSEC

IPSEC is an industry standard that is used to secure IP communications. It allows you to encrypt IP traffic and perform mutual authentication. It also includes protocols for establishing keys. IPSEC operates at the network layer—layer 3 in the Open Systems Interconnection (OSI) model. It is not a tunneling protocol that runs over TCP/IP. The best way to think about it is as secure IP

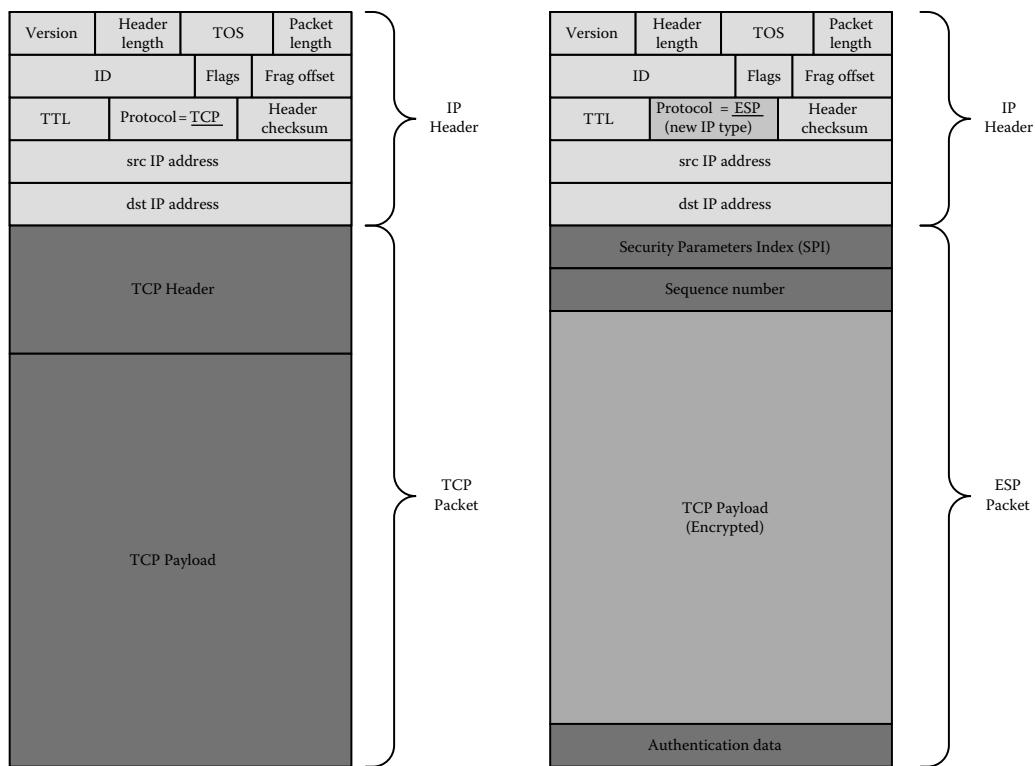
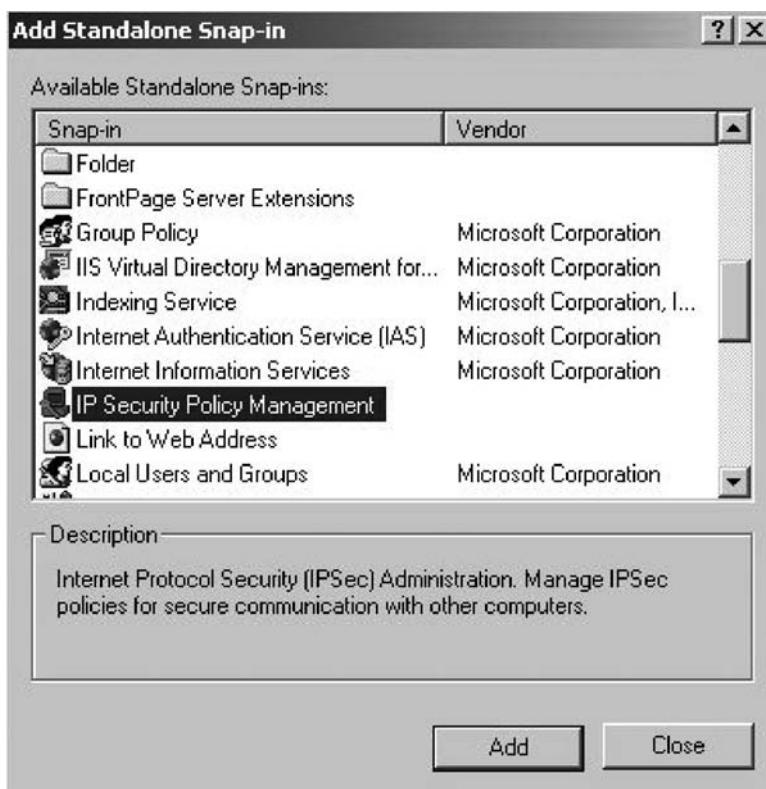


Figure 7.4 IPSEC packet structure.

(and TCP) can be “tunneled over” IPSEC. Anything that you can run over IP [e.g., TCP and User Datagram Protocol (UDP)] you can run over IPSEC. The packet structure used in an ESP IPSEC communication sits directly after the IP header as Figure 7.4 shows. The IPSEC packet structure is part of the IP payload as a new protocol type as shown by the box with the different color in the IP header. The TCP packet then travels as the ESP payload and is completely encrypted.

IPSEC is an industry standard. This makes IPSEC easy to use and available on any operating system you have. It is also supported by the switch vendors and there are even hardware manufacturers that create network cards that do the IPSEC encryption/decryption. As an example, to set up IPSEC on windows follow these steps:

1. Run mmc.
2. Click on Console- > Add/Remove Snap-in..
3. Click on Add...
4. Select IP Security Policy Management as shown in Figure 7.5 and click Add.
5. Select Local computer or a domain and click Finish.
6. Click on Close, and OK.
7. Right click on IP Security Policies on Local Machine and go through the wizard to set up the policy. The wizard has only three steps so there are really no selections on it (apart from whether or not to make this the default rule—which you probably shouldn’t if this is your first time).




---

**Figure 7.5 Adding the IPSEC Snap-in.**

8. In the properties editor you can build rules. As an example, click on Add.. to add a rule. You get a set of tabs that allow you to configure your rule as shown in Figure 7.6. When you're finished click OK and make sure your rule is checked. Click OK again.

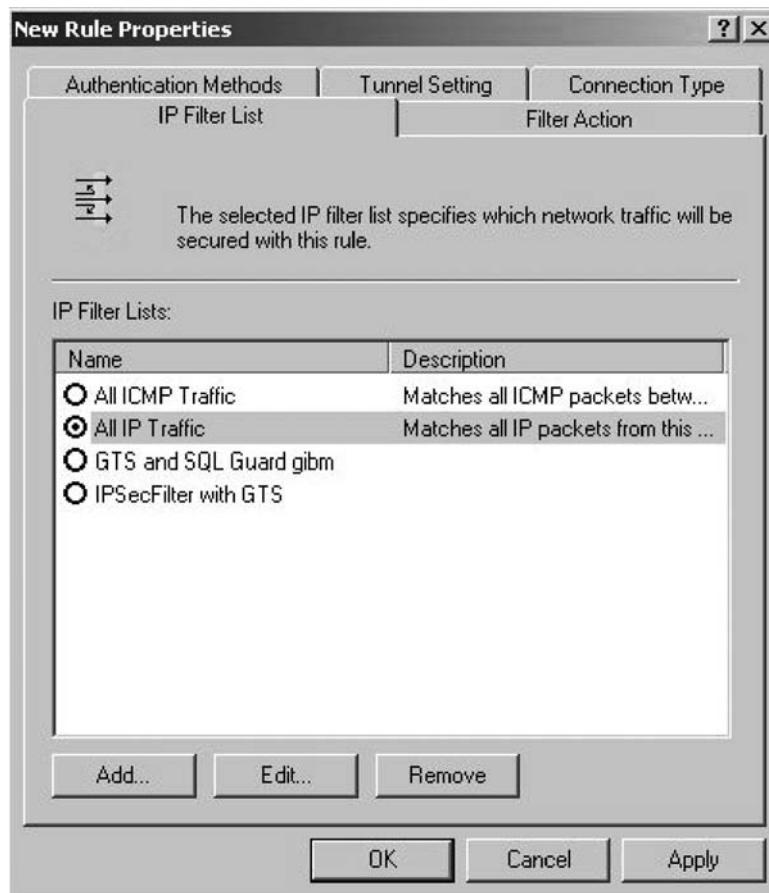
Having seen how this is done on Windows, let's look at the procedure to setup IPSEC on a Solaris operating system to set up encrypted traffic with the Windows workstation. The procedure will set up IPSEC between 192.168.2.62 and 192.168.1.201:

Step 1: Create /etc/inet/ipsecinit.conf

Step 2: Enter two entries in this file—one for each direction:

```
{ saddr 192.168.2.62 daddr 192.168.1.201 } ipsec { encr_algs 3des
encr_auth_algs md5 sa unique }
{ saddr 192.168.1.201 daddr 192.168.2.62 } ipsec { encr_algs 3des
encr_auth_algs md5 sa unique }
```

Step 3: Create /etc/inet/secret/ike.preshared. The preshared key must be converted from ASCII format (used in Windows) to a hex format (used in Solaris). You can use the od command to do that. The file should include one entry of the form:




---

Figure 7.6 Editing the properties for a new rule.

```
{
lo calidtype IP
l ocalid 192.168.2.62
re moteidtype IP
re moteid 192.168.1.201
k ey 697073656374657374
}
```

Step 4: Create /etc/inet/ike/config:

```
{
l abel "Windows"
l ocal_id_type IP
l ocal_addr 192.168.2.62
re mote_addr 192.168.1.201
p l_xform {
a uth_method preshared
o akley_group 2
}
```

```
a      uth_alg md5  
e      ncr_alg 3des  
}  
}
```

Step 5: Flush the current IPSEC configuration:

```
/usr/sbin/ipsecconf -f
```

Step 6: Load the policies defined in ipsecinit.com:

```
/usr/sbin/ipseccconf -a /etc/inet/ipsecinit.conf
```

#### Step 7: Restart the IKE daemon:

```
pkill in.iked  
/usr/lib/inet/in.iked
```

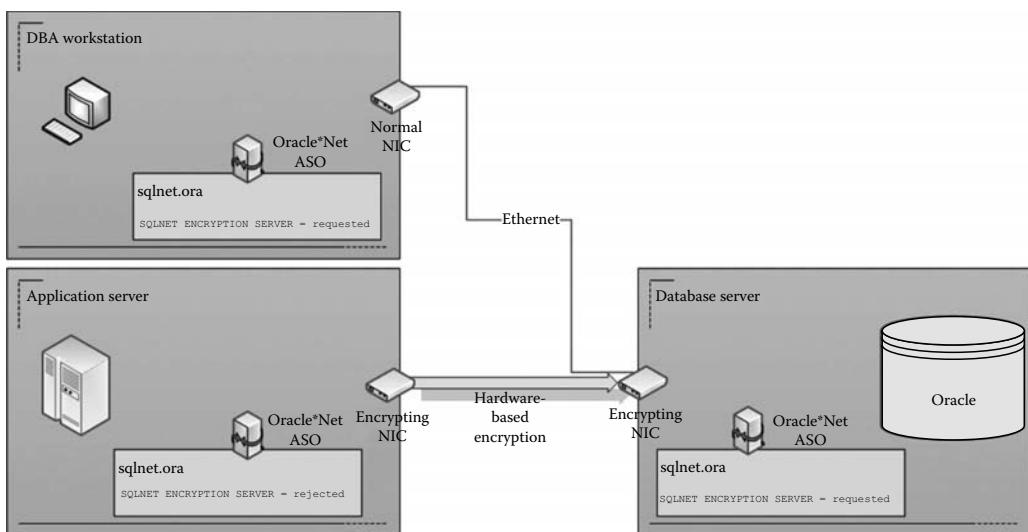
## Hardware Acceleration

All the methods described up until now used software to perform encryption and decryption operations. When you use ASO it is the Oracle software that does the work. When you use SSH tunnels, it is the SSH client and the SSH daemon that do the work. In all these cases you are adding load to the client, and more importantly to the database server host. Encryption is not a trivial operation and the amount of additional work you are doing is directly proportional to the network traffic that is being processed by the database server.

One way to avoid this additional load is to use hardware acceleration—i.e., do the encryption/decryption operations in hardware. There are various kinds of encryption accelerators and they come in many form factors. The most common form factors are either encryption appliances or specialized network cards. Encryption appliances are dedicated boxes that are plugged just before your database server host. There is a short network segment (a wire that connects your host to the accelerator) where data is unencrypted but this is usually not an issue. All traffic from the network comes in through the accelerator and is decrypted and all responses are encrypted by the accelerator so there is no additional work that the database host needs to do.

The second option is to use encrypting network cards. These cards have specialized hardware on the card itself—usually a dedicated CPU that is responsible for all encryption/decryption operations. These are simple to deploy—you don't have to change your network topology and add new machines to your racks—you just replace an existing network card with a slightly more expensive network card and you're good to go. There are many such cards from many vendors and they often have varying attributes so the selection should be done by the network administrators. Different cards have different encryption properties. For example, the 3COM Fast Ethernet NIC 3XP Encrypt card has a 3XP coprocessor that performs 3DES168 encryption. When you use this card you must ensure that the other side also encrypts using 3DES168. The Intel PRO/100S card has an integrated coprocessor that does IPSEC encryption/decryption. It allows you to manage your network encryption using IPSEC but do the work in hardware.

Using such network cards is very elegant and has the distinct advantage of not impacting performance. However, it is an additional investment—it seems to imply that you now need to change all your network cards on all machines. This can indeed be a prohibitive cost and this is



**Figure 7.7 A hybrid approach using ASO and hardware acceleration.**

where a hybrid approach can be very useful. The impact to system load is directly proportional to the amount of traffic that you need to encrypt/decrypt. In most environments, most of the traffic to the database comes from one application server (or a few application servers). It therefore makes a lot of sense to use hardware acceleration to encrypt the traffic between the application server and the database server and use software-based encryption for all other clients (which do not generate a lot of traffic and therefore will not impact load even if decrypting using the database). One example could be to use IPSEC for encryption and use IPSEC-aware network cards for the database server and the application server. This way all communications with the database server will be encrypted using IPSEC but for the traffic coming from the application server (which can easily account for 95 percent of the traffic) all encryption processing is done in hardware. Another example is shown in Figure 7.7 where ASO is used in tandem with hardware acceleration. In this example the server is set with `SQLNET.ENCRYPTION_SERVER=requested` as are all the clients apart from the application server. The application server is set with `SQLNET.ENCRYPTION_CLIENT=rejected` to ensure that ASO does not set up encrypted connections (no point in encrypting twice). There will be no software-based encryption for the application server traffic. The traffic between the application server and the database server is encrypted using specialized network cards to ensure no performance penalty.

#### Two Things to Remember about Other Methods to Achieve Network Encryption

1. Network encryption is a perfect example of a case where you might be able to delegate to other groups. The database may not be the best place to do this encryption. Check if there is an initiative to encrypt other communications (e.g., using IPSEC).
2. All software-based network encryption require CPU cycles—make sure to test the impact on high-volume servers.

## 7.5 Discussion: Performance Impact When Encrypting Data-in-Transit

Encryption is an expensive operation and it is CPU-bound. It is a nontrivial computation that needs to be performed for every block of data. The key size and the algorithm you choose determine the number of operations and the computation required per number of bytes. For example, 3DES is three times more expensive than DES because it runs three loops of DES—but DES is too weak to be used. Even though the algorithm and key size can matter, generally you don't have to get into this level of detail when you perform your analysis of the performance impact. There will not be enough of a difference between different algorithms or even between whether or not you use ASO or SSH tunnels or IPSEC—all software-based encryption will have an impact on performance, but it tends to be similar. The only case in which there is a considerable difference is when you start using hardware acceleration—then you don't have an impact on the host's CPU. In most cases, when you do use software-based encryption, this impact will be noticeable but not prohibitive. It can add 5 percent or even 10 percent to your load sometimes but you will never see numbers higher than this and usually this is a reasonable price to pay for the added protection.

It is very hard to give you precise numbers as to the impact of turning on ASO on your application because like so many things it depends on the ratios between your communication times, your disk I/O times, query processing times etc. Encryption is a CPU-intensive operation. Therefore, if your application is CPU-bound, adding encryption and decryption can have a noticeable impact in terms of machine load. If you have a mostly I/O bound system with plenty of spare CPU, encryption/decryption is less of an issue. If your application is sending and receiving large amounts of data, encryption will likely have a larger impact than if you are sending just a little data back and forth. If you know how much time you are processing at the Oracle\*NET level then you can use that as a rule of thumb and triple that time (it could even be more than that depending on the software). For example, if within your total database processing time you spend 5 percent of your CPU at the communications layer, then you can expect to spend 15 percent of your CPU when you start using encryption. If you spend 2 percent at the communication layer then you can expect it to go up to 6 percent. This is a very coarse rule-of-thumb and may not be true in your environment—you have to try it and see how it behaves.



## *Chapter 8*

---

# Encrypting Data-at-Rest

---

An Oracle database manages your data but at the end of the day the data resides within a set of operating system files. These files are protected through file permissions. A privileged user or an attacker who has managed to bypass the file system that is enforcing these permissions has access to these files and can access the data even if they have no permission to do so directly through an Oracle connection. The files contain the data, and access to the files implies access to the data. This is the fundamental problem that encryption of data-at-rest within an Oracle environment solves. If you keep the sensitive data encrypted, then access to the files does not mean that the sensitive data is available.

In Chapter 7 you learned about encrypting the database communications over the network. This type of encryption is used when you want to secure your communications from an eavesdropper who is not privileged to see sensitive data. In that case, data is encrypted before it is sent over the wire and decrypted by the client database driver (and vice versa). The data itself is not necessarily encrypted within the database because encryption of data-in-transit secures against an attacker who has access to the communication channel. Encryption of data-at-rest is complementary to encryption of data-in-transit—it encrypts the data on disk (in the database and in backups). It addresses securing against an attacker who has access to the media itself.

A “media eavesdropper” can take many forms. It can include a system administrator who has root access to a Unix server and thus can copy any file; it can include a person who has access to the data center and can physically pull out one of a number of RAID disks; it can include someone who maintains your hardware.

One of the most publicized and amusing reports in this area is a study called “A Remembrance of Data Passed: A Study of Disk Sanitization Practices” which was published in 2003 by two MIT graduate students. They purchased 158 disk drives (investing a very small amount of money) and analyzed them. They found that 117 (74 percent) of the drives contained old data that could be recovered and only 12 drives (9 percent) had been properly cleaned or sanitized (the rest of the drives simply did not work). Among the sensitive information retrieved from the disk drives were detailed personal and corporate financial records, medical records, etc. As an example, they found 2868 credit card numbers in addition to bank account numbers, dates of transactions, and account balances. According to the study, that particular drive came from an automatic teller machine in Illinois and no effort was made to remove any of the financial information on the drive prior to

resale. In addition—one of the students noted that this avenue of data theft may be one that is being utilized actively by attackers, adding—“All I can say is that when I bought [used] drives on eBay there was always somebody bidding against me. Now I don’t know who’s buying them or why, but if I were a foreign government interested in doing economic espionage on the United States, it seems to me that buying disk drives would be a cheap way to get a lot of good stuff.”

Destroying disks has of course become a practice that is followed by all corporations and most organizations. The issue that data encryption deals with is not specifically that of disk cleansing. The issue is that you can never reliably enumerate all the different ways through which an attacker may get access to your data files or backups. Therefore, although you should always invest in good physical security, system administrator auditing, and other techniques to help you combat such threats, the simplest and perhaps most effective way to deal with this issue is to make these files “unreadable” in terms of the sensitive data. That’s what encryption of data-at-rest in the Oracle world is all about—making your sensitive data unreadable and especially if they are not going through the database layer. For such cases, encryption buys you protection where, even if someone were able to get access to your files, they would not have access to your sensitive data.

The need for encryption of data-at-rest is not new, but it has bubbled up to the top of almost everyone’s list in terms of Oracle security. The reasons people have an urgent need to encrypt falls within one of two groups. For most companies both are relevant. The first group of reasons falls within the category of elevated security. Being more secure is on everyone’s mind and encryption is certainly one of the technologies that can help you be more secure. Another reason that falls within this category is the desire not to be in the headlines (not to have bad publicity). For example, in 2006 the Veterans Affairs Department (Virginia) in the United States was in the headlines because an employee’s laptop was stolen. Contrary to VA policies, this laptop had private data on more than 26 million veterans—data that can be used to launch identity theft. It was unclear whether the thieves actually used the information but the cost to the VA was huge and the headlines were awful. That same year had many other incidents involving loss or theft of media—and it prompted many companies to invest heavily in encryption—database encryption, file encryption, media encryption, etc.

The second main reason to encrypt is because you are told to do it—you have to comply with some regulation or some internal audit finding. Because encryption is so intuitive and well-known, it is very often mandated by regulations. In fact, sometimes it is mandated too quickly or too explicitly. There are often methods that can address a security threat in a way that is better or easier to implement than encryption and yet for compliance reasons you will find yourself implementing encryption.

A good example of a regulation that many companies need to abide by, and that causes many companies to implement encryption of data-at-rest is Payment Card Industry (PCI). The PCI Data Security Standard (DSS) is a set of industry tools, specifications, and measurements to help ensure the safe handling of sensitive information involving payment cards. PCI DSS (usually just called PCI) provides a framework for implementing an account data security process including preventing, detecting, and reacting to security incidents. The current version of the standard is version 1.1. If you work for a company in the retail space, work for an online merchant, work for a card processor, or work for a financial institution that manages any type of card information, you will have heard about PCI and possibly even have had to comply with some aspect within PCI.

The PCI standard is comprised of 12 general requirements designed to

- Build and maintain a secure network
- Protect cardholder data

- Ensure the maintenance of vulnerability management programs
- Implement strong access control measures
- Regularly monitor and test networks
- Ensure the maintenance of information security policies

There are many references to encryption within the PCI standard but perhaps the most relevant to encryption of data-at-rest is Section 3.4 of the PCI standard which states (PAN stands for Primary Account Information):

Render PAN, at minimum, unreadable anywhere it is stored (including data on portable digital media, backup media, in logs, and data received from or stored by wireless networks) by using any of the following approaches:

- Strong one-way hash functions (hashed indexes)
- Truncation
- Index tokens and pads (pads must be securely stored)
- Strong cryptography with associated key management processes and procedures

The MINIMUM account information that must be rendered unreadable is the PAN. If for some reason, a company is unable to encrypt cardholder data, refer to Appendix B: "Compensating Controls for Encryption of Stored Data."

Note that PCI states that the PAN (at a minimum) must be rendered unreadable—it does not state it has to be encrypted. Still, the common interpretation and implementation involves encryption. Note also that the spec also allows for compensating controls in cases where this requirement cannot be met. This is an important addition made to version 1.1 of the spec as a result of real-world experience. When PCI originally came out, many companies embarked on massive data encryption projects. The majority of these projects failed miserably; they started out as three- to six-month projects and it quickly became clear that they would go on for three years (if they ended up successfully at all). These delays were often caused by misinterpretation of what had to be encrypted and how to do it. Most of the problems occurred when organizations tried to encrypt the card holder and account information in ways that involved application changes. Encrypting these values sometimes meant that you had to change the schemas, all application code that touches these fields, all extraction, transformation, and loading (ETL) scripts that extract these values, all target databases where they may reside, all stored procedures that access these fields, etc. This experience was the basis of two evolutionary changes that were incorporated into PCI 1.1—much to the relief of the community. First, the specification clearly defines acceptance for compensating controls in cases where an organization finds that meeting a specific PCI requirement is too burdensome. Second, the specification (and the interpretations that followed) provided some analysis and distinction between disk, file, and field-level encryption. Based on some of the early failed attempts at implementing PCI, people understood that for an encryption project to be successful the data must be encrypted in a way that is transparent to the application—most commonly at the storage level rather than the application level. We will talk more about this when we talk about Oracle Transparent Data Encryption (TDE).

In this chapter, we'll review the tools available to you within Oracle for encrypting the data itself. You'll learn about packages that provide you with cryptographic functions and how to use them to encrypt and decrypt data programmatically, as well as about advanced security options such as TDE. You'll learn how to encrypt selective columns or full tablespaces. You'll also learn how to encrypt your backups. In all cases, you'll see that almost always, the hardest part of encryption is how to manage the encryption keys—and you'll learn what your options are there too.

## 8.1 Application-, Database-, and Storage-Based Encryption

Before we delve into the facilities that allow you to encrypt data within Oracle, let's take a step back and review the bigger picture and the various places that you can choose to encrypt. In all cases, remember that we're trying to ensure that the data saved within the database is encrypted. As Figure 8.1 shows, there are many ways to achieve the same result.

The methods can be classified into three categories depending on which layer does the encryption/decryption. In the first line of Figure 8.1, the application performs all the encryption/decryption operations. In this case, the data-at-rest is definitely encrypted but in addition to that, the sensitive data as it is traveling over the network is also encrypted even if you don't enable network encryption. All access to the data that does not originate within the application will find the data unreadable. For example, a database administrator (DBA) can only get access to the encrypted data. The problem is that so will any stored procedure, any replication job, any script, etc.—the amount of rework that will typically be required may be huge. Due to this issue, the second method is to let the database do the encryption—as shown on the second line in Figure 8.1. In this case encryption is transparent to the application and the data is encrypted by the database



**Figure 8.1 Options for data encryption—application, database, and storage encryption.**

just before it is stored and decrypted when someone accesses the data (through the database). If someone accesses the data bypassing the database, the data will not be readable. Finally, the third option relies on the file system or the storage system. Rather than have the database encrypt/decrypt the data, you can use a storage system that encrypts the files. From an application and database perspective there is no change whatsoever. Instead, all the files that are used by the persistence layer of the database are encrypted transparently (transparently even to the database). Although we will not discuss these solutions in this chapter, it turns out that this approach is often the easiest to implement and the most attractive solution—but this is a subjective assessment. Being an Oracle HOWTO, most of the techniques you'll see within this chapter belong to the second line in Figure 8.1 where the database performs all the encryption transparently. But let's start with a HOWTO that is somewhere between the first and second lines—in which Oracle encryption functions are used to support application-level encryption.

## 8.2 HOWTO Use DBMS\_CRYPTO

DBMS\_CRYPTO is a PL/SQL package provided as part of the Oracle database that allows you to selectively encrypt and decrypt data. DBMS\_CRYPTO allows you to perform cryptographic operations on data, for example, to encrypt data or decrypt data. As such, it provides the building blocks for writing applications which secure data-at-rest. It is up to you to use these capabilities—what Oracle gives you in the form of DBMS\_CRYPTO are the atomic cryptographic functions.

Let's look at a very simple example. Assume that you are building an application that stores credit card numbers. Assume also that you've been given a security directive that credit card data must not be stored unencrypted and that you've decided to encrypt the data using the Advanced Encryption Standard (AES); if you're embarking on a new project and have the choice among algorithms—choose AES.

You plan on getting a varchar2 for the credit card number (as 16 digits for example) and you need to store an encrypted representation rather than the number itself. Later, when the application needs to use the number for billing or for validation purposes, you will call upon a decryption function that accepts the encrypted version and returns the credit card number.

Listing 8.1 shows the encryption function. The function uses ENCRYPT (line 11) within the DBMS\_CRYPTO package to generate the encrypted value. ENCRYPT takes two RAW values—the plain text that you want to encrypt and the key. DBMS\_CRYPTO only works with RAW data so you have to cast your data to RAW as shown at line 12. Finally, you need to tell ENCRYPT which cryptographic algorithm you want to use—in this case AES with a key length of 192 bits using CBC chaining and PKCS5 padding as shown at lines 6 through 8 (see Chapter 5 for more information on CBC and PKCS5). The same applies to decrypting data. Listing 8.2 shows the use of DECRYPT (line 11) to convert a raw ciphertext into decrypted plaintext using the key and the appropriate cryptographic specification.

In this example you used AES with a key length of 192 bits. Within DBMS\_CRYPTO AES is supported with key lengths of 128, 192, and 256 bits. Unless you have special requirements (either in terms of security where you'll want 256 bits or in terms of performance where you may opt for 128 bits), using 192 bits for a key is a good bet. If you decide to use Triple Data Encryption Standard (3DES), DBMS\_CRYPTO gives you the options to use two keys (effective bit length of 112) or three keys (effective bit length of 156).

As you saw in Listing 8.1, you also have to supply a chaining scheme and a padding scheme for your encryption. DBMS\_CRYPTO allows you to chain based on Electronic Code Book

```

SQL> create or replace function encrypt_value (
  2   p_in      in varchar2,
  3   p_key     in raw)
  4 return varchar2 is
  5   l_encrypted  raw(512);
  6   l_type      number := DBMS_CRYPTO.ENCRYPT_AES192+
  7           DBMS_CRYPTO.CHAIN_CBC+
  8           DBMS_CRYPTO.PAD_PKCS5;
  9 begin
 10   l_encrypted  :=
 11     DBMS_CRYPTO.ENCRYPT(
 12       UTL_RAW.CAST_TO_RAW(p_in),
 13       l_type,
 14       p_key);
 15   return  UTL_RAW.CAST_TO_VARCHAR2(l_encrypted);
 16 end;
 17 /

```

Function created.

---

**Listing 8.1 Encrypting functions using DBM\_CRYPTO.**

```

SQL> create or replace function decrypt_value (
  2   p_in      in varchar2,
  3   p_key     in raw)
  4 return varchar2 is
  5   l_decrypted  raw(512);
  6   l_type      number := DBMS_CRYPTO.ENCRYPT_AES192+
  7           DBMS_CRYPTO.CHAIN_CBC+
  8           DBMS_CRYPTO.PAD_PKCS5;
  9 begin
 10   l_decrypted  :=
 11     DBMS_CRYPTO.DECRYPT(
 12       UTL_RAW.CAST_TO_RAW(p_in),
 13       l_type,
 14       p_key);
 15   return UTL_RAW.CAST_TO_VARCHAR2(l_decrypted);
 16 end;
 17 /

```

Function created.

---

**Listing 8.2 A decryption function using DBM\_CRYPTO.**

(CHAIN\_ECB—which you should never use), Cipher Block Chaining (CHAIN\_CBC—which is what you should almost always use), Cipher Feedback Chaining (CHAIN\_CFB), or Output Feedback Chaining (CHAIN\_OFB). All these chaining methods come into play when you encrypt a lot of data and have to determine how blocks of ciphertext interact. Encryption algorithms work in blocks; chaining methods determine how these blocks are put together. ECB is the weakest form because each plaintext is encrypted independently. It is susceptible to statistical attacks and replay attacks and should not be used. CBC is the method most commonly used—it XORs the plaintext of each block with the ciphertext of the previous block before applying the encryption algorithm. CFB allows you to encrypt units of data smaller than a block size and OFB allows running a block cipher as a synchronous stream cipher.

Padding is relevant when some of your data does not fill a block completely. This almost always happens—your data may be too short to fill a block or, as it is broken up into blocks, the last part is too small to fill a block. Because encryption algorithms always work on blocks, the question then becomes what you fill that last partly empty block with. DBMS\_CRYPTO supports three padding methods—PAD\_NONE (which will not pad and will return an error unless you manage padding and ensure that all blocks are full), PAD\_ZERO (which pads the block with zeros), and PAD\_PKCS5 (which is the option you should always use). PKCS5 refers to PKCS #5—The Password-Based Cryptographic Standard which you can view at <http://tools.ietf.org/html/rfc2898>. The padding method is described in Section 6.1.1 of this document but it isn't really important to fully understand it.

So now that we have functions for encrypting and decrypting our credit card, let's look at an example that uses these functions in Listing 8.3. In line 6 you generate the key. DBMS\_CRYPTO gives you various cryptographic primitives—but the rest is up to you. You have to generate keys, manage keys (more on that later), and manage the whole encryption process. The key that you generate in line 6 has 192 random bits—which is the key length matching the algorithm chosen in Listings 8.1 and 8.2. You could of course generate these bits yourself, but random bits usually work well for encryption keys. Once you have the key you can call the encryption function as shown in line 8—you can see the ciphertext output at the end of the listing. If you then run this ciphertext through the decryption function at line 12 (using the same key of course), you get the original credit card number as expected.

Now that you've seen a simple example, let's delve deeper into DBMS\_CRYPTO and its uses. First, you should know that DBMS\_CRYPTO is only available in Oracle 10g and 11g. In version 8i and 9i there is another package for performing cryptographic operations—DBMS\_OBFUSCATION\_TOOLKIT. This package is still available in 10g and 11g but DBMS\_CRYPTO was not available prior to 10g. Starting from 10g DBMS\_OBFUSCATION\_TOOLKIT is deprecated and you should not use it—you should always prefer DBMS\_CRYPTO. It is more secure, has more functionality, and is overall better. As an example, Table 8.1 lists the supported cryptographic functions within DBMS\_CRYPTO and DBMS\_OBFUSCATION\_TOOLKIT—you can see

```
SQL> declare
  2   my_key    raw(192);
  3   encrypted_data  varchar(512);
  4   decrypted_data  varchar(512);
  5   begin
  6     my_key := DBMS_CRYPTO.RANDOMBYTES(192/8);
  7     encrypted_data :=
  8       encrypt_value('11112222333444',my_key);
  9     dbms_output.put_line(
 10       'Encrypted data is ' || encrypted_data);
 11     decrypted_data :=
 12       decrypt_value(encrypted_data, my_key);
 13     dbms_output.put_line(
 14       'Decrypted data is ' || decrypted_data);
 15   end;
 16 /
Encrypted data is ï°o?ûj±8^.šb$cfg
Decrypted data is 11112222333444
PL/SQL procedure successfully completed.
```

---

**Listing 8.3** Getting the encrypted credit card number.

**TABLE 8.1 Comparing DBMS\_CRYPTO and DBMS\_OBFUSCATION\_TOOKIT Cryptographic Options**

Cryptographic Option	DBMS_CRYPTO	DBMS_OBFUSCATION_TOOLKIT
Available in which Oracle version	10g, 11g	8i, 9i, 10g, 11g
Cryptographic algorithms	DES, 3DES, 3DES_2KEY (Triple DES with two keys), AES, RC4	DES, 3DES
Cryptographic hash algorithms	MD5, SHA-1, MD4	MD5
MAC algorithms	HMAC_MD5, HMAC_SH1	Unsupported
Cryptographic pseudorandom number generator	RAW, NUMBER, BINARY_INTEGER	RAW, VARCHAR2
Block cipher chaining modes	CBC, CFB, ECB, OFB	CBC
Padding	PBCS5, zeros	Unsupported

that there really is no reason to use DBMS\_OBFUSCATION\_TOOLKIT any more. Another thing to remember is that although DBM\_OBFUSCATION\_TOOLKIT is deprecated in 10g and 11g, not only is it still available—it is even granted to PUBLIC as shown below! Therefore make sure to revoke this grant on your 10g and 11g systems.

```
SQL> select GRANTEE, TABLE_NAME, PRIVILEGE from DBA_TAB_PRIVS where TABLE_NAME like '%OBFUSC%';
```

GRANTEE	TABLE_NAME	PRIVILEGE
PUBLIC	BMS_OBFUSCATION_TOOLKIT	EXECUTE

DBMS\_CRYPTO is useful for more than just encrypting and decrypting data as Table 8.1 shows. As its name implies, it gives you all sorts of cryptographic functions and you can use it in applicative scenarios beyond encryption. As an example, let's look at how you can use DBMS\_CRYPTO hash functions and message authentication codes (MAC) functions. You use a hash function when you want to map an arbitrary-length input data (e.g., a message) to a fixed-length value. The important attribute of the cryptographic hash functions provided by DBMS\_CRYPTO is that collisions are statistically impossible (or, more precisely, very unlikely and impossible to manually create). This attribute allows you to rely on the hash value as a form of signature to know (among other things) whether the data was altered. As an example, if you need to store passwords for your application, you can use a message digest 5 (MD5) or secure hash algorithm-1 (SHA-1) hash function to store the hash signature rather than store the password itself. Then, when a user logs on, you can get the password, recompute the hash value, and compare it with what you have stored. An example is shown in Listing 8.4.

Next, let's see how you use DBMS\_CRYPTO to generate MAC. Let's stick with the password example. Suppose that you want to protect yourself against an attack on the application that can be performed by someone who has access to the database. As an example, suppose that I am a DBA and have access to the table where the hash values are stored and suppose that Jane is a user of the

```

SQL> set serveroutput on
SQL>
SQL> DECLARE
  2   l_pwd VARCHAR2(16) := 'm7p4ssw0rd';
  3   l_raw_pwd RAW(128) := utl_raw.cast_to_raw(l_pwd);
  4   l_hash_val  RAW(2048);
  5 BEGIN
  6   dbms_output.put_line('Password: ' || l_pwd);
  7   dbms_output.put_line('Raw Password: ' || l_raw_pwd);
  8
  9   l_hash_val := DBMS_CRYPTO.HASH(l_raw_pwd, DBMS_CRYPTO.HASH_MD4);
10   dbms_output.put_line('MD4: ' || l_hash_val);
11
12   l_hash_val := DBMS_CRYPTO.HASH(l_raw_pwd, DBMS_CRYPTO.HASH_MD5);
13   dbms_output.put_line('MD5: ' || l_hash_val);
14
15   l_hash_val := DBMS_CRYPTO.HASH(l_raw_pwd, DBMS_CRYPTO.HASH_SH1);
16   dbms_output.put_line('SHA-1: ' || l_hash_val);
17 END;
18 /
Password: m7p4ssw0rd
Raw Password: 6D377034737377307264
MD4: A87B23042DC183B0D59B443EA2EC25B1
MD5: 32D90F8B1949C7B4A16AC0D3A13CB11A
SHA-1: 1FCE2298734FE3370F2F77EAC3BCA5F1E9E7BA4A

PL/SQL procedure successfully completed.

```

#### **Listing 8.4 Using DBMS\_CRYPTO hash functions.**

application. I can replace the hash value stored for Jane by a hash value that I generate based on a password that I choose. Now I have access to Jane's account within the application. I will of course do all this at night when Jane is not around and replace the original hash value by the time she's back so that no one is the wiser. If you are the application owner and you want to protect yourself from this type of attack, you can use a MAC function instead of a hash function. You still create a value which acts as a signature for the password, but this time you use a key as shown in Listing 8.5. This is where the difference is—the signature can only be generated if I have the appropriate key. So if I am the wily attacker, I can't replace Jane's password hash with a value that will match the application check because I don't have access to the key. Of course, the password scenario is not the typical use of a MAC function. A more common use of MAC functions occur when you need to verify that a message you receive is correct and comes from the party you think you're communicating with, when you need to verify contents of a file following a concern that someone may have tampered with it, etc. In all these cases you need the key to compute a value that you compare to a previously computed value.

Let's look at keys and what key management looks like within DBMS\_CRYPTO. Simply put—DBMS\_CRYPTO does nothing in terms of key management—you're on your own! Key management is by far the most complex part of a good encryption implementation, which explains why DBMS\_CRYPTO is viewed as a “low-value” offering, and why TDE (which does manage the keys for you) is considered a “high-value” offering.

So what is key management and why is it so difficult? If you're encrypting data, you're doing it so someone will not be able to read the data. You use keys to encrypt the data and you need the keys to decrypt it later to use it within your application. You need to store the keys somewhere and

```

SQL> set serveroutput on
SQL>
SQL> DECLARE
  2      l_pwd VARCHAR2(16) := 'm7p4ssw0rd';
  3      l_raw_pwd RAW(128) := utl_raw.cast_to_raw(l_pwd);
  4      l_key     RAW(128) := DBMS_CRYPTO.RANDOMBYTES(64);
  5      l_mac_val RAW(2048);
  6  BEGIN
  7      dbms_output.put_line('Password: ' || l_pwd);
  8      dbms_output.put_line('Raw Password: ' || l_raw_pwd);
  9      dbms_output.put_line('Key: ' || l_key);
 10
 11      l_mac_val := DBMS_CRYPTO.MAC(l_raw_pwd,
DBMS_CRYPTO.HMAC_SH1, l_key);
 12      dbms_output.put_line('SHA-1 MAC: ' || l_mac_val);
 13
 14      l_mac_val := DBMS_CRYPTO.MAC(l_raw_pwd,
DBMS_CRYPTO.HMAC_MD5, l_key);
 15      dbms_output.put_line('MD5 MAC: ' || l_mac_val);
 16  END;
 17 /
Password: m7p4ssw0rd
Raw Password: 6D377034737377307264
Key:
C7E530226DF1C3DBAC759934EE894880FC430B1642548DC44AA74D3B619
2F96214BDF5D65E10A9B2
8AC1BF0B76CE2CB215317F4F431C0F441DC9006EF1952904
SHA-1 MAC: D1336C1970B427BF3DFD0DB447047C614BFB3A0C
MD5 MAC: 4BE41D71F84D918E616011FA20A50965
PL/SQL procedure successfully completed.

```

### **Listing 8.5 Using DBMS\_CRYPTO MAC functions.**

manage their retrieval and use. Of course, if the people you want to hide the data from have access to the keys, they can decrypt and view the data. Your encryption implementation is therefore only as good as your key management. Unless your implementation ensures that you protect your keys well and that only users and applications authorized to use the data have access to the keys, you might as well not encrypt at all. As an example, if you're trying to protect against someone stealing the disk, and you keep the keys unprotected in a file on disk, then you've done nothing—the attacker can steal the disk, get the keys, and then use the keys to decrypt the data. Similarly, if you are trying to ensure that DBAs cannot use decrypted data, don't store the keys unprotected within the database.

The two most important decisions you have to make when managing the keys are

1. Where do you keep your keys?
  - a. In the database?
  - b. On a file system—and if so, which file system?
  - c. Once the application is running—in which memory space do the keys live in?
2. How do you assign keys to data? Do you maintain different keys for different data?

Where and how you keep your keys is of utmost importance. You can place your keys in a database table or in a file. If you keep it in a file you can choose to have the file on the host where the database resides or on another host—for example, the host where the application server resides.

If only the application should be able to decrypt sensitive data then the later makes a lot of sense. You can even think of schemes where the file resides on a USB key or some other removable media—to ensure that only you have access to the key and thus to the unencrypted data. You must ensure good security for such files, not only using file permissions; you should consider encrypting this file using another key. Of course, you then have to store that key somewhere/somehow. Don't be fooled into thinking that you can embed the key in your code. This is a very bad idea—code can be easily disassembled to extract keys and hard coding keys is always a bad idea. Also, don't forget that you may need this key for other access types—e.g., ETL procedures that move data to a nother database, interfaces with other programs, jobs that run within the database only, etc.

Normally, you would open such a file or access a certain database table, extract the key, and store it somewhere in memory. Here too there are multiple options—you can choose to store this in the application address space or somewhere in the database address space. In either case there may be ways to extract the key from the address space itself—for example, using a memory dump. If you store the key within a database table or a database context, even temporarily, then you must consider that a DBA may have access to your key. If you store it in the application address space don't forget that because the cryptographic function runs within the database, calls to DBMS\_CRYPTO (which have the key) are sent to the database from the client. This means that the key itself travels within the Oracle\*Net communications stream and an attacker can extract this from the stream. If an attacker sniffs the network communications using something as simple as a tcp dump, and the client issues a statement as shown in Listing 8.6, then the attacker can see the key value as shown on the output in Figure 8.2. Only one packet of the Transmission Control Protocol (TCP) stream is shown—notice that the key is clearly visible within the packet at byte 0x0130. The lesson here is that if the key is used from within the client address space you should consider encrypting the network traffic using Advanced Security Option (ASO) network encryption, using IPSEC, or using any other network encryption method (Chapter 7).

```
SQL> declare
  2   my_key    raw(192);
  3   encrypted_data  varchar(512);
  4   decrypted_data  varchar(512);
  5   begin
  6     my_key  :=
'51F397E689753FD7C72FD74BDD9C99E889B0266524EC8DA4';
  7     encrypted_data  :=
  8     encrypt_value('11112222333444',my_key);
  9     dbms_output.put_line(
10       'Encrypted data is ' || encrypted_data);
11     decrypted_data  :=
12     decrypt_value(encrypted_data,    my_key);
13     dbms_output.put_line(
14       'Decrypted data is ' || decrypted_data);
15   end;
16 /
Encrypted data is ???"W:??II?????iO
Decrypted data is 11112222333444
PL/SQL procedure successfully completed.
```

---

**Listing 8.6 Encrypting data using a fixed key after it is retrieved from a file.**

```

0x0000: 4500 028b 4933 4000 4006 6885 c0a8 02db E...I3@.0.h.....
0x0010: c0a8 0289 fbbd 05f2 44d4 5129 c14c 476e .....D.Q).LGn
0x0020: 5018 8000 d667 0000 0263 0000 0600 0000 P....g....c.....
0x0030: 0000 1169 1fff ffff ffff fe01 0201 ....i.....
0x0040: 0701 0103 5e20 0304 0021 00ff ffff ffff ....^....!
0x0050: ffff fe02 01a0 ffff ffff ffff fffe 010d .....
0x0060: ffff ffff ffff fffe ffff ffff fffe ..... .
0x0070: 0001 0100 0000 0000 0000 0000 0000 0000 ..... .
0x0080: 0000 0000 00ff ffff ffff fe00 0000 ..... .
0x0090: 0000 0000 00ff ffff ffff feff ffff ..... .
0x00a0: ffff ffff feff ffff ffff fe00 00ff ..... .
0x00b0: ffff ffff ffff feff ffff ffff fe00 ..... .
0x00c0: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x00d0: 0000 fe40 6465 636c 6172 650a 206d 795f ...@declare..my_
0x00e0: 6b65 7920 7261 7728 3139 3229 3b0a 2065 key.raw(192);..e
0x00f0: 6e63 7279 7074 6564 5f64 6174 6120 7661 ncrypted_data.va
0x0100: 7263 6861 7228 3531 3229 3b0a 2064 6563 rchar(512);..dec
0x0110: 7279 7074 4065 645f 6461 7461 2076 6172 rypt@ed_data.var
0x0120: 6368 6172 2835 3132 293b 0a62 6567 696e char(512);.begin
0x0130: 0a20 6d79 5f6b 6579 203a 3d20 2735 3146 ..my_key.:=.'51F
0x0140: 3339 3745 3638 3937 3533 4644 3743 3732 397E689753FD7C72
0x0150: 4644 3734 4240 4444 3943 3939 4538 3839 FD74B@DD9C99E889
0x0160: 4230 3236 3635 3234 4543 3844 4134 273b B0266524EC8DA4';
0x0170: 0a20 656e 6372 7970 7465 645f 6461 7461 ..encrypted_data
0x0180: 203a 3d0a 2020 2065 6e63 7279 7074 5f76 .:=....encrypt_v
0x0190: 616c 7565 2827 4031 3131 3132 3232 3233 alue('@111122223
0x01a0: 3333 3434 3427 2c6d 795f 6b65 7929 3b0a 33444',my_key);.
0x01b0: 2064 626d 735f 6f75 7470 7574 2e70 7574 .dbms_output.put
0x01c0: 5f6c 696e 6528 0a20 2020 2745 6e63 7279 _line('Encry
0x01d0: 7074 6564 2064 6140 7461 2069 7320 2720 pted.da@ta.is.'..
0x01e0: 7c7c 2065 6e63 7279 7074 6564 5f64 6174 ||.encrypted_dat
0x01f0: 6129 3b0a 2064 6563 7279 7074 6564 5f64 a);..decrypted_d
0x0200: 6174 6120 3a3d 0a20 2020 6465 6372 7970 ata.:=....decryp
0x0210: 745f 7661 6c75 6528 4065 6e63 7279 7074 t_value(@encrypt
0x0220: 6564 5f64 6174 612c 206d 795f 6b65 7929 ed_data,.my_key)
0x0230: 3b0a 2064 626d 735f 6f75 7470 7574 2e70 ;..dbms_output.p
0x0240: 7574 5f6c 696e 6528 0a20 2020 2744 6563 ut_line('Dec
0x0250: 7279 7074 6564 2064 6120 7461 2069 7320 rypted.da.ta.is.
0x0260: 2720 7c7c 2064 6563 7279 7074 6564 5f64 '.||.decrypted_d
0x0270: 6174 6129 3b0a 656e 643b 0001 0101 0100 ata);.end;.....
0x0280: 0000 0000 0109 0000 0000 00 ..... .

```

**Figure 8.2** Encryption key in cleartext on the wire.

The second decision you have to make involves how you map data to keys. You may for example decide that you encrypt all sensitive data using a single encryption key. At the other extreme, you may opt to encrypt different data using different keys—even have a different key for every column that you decide to encrypt. Multiple keys give you more granularity in terms of who can view which data. For example, you can ensure that some users can see credit card numbers and other users can see social security numbers, but not vice versa. Multiple keys, especially if stored separately, imply that if one key is compromised not all sensitive data is exposed. Of course, multiple keys imply more challenges in terms of key management and possibly more security slips. One common scheme (which we will see again in the next HOWTO) uses different keys for different data and uses a single master key to encrypt these individual keys.

There is much more that can be said about key management and there are books and products devoted to this topic alone. Key management is hard—so if you’re going to use DBMS\_CRYPTO make sure to devote enough time to design your key management approach, and make sure to do a design review with your peers to flush out design flaws. Otherwise, opt to use TDE which manages the keys for you—as you’ll see in the next HOWTO.

### Seven Things to Remember about DBMS\_CRYPTO

1. Use DBMS\_CRYPTO when you need cryptographic functions; these are available to you out of the box in Oracle 10g and 11g.
2. Use DBMS\_CRYPTO; do not use DBMS\_OBFUSCATION\_TOOLKIT.
3. Revoke privileges to both DBMS\_CRYPTO and DBMS\_OBFUSCATION\_TOOLKIT from PUBLIC.
4. Use DBMS\_CRYPTO when you need to programmatically encrypt/decrypt, create cryptographic hash values, and create MACs.
5. Prefer the AES algorithm, CBC chaining, and PKCS#5 padding.
6. Prefer random bytes to generate your keys.
7. When using DBMS\_CRYPTO, you are the one responsible for the scheme—pay special attention to where/how you store keys and invest in a design review to flush out vulnerabilities.

## 8.3 HOWTO Use TDE to Encrypt Columns

TDE was introduced in Oracle 10g Release 2 and further improved in Oracle 11g. As its name suggests, TDE allows you to encrypt data in a transparent manner—that is, rather than you having to programmatically encrypt and decrypt the data, you specify which data needs to be encrypted and TDE does the work for you. TDE also has a built-in scheme for managing the encryption keys so that burden is not on you. TDE is a part of the Oracle ASO.

The transparency of TDE is most apparent in that anyone who accesses the data through the database sees the data in its unencrypted form. For example, if you have a column that is encrypted in the database it is evident if you describe the table:

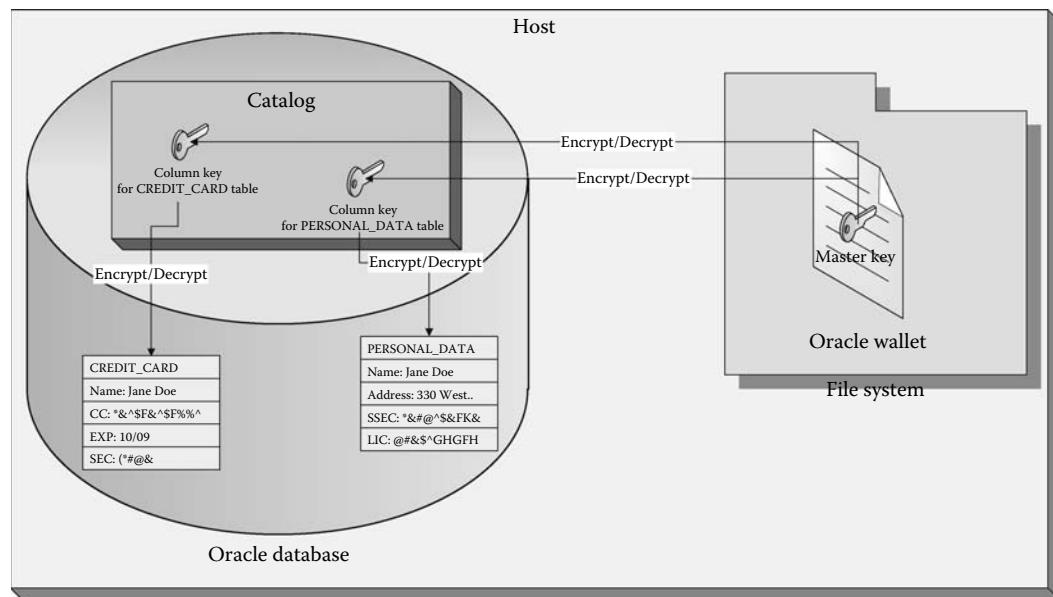
```
SQL> desc emp;
Name          Null?    Type
-----  -----
EMPNO        NO      NUMBER(4)  ENCRYPT
ENAME         V       CHAR(10)
JOB          V       CHAR(9)
MGR          N      NUMBER(4)
HIREDATE     DA     DATE
SAL           N      NUMBER(7,2)
COMM          N      NUMBER(7,2)
DEPTNO        N      NUMBER(2)
```

And yet, if you select from the table you will get the unencrypted data returned to you:

```
SQL> select empno from emp;
EM      PNO
-----
77      82
78      44
79      02
73      69
...
...
```

Note that it doesn't matter who selects from the table. TDE is not an access control mechanism. It was not built to help you ensure that DBAs don't look at sensitive data. It was built to ensure that if someone steals the files they will not have access to sensitive information. The encryption is done at the storage level of the database.

The second piece of transparency that TDE provides involves key management. As shown in Figure 8.3, TDE uses a different encryption key for each table that has columns that you encrypt. There is one key per table—regardless of whether you choose to encrypt one column or multiple columns within that table. This key is often called a column encryption key. Column encryption keys are stored within the data dictionary—they reside within the database. However, they are not stored in the clear—if they were, then anyone who steals the database files would also have the keys and therefore could decrypt the data. Instead, the keys themselves are encrypted using a master key. When you need to access encrypted data, the master key is used to decrypt the column



**Figure 8.3 Key management in TDE.**

key which is then used to decrypt the data itself. The master key does not sit within the database. Instead, it sits within an Oracle wallet—usually in a file on the file system. This allows for separation of duties as we'll see later. The wallet itself is usually secured with a password—so even if someone steals both the data files and the wallet they still would not have access to the data—they would need the password to open the wallet, to extract the master key, to decrypt the column key, and to decrypt the data.

Let's move on and see how to enable TDE. First—you need to create a wallet. Edit your sqlnet.ora file and add an entry specifying where the wallet resides:

```
ENCRYPTION_WALLET_LOCATION = (
  SOURCE = (METHOD = FILE)
  (METHOD_DATA = (DIRECTORY = /var/security/wallet/)))
```

Note: There are two wallet definitions possible within sqlnet.ora. You can use the WALLET\_LOCATION parameter to specify a default wallet that can be shared among various database components or you can create a separate wallet that is used only by TDE using the ENCRYPTION\_WALLET\_LOCATION location. It is better to define a separate wallet for TDE—this way you can better manage (e.g., regenerate) the master key as needed without affecting or being limited by other components.

Now, log onto the database and set up the master key within the wallet as follows:

```
SQL> alter system set encryption key identified by "0okmNHY^";
System altered.
```

Note that you must use double quotes—single quotes will not do:

```
SQL> alter system set encryption key identified by '0okmNHY^';
alter system set encryption key identified by '0okmNHY^'
*
ERROR at line 1:
ORA-28358: improper set key syntax
```

At this point you've set up a wallet with the master key. If you check the directory that you specified in sqlnet.ora you'll notice that the wallet file has been created for you and is owned by the instance owner account:

```
$ ls -l /var/security/wallet/
total 4
-rw-r--r-- 1 oracle11 dba 1693 Dec 28 12:48 ewallet.p12
```

The .p12 extension on the wallet file comes from the PKCS#12 standard which defines a standard file format for storing keys and protecting them with a password—Oracle conforms to the standard and maintains the wallet in this format.

You are ready to start encrypting data. To encrypt the empno column and the sal column within Scott's emp table do:

```
SQL> alter table emp modify empno encrypt no salt;
Table altered.
SQL> alter table emp modify sal encrypt salt;
Table altered.
SQL> desc emp;
```

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4) ENCRYPT
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2) ENCRYPT
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)

You've specified that you want both the SAL and the EMPNO information to be encrypted. The two alter table commands look a bit different—in the first you specified NO SALT and in the second SALT. "With salt" is the default. If you specify SALT, TDE adds random bits to the cleartext before performing the encryption. This makes the encryption solution stronger because one form of attacks that can be launched on encrypted data involves making use of statistical information—e.g., T and H often go together, S appears much more often than Z etc.

When you describe a table (as shown above) you do not see whether a column is encrypted with salt, but you can get this information from USER\_ENCRYPTED\_COLUMNS:

```
SQL> select table_name, column_name, encryption_alg, salt from user_encrypted_columns;
```

TABLE_NAME	COLUMN_NAME	ENCRYPTION_ALG	SAL
EMP	EMPNO	AES 192 bits key	NO
EMP	SAL	AES 192 bits key	YES

Similarly, to create a table with encrypted columns simply use the ENCRYPT modifier:

```
SQL> create table credit_card(
  2 cid number,
  3 cc_number varchar2(16) encrypt salt,
  4 exp_month varchar2(2),
  5 exp_year varchar2(2),
  6 cc_last_four varchar2(4),
  7 sec_code varchar2(3) encrypt salt
  8 );
Table created.
```

The default algorithm used within TDE unless you specify otherwise is AES with a key length of 192 bits, but you can choose among AES128, AES192, AES256, and 3DES168 (triple DES with three keys for an effective key length of 168 bits). You control the algorithm used and whether or not to use salt in the ALTER and the CREATE commands as shown below:

```
SQL> create table credit_card(
  2 cid number,
  3 cc_number varchar2(16) encrypt no salt,
  4 exp_month varchar2(2) encrypt salt,
  5 exp_year varchar2(2) encrypt salt,
```

```

6 cc_last_four varchar2(4),
7 sec_code varchar2(3) encrypt using 'AES256'
8 );
Table created.

```

If you later want to keep data for a column in the unencrypted form:

```
SQL> alter table emp modify empno decrypt;
Table altered.
```

Similarly you can add salt or remove salt and change the encryption algorithm:

```

SQL> alter table emp modify sal encrypt no salt;
Table altered.
SQL> alter table emp modify sal encrypt salt;
Table altered.
SQL> alter table emp rekey using 'AES256';
Table altered.

```

Note that the last command also changes the column encryption key for EMP. You can also change the column encryption key without changing the algorithm:

```
SQL> alter table emp rekey;
Table altered.
```

## TDE Overhead

Two things happen when you start encrypting/decrypting data. First, operations take longer because Oracle needs to decrypt data when you SELECT it and needs to encrypt it when you INSERT or UPDATE it. Second, storage requirements grow because data that is encrypted takes up more bytes than the original data, especially when the data that needs to be encrypted is shorter than the encryption algorithm block size (which is almost always the case).

The effect on performance depends on many things—especially if you want to state it as a percentage of the original performance. Encryption and decryption is not overly expensive, but it is also not a benign operation and it does take time. A rule of thumb that you can use is that operations will take 5 percent more time if it involves encrypting/decrypting a sensitive column. However, this is incremental—so if you have three or four columns that need to be encrypted you can expect an overhead closer to 15 percent over the nonencrypted version.

The storage overhead is easier to compute and observe. Each encrypted value is padded to 16 bytes and if salt is used for encryption, 16 more bytes are added for the salt. In addition, each encrypted value is associated with a 20-byte integrity checksum. Each row therefore requires an average of 28 additional bytes when not using salt ( $20 + (16 - 0)/2$ ) and 44 additional bytes when using salt ( $20 + 16 + (16 - 0)/2$ ). This is per encrypted column. As an example, if you have a credit card table and decide to encrypt the number and the security code with salt you can easily go from each row requiring 40 bytes to each row requiring 120 bytes—i.e., triple the storage. By the way, the only way that you can gauge the storage overhead is by looking directly at the blocks used on disk. TDE maintains transparency so even if you use dbms\_stats or avg\_row\_len or vsize, you will not see the storage overhead—as shown below when looking at these stats on the same table before and after encryption:

```

SQL> desc emp;
Name          Null?    Type
-----
EMPNO          NOT NULL NUMBER(4) ENCRYPT
ENAME           VARCHAR2(10)
JOB             VARCHAR2(9)
MGR             NUMBER(4)
HIREDATE        DATE
SAL              NUMBER(7,2) ENCRYPT
COMM            NUMBER(7,2)
DEPTNO          NUMBER(2)

SQL> select vsize(sal) from emp where EMPNO=7900;
VSIZE(SAL)
-----
      3
SQL> select avg_row_len from tabs where table_name='EMP';
AVG_ROW_LEN
-----
      37
SQL> alter table emp modify sal decrypt;
Table altered.
SQL> select vsize(sal) from emp where EMPNO=7900;
VSIZE(SAL)
-----
      3
SQL> select avg_row_len from tabs where table_name='EMP';
AVG_ROW_LEN
-----
      37

```

## TDE Limitations

TDE encrypts at the SQL layer before the data reaches the disk. Utilities that bypass the SQL layer will not work when you use TDE. As an example, TDE will not work with synchronous Change Data Capture (CDC) and Oracle Streams. This affects the use of Oracle Audit Vault and the REDO Collector as you'll see in Chapter 13. As an example, if you set up a capture process on table EMP and try to add encryption you will get the following errors:

```

SQL> alter table emp modify empno encrypt;
alter table emp modify empno encrypt
*
ERROR at line 1:
ORA-28350: cannot encrypt the specified column recorded in CDC synchronized
change table

```

Other facilities that you cannot use in tandem with TDE are

- Index types other than B-tree.
- Original import/export utilities.

- External large objects, BFILE. In Oracle 10gR2 you cannot use BLOBS and CLOBS either. In Oracle 11g support for internal large data types was added but you still cannot use BFILE LOBs when using TDE.
- Materialized View Logs.
- Transportable tablespaces.

## Handling of Encrypted Data

TDE takes care of the entire data life cycle in ensuring that your sensitive data remains encrypted. As an example, the data remains encrypted in temp files, undo-segments, redo-logs, and the buffer cache of the SGA. If you use Log Miner to view records that have encrypted data you will find that those columns are replaced with “Unsupported type” (which is precisely why TDE will not work with CDC and Oracle Streams). For example, let’s insert into a table with the following specification:

```
SQL> desc emp;
```

Name	N	ull?	Type
EMPNO		NOT NULL	NUMBER(4) ENCRYPT
ENAME			VARCHAR2(10)
JOB			VARCHAR2(9)
MGR			NUMBER(4)
HIREDATE			DATE
SAL			NUMBER(7,2) ENCRYPT
COMM			NUMBER(7,2)
DEPTNO			NUMBER(2)

Inspecting SQL\_REDO after an insert will show records of the form:

```
SQL _REDO
-----
INSERT INTO emp VALUES (Unsupported type,'RON','DBA',4444,'01-JAN-2008',
Unsupported type,0,20);
```

In Oracle 11g data that is created within temporary tablespaces that comes from join and sort operations on TDE-encrypted data remains encrypted. In Oracle 10gR2 this is not so. Oracle 11g therefore provides better security in case an attacker inspects temporary tablespaces. However, even with Oracle 11g decrypted data is still present in swap files on disk and operating system administrators may be able to see decrypted data through these files—you should be aware of this. Also, when you alter a table to enable encryption, old data may still be available in the data files in an unencrypted format. This is similar to finding old data on a disk even after it has been formatted. Eventually, the database will overwrite these blocks with new data but you have no way to know whether this happened or when this will happen. If this is a concern, you should create a new tablespace on a new data file, move the encrypted table to the new tablespace, drop the original tablespace using the INCLUDING CONTENTS KEEP DATAFILES modifier, and then use an operating system-specific command to delete the file (don’t use rm or such commands—use commands that really “shred” the files).

### Seven Things to Remember about Encrypting Columns with TDE

1. TDE protects data in cases of media-theft and access to the data files—it does not secure when access is made through the database. TDE is not a granular access control mechanism; it encrypts data on storage.
2. There is one master key and one key per table (the column key). The column key is stored in the data dictionary and is encrypted using the master key. The master key is stored in a wallet.
3. Each table uses a single key—even if you encrypt multiple columns. You control the encryption properties using ALTER TABLE and CREATE TABLE commands.
4. SALT adds random bits to the data before the data is encrypted. Encrypting with SALT is more secure because it is more resilient to frequency and statistical attacks. There is more overhead when using SALT.
5. Performance overhead tends to be 5 percent per encrypted column and can add up when encrypting multiple columns. Be selective in terms of what you encrypt.
6. TDE can easily double or triple your storage requirements—depending on how much you decide to encrypt and whether or not you are using SALT.
7. You cannot use Oracle Streams, CDC, or the Oracle Audit Vault REDO collector on data that is encrypted using TDE column encryption.

## 8.4 HOWTO Encrypt Foreign Keys and Columns Used for Indexes

How would you use TDE to encrypt columns that are used in foreign key constraints? This is a tricky question—you can't. You can't do that because every table has its own key that is used to encrypt columns—so how can the two values ever match? If you try to do that, you'll get an error. As an example, if you have a foreign key:

```
SQL> alter table emp add constraint emp_dept_fk
  2 foreign key (deptno) references dept;
```

and then try to encrypt the column, you will get an error:

```
SQL> alter table dept modify deptno encrypt no salt;
alter table dept modify deptno encrypt no salt
*
ERROR at line 1:
ORA-28335: referenced or referencing FK constraint column cannot be encrypted
```

What if you want to encrypt data but you do have foreign keys? You either use TDE with tablespace encryption (see the next HOWTO) or you abandon TDE and manage all the encryption yourself using DBMS\_CRYPTO (see the previous HOWTO).

What about indexes? TDE allows encrypting primary keys and columns used within indexes. It does limit you a bit—you can encrypt such columns as long as you don't use salt. This should be obvious—if you are going to lookup a value that has been encrypted, the database will take your value from the WHERE clause, encrypt it using the table key, and then look it up in the index. That will work fine—unless there is salt. If there is salt then some random bits have been added and therefore the match will not be made. TDE protects you from

making this mistake. As an example, it will let you encrypt an index column but only if it is without salt:

```
SQL> alter table emp modify empno encrypt salt;
alter table emp modify empno encrypt salt
*
ERROR at line 1:
ORA-28338: cannot encrypt indexed column(s) with salt
SQL> alter table emp modify empno encrypt no salt;
Table altered.
```

In addition, remember that you should never encrypt data that is used for range scans unless the table is so small that you can live with full table scans all the time. Data that is only used for identity searches can be encrypted and matched without significant performance penalties, but data that is used for range searches cannot. The reason is that when data is encrypted it does not possess the same ordering (or any other attributes) as the unencrypted data. If Oracle has to search for the record that matches credit card 11112222333444, it can encrypt that data into (as an example) 8768FD098A098B0AFC6D and search the index for the encrypted value—that will work fine. But if you ask Oracle to search for all records that have a credit card number that ends with 4444 then it has no good way of performing this query. For example, it is not true that you can look for all records where the encrypted value is AFC6D—it just doesn't work this way. If you did a range search on such a query, Oracle would have to do a full table scan! For large tables this is not an option—and this is probably the biggest issue with column-level encryption in real scenarios. Finally, one more limitation—you can't use non-B-tree index types with TDE.

#### Five Things to Remember about TDE and Keys

1. You can't encrypt foreign keys using column-level TDE.
2. If you need this feature—use tablespace-level TDE or DBMS\_CRYPTO.
3. You can't encrypt indexes with salt.
4. Never encrypt columns used for indexes if you perform range searches.
5. You can't use index types other than B-tree indexes.

## 8.5 HOWTO Use TDE to Encrypt Tablespaces

If you're using Oracle 11g you can use tablespace encryption rather than selective column encryption. The benefit of this approach is less maintenance, planning, design, and setup. You don't have to decide which columns are encrypted—entire tablespaces are being encrypted for you. The overall scheme is similar to that shown by Figure 8.3—there is a master tablespace encryption key in the wallet and that master key is used to encrypt tablespace keys. If you use TDE for both column encryption and tablespace encryption you will have two master keys—one master key for column encryption and one master key for tablespace encryption. Each tablespace that is encrypted has its own encryption key and that key is encrypted by the tablespace master key. The tablespace encryption keys live in the data dictionary as do the column encryption keys.

Execute the command:

```
SQL> alter system set encryption key identified by "0okmNHY^";
System altered.
```

Oracle 11g creates both the column and tablespace master keys automatically. If you were using TDE within Oracle 10gR2 and upgrade the database and compatibility to Oracle 11g then the tablespace master key does not yet exist. To enable tablespace encryption you must reissue the ALTER SYSTEM SET ENCRYPTION KEY command—this will create a tablespace master key for you.

Once you have the tablespace master encryption key you can go ahead and create encrypted tablespaces by adding the ENCRYPTION clause to the CREATE TABLESPACE command:

```
SQL> create tablespace secretdata
  2 datafile '/home/oracle11/oradata/onlsatur/secret01.dbf'
  3 size 100M
  4 ENCRYPTION USING 'AES256'
  5 default storage(ENCRYPT);
Tablespace created.
```

If you don't specify ENCRYPTION USING as in line 4 above, the default encryption used is AES128 (this is different from column encryption where the default is AES192 and is meant to reduce the overhead because now you're encrypting much more). You can now view that indeed the tablespace is encrypted using DBA\_TABLESPACES:

```
SQL> select TABLESPACE_NAME, ENCRYPTED from dba_tablespaces;

TABLESPACE_NAME E
-----
SYSTEM N O
SYSAUX N O
UNDOTBS1 N O
TEMP NO O
USERS N O
SECRETDATA Y ES

6 rows selected.
```

Note that you cannot encrypt an existing tablespace—the tablespace has to be encrypted at creation time. If you have existing data in an existing tablespace that is unencrypted and you want to manage it as sensitive information within an encrypted tablespace you need to create the encrypted tablespace and then create a new table in the encrypted table space using CREATE TABLE ... AS SELECT ... or move the table to the new tablespace using ALTER TABLE ... MOVE.

### The e Things to Remember about Encrypting Tablespaces with TDE

1. Tablespace encryption is only available starting with Oracle 11g.
2. You cannot encrypt an existing tablespace—you have to create an encrypted tablespace and then move data to it.
3. There is a separate master key used for tablespace encryption—it is not the same master key used for TDE column encryption. If you upgrade from Oracle 10g you need to re-create the master key so that the tablespace master key will be created.

## 8.6 HOWTO Manage TDE Master Keys

Recall from HOWTO 8.3 that the wallet contains the master key which is used to encrypt/decrypt all data-related keys. Remember also that when you first performed the `alter system set encryption key` command you set a password to protect the master key within the wallet. You use the command once to create the master key. The next time the database starts, you need to reopen the wallet and let Oracle fetch the master key. If you don't, you will not be able to read the encrypted data:

```
SQL*Plus: Release 11.1.0.6.0 - Production on Sat Dec 29 10:11:36 2007
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Connected to an idle instance.
SQL> ORACLE instance started.

Total System Global Area        417546240  bytes
Fixed Size                      2144984  bytes
Variable Size                   301991208  bytes
Database Buffers                109051904  bytes
Redo Buffers                     4358144  bytes

Database mounted.
Database opened.

SQL> Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

$ sqlplus scott
SQL*Plus: Release 11.1.0.6.0 - Production on Sat Dec 29 10:12:32 2007
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Enter password:
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL> select * from emp;
select * from emp
*
ERROR at line 1:
ORA-28365: wallet is not open
```

To read the encrypted data you need to open the wallet using the password every time you start the database, after which you will be able to decrypt the column keys and decrypt the data:

```
SQL> connect isa
Enter password:
Connected.
SQL> alter system set encryption wallet open identified by "0okmNHY^";
System altered.
SQL> connect scott
Enter password:
Connected.
SQL> select * from emp;
```

EMPNO	E	NAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SM	ITH	CLERK	7902	17-DEC-80	800		20
7499	A	LLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WA	RD	SALESMAN	7698	22-FEB-81	1250	500	30

To ensure good security, you should adopt a separation of duties approach. This means that the person who has the wallet password is not the DBA responsible for starting up the instance—just like in the movies you see two people who have to turn a key to fire a ballistic missile.

By default, the master key is randomly generated for you by TDE. However, you can use a public key infrastructure (PKI) key pair as a master key if this is required by your corporate policy. In this case, generate a certificate as shown in Chapter 5 and, with the certificate id, issue the alter system command:

```
SQL> alter system set encryption key <put your certificate id here> identified by "0okmNHY^";
System altered.
```

If you use PKI for the master key you can use an auto-login wallet. Auto-login wallets remain open all the time and do not need to be opened by a person when the database starts up. Enabling the auto-login option creates an obfuscated copy of the wallet that is then used by the database without requiring you to explicitly open it. Auto-login wallets are also useful when you want to use the same wallet from within multiple databases. You set a wallet to enable auto-login either by using the orapki utility as discussed in Chapter 5 or from within the Oracle Wallet Manager by selecting Wallet → Auto Login from the menu bar.

What should you do if you believe that your master key has been compromised? Before answering this question you have to understand that there are really two questions hiding here. The first question is whether you can reset the master key—i.e., reset the key stored within the wallet without changing the wallet password. The second question is whether you can change the password on the wallet.

Resetting the master key is possible without changing the password. Assume that you set the password for the wallet, and make another call of the form:

```
SQL> alter system set encryption key identified by "9ijnBGT%";
System altered.
```

Oracle generates a new master key for you and stores it in the wallet. It also re-encrypts the columns keys using the new master key. The wallet at this point contains the new master key as well as all the old master keys that have been retired. This is very important for RMAN encryption where you may need to bring back old backup files that were encrypted using the key within the wallet—a key that has since been retired.

The harder question is what to do if you think the wallet password has been compromised. The master key is not really ever visible so usually when the question of “what to do if the key is compromised” is really a question of “what to do if the password to the wallet has been compromised.” With the password and the wallet file an attacker has access to the current master key, all retired master keys and through those, access to all column keys and all backups—i.e., all sensitive data. In this case, you really need to reset everything—the keys, the master keys, and especially the password. It is possible to change just the password on the wallet without changing the master key (using Oracle Wallet manager) but for safety purposes you should assume that if the password was compromised so were the keys. You can’t just run the alter system command:

```
SQL> alter system set encryption key identified by "9ijnBGT%";
System altered.
```

If you just execute this command—you’ll get the following error message:

```
SQL> alter system set encryption key identified by "9ijnBGT%";
alter system set encryption key identified by "9ijnBGT%"
*
ERROR at line 1:
ORA-28353: failed to open wallet
```

This is happening because you already have a wallet and the password is incorrect. This error occurs regardless of whether or not the wallet is open or closed.

The right sequence that you should follow if you suspect your encryption has been compromised is elaborate and if you have a lot of encrypted data, long. The issue here is that you can't just limit what you're doing to the password or the master key. If you think someone has compromised your master key then you have to assume that they may already have access to the column encryption keys. In that case, the safe sequence of events that you need to perform is as follows:

1. Decrypt the data
2. Close the wallet
3. Move the wallet to a different file (keep it as a backup)
4. Regenerate the master key and the wallet using the new password
5. Re-encrypt the data

```
SQL> alter table emp modify empno decrypt;
Table altered.
SQL> alter table emp modify sal decrypt;
Table altered.

SQL> connect/as sysdba
Connected.
SQL> alter system set encryption wallet close;
System altered.
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

$ cd /var/security/wallet/
$ mv ewallet.p12 ewallet.p12.old

$ sqlplus/as sysdba
SQL*Plus: Release 11.1.0.6.0 - Production on Sat Dec 29 13:55:18 2007
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL> alter system set encryption key identified by "9ijnBGT%";
System altered.

SQL> connect scott
Enter password:
Connected.
SQL> alter table emp modify empno encrypt no salt;
Table altered.
SQL> alter table emp modify sal encrypt salt;
Table altered.
```

Always keep the old version of the wallet when you do this. If you forget to decrypt one of the columns and lose the original master key that data is gone forever. More importantly, the old wallet may contain many keys that were used by RMAN to encrypt backups—you will need this

wallet to restore these backups. You must have a process by which you manage the wallets with the master keys—in almost all organizations the risk of losing data because of a broken process is by far greater than the risk of losing sensitive data to unauthorized use.

You need to back up the wallet with the master key—but don’t back it up along with the data. The whole point of TDE is to ensure that having access to the data files does not automatically mean that the data is accessible. Don’t keep the backups of the wallet with the master key along with the data backups. This is especially true for auto-login wallets which do not require a password to open. You’ll see in HOWTO 8.10 that RMAN automatically excludes the wallet from database backups. Oracle Secure Backup (OSB) will only exclude auto-login wallets so if you want it to exclude all wallets you should add an exclude statement to your OSB configuration:

```
exclude name *.p12
```

### Five Things to Remember about Managing TDE Master Keys

1. The wallet must be open and the master keys available to read data encrypted using column encryption or tablespace encryption. When the database is started you must open the wallet or encrypted data cannot be used.
2. Auto-login wallets don’t require manual entry of the password when the database starts up. They are convenient but less secure because theft of the data files and the wallet implies access to the data. Prefer not to use auto-login wallets.
3. Recreate the wallet only if it has been compromised—many master keys do not necessarily imply stronger security.
4. ALWAYS backup the wallet and passwords in a safe place. This is especially true if you modify the master keys—without the old keys you will lose data if you have not migrated it over to use the new key.
5. The risk due to losing keys is usually larger than the risk due to data theft—make sure to put a comprehensive process for key management.

## 8.7 HOWTO Use HSMs and TDE

A Hardware Security Module (HSM) is a device that provides secure storage of keys and sometimes also provides a computational engine for performing cryptographic operations such as encryption/decryption. The device can come in a variety of forms: for example, a PCI card, a USB card, a Small Computer Systems Interface (SCSI) card, etc. The main purpose of the HSM is to securely generate or store keys. HSMs provide better security of keys than do Oracle wallets and you would use HSMs if your sensitivity levels are high. Many HSMs are also cryptographic accelerators—i.e., they perform the encryption/decryption themselves. The main reason for this is that HSMs do not allow removing the keys from the device in an unencrypted form and thus only they can do the cryptographic operations. Because HSMs are hardware/software devices that are produced by different manufacturers, there needs to be some application programming interfaces (APIs) with which other software (such as an Oracle database) can talk to the HSM. There are a number of such APIs but the most common is PKCS#11 (also known as cryptoki)—an API developed by RSA (now EMC). TDE supports HSM devices using the PKCS#11 API.

To enable HSM-based TDE you first have to let Oracle know that you want to use an HSM rather than an Oracle wallet. Change the METHOD attribute to HSM rather than FILE in the ENCRYPTION\_WALLET\_LOCATION entry in sqlnet.ora:

```
ENCRYPTION_WALLET_LOCATION = (
  SOURCE = (METHOD = HSM)
  (METHOD_DATA = (DIRECTORY = /var/security/wallet/)))
```

Although you do not need the DIRECTORY for the HSM, leave it there for previously encrypted data, for RMAN encryption, and for tablespace encryption (you'll see that later in this HOWTO).

TDE uses PKCS#11. Your HSM vendor needs to give you a PKCS#11 library that Oracle can call. You need to copy this library for Oracle to be able to make the calls—e.g., on Unix to:

```
/opt/oracle/extapi/[32,64]/hsm/[VENDOR]/[VERSION]/[LIBFILENAME]
```

32 and 64—depending if the library is a 32 bit or 64 bit library.

Once Oracle can talk to the HSM device you need to generate a user ID and password for Oracle within the HSM. This is vendor specific. Then you can issue the ALTER SYSTEM SET ENCRYPTION command to generate the master key directly within the HSM. As an example, if the user id given to you is orahsm and the password is 4rfvBHU\* then execute:

```
SQL> alter system set encryption key identified by "orahsm:4rfvBHU*";
System altered.
```

If you are already using TDE with an Oracle wallet and want to start using an HSM you run:

```
SQL> alter system set encryption key identified by "orahsm:4rfvBHU*" migrate
using "0okmNHY^";
System altered.
```

This will decrypt the existing column encryption keys and re-encrypt them using the new master key that is stored within the HSM (by making the calls to the API, and typically by doing the encryption on the HSM device itself).

TDE only supports HSM-based master keys for TDE column encryption. Tablespace encryption always uses a master key stored in the Oracle wallet. If your security requirements force you to use HSM you cannot use tablespace encryption in TDE (at least not in 11gR1) and you'll have to stay with column-level encryption. You can do both at the same time—i.e., use HSM-based columns encryption for some data and tablespace encryption for other data. You simply need to issue the `system set encryption key identified by "orahsm: 4rfvBHU*" command`—the column keys will be encrypted using a master key stored in the HSM but a tablespace master key will be generated within the Oracle wallet (with a password of `orahsm: 4rfvBHU*`—that's perfectly fine). This is another reason to leave the DIRECTORY section within ENCRYPTION section:

```
ENCRYPTION_WALLET_LOCATION = (
  SOURCE = (METHOD = HSM)
  (METHOD_DATA = (DIRECTORY = /var/security/wallet/)))
```

### The e Things to Remember about TDE and HSMs

1. HSMs give you increased security by managing keys on a specialized device rather than a file and normal (intercept-able) memory.
2. HSMs are a brand of products supplied by multiple vendors. You need to pick an HSM that supports PKCS#11 for it to work with TDE. The setup of the HSM is vendor specific and may involve some troubleshooting; you will often need help from the infosec group who may be familiar with this product.
3. Tablespace encryption does not support HSMs—in 11gR1 the tablespace master key can only reside in an Oracle wallet.

## 8.8 HOWTO Use TDE with External Tables (Oracle Data Pump)

If you look back at Figure 8.3 and HOWTO 8.3 you'll remember that each column encryption key is encrypted with the master key stored in the wallet. However, TDE also allows you to encrypt a column key directly using a password rather than using the master key—using ENCRYPT IDENTIFIED BY:

```
SQL> alter table emp modify sal encrypt identified by '1qazXSW@';
Table altered.
```

This feature comes in handy when you want to encrypt an external table defined using Oracle Data Pump. In this case, a file-based table may be accessed from multiple databases and these databases may not share single master key. If you encrypt the table using a password then you can access Data Pump tables regardless of the master key. Create the external table and specify the encryption password:

```
SQL> create table emp_ext(
  2  empno,
  3  ename,
  4  job,
  5  mgr,
  6  hiredate,
  7  sal encrypt identified by '1qazXSW@',
  8  comm,
  9  deptno)
10 organization external(
11 type oracle_datapump
12 default directory "DP"
13 location('emp_ext.dat'))
14 reject limit unlimited
15 as
16 select empno,ename,job,mgr,hiredate,sal,comm,deptno from scott.emp
17 /
Table created.
```

At this point the external table is created and the data is encrypted. To access the data from another database you can create a table based on the external file—but you must provide the password or you will not be able to read any data from this table:

```
SQL> create table emp_ext_read(
  2  empno NUMBER(4),
  3  ename VARCHAR2(10),
  4  job VARCHAR2(9),
  5  mgr NUMBER(4),
  6  hiredate DATE,
  7  sal NUMBER(7,2) encrypt identified by '1qazXSW@',
  8  comm NUMBER(7,2),
  9  deptno NUMBER(2) )
10 organization external(
11 type oracle_datapump
12 default directory "DP"
13 location('emp_ext.dat'))
14 reject limit unlimited
15 /
Table created.
```

Note that you can't create the table first and then alter the table to encrypt the column:

```
SQL> create table emp_ext
  2 organization external(
  3 type oracle_datapump
  4 default directory "DP"
  5 location('emp_ext.dat'))
  6 reject limit unlimited
  7 as
  8 select empno,ename,job,mgr,hiredate,sal,comm,deptno from scott.emp
  9 /
Table created.

SQL> alter table emp_ext modify sal encrypt identified by '1qazXSW@';
alter table emp_ext modify sal encrypt identified by '1qazXSW@'
*
ERROR at line 1:
ORA-30657: operation not supported on external organized table
```

#### Two Things to Remember about TDE and Data Pump External Tables

1. TDE encryption supports external tables.
2. Use ENCRYPTED IDENTIFIED BY if the data file needs to be used in a database with different keys.

## 8.9 HOWTO Keep Data Encrypted When You Export It Using Oracle Data Pump Utilities

TDE protects your sensitive data as long as it is within the database—what should you do when you need to export it? Use the Oracle Data Pump export utility. When you use EXPDP you can use the ENCRYPTION parameter to specify what data should be encrypted in the export file. The ENCRYPTION parameter allows you to “match-up” the encryption with the TDE-defined encrypted columns or specify that you want everything encrypted. More specifically, the values that the ENCRYPTION parameter can take are

- NONE—Nothing will be encrypted in the dump file.
- ENCRYPTED\_COLUMNS\_ONLY—Only columns encrypted through TDE are encrypted in the dump file.
- DATA\_ONLY—All data is encrypted, but only the data is.
- METADATA\_ONLY—All metadata is encrypted.
- ALL—All data and all metadata are encrypted.

There are two more parameters that affect encryption of the data in the dump files: ENCRYPTION\_PASSWORD and ENCRYPTION\_MODE. The ENCRYPTION\_PASSWORD is the password used to encrypt the data. As an example, let's look at the command:

```
$ expdp scott tables=emp directory=dpump_dir dumpfile=emp.dmp
  encryption=ENCRYPTED_COLUMNS_ONLY
  encryption_password=dfyug3476ytg
```

Here, you are exporting the emp table and encrypting only the columns that are encrypted using TDE. By default, you are not using the column keys that TDE uses within the dump file—what Oracle will do for you is decrypt the data as it is being extracted, and then re-encrypt it using another key which is generated based on the ENCRYPTION\_PASSWORD value. This is necessary because often the data will need to be imported to another database which does not share the same column keys. In this case, when you run IMPDP, you have to supply the password to decrypt the data in the dump file. Alternatively you can use the ENCRYPTION\_MODE parameter to get two other possible behaviors:

- ENCRYPTION\_MODE = TRANSPARENT means that the master encryption key will be used to encrypt the exported data. This means that the data can only be reimporrted to a database with the same master key and the wallet must be open.
- ENCRYPTION\_MODE = DUAL means that when the export is performed, the sensitive data is encrypted both using the supplied password as well as using the wallet's master encryption key. Note that as in tablespace encryption, the master key used here will be the master key within the wallet—if you're using HSMs EXPDP will use the master key from the Oracle wallet and not the master key within the HSM. When you reimport the data you can either supply the password or leave it out if the target database has the same master key and the wallet is open. This is the most flexible option in that it allows you to use keys where possible with passwords being a fallback. You pay for this flexibility in longer export time and larger dump files.

### The 5 Things to Remember about Import/Export Using Oracle Data Pump Utilities

1. Prefer to use DUAL\_MODE if you may need to import data to the same database from which the data was exported as well as databases that do not share the keys.
2. DUAL\_MODE implies more overhead because the data needs to be encrypted with and without the password.
3. Choose the ENCRYPTION parameter wisely—if you only need it to encrypt the TDE columns encrypted, use ENCRYPTED\_COLUMNS\_ONLY. For better security, use ALL.

## 8.10 HOWTO Encrypt Backups with RMAN

Encrypting backups has received a lot of attention in the past couple of years. What good is encrypting the online data if it is possible to access the sensitive data from the backups? The issue of backup encryption came to the forefront as a result of a number of breaches and losses that involved backup. Perhaps the highest visibility incident occurred back in late 2004 and disclosed in early 2005 when one of the U.S.'s largest banks lost tapes containing personal information on 1.2 million federal employees, including some members of the U.S. Senate. More ironic is the fact that one of these members was Senator Patrick Leahy. Patrick Leahy is one of the two senators (along with Arlen Specter) who introduced the Personal Data Privacy And Security Act Of 2005—proposed legislation that would help consumers better protect the privacy of their personal information in the face of recurrent data security breaches across the country. The reports on that incident were that the backup tapes were likely stolen off of a commercial plane by baggage handlers. If these tapes were encrypted, no harm done.

Luckily, you can easily encrypt RMAN backups—whether you are using TDE or not. RMAN backup encryption is simple to do; you should always encrypt your backups if you have sensitive data—there's really no excuse to not do it. Performance too is less of an issue. It is true that backups take longer if data has to be encrypted, and that restoring the data takes longer. It is also true that the encrypted backup takes more space. But these costs are negligible compared to the impact of lost sensitive data.

Like Oracle Data Pump, RMAN encryption has multiple modes—transparent mode, password mode, or dual mode. Transparent mode is the default. It relies on the wallet and does not require you to enter a password. If you use password mode you need to supply the password when backing up and later when restoring. If you use dual mode and you restore back to a database with the correct master key within the wallet then you do not have to supply the password. You can still supply the password if you are restoring to another database. Transparent and dual modes are more secure and you should only use password mode when your backups need to be transportable. In both of these modes, a new encryption key is generated for every backup. Obviously, when you reset the master key Oracle saves the previous version of the master key so that you can restore your backups even if you reset your master key many times since the backup was made. If you use passwords you are responsible to remember which password was used for which backup. When using a wallet, the master key is used to encrypt the backup key whereas when using passwords the password is used to protect the backup key.

Like any other RMAN property, you can CONFIGURE ENCRYPTION to change your default RMAN configuration and affect all RMAN session or use SET ENCRYPTION to affect only a single RMAN session. As an example, if you want all backups to be encrypted, use

```
RMAN> CONFIGURE ENCRYPTION FOR DATABASE ON;
```

to set the RMAN configuration as shown in Listing 8.7. To see which algorithms you can use for RMAN look at v\$*rman\_encryption\_algorithms*:

```
SQL> select ALGORITHM_DESCRIPTION from v$rman_encryption_algorithms;
```

```
RMAN> show all;
RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 8 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE ON;
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON STANDBY;
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
```

---

### **Listing 8.7 RMAN configuration sample for encrypting backups.**

```
ALGORITHM_DESCRIPTION
-----
AES 128-bit key
AES 192-bit key
AES 256-bit key
```

After you've configured RMAN to use encryption, make sure that the wallet is open and backup your data:

```
SQL> ALTER SYSTEM SET WALLET OPEN identified by "9ijnBGT%";
System altered.
RMAN> backup tablespace sysaux;
Starting backup at 06-MAY-07
...
Finished backup at 06-MAY-07
```

Later, when you want to restore your backup, make sure your wallet is open or you'll get a decryption error as follows:

```
SQL> alter system set wallet close;
System altered.

RMAN> restore tablespace sysaux;
Starting restore at 07-MAY-07
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile backupset restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
...
RMAN-00571: =====
RMAN-00569: ====== ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
RMAN-03002: failure of restore command at 07/05/2007 17:08:11
ORA-19870: error reading backup piece
C:\ORACLE\FLASH_RECOVERY_AREA\SAMPLE\BACKUPSET\2007_05_0601_MF_NNNDF_TAG2007
```

```
0506T165957_1NBGZG0L_.BKP
ORA-19913: unable to decrypt backup
ORA-28365: wallet is not open
```

Open your wallet, and the restore will decrypt the data:

```
SQL> ALTER SYSTEM SET WALLET OPEN identified by "9ijnBGT%";
System altered.
RMAN> restore tablespace sysaux;
Starting restore at 07-MAY-07
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
...
channel ORA_DISK_1: restore complete, elapsed time: 00:00:46
Finished restore at 07-MAY-07
RMAN> recover tablespace sysaux;
Starting recover at 07-MAY-07
using channel ORA_DISK_1
starting media recovery
media recovery complete, elapsed time: 00:00:07
Finished recover at 07-MAY-07

RMAN> sql "alter tablespace sysaux online";
sql statement: alter tablespace sysaux online
```

If you absolutely need to use a password you can set a password for a single session and encrypt/decrypt your backups:

```
RMAN> set encryption on identified by welcome1 only;
executing command: SET encryption
RMAN> backup tablespace sysaux;
Starting backup at 07-MAY-07
using channel ORA_DISK_1
...
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:38
Finished backup at 07-MAY-07

RMAN> set decryption identified by welcome1;
executing command: SET decryption
RMAN> restore tablespace sysaux;
Starting restore at 07-MAY-07
...
channel ORA_DISK_1: restore complete, elapsed time: 00:00:39
Finished restore at 07-MAY-07
```

One important note that was already mentioned in HOWTO 8.6 is that RMAN automatically excludes the wallet from database backups. However, if you use OSB, it will only exclude auto-login wallets. If you want it to exclude all wallets from your backups (and you should!), you should add an exclude statement to your OSB configuration:

```
exclude name *.p12
```

### The ~~e~~ Things to Remember about Encrypting Backups

1. Prefer to use TRANSPARENT MODE or DUAL MODE; using keys in a wallet is more secure than using passwords. Use DUAL MODE if your backups may need to be restored to the same database but must also be transportable.
2. Prefer to specify encryption as part of your RMAN configuration rather than per session. This way, there is less chance that you'll forget to encrypt. Verify your configuration periodically to ensure that changes have not been made. Better yet, use a change tracking system that checks your RMAN configuration for changes.
3. If you use OSB, add an exclude directive to ensure that wallets are not backed-up with the data.

## 8.11 Discussion: Why Did Oracle Pick the TDE Approach?

There are very good reasons to encrypt the data. These reasons span a broad spectrum from security to compliance. And yet, encryption may be one of the most misunderstood options. This misunderstanding is counterintuitive. Encryption is one of the better known and one of the most prominent security technologies. It appears in all our spy and action movies and is very visible. It also is often perceived as a simple thing to do—just encrypt it all and you're safe and happy. Obviously—this is not the case. To do encryption right requires a very large investment—encryption of data-at-rest is actually one of the harder security technologies to implement. In the Oracle world, you've seen that the options you have are abundant—starting with DBMS\_CRYPTO when you want to do it manually, through TDE when you want Oracle's help, and RMAN encryption to ensure that your backups will not become the source of the leak.

When people think about encryption of data within the database they often think that such encryption does two things—it helps you secure the data in scenarios of media theft and it allows you to hide data from privileged users such as DBAs. There are some third-party products that implement encryption of data-at-rest for Oracle databases and address both of these goals. For many people it is therefore surprising when they learn that TDE only addresses the first problem and is not meant to be a technology for granular hiding of data.

There are some very good reasons why Oracle picked this approach and they are also the reasons why all third-party products that address both goals have weak adoption. You saw some of these reasons at the beginning of this chapter. For example, the fact that TDE is “transparent” means that applications don't have to change—you don't have to change your schema in terms of size, you don't have to change your stored procedures, triggers, etc. If you had to check every access to the data and understand how encryption might affect it, you would end up with a very long project indeed. Instead, TDE allows you to know that your data is encrypted in terms of the data files, logs, etc., and all this with relatively small impact to your applications. You still have to plan for testing and changes that may have to be made due to the additional overhead, but overall it can be done in a relatively short time.

The “big picture” is that trying to address both goals using encryption causes the encryption scheme to be too complex and not worth it. Encryption (and especially the key management

aspect) tends to be complex enough as it is even if you just need to solve media theft. When you try to pile up the requirements and also address hiding information from different users, the reward is not worth the investment. It becomes too costly to implement. More important, there are many ways to solve the problem of hiding data from privileged users including authorizations (Chapters 15 and 17), monitoring and auditing (Chapters 9 through 14), and more. Oracle decided that they would use encryption for what encryption is best suited for, create a simple scheme that is not too difficult to implement, and solve the other problem using other technologies. This has worked out quite well.



## *Chapter 9*

---

# Standard Auditing

---

This is the first of six chapters of this book devoted to auditing. As a single topic, auditing is discussed more than any other topic in this book. This is no coincidence—auditing is usually the first activity to be performed in any Oracle security initiative, it is usually the first on auditor's minds, and it is one of the most effective tools in combating wrongdoing. Among other reasons, you need to audit because:

- Auditors require it and you will not pass an audit without a good audit trail which includes (at the very least) activities performed by privileged users and access to sensitive objects. This is now required by multiple regulations (such as Sarbanes Oxley and PCI) and by most internal and external auditors.
- An audit trail is a very effective deterrent which can stop people from doing the wrong thing. It is like putting a camera in places like casinos, ATMs, and subways.
- An audit trail is mandatory if you ever have to do an investigation to find out what went wrong.
- An audit trail can alert you on suspicious activity. You can't analyze what you don't know.
- An audit trail is important when defining access control—especially for existing applications. It is very hard to define access control policies that will not break business processes unless you know what the current state is in terms of who is doing what.

The act of auditing generates an audit trail. The audit trail includes records that tell you what was done in your database. You can base auditing on what activities were done, what certain users did, what privileges were used, etc. For each record in the audit trail you have important information such as the five Ws—who did it, what they did, where they did it (which schema and which object), when they did it, how they did it (what the Structured Query Language (SQL) command was) as well as additional information that can be useful in an investigation.

In this chapter you'll learn about Oracle standard auditing. This is the most general and comprehensive auditing mechanism that Oracle provides and you can use standard auditing for many of your auditing needs. In Chapter 10 you'll learn about some activities that are always audited, even when you don't ask for it, and about administrator auditing. Chapter 11 discusses fine-grained auditing (FGA)—a facility that lets you define very granular conditions for generating a audit

records. In Chapter 12 you'll learn how to use Oracle Streams to record before and after values for transactions on sensitive objects and how to use audit-generated information to infer what data a user retrieved from the database. In Chapter 13 you'll learn about Oracle Audit Vault and in Chapter 14 you'll learn about Database Activity Monitoring (DAM) and the different architectures that can help you address your auditing requirements with less work and less overhead.

## 9.1 HOWTO Enable Standard Auditing

Standard auditing is the most comprehensive and complete auditing facility within the Oracle database. It allows you to audit activities based on activity type, object, privilege, or user. There are two stages involved when using standard auditing. The first is to enable the audit mechanism and the second is to define the audit categories—i.e., define what activity should produce records in the audit trail.

To enable (or disable) the audit trail you need to set the AUDIT\_TRAIL initialization parameter. You then need to restart the database. For example:

```
SQL> alter system set audit_trail=db scope=spfile;
System altered.

SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup
ORACLE instance started.

Total System Global Area 2097152000 bytes
Fixed Size           1979968 bytes
Variable Size        1124075968 bytes
Database Buffers    956301312 bytes
Redo Buffers         14794752 bytes

Database mounted.
Database opened.
```

Turning on the audit facility lets Oracle start recording audit records based on the AUDIT specifications you give it. HOWTOs 9.2 through 9.7 review how you specify what to audit. The audit trail is either a data dictionary table or a set of operating system (OS) files. You control where the audit records end up through the value you use for the AUDIT\_TRAIL parameter. If you set the parameter to DB or DB\_EXTENDED then the audit trail is maintained in the AUD\$ system table. If you set the parameter to OS, eXtensible Markup Language (XML), or XML\_EXTENDED then the audit trail is maintained by the OS.

From a separation of duties perspective, using OS files is a better option. Auditors will often frown on an implementation where the audit trail is maintained in a database table—especially if there is a requirement to audit privileged users such as database administrators (DBAs) (and in today's world—this is always a requirement). How can an audit trail for DBA activity be used to monitor for inappropriate activity if DBAs can modify the audit trail? Therefore, you should prefer OS audit trails even though they are harder to manage and less convenient to use. Note that because the files are owned by the Oracle instance account owner, a DBA who has access to the instance account is still able to modify the audit trail.

The difference between DB and DB\_EXTENDED is that in DB\_EXTENDED you also get the SQL text and the bind values for each audit record. With DB as the value, the audit record includes the command performed but not the full text. This is also the difference between XML and XML\_EXTENDED. With both options the audit trail is written to OS files in the form of XML documents, but in the later case the SQL text and bind values are part of each audit record. When you choose OS the audit trail is still in the OS but is not an XML format, just a text delimited format.

First set the audit trail to DB as shown above and audit all activity on SCOTT.EMP:

```
SQL> audit all on scott.emp by access;
Audit succeeded.
```

Don't worry about this syntax yet—we'll discuss it in the HOWTOs that follow. If you now perform any action on this table:

```
SQL> update emp set sal=sal*0.95 where job='MANAGER';
3 rows updated.
```

You will get the following record in the audit trail:

```
select username, owner, obj_name, action_name, sql_text from dba_audit_trail

USERNAME OW NER     OBJ_NAME      ACTION_NAME      SQL_TEXT
-----  ---  ---  -----  -----
SCOTT   S    COTT    EMP          UPDATE        (null)
```

If however you change the audit trail to DB\_EXTENDED and restart the database:

```
SQL> alter system set audit_trail='db_extended' scope=spfile;
System altered.
```

The same activity will produce a record with the text:

```
select username, owner, obj_name, action_name, sql_text from dba_audit_trail

USERNAME OW NER     OBJ_NAME      ACTION_NAME      SQL_TEXT
-----  ---  ---  -----  -----
SCOTT      SCOTT    EMP          UPDATE        update emp set
sal=sal*0.95 where job='MANAGER'
```

You can also use the following syntax:

```
SQL> alter system set audit_trail='db','extended' scope=spfile;
System altered.
```

Or:

```
SQL> alter system set audit_trail=db,extended scope=spfile;
System altered.
```

But you can't use 'DB,EXTENDED'—which is a little confusing at first:

```
SQL> alter system set audit_trail='db,extended' scope=spfile;
alter system set audit_trail='db,extended' scope=spfile
*
ERROR at line 1:
ORA-00096: invalid value xml,extended for parameter audit_trail, must be from
among extended, xml, db_extended, false, true, none, os, db
```

## Operating System Audit Trail

To set the audit trail to OS, use the following and restart the database:

```
SQL> alter system set audit_trail='os' scope = spfile;
System altered.
```

The audit record will be part of an OS file normally in the adump directory and each audit file will include records of the form (in Unix):

```
[oracle10@rh4u2x32p adump]$ cat ora_19158.aud
Audit file /home/oracle10/admin/on0r4231/adump/ora_19158.aud
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
ORACLE_HOME = /home/oracle10/oracle/product/10.2.0/db_1
System name: Linux
Node name: rh4u2x32p
Release: 2 .6.9-22.ELsmp
Version: #1 SMP Mon Sep 19 18:32:14 EDT 2005
Machine: i 686
Instance name: on0r4231
Redo thread mounted by this instance: 1
Oracle process number: 15
Unix process pid: 19158, image: oracle@rh4u2x32p (TNS V1-V3)

Wed Mar 5 22:24:03 2008
SESSIONID: "360568" ENTRYID: "1" STATEMENT: "1" USERID: "SCOTT" USERHOST: "rh4u2x32p"
TERMINAL: "pts/4" ACTION: "100" RETURNCODE: "0" COMMENT$TEXT: "Authenticated by:
DATABASE" OS$USERID: "oracle10" PRIV$USED: 5

Wed Mar 5 22:24:10 2008
SESSIONID: "360568" ENTRYID: "2" STATEMENT: "7" USERID: "SCOTT" USERHOST: "rh4u2x32p"
TERMINAL: "pts/4" ACTION: "6" RETURNCODE: "0" OBJ$CREATOR: "SCOTT" OBJ$NAME: "EMP"
OS$USERID: "oracle10"

Wed Mar 5 22:24:12 2008
SESSIONID: "360568" ENTRYID: "1" ACTION: "101" RETURNCODE: "0" LOGOFF$PREAD: "28"
LOGOFF$LREAD: "287" LOGOFF$LWRITE: "18" LOGOFF$DEAD: "0" SESSIONCPU: "13"
```

To know what the ACTION (e.g., the value 101 above) means, check the AUDIT\_ACTIONS data dictionary table.

On Unix, setting AUDIT\_TRAIL = OS has another related function. You can set the AUDIT\_SYSLOG\_LEVEL initialization parameter to a value of the form <facility>. <priority> in which case the audit records will be written to syslog. This can be very useful because many log aggregators accept syslog feeds. Check with your system administrator before you decide to use this feature

because SAs do not like unknown messages in their syslog files. On Windows, audit records are written to the Windows event log and can be viewed by the Windows Event Viewer.

Let's look at the XML options. Set the audit trail to XML and restart the system:

```
SQL> alter system set audit_trail='xml' scope=spfile;
System altered.
```

An audit record will appear in OS files (normally in the adump directory) and each audit record will look like:

```
[oracle10@rh4u2x32p adump]$ cat ora_3221190068.xml
<?xml version="1.0" encoding="UTF-8"?>
<Audit xmlns="http://xmlns.oracle.com/oracleas/schema/dbserver_audittrail-10_2.xsd"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://xmlns.oracle.com/oracleas/schema/dbserver_audittrail-
10_2.xsd">
  <Version>10.2</Version>
  <AuditRecord><Audit_Type>1</Audit_Type><Session_Id>360569</Session_Id><StatementId>1</StatementId><EntryId>1</EntryId><Extended_Timestamp>2008-03-05T22:28:05.826591</Extended_Timestamp><DB_User>SCOTT</DB_User><OS_User>oracle10</OS_User><Userhost>rh4u2x32p</Userhost><OS_Process>19217</OS_Process><Terminal>pts/4</Terminal><Instance_Number>0</Instance_Number><Action>100</Action><Returncode>0</Returncode><Comment_Text>Authenticated by: DATABASE</Comment_Text>
  </AuditRecord>
  <AuditRecord><Audit_Type>1</Audit_Type><Session_Id>360569</Session_Id><StatementId>7</StatementId><EntryId>2</EntryId><Extended_Timestamp>2008-03-05T22:08.911567</Extended_Timestamp><DB_User>SCOTT</DB_User><OS_User>oracle10</OS_User><Userhost>rh4u2x32p</Userhost><OS_Process>19217</OS_Process><Terminal>pts/4</Terminal><Instance_Number>0</Instance_Number><Object_Schema>SCOTT</Object_Schema><Object_Name>EMP</Object_Name><Action>6</Action><TransactionId>060027006C030000</TransactionId>
  <Returncode>0</Returncode><Scn>15542219</Scn>
  </AuditRecord>
</Audit>
```

If you set the audit trail to XML, EXTENDED you can use one of the following syntaxes:

```
SQL> alter system set audit_trail='xml','extended' scope=spfile;
System altered.
```

Or:

```
SQL> alter system set audit_trail=xml,extended scope=spfile;
System altered.
```

But you can't use 'XML,EXTENDED':

```
SQL> alter system set audit_trail='xml,extended' scope = spfile;
alter system set audit_trail='xml,extended' scope = spfile
*
ERROR at line 1:
ORA-00096: invalid value xml,extended for parameter audit_trail, must be from
among extended, xml, db_extended, false, true, none, os, db
```

An audit record will look like:

```
cat ora_3221190052.xml
<?xml version="1.0" encoding="UTF-8"?>
<Audit xmlns="http://xmlns.oracle.com/oracleas/schema/dbserver_audittrail-10_2.xsd"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://xmlns.oracle.com/oracleas/schema/dbserver_audittrail-
10_2.xsd">
  <Version>10.2</Version>
  <AuditRecord><Audit_Type>1</Audit_Type><Session_Id>360570</Session_Id><StatementId>1</StatementId><EntryId>1</EntryId><Extended_Timestamp>2008-03-05T22:45:07.752061</Extended_Timestamp><DB_User>SCOTT</DB_User><OS_User>oracle10</OS_User><Userhost>rh4u2x32p</Userhost><OS_Process>19474</OS_Process><Terminal>pts/1</Terminal><Instance_Number>0</Instance_Number><Action>100</Action><Returncode>0</Returncode><Comment_Text>Authenticated by: DATABASE</Comment_Text>
  </AuditRecord>
  <AuditRecord><Audit_Type>1</Audit_Type><Session_Id>360570</Session_Id><StatementId>8</StatementId><EntryId>2</EntryId><Extended_Timestamp>2008-03-05T22:46:04.421663</Extended_Timestamp><DB_User>SCOTT</DB_User><OS_User>oracle10</OS_User><Userhost>rh4u2x32p</Userhost><OS_Process>19474</OS_Process><Terminal>pts/1</Terminal><Instance_Number>0</Instance_Number><Object_Schema>SCOTT</Object_Schema><Object_Name>EMP</Object_Name><Action>6</Action><TransactionId>01001C0001030000</TransactionId>
  <Returncode>0</Returncode><Scn>15543974</Scn>
  <Sql_Text>
    update emp set sal=sal*0.95 where job='MANAGER'
  </Sql_Text>
  </AuditRecord>
</Audit>
```

Note that if you want the SQL text and you want to use OS files for better separation of duties you need to use the XML\_EXTENDED option.

To see the current value for the audit\_trail parameter:

```
show parameter audit_trail
NAME          TYPE        PE          VALUE
-----        -----      -----
audit_trail  string      XML
```

The location of the audit files (when using OS files) is governed by the AUDIT\_FILE\_DEST initialization parameter, e.g.:

```
#####
# Security and Auditing
#####
audit_file_dest=/home/oracle10/admin/on0r4231/adump
```

To set the value from within the database use `alter system set audit_file_dest = <new location> deferred`. All sessions that are active will keep writing to the old directory—only new sessions will start writing audit records to the new directory.

To see the location of the audit directory:

```
SQL> show parameter audit_file_dest;
```

NAME	TY	PE	VALUE
audit_file_dest	s	tring	C:\ORACLE\PRODUCT\10.2.0\ADMIN\ONORON00\ADUMP

### The e Things to Remember about Standard Audit

1. To enable standard audit you need to set the AUDIT\_TRAIL initialization parameter and restart the database.
2. Standard audit produces audit records that can be written either to a database table (AUD\$) or to OS files (in a directory controlled by AUDIT\_FILE\_DEST). Prefer to use OS files for better separation of duties.
3. OS audit files can be formatted as XML files or as a name-value delimited text file. If you need to audit the SQL text and need OS files you will need to use the more verbose XML format.

## 9.2 HOWTO Use Audit Qualifiers

Before we delve into the audit commands themselves, there are two qualifiers that you can add to any audit statement.

### Auditing When Successful, When Unsuccessful, and Both

When you audit, you can use WHENEVER SUCCESSFUL, WHENEVER NOT SUCCESSFUL, or both. If you specify WHENEVER SUCCESSFUL you will only get an audit record when the activity being audited was successful. It is also common to audit WHENEVER NOT SUCCESSFUL—a failure to execute a command can indicate a security issue. The default is to create an audit record in both cases—this is what happens when you don't specify anything. Let's look at an example:

First let's audit all updates on SCOTT.EMP, but only when not successful:

```
SQL> audit update on scott.emp by access whenever not successful;
Audit succeeded.
```

You'll see what BY ACCESS means in a short while—simply put, it means an audit record per access. Now perform two updates, one of which is going to fail:

```
SQL> update emp set sal=sal*1.1 where empno=4444;
1 row updated.
SQL> update emp set empno=5555 where empno=4444;
update emp set empno=5555 where empno=4444
*
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.PK_EMP) violated
```

Let's see what you get in the audit trail (in this case using DB\_EXTENDED):

```
select timestamp,username,action_name,sql_text,returncode
from dba_audit_trail
where obj_name='EMP' and owner='SCOTT'

TIMESTAMP          USERNAME      ACTION _ NAME      SQL _ TEXT
RETURNCODE

-----
-----  -----  -----  -----
3/31/2008 4:13:47 PM  SCOTT        UPDATE       update emp set empno=5555 where
empno=4444 1
```

Returncode has a value of zero when successful and the Oracle error code when unsuccessful—hence the 1 in return code (ORA-00001). Let's look at another example. As another user without privileges, try to update the table—you'll get a privilege error:

```
SQL> connect ronb
Enter password: *****
Connected.

SQL> update scott.emp set sal=sal*1.1 where empno=4444;
update scott.emp set sal=sal*1.1 where empno=4444
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

The audit record is:

```
3/31/2008 4:53:12 PM RONB          UPDATE       update scott.emp set
sal=sal*1.1 where empno=4444 2004
```

What's going on here—this is 2004, not 1031. ORA-02004 is a security violation error. This error code is an abstraction that only appears in the audit trail's return code—it is never the error seen by the user within their session.

If you want both successful and unsuccessful updates to be logged:

```
SQL> audit update on scott.emp by access;
Audit succeeded.
```

So now:

```
SQL> update emp set sal=sal*1.1 where empno=4444;
1 row updated.

SQL> update emp set empno=5555 where empno=4444;
update emp set empno=5555 where empno=4444
*
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.PK_EMP) violated
```

And the resulting audit records:

```

select timestamp,username,action_name,sql_text,returncode
from dba_audit_trail
where obj_name='EMP' and owner='SCOTT'

TIMESTAMP          USERNAME      ACTION_NAME    SQL_TEXT
RETURNCODE

-----
-----      -----      -----
3/31/2008 4:32:11 PM    SCOTT        UPDATE       update emp set sal=sal*1.1 where
empno=4444 0
3/31/2008 4:32:17 PM    SCOTT        UPDATE       update emp set empno=5555 where
empno=4444 1

```

WHEN U NSUCCESSFUL refers to whether the action succeeded, not whether the action persists.

Audit records are written out as the actions occur and are NOT rolled back with a transaction. For example:

```

SQL> select * from emp where empno=4444;
      EMPNO  ENAME   JOB          MGR  HIREDATE      SAL     COMM  DEPTNO
-----  -----  -----  -----  -----  -----  -----  -----
      4444    RON     DEV       7902 02-NOV-02  3061.3    200      20
1 row selected.

SQL> update emp set comm=300 where empno=4444;
1 row updated.

SQL> rollback;
Rollback complete.

```

But when you look at the audit trail you will find the following:

```

3/31/2008 5:17:38 PM    SCOTT        UPDATE       update emp set comm=300
where empno=4444          0

```

This may be surprising at first—but it makes sense from an audit trail and a security perspective. An audit trail is not a transaction log and is not used to recover data, roll-forward activities etc. It is a security tool for monitoring what users and applications are doing—and these records should not “disappear” when a rollback occurs. Take a look at Section 15.8. It shows you a technique where you can use DELETEs and rollbacks to determine data values even if it is not authorized to select these values (if a certain parameter has the wrong value). This is a very specific example, but more generally, if the user did it, it has to be in the audit trail.

## BY SESSION and BY ACCESS

The second qualifier available in any audit command allows you to specify whether you want an audit record per access or per session. If you specify BY ACCESS, Oracle creates one audit record for each access. If you specify BY SESSION, Oracle writes a single audit record per combination of session, object, and SQL statement type. If the same statement is repeated

multiple times within the same session there is no indication of this fact. Let's look at a quick example.

First, audit all activity on SCOTT.EMP and request that an audit record be written per access:

```
SQL> audit all on scott.emp by access;
Audit succeeded.
```

Connect and do the following:

```
SQL> connect scott
Enter password: *****
Connected.

SQL> select max(sal) from emp;
MAX (SAL)
-----
5     000
1 row selected.

SQL> update emp set sal=6000 where sal=5000;
1 row updated.
```

Let's look at the relevant records in the audit trail:

```
3/31/2008 7:04:16 PM    SCOTT      UPDATE      update emp set sal=6000 where sal=5000
0
3/31/2008 7:04:02 PM    SCOTT      SELECT      select max(sal) from emp
0
```

Now change the audit definition:

```
SQL> audit all on scott.emp by session;
Audit succeeded.
```

Do the same sort of activities:

```
SQL> select max(sal) from emp;
MAX (SAL)
-----
6     000
1 row selected.

SQL> update emp set sal= 6500 where sal= 6000;
1 row updated.
```

The audit trail looks very different this time:

```
3/31/2008 7:20:36 PM    SCOTT      SESSION REC      select max(sal) from emp
0
```

There's only one record even though there were two activities. The action name for this is SESSION REC. This indicates that this audit record represents all activities on the object. Don't look at the SQL Text—it is meaningless when you use BY SESSION (it is simply the SQL text for the first access). What's more important to look at is the SES\_ACTIONS column:

```
select timestamp,username,action_name,ses_actions
from dba_audit_trail
where obj_name='EMP' and owner='SCOTT'
3/31/2008 7:20:36 PM      SCOTT      SESSION REC      -----SS-----
```

The SES\_ACTIONS column is a “bit map” of sorts—each position represents a certain activity type as shown in Table 9.1. In the example above you can see that the tenth and eleventh positions are marked with an “S.” “S” stands for success, “F” stands for failure, and “B” stands for both. Issue an update that fails and one insert that fails:

```
SQL> update emp set empno=5555 where empno=4444;
update emp set empno=5555 where empno=4444
*
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.PK_EMP) violated

SQL> insert into emp values(4444,'foo','bar',9999,'02-NOV-02',0,0,20);
insert into emp values(4444,'foo','bar',9999,'02-NOV-02',0,0,20)
*
```

**Table 9.1 What Each Position in the SES\_ACTIONS Column Means**

Position	Action
1	ALTER
2	AUDIT
3	COMMENT
4	DELETE
5	GRANT
6	INDEX
7	INSERT
8	LOCAL
9	RENAME
10	SELECT
11	UPDATE
12	REFERENCE
13	EXECUTE

```
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.PK_EMP) violated
```

The audit record shows S for SELECT, B for UPDATE and F for INSERT:

3/31/2008 7:20:36 PM	SCOTT	SESSION REC	-----F-SB-----
----------------------	-------	-------------	----------------

Because there is only one audit record where everything is encoded, don't expect meaningful values for any column that can change such as SQL text or return code.

When you ask for BY SESSION, Oracle creates a single record by session id and object id (AUD\$.OBJ\$CREATOR and OBJ\$NAME) and per system privilege used. If any of these is different a new audit record is created. For example, if you drop a table and recreate it within the same session then multiple records with the same type will be written because the object ID has changed. Finally, BY SESSION auditing only applies to database audit trails—not to OS audit trails. If you use the BY SESSION qualifier, the OS files will still have one record per access—this is because Oracle can't afford to read the files and update these records—something that it can easily do when the data is stored in a database. Note that the main advantage of BY SESSION auditing is not in performance. The main advantage is that if you're only required to know what type of activities were performed by each user or session (as opposed to what precise SQL was sent, how many times each command was run, etc.) then BY SESSION auditing is by orders of magnitude more compact, easier to manage and does not require you to deal with potentially millions of audit records. This qualifier simply gives you a tool to audit at the granularity level that best matches your requirements.

#### **The e Things to Remember about Audit Qualifiers**

1. You can specify whether you want an audit record to be created only when the action succeeded, only when it did not succeed or in both cases.
2. The audit trail can include an audit record for every action performed or an audit record that records only what type of action was performed within a session. If you use BY SESSION qualifiers with an OS audit trail, multiple audit records will still be written.
3. Audit records are written to the audit trail even if you rollback a transaction. From a security perspective this is the right thing to do—you need to know what a user did even if they rolled back the transaction at the end.

## **9.3 HOWTO Use Statement Auditing**

You can audit based on three separate categories (or any combination thereof)—statements, privileges, and objects. When using statement auditing you specify which statements you want to be audited and an audit record is created due to an execution of a certain statement type—e.g., a CREATE TABLE. The audit command requires you to specify which statement to audit, e.g.,

```
SQL> audit create table by scott;
Audit succeeded.
```

```
SQL> audit create table;
Audit succeeded.
```

To audit all statements:

```
SQL> audit all;
Audit succeeded.
```

To audit a set of statements:

```
SQL> audit create table, create procedure;
Audit succeeded.
```

Or, use statement groups as a shorthand—for example, rather than specify:

```
SQL> audit alter user;
Audit succeeded.
```

```
SQL> audit drop user;
Audit succeeded.
```

```
SQL> audit create user;
Audit succeeded.
```

You can do:

```
SQL> audit user;
Audit succeeded.
```

You can use audit qualifiers to limit the cases when audit records are created, e.g.,

```
SQL> audit create table by access;
Audit succeeded.
SQL> audit create table by access whenever successful;
Audit succeeded.
```

Finally, you can qualify the user for which statement auditing applies:

```
SQL> audit create table by scott;
Audit succeeded.
SQL> audit create table by scott by access;
Audit succeeded.
```

Note that the user qualifier is not a fourth type by which you audit—it is a qualifier that you can add when using one of the three audit specifications.

## AUDIT SESSION

AUDIT SESSION allows you to audit creation of sessions. An audit record is created for each connection to the instance. It inserts a record into the audit trail at connection time and updates the audit record at disconnect time. When in the database, the record includes the connection time, the disconnection time, I/O processing time, etc. This audit option is of course only available BY SESSION. You can qualify AUDIT SESSION by user, e.g.,

```
SQL> audit session by scott;
Audit succeeded.
```

## AUDIT NOT EXIST

AUDIT NOT EXIST allows you to audit statement failures that occur because the target object does not exist, e.g.,

```
SQL> audit not exists by access;
Audit succeeded.
```

Now access an object that does not exist:

```
SQL> select * from foobar;
select * from foobar
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

An audit record is generated. For example, if using XML\_EXTENDED:

```
<AuditRecord><Audit_Type>1</Audit_Type><Session_Id>16219456</Session_Id><StatementId>7</StatementId><EntryId>2</EntryId><Extended_Timestamp>2008-04-02T08:30:27.243616</Extended_Timestamp><DB_User>SCOTT</DB_User><OS_User>oracle10</OS_User><Userhost>trex</Userhost><OS_Process>21919</OS_Process><Terminal>pts/2</Terminal><Instance_Number>0</Instance_Number><Object_Schema>SCOTT</Object_Schema><Object_Name>FOOBAR</Object_Name><Action>3</Action><Returncode>942</Returncode>
<Sql_Text>
select * from foobar
</Sql_Text>
</AuditRecord>
```

### Four Things to Remember about Statement Auditing

1. Use statement auditing when you need to audit certain activity such as Data Definition Language (DDL) statements.
2. Use audit qualifiers and user qualifiers to make sure you are auditing only those activities that you require an audit trail for.
3. Use AUDIT SESSION to audit connects and disconnects.
4. Use AUDIT NOT EXIST to audit statements that use an object that does not exist.

## 9.4 HOWTO Use Object Auditing

You can audit based on three separate things (or any combination thereof)—statements, privileges, and objects. When using object auditing you specify that an audit record should

be created based on the fact that a certain action was done on a certain object. You specify objects and statements and you can use the standard qualifiers, but you can't qualify by user. For example:

```
SQL> audit select on scott.emp by access;
Audit succeeded.

SQL> audit update on scott.emp by access;
Audit succeeded.

SQL> audit all on scott.emp by access;
Audit succeeded.

SQL> audit insert, update, delete on scott.emp by session;
Audit succeeded.

SQL> audit select on scott.emp by scott;
audit select on scott.emp by scott
*
ERROR at line 1:
ORA-01708: ACCESS or SESSION expected
```

This is useful for objects that already exist—but how do you handle objects that have not been created yet? It is not feasible to tie every object creation with a new AUDIT call manually. Instead, use ON DEFAULT. For example, to specify that any new object created should have an audit trail for INSERT, UPDATE, and DELETE:

```
SQL> audit update,insert,delete ON DEFAULT by access;  
Audit succeeded.
```

Any new object will be “born” with these and its definitions (assuming that this object can be audited for these activities). To view what the defaults are

What this “bitmap” shows is that any new object will automatically be audited for DELETE (DEL)—by access for success (the left side of A/A) and by access for failure (the right side of A/A), and the same for INSERT (INS) and UPDATE (UPD). This is a cumulative definition—for example,

## Auditing Views

You can audit views as well as base tables. If you audit both the view and the base table, and access the view then you'll get multiple records. For example, create a view on EMP and then audit both the view and the base table:

```
SQL> create view emp_details as
  2  select EMPNO,ENAME,JOB,MGR,HIREDATE,DEPTNO
  3  from emp;
View created.

SQL> audit select on emp by access;
Audit succeeded.

SQL> audit select on emp_details by access;
Audit succeeded.
```

If you then query the view you will get two audit records.

```
SQL> select count(*) from emp_details;
CO UNT(*)
-----
18219

select USERNAME,TIMESTAMP,ACTION_NAME,OWNER,OBJ_NAME,SQL_TEXT from dba_audit_trail
SCOTT      4/2/2008 11:42:18 AM    SELECT          SCOTT      EMP        select
count(*) from emp_details
SCOTT      4/2/2008 11:42:18 AM    SELECT          SCOTT      EMP_DETAILS  select
count(*) from emp_details
```

### The e Things to Remember about Object Auditing:

1. Use object auditing when you need to audit usage of certain sensitive objects.
2. You can use the standard audit qualifiers but you can't qualify by user.
3. Use ON DEFAULT to specify what to audit for objects that will be created and use ALL\_DEF\_AUDIT\_OPTS to view what the default is.

## 9.5 HOWTO Use Privilege Auditing

You can audit based on three separate things (or any combination thereof)—statements, privileges, and objects. When using privilege auditing you define the privileges that, when used, cause an audit record to be created. An audit record is created only when the privilege is used—regardless of which statement was used. For example, if you audit based on the use of a CREATE ANY TABLE privilege, then an audit record is only created when a user does a CREATE TABLE outside their schema—but if they create a table in their own schema (and utilize the CREATE TABLE privilege and not the CREATE ANY TABLE privilege) then an audit record is not written. For example:

```
SQL> connect sys as sysdba
Enter password:
Connected.
```

```

SQL> grant create any table to scott;
Grant succeeded.

SQL> audit create any table by scott by access;
Audit succeeded.

SQL> connect scott
Enter password: *****
Connected.

SQL> create table TAB_WONT_SHOW(i int);
Table created.

SQL> create table HR.TAB_WILL_SHOW(i int);
Table created.

```

The first CREATE TABLE does not use the CREATE ANY TABLE privilege so it will not show up in the audit trail. The PRIV\_USED column in DBA\_AUDIT\_TRAIL shows you the privilege used:

```

SQL> select USERNAME,OWNER,OBJ_NAME,ACTION_NAME,PRIV_USED from dba_audit_trail;
USERNAME          OWNER
-----
OBJ_NAME
-----
ACTION_NAME P      PRIV_USED
-----
SCOTT H           R
TAB_WILL_SHOW     CREATE ANY TABLE
CREATE TABLE

```

To audit all privileges (in this case with a user qualifier):

```
audit all privileges by scott by access;
```

### The e Things to Remember about Privilege Auditing

1. Use privilege auditing when you need to audit all activity that, to be performed, had to use an elevated privilege. This is especially important for system privileges that you'll learn more about in Chapter 15.
2. The PRIV\_USED column records what privilege was used to perform the audited statement.
3. You can qualify privilege auditing with a user.

## 9.6 HOWTO Audit for Unexpected Errors in the Network Layer

This is such a small topic that hardly warrants a separate HOWTO—but it is an important audit capability and is needed for completeness. You use network auditing to audit networking errors. These are internal or setup errors in the Oracle networking layers—not connect errors.

Examples include TNS-02507—encryption algorithm not installed, TNS-12648—encryption or data integrity algorithm list is empty, and TNS-12650—no common encryption or data integrity algorithm.

You enable this category using

```
SQL> audit network;
Audit succeeded.
```

You disable it using:

```
SQL> noaudit network;
Noaudit succeeded.
```

## 9.7 HOWTO Read Audit Records

In HOWTO 9.1, you already saw the structure of audit records in the database and in OS files. OS files are created per session and include self-describing audit records if stored using an XML format, or a delimited format when using AUDIT\_TRAIL=OS. When using DB or DB\_EXTENDED there are many database views that you can use to mine data from audit records easier. The audit trail itself is stored in AUD\$ which is generated when the database is created. However, you will almost never select from AUD\$ as the data there is hardly readable—e.g.,

```
select
SESSIONID,ENTRYID,STATEMENT,USERID,USERHOST,TERMINAL,ACTION#,RETURNCODE,OBJ$CREATOR,OBJ$NAME,
SPARE1,NTIMESTAMP#,PROCESS#,XID,SCN
from sys.aud$
```

SESSIONID	E	NTRYID	STATEMENT	USERID	USERHOST	TERMINAL	ACTION#
RETURNCODE							
OBJ\$CREATOR	OBJ	\$NAME	SPARE1	NTIMESTAMP#		PROCESS#	XID
SCN							
1759404	1		7	SCOTT	saturn	pts/0	3
0							
SCOTT		EMP	oracle10	2007-09-17 12:43:43.317801	23086		(null)
36825881							
1759404	2		8	SCOTT	saturn	pts/0	6
0							
SCOTT	E	MP	oracle10	2007-09-17 12:43:46.998292		23086	
0200230056390000	36825883						
1759404	3		9	SCOTT	saturn	pts/0	6
0							
SCOTT	E	MP	oracle10	2007-09-17 12:43:51.380061		23086	
0200230056390000	36825885						

Many of the details here are coded; Oracle gives you many views and code tables that are much more useful. These are created by running \$ORACLE\_HOME/rdbms/admin/cataudit.sql and include:

- DBA\_AUDIT\_TRAIL—A user friendly version for AUD\$ with more readable data, for example
  - DBA\_AUDIT\_TRAIL.OS\_USER comes from AUD\$.SPARE1
  - DBA\_AUDIT\_TRAIL.ACTION\_NAME has a readable version of the code stored in AUD\$.ACTION#
- USER\_AUDIT\_TRAIL—A view based on DBA\_AUDIT\_TRAIL that shows audit records caused by the current connected user.
- DBA\_AUDIT\_EXISTS—A view based on DBA\_AUDIT\_TRAIL that shows failed operations due to nonexisting objects.
- DBA\_AUDIT\_SESSION—A view based on DBA\_AUDIT\_TRAIL that shows connects (including failed connects) and disconnects.
- DBA\_AUDIT\_STATEMENT—A view based on DBA\_AUDIT\_TRAIL that shows statements that do not access data in objects such as ALTER SYSTEM, GRANT, etc.
- DBA\_AUDIT\_OBJECT—A view based on DBA\_AUDIT\_TRAIL that shows statements that DO access objects such as DML statements.

There is a lot of information in DBA\_AUDIT\_TRAIL. Some of the columns are self-explanatory. Others include

- TERMINAL—On Unix this is the pts, on Windows this is the machine name.
- ACTION—The numeric code of the action performed by the user (see AUDIT\_ACTIONS).
- ACTION\_NAME—The name used for the action performed by the user (see AUDIT\_ACTIONS).
- NEW\_OWNER—if the object is renamed, the owner of the new object.
- NEW\_NAME—if the object is renamed, the new name of the object.
- OBJ\_PRIVILEGE—if the audited action was a grant or revoke of a object privilege, records what object privileges were acted upon.
- SYS\_PRIVILEGE—Same but for system privilege.
- ADMIN\_OPTION—Indicates if the privilege was granted with ADMIN\_OPTION (see Chapter 15).
- GRANTEE—if the audited action is a grant or privilege or role, the name of the grantee (user or role).
- AUDIT\_OPTION—When the audited action is a AUDIT statement the option is recorded.
- SES\_ACTIONS—if the auditing action is on an object, the audit information is recorded here.
- LOGOFF\_TIME—if the auditing was for a user session, the logoff time.
- LOGOFF\_LREAD—Logical blocks read by the session.
- LOGOFF\_PREAD—Physical blocks read by the session.
- LOGOFF\_LWRITE—Logical blocks written by the session.
- LOGOFF\_DLOCK—Number of deadlocks detected while in the session.
- COMMENT\_TEXT—Authentication details such as whether it was the database that authenticated the user or some external authentication server, client connection information etc. An example value is
  - Authenticated by: DATABASE; Client address: (ADDRESS=(PROTOCOL=tcp)(HOST=192.168.222.1)(PORT=1325))

- SESSIONID—Audit session id (this is not the same as session id from V\$SESSION).
- ENTRYID—If a single statement causes multiple audit events, this is the sequence.
- STATEMENTID—A unique id for statements per session. 0 if the audit event is not related to a statement.
- RETURNCODE—Zero if no error or the Oracle error number.
- PRIV\_USED—The exact privilege used to perform the action.
- CLIENT\_ID—Client identifier for the session—relevant to identity propagation. If the application calls `exec dbms_session.set_identifier('RON')` and propagates the end-user credentials (at the application level) then the audit trail will include this information and you can use it to know not only which user was logged onto the database but also which user was logged onto the application:

```
select OS_USERNAME,USERNAME,CLIENT_ID,OBJ_NAME,ACTION_NAME from
dba_audit_object where OBJ_NAME='EMP'
```

OS_U	SERNAME	USERNAME	CLIENT_ID	OBJ_NAME	ACTION_NAME
L	3AD029\RBNatn	SCOTT	<u>RON</u>	EM	P

- The System Change Number (SCN)—Important when you want to use flashback queries (see Chapter 12).

In addition to DBA\_AUDIT\_TRAIL, you can use DBA\_COMMON\_AUDIT\_TRAIL. As Figure 9.1 shows, this view is a master view that aggregates all audit information that you may have. DBA\_AUDIT\_TRAIL is the view where all standard auditing records appear when you

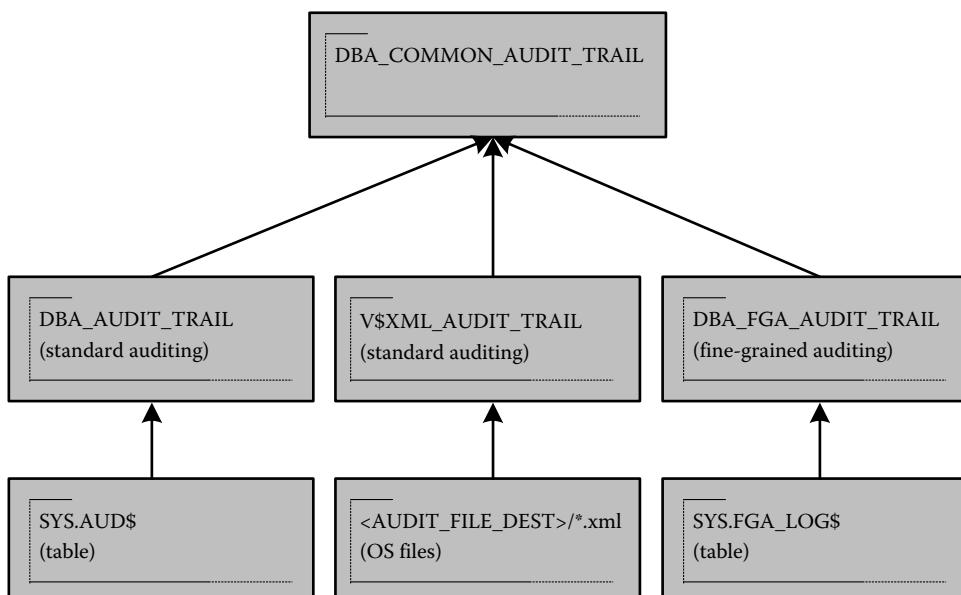


Figure 9.1 Tables and views used to view audit records.

use DB auditing. V\$XML\_AUDIT\_TRAIL is a view that is built on the OS XML files created when you use XML auditing. DBA\_FGA\_AUDIT\_TRAIL is a view that shows all audit records generated by FGA (see Chapter 11). DBA\_COMMON\_AUDIT\_TRAIL allows you to view audit records in one central place regardless of where they physically sit. Records in this view will specify their source in the AUDIT\_TYPE column:

AUDIT_TYPE	S ESSION_ID	P RXY_SESSIONID	S TATEMENTID	E NTRYID
EXTENDED_TIMESTAMP	G LOBAL_UID	D B_USER	C LIENT_ID	E CONTEXT_ID
EXT_NAME	O S_U SER	U SERHOST	O S_PROCESS	T ERMINAL
INSTANCE_NUMBER	O BJ_ECT_SCHEMA	O BJECT_NAME	P OLICY_NAME	N EW_OWNER
NEW_NAME	A CTION	S TATEMENT_T YPE	A UDIT_O PTION	T RANSACTIONID
RETURNCODE	S CN	C OMMENT_T EXT		
SQL_BIND	S QL_T EXT			
OBJ_PRIVILEGE	S YS_P RIVILEGE		A DMIN_O PTION	O S_P RIVILEGE
PRIV_USED	S ES_A CTIONS		L OGOFF_T IME	L OGOFF_L READ
LOGOFF_PREAD	L OGOFF_L WRITE		L OGOFF_DLOCK	S ESSION_C PU
<hr/>				
<b>Standard Audit</b>	1 000	(null)	195	5
9/19/2007 12:57:02 AM	(null)	S YSTEM	(null)	(null)
(null) L	3AD029	3028:660	(null)	0
EMP (n	ull)	(null)	(null)	FLASHBACK
(null) (n	ull)	0	4116511	(null)
(null) (n	ull)	(null)	N ONE	(null)
ANY TABLE	(null)	(null)	(null)	(null)
(null) (n	ull)	(null)		
<b>Fine Grained Audit</b>	633	(null)	15	2
6/7/2007 1:39:36 PM	(null)	S COTT	(null)	(null)
L3AD029\RBNatan	L3AD029\RBNatan	WORKGROUP\L3AD029	2460:4412	(null)
SCOTT E	M P	E MP_P OL	(null)	(null)
UPDATE (	null)	09001c0058020000	(null)	1763513
(null)	update emp set sal=901 where empno=7369			(null)
(null) (n	ull)	(n ull)	N ONE	(null)
(null) (n	ull)	(null)	(null)	(null)
(null) (n	ull)			

### Three Things to Remember about Viewing Audit Records:

1. Use DBA\_AUDIT\_TRAIL to view audit records generated by standard auditing configured as DB or DB, EXTENDED.
2. Use V\$XML\_AUDIT\_TRAIL to view audit information if you use XML-based OS audit files.
3. When you store the audit records in a database table you have views that show you subsets of the audit trail such as DBA\_AUDIT\_STATEMENT and DBA\_AUDIT\_OBJECT.

## 9.8 HOWTO View What Is Currently Being Audited

There are three views that show you what is currently being audited by the standard audit facility. DBA\_STMT\_AUDIT\_OPTS shows you statement auditing DBA\_PRIV\_AUDIT\_

OPTS sh ows yo u p rivilege aud iting a nd D BA\_OBJ\_AUDIT\_OPTS sh ows yo u o bject auditing.

DBA\_STMT\_AUDIT\_OPTS:

```
select * from dba_stmt_audit_opts
```

USER_NAME_P	ROXY_NAME	AUDIT_OPTION	SUCCESS	FAILURE
(null)	(null)	ALTER SYSTEM	BY ACCESS	BY ACCESS
(null)	(null)	SYSTEM AUDIT	BY ACCESS	BY ACCESS
(null)	(null)	CREATE SESSION	BY ACCESS	BY ACCESS
(null)	(null)	TABLE	BY ACCESS	BY ACCESS
(null)	(null)	CLUSTER	BY ACCESS	BY ACCESS
(null)	(null)	TABLESPACE	BY ACCESS	BY ACCESS
(null)	(null)	USER	BY ACCESS	BY ACCESS
(null)	(null)	ROLLBACK SEGMENT	BY ACCESS	BY ACCESS
(null)	(null)	TYPE	BY ACCESS	BY ACCESS
(null)	(null)	INDEX	BY ACCESS	BY ACCESS
(null)	(null)	CREATE USER	BY ACCESS	BY ACCESS
(null)	(null)	ALTER USER	BY ACCESS	BY ACCESS
(null)	(null)	DROP USER	BY ACCESS	BY ACCESS
(null)	(null)	SYNONYM	BY ACCESS	BY ACCESS
(null)	(null)	PUBLIC SYNONYM	BY ACCESS	BY ACCESS

## DBA PRIV AUDIT OPTS:

```
select * from dba_priv_audit_opts
```

USER_NAME_P	ROXY_NAME	PRIVILEGE	SUCCESS	FAILURE
(null)	(null)	ALTER SYSTEM	BY ACCESS	BY ACCESS
(null)	(null)	AUDIT SYSTEM	BY ACCESS	BY ACCESS
(null)	(null)	CREATE SESSION	BY ACCESS	BY ACCESS
(null)	(null)	CREATE USER	BY ACCESS	BY ACCESS
(null)	(null)	ALTER USER	BY ACCESS	BY ACCESS
(null)	(null)	DROP USER	BY ACCESS	BY ACCESS
(null)	(null)	CREATE TABLE	BY ACCESS	BY ACCESS
(null)	(null)	CREATE ROLE	BY ACCESS	BY ACCESS
(null)	(null)	CREATE PROCEDURE	BY ACCESS	BY ACCESS
SCOTT	(null)	CREATE TABLE	BY ACCESS	BY ACCESS
SCOTT	(null)	CREATE ANY TABLE	BY ACCESS	BY ACCESS

## DBA OBJ AUDIT OPTS:

```
select * from dba_obj_audit_opts;
```

Each record shows which statements are audited per object. Each value takes the form of [A|S]/[A|S] where A encodes BY ACCESS and S encodes BY SESSION, the left hand side value specifies what happens on success and the right hand side value specifies what happens on failure. For example, given the above record, if you stop auditing ALTER statements:

```
SQL> noaudit alter on scott.emp;  
Noaudit succeeded.
```

Then the record changes to

```
select * from dba_obj_audit_opts;
```

OWNER	OBJ	ECT_NAME	OBJ	ECT_TYPE	ALT	AUD	COM	DEL	GRA	IND	
INS	L	OC	REN	SEL	UPD	REF	EXE	CRE	REA	WRI	FBK
SCOTT	E	MP		TABLE	-/-	/A	A/A	S/S	A/A	A/A	
S/S	A	/A	A/A	A/A	S/S	-/-	-/-	-/-	-/-	A/A	

## 9.9 HOWTO Use NOAUDIT

Disable audit commands using NOAUDIT. For example, if you execute AUDIT ALL and then decide this is not required, you run NOAUDIT ALL. You don't use the access/session qualifier when you use NOAUDIT, for example:

```
SQL> noaudit create table by access;
noaudit create table by access
      *
ERROR at line 1:
ORA-01718: BY ACCESS | SESSION clause not allowed for NOAUDIT
SQL> noaudit create table;
Noaudit succeeded.
```

You can use the WHENEVER qualifier to change the cases in which an audit record is created:

SCOTT E MP\_DETAILS VIEW  
-/- -/- -/- A/A -/- -/- A/A -/- -/- **A/A** S/A -/- -/- -/- -/- -/- -/-

```
SQL> noaudit select on scott.emp_details whenever successful;  
Noaudit succeeded.
```

SCOTT\_E MP\_DETAILS VIEW  
-/- -/- -/- A/A -/- -/- A/A -/- -/- -/A S/A -/- -/- -/- -/- -/- -/-

What is very confusing for most people that start using standard auditing is that the AUDIT and NOAUDIT statements are not cumulative. Oracle does not generate a single concise profile of what to audit and adjusts it when you use AUDIT and NOAUDIT. The clearest example of this occurs when you execute NOAUDIT ALL. Contrary to what you may expect, this will not stop all auditing. It will only stop global AUDIT statements such as AUDIT ALL or AUDIT NOT EXIST. But if for example you have the following:

```
SQL> select * from dba_stmt_audit_opts;
USER_NAME_P          ROXY_NAME
-----
AUDIT_OPTION SU      CCESS      FAILURE
-----
SCOTT
CREATE ANY TABLE     BY ACCESS   BY ACCESS
SCOTT
CREATE SESSION       BY ACCESS   BY ACCESS
```

And you execute:

```
SQL> noaudit all;
Noaudit succeeded.
```

You will still be auditing:

```
SQL> select * from dba_stmt_audit_opts;
USER_NAME_P          ROXY_NAME
-----
AUDIT_OPTION SU      CCESS      FAILURE
-----
SCOTT
CREATE ANY TABLE     BY ACCESS   BY ACCESS
SCOTT
CREATE SESSION       BY ACCESS   BY ACCESS
```

Oracle handles each of your audit specifications individually. Therefore, when you want to disable certain audits you need to reverse your calls to AUDIT with equivalent calls to NOAUDIT. In the previous example, you have to explicitly call:

```
SQL> noaudit create session by scott;
Noaudit succeeded.

SQL> noaudit create any table by scott;
Noaudit succeeded.
```

So Now:

```
SQL> select * from dba_stmt_audit_opts;
no rows selected
```

This is even true when you use user names as qualifiers, e.g.,

```
SQL> select * from dba_stmt_audit_opts;
no rows selected

SQL> audit create session by scott;
Audit succeeded.
```

```

SQL> select * from dba_stmt_audit_opts;
USER_NAME_P          ROXY_NAME
-----
AUDIT_OPTION_SU      CCESS      FAILURE
-----
SCOTT
CREATE SESSION        BY ACCESS   BY ACCESS

SQL> noaudit create session;
Noaudit succeeded.

SQL> select * from dba_stmt_audit_opts;
USER_NAME_P          ROXY_NAME
-----
AUDIT_OPTION_SU      CCESS      FAILURE
-----
SCOTT
CREATE SESSION        BY ACCESS   BY ACCESS

SQL> noaudit create session by scott;
Noaudit succeeded.

SQL> select * from dba_stmt_audit_opts;
no rows selected

```

## 9.10 Discussion—Auditing and Performance

The first question everyone asks is how auditing affects database performance. Understanding the impact that audit trails have on your database is one of the most important things—especially if you have a high-volume environment and you have stringent auditing requirements that include auditing activities that happen a lot. The bad news is that it is hard to get a single number or formula that you can always equate to your environment and know up-front what to expect. For example, in a presentation titled “Does VPD, FGA or audit really cause performance issues” delivered by Pete Finnigan at the Oracle UK User Group in 2006 ([www.insight.co.uk/files/presentations/Does%20VPD,%20FGA%20or%20Audit%20Cause%20Performance%20Issues.pdf](http://www.insight.co.uk/files/presentations/Does%20VPD,%20FGA%20or%20Audit%20Cause%20Performance%20Issues.pdf)) he quotes the performance impact of standard auditing to be anywhere between 8 and 200 percent. You were probably hoping for a straight answer, but unfortunately life is more complicated than that—it means that you need to evaluate how much you will be auditing and that you need to run your own tests when you change your auditing profile. More work every time an auditor changes their mind. That’s the bad news. The good news is that auditing is a fairly simple operation so the impact is directly proportional to how much you audit.

There are some other numbers out there—but surprisingly not too many. Another set of numbers come from a presentation by Dave Moore presented at the Twin Cities Oracle user Group ([www.tcoug.org/Archive/Winter2007/Auditing\\_Options.ppt](http://www.tcoug.org/Archive/Winter2007/Auditing_Options.ppt)) where he shows that a database that did 50K inserts in 17 s took 2 min to perform the same 50K inserts when auditing all these inserts using DB\_EXTENDED. This is the extreme case, where everything is logged. Finally here’s another set of measurements that I did on a 28 CPU Sun E6500 with 28 GB of memory running Solaris 9. In this test I had 100 concurrent connections each doing real application-level transactions (that sometimes involve multiple tables) and that sleep for 100 ms between each such

transaction. Without auditing the throughout was 19,000 per minute and the load average on the machine was 7.2. With auditing of a fourth of the transaction types the throughout went down to 13,000 transactions per minute (a 30 percent drop) and the load average on the server was 14.5 (double).

None of these numbers are very scientific, so don't read too much into them. The bottom line is that auditing does affect performance and that the impact is proportional to how much you audit. This is not surprising—Oracle needs to write audit records to the file systems and that takes time. I wish there was a simpler answer; the important points to remember are

- Understand your requirements and make sure you audit what you need. Impact on performance is related to how much you audit so try to audit less (as long as it does not have an adverse impact to database security or your ability to pass an audit). This will also make your reviewing process less tedious, will require less disk space, etc.
- If your requirements or implementation changes and you change your audit policy in a substantial way, you must go through an entire change management process and comprehensive testing to avoid surprises in production.
- If you want to avoid this overhead (both on the server and more importantly on the rollout of changes to policies) or if you have extreme audit requirements including auditing of DML or SELECT you either have to accept the performance impact or consider an auditing solution that is not based on the database doing more I/O. You'll learn about such Database Activity Monitoring (DAM) solutions in Chapter 14.
- Finally, remember that the performance impact is not only in the generation of the audit trail itself. The audit trail needs to be moved elsewhere—it can't stay in the database or on the OS. It should be moved fairly quickly for better separation of duties. This means that you also need a process that keeps reading these records, copying them elsewhere, and deleting them. That will add to the impact on performance.

## *Chapter 10*

---

# Mandatory and Administrator Auditing

---

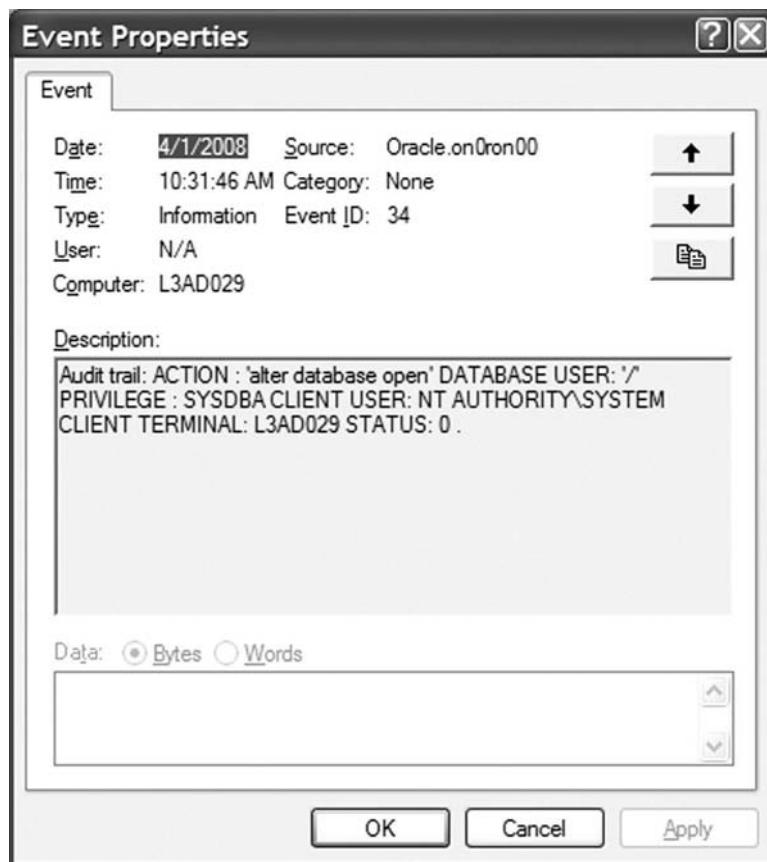
This is the shortest chapter in the book and it covers two simple categories of auditing. Although simple, they are of utmost importance. In all initiatives you will implement or you will be required to audit privileged user activity—and who more privileged than SYS or other users with SYSDBA privileges? This chapter shows you how to audit SYS activities and other important activities such as system startup and shutdown.

These HOWTOs were not included in Chapter 9 because administrator and mandatory auditing are indeed separate mechanisms; they are not part of standard auditing nor part of fine-grained auditing. You can enable standard auditing and you will still not get any audit records related to SYS activity. On the flip side, you can disable all auditing using NOAUDIT, but if you're auditing administrators then all SYS activities will still be logged. Finally, irrespective of where you choose to store the standard audit trail (DB or OS files) and what format you choose (e.g., XML), SYS auditing is always generated in files using a fixed format or sent to operating system log daemons such as syslog on Unix and the event log on Windows.

### **10.1 HOWTO Use Mandatory Auditing**

Some activities are always audited even if you don't explicitly ask for them. These activities include all database startups, all database shutdowns, and all sysdba or sysoper logons. This is called mandatory auditing. On Unix these audit records go into the adump directory (by default) and on Windows these audit records are written to the Windows event log regardless of what you set the AUDIT\_TRAIL parameter to.

Figure 10.1 shows a sample event in the Windows Event Viewer, in this case for a database startup. Figure 10.2 shows an event generated for a “/ as sysdba” connection on Windows. On Unix a shutdown after connecting as sysdba will produce the following two audit records in a single file in the adump directory:




---

**Figure 10.1 Mandatory auditing—Windows event for database startup.**

```

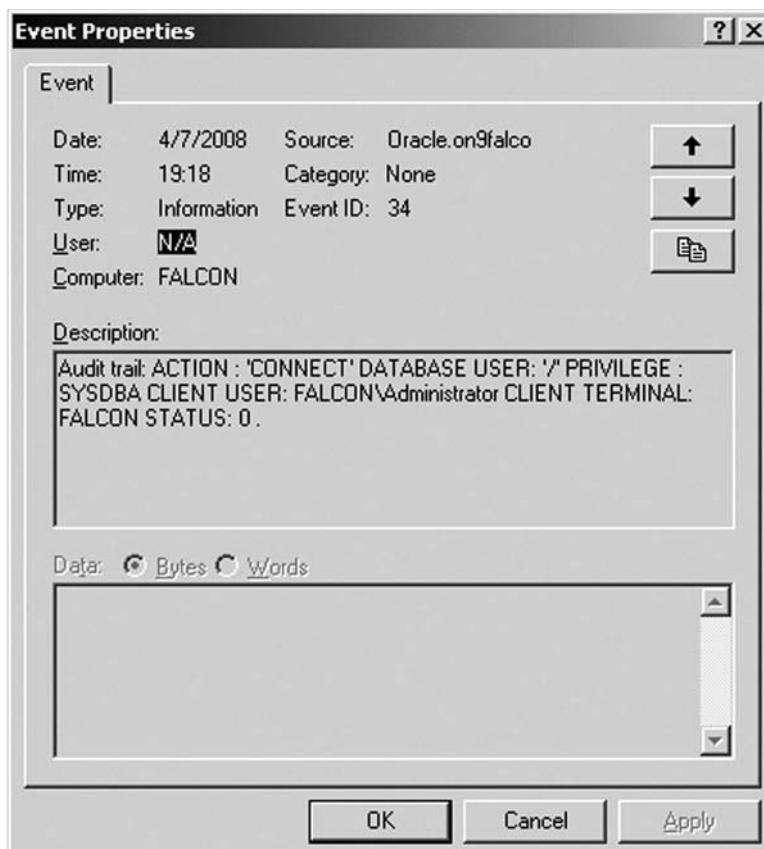
Audit file /home/oracle10/admin/on0jumbo/adump/ora_25953.aud
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
ORACLE_HOME = /home/oracle10/product/10.2.0/db_1
System name: Linux
Node name: jumbo.guardium.com
Release: 2 .6.9-55.0.2.ELsmp
Version: #1 SMP Tue Jun 12 17:59:08 EDT 2007
Machine: i 686
Instance name: on0jumbo
Redo thread mounted by this instance: 1
Oracle process number: 17
Unix process pid: 25953, image: oracle@jumbo.guardium.com

```

```

Mon Apr 7 19:13:02 2008
ACTION : 'CONNECT'
DATABASE USER: '/'
PRIVILEGE : SYSDBA
CLIENT USER: oracle10
CLIENT TERMINAL: pts/1
STATUS: 0

```



**Figure 10.2 Mandatory auditing—Windows event for sysdba logon.**

```
Mon Apr 7 19:15:12 2008
ACTION : 'SHUTDOWN'
DATABASE USER: '/'
PRIVILEGE : SYSDBA
CLIENT USER: oracle10
CLIENT TERMINAL: pts/1
STATUS: 0
```

Connecting using “/ as sysoper” produces an audit record of the form:

```
Audit file /home/oracle10/admin/on0jumbo/adump/ora_26182.aud
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
ORACLE_HOME = /home/oracle10/product/10.2.0/db_1
System name: Linux
Node name: jumbo.guardium.com
Release: 2 .6.9-55.0.2.ELsmp
Version: #1 SMP Tue Jun 12 17:59:08 EDT 2007
Machine: i 686
Instance name: on0jumbo
Redo thread mounted by this instance: 1
Oracle process number: 15
Unix process pid: 26182, image: oracle@jumbo.guardium.com (TNS V1-V3)
```

```
Mon Apr  7 19:17:15 2008
ACTION : 'CONNECT'
DATABASE USER: '/'
PRIVILEGE : SYSOPER
CLIENT USER: oracle10
CLIENT TERMINAL: pts/1
STATUS: 0
```

Note that these audit records do not include any details on the activity done by those connected with sysdba or sysoper privileges, just the fact that the connections occurred. For this you will have to turn to the next HOWTO.

### Other Audit Records That Are Always Written

Another set of activities that are always recorded no matter what you configure are changes to the audit trail itself. Any Data Manipulation Language (DML) statements performed on SYS.AUD\$ and SYS.FGA\_LOG\$ (see Chapter 11) are logged to SYS.AUD\$.

#### Four Things to Remember about Mandatory Audit:

1. Startup of the database, shutdown, and logon using sysdba and sysoper privileges are always logged in OS files regardless of what audit trail you configure.
2. In Windows, these audit records are written to the application event log.
3. Mandatory auditing only audits the fact that an administrator logged on, it does not audit what they did.
4. Although these audit files are small, they can add up. Make sure to rotate these files and archive/purge them on a regular basis.

## 10.2 HOWTO Enable Administrator Auditing

Mandatory auditing logs SYS connections but does not show you what the user SYS did. Standard auditing itself does not write audit records for SYS. For example, set an audit trail:

```
SQL> show parameter audit_trail
```

NAME	TYPE	PE	VALUE
audit_trail	string		DB

Now logon as SYS and see how many audit records are there

```
-bash-3.00$ sqlplus / as sysdba
SQL*Plus: Release 11.1.0.6.0 - Production on Mon Apr  7 19:35:13 2008
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL> select count(*) from aud$;
COUNT(*)
-----
-----
```

Set an audit rule:

```
SQL> audit all on scott.emp;
Audit succeeded.
```

Now access the table (still as the user SYS):

```
SQL> select count(*) from scott.emp;
COUNT(*)
-----
14
```

The audit trail will not be updated:

```
SQL> select count(*) from aud$;
COUNT(*)
-----
1578
```

To enable administrator auditing you need to set another parameter:

```
SQL> show parameter audit;
```

NAME	TY	PE	VALUE
audit_file_dest	s	tring	/var/oracle11/admin/onlcondo/a dump
<u><b>audit_sys_operations</b></u>		<u><b>boolean</b></u>	<u><b>FALSE</b></u>
audit_syslog_level	s	tring	
audit_trail	s	tring	DB

Enable administrator auditing by setting AUDIT\_SYS\_OPERATIONS:

```
SQL> alter system set audit_sys_operations=true scope=spfile;
System altered.
```

Then, shutdown and startup the database. At this point ALL SYS-issued statements are audited indiscriminately. The data is not going to the standard audit trail, it is going to separate files in the audit\_file\_dest directory. For example, if after setting AUDIT\_SYS\_OPERATIONS to TRUE and restarting the database you perform:

```
SQL> select count(*) from aud$;
COUNT(*)
-----
1582
SQL> select count(*) from scott.emp;
COUNT(*)
-----
14
```

```
SQL> select count(*) from aud$;
COUNT(*)
-----
1582
```

However, you get the audit trail shown in Listing 10.1.

#### **The e Things to Remember about Mandatory Audit:**

1. Both mandatory auditing and standard auditing do not log activity performed by SYS. To enable SYS auditing set AUDIT\_SYS\_OPERATIONS to TRUE and restart the database.
2. SYS auditing is always logged to operating system files and defaults to the dump directory.
3. SYS auditing logs indiscriminately—make sure that you have enough space in AUDIT\_FILE\_DEST, that you don't run heavy batches or jobs using sysdba or sysoper privileges, and that you rotate these files periodically.

### **10.3 HOWTO Use Syslog Auditing**

Syslog is a standard for forwarding log messages in an Internet Protocol (IP) network. The term “syslog” is often used for both the actual syslog protocol, as well as the application or library sending syslog messages. Syslog is often used for system management and security auditing. It is supported by a wide variety of devices and receivers across multiple platforms and is therefore often used to integrate log data from many different types of systems into a central repository.

On Unix you can send administrator audit records (as well as standard audit records) to syslog. This is the Unix equivalent to the use of the Windows event log. The advantage of using syslog is that the adump files can be modified by the instance account owner whereas syslog files are usually owned by root. Moreover, syslog is often used by Security Event Managers (SEMs) and other log aggregators and correlation engines.

The syslog protocol is a client/server protocol. The syslog client sends a small textual message (less than 1024 bytes) to the syslog server. The receiver is commonly called “syslogd,” “syslog daemon,” or “syslog server.” Syslog messages can be sent via User Datagram Protocol (UDP) or Transmission Control Protocol (TCP). The data is sent in cleartext however although wrappers such as Stunnel, sslio, or sslwrap can be used to encrypt syslog messages. You can also use syslog-*ng* to encrypt the messages. When you configure Oracle to use syslog, Oracle calls a syslog library to write its audit records. From a syslog perspective Oracle is acting as a client and syslogd is acting as the server.

Before you configure syslog auditing you should know what you have configured in /etc/syslog.conf. Syslog.conf controls how the syslog daemon logs events. Every syslog message has two attributes—a facility and a priority. Syslogd.conf defines what to do with messages per facility and priority. A sample config file is shown below:

```
# cat /etc/syslog.conf
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
kern.*                                     /dev/console
```

```
Audit file /var/oracle11/admin/onlcondo/adump/ora_1060894.aud
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
ORACLE_HOME = /var/oracle11/product/11.1.0/db_1
System name: AIX
Node name: condor
Release: 3
Version: 5
Machine: 0000853F4C00
Instance name: onlcondo
Redo thread mounted by this instance: 1
Oracle process number: 18
Unix process pid: 1060894, image: oracle@condor (TNS V1-V3)

Mon Apr 7 19:51:47 2008
ACTION : 'CONNECT'
DATABASE USER: '/'
PRIVILEGE : SYSDBA
CLIENT USER: oracle11
CLIENT TERMINAL: pts/0
STATUS: 0

Mon Apr 7 19:52:22 2008
ACTION : 'ALTER DATABASE OPEN'
DATABASE USER: '/'
PRIVILEGE : SYSDBA
CLIENT USER: oracle11
CLIENT TERMINAL: pts/0
STATUS: 0

Mon Apr 7 19:52:29 2008
ACTION : 'select count(*) from aud$'
DATABASE USER: '/'
PRIVILEGE : SYSDBA
CLIENT USER: oracle11
CLIENT TERMINAL: pts/0
STATUS: 0

Mon Apr 7 19:52:38 2008
ACTION : 'select count(*) from scott.emp'
DATABASE USER: '/'
PRIVILEGE : SYSDBA
CLIENT USER: oracle11
CLIENT TERMINAL: pts/0
STATUS: 0

Mon Apr 7 19:52:44 2008
ACTION : 'select count(*) from aud$'
DATABASE USER: '/'
PRIVILEGE : SYSDBA
CLIENT USER: oracle11
CLIENT TERMINAL: pts/0
STATUS: 0
```

---

**Listing 10.1 Sample administrator audit trail.**

```

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none /var/log/messages

# The authpriv file has restricted access.
authpriv.* /var/log/secure

# Log all the mail messages in one place.
mail.* -/var/log/maillog

# Log cron stuff
cron.* /var/log/cron

# Everybody gets emergency messages
*.emerg *

# Save news errors of level crit and higher in a special file.
uucp,news.crit /var/log/spooler

# Save boot messages also to boot.log
local7.* /var/log/boot.log

```

Each line in this file starts with a selector. A selector specifies a combination of a facility and a priority and then specifies where the messages are to be written. In this example, each line specifies a file but syslog can also redirect messages to remote syslog servers. For example, the first line specifies that all messages sent to the KERN facility are logged to the console no matter what their priority. The next line specifies that all messages sent to any facility with the info priority and that all messages sent to the mail, cron, or authpriv facilities with no priority are written to /var/log/messages. Priorities (also called severities) include EMERG, ALERT, CRIT, ERR, WARNING, NOTICE, INFO, and DEBUG. Facilities include AUTH, AUTHPRIV, CRON, DAEMON, FTP, KERN, LPR, MAIL, MARK, NEWS, SECURITY, SYSLOG, USER, UUCP, and LOCAL0–LOCAL7. To configure syslog auditing for Oracle you simply specify which facility and which priority Oracle should use for its messages.

Audit records in Oracle are written to syslog in two cases. For standard auditing you can set AUDIT\_TRAIL to OS and then set AUDIT\_SYSLOG\_LEVEL to a value specifying a facility and a priority. For administrator auditing (AUDIT\_SYS\_OPERATIONS = TRUE) just set the AUDIT\_SYSLOG\_LEVEL. For example, if your syslog.conf has a line of the form:

```
*.info;mail.none /var/adm/syslog/syslog.log
```

Logon to the database, set the two initialization parameters and restart the database:

```

-bash-3.2$ sqlplus / as sysdba
SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 8 17:55:24 2008
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL> alter system set audit_sys_operations=true scope=spfile;
System altered.
SQL> alter system set audit_syslog_level='local1.info' scope=spfile;
System altered.
SQL> shutdown
Database closed.

```

```

Database dismounted.
ORACLE instance shut down.
SQL> startup
ORACLE instance started.
Total System Global Area 521936896 bytes
Fixed Size          2124664 bytes
Variable Size       406848648 bytes
Database Buffers   109051904 bytes
Redo Buffers        3911680 bytes
Database mounted.
Database opened.

```

If you made a mistake in the facility or priority you get the following error:

```

SQL> startup
ORA-32028: Syslog facility or level not recognized

```

At this point, all SYS activity will be recorded by syslog, e.g.,

```

-bash-3.2$ sqlplus / as sysdba
SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 8 18:41:12 2008
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL> delete from aud$;
3460 rows deleted.

```

In the syslog file you get an audit record of the form:

```

Apr 4 09:44:59 vireo Oracle Audit: SESSIONID: "6784"
ENTRYID: "1" STATEMENT: "6" USERID: "SYS" USERHOST: "vireo"
TERMINAL: "pts/2" ACTION: "7" RETURNCODE: "0" OBJ$CREATOR: "SYS" OBJ$NAME:
"AUD$" SES$TID: "4564" OS$USERID: "oracle11"

```



## *Chapter 11*

---

# Fine-Grained Auditing

---

Fine-grained auditing (FGA) was introduced as a granular audit option for SELECT statements in Oracle 9i. It was then enhanced in 10g to also include Data Manipulation Language (DML). FGA was added to the Oracle database feature set not as a replacement for standard auditing, but as a way to address audit requirements which explicitly specify conditions for which an audit record needs to be created. Such requirements can sometimes be articulated based on data within certain columns or simply by which columns are being accessed. Using standard auditing you can specify what you want audited based on commands and objects. It's an "all or nothing" specification—you can't specify that you only want an audit record if the user accessed certain rows or certain columns. FGA augments this capability by allowing you to specify granular conditions which determine whether or not an audit record should be written to the audit trail.

FGA is delivered through the DBMS\_FGA package and setting up FGA policies requires EXECUTE privileges on this package. The audit records are written to a different table than standard auditing or to operating system files. When written to a database table they are written to SYS.FGA\_LOG\$. You can then use the DBA\_FGA\_AUDIT\_TRAIL view to look at the audit records.

One of the nice things with FGA is that you don't have a two-step initiation process. You don't need to set an initialization parameter and you do not need to restart the database, you just have to define policies.

Let's look at a simple example of the mechanics involved. In this example you won't even add a granular audit specification yet, so this type of auditing can be accomplished even with standard auditing. You'll just define an audit trail on the SCOTT.EMP table regardless of the access (SELECT or DML). Define an FGA audit policy:

```
SQL> begin
 2 dbms_fga.add_policy(
 3 object_schema=>'SCOTT',
 4 object_name=>'EMP',
 5 policy_name=>'EMP_ACCESS'
 6 );
 7 end;
 8 /
```

PL/SQL procedure successfully completed.

This tells Oracle to audit the SCOTT.EMP table. Since there is no condition, Oracle will audit all DML and SELECT access. Connect to the database and select from the table:

```
SQL> connect scott
Enter password: *****
Connected.

SQL> select * from emp where deptno=10;

  EMPNO ENAME      JOB              MGR HIREDATE        SAL     COMM   DEPTNO
----- -----      ---             ---    ---        ---     ---     ---
    7782 CLARK       MANAGER        7839 09-JUN-81     2450
    7839 KING        PRESIDENT      7782 17-NOV-81     5000
    7934 MILLER      CLERK         7782 23-JAN-82     1300
3 rows selected.
```

Now look at the audit record that is produced:

DB_USER	OS_U	SER	POLICY_NAME	TIMESTAMP	SQL_TEXT
SCOTT	L3AD029\RBNatAn		EMP_ACCESS	9/19/2007 12:08:06 AM	select * from emp where deptno=10

FGA only audits DML and SELECT. If you were to alter the table or do any other Data Definition Language (DDL) statement it would not appear in the FGA trail. SELECTs are audited even if access is done through views, for example:

```
SQL> create view v_emp as select * from emp;
View created.
```

```
SQL> select * from v_emp where deptno=20;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	02-APR-81	3599.75		20
7788	SCOTT	ANALYST	7566	19-APR-87	3630		20
7876	ADAMS	CLERK	7788	23-MAY-87	1331		20
7902	FORD	ANALYST	7566	03-DEC-81	3630		20
4444	RON	DEV	7902	01-NOV-02	1161.6	100	20

5 rows selected.

```
select db_user,os_user,policy_name,timestamp,object_name,sql_text from
dba_fga_audit_trail where policy_name='EMP_ACCESS'
```

DB_USER	OS_USER	POLICY_NAME	TIMESTAMP	OBJECT_NAME
SCOTT	L3AD029\RBNatAn	EMP_ACCESS	9/19/2007 12:08:06 AM	EMP
select * from emp where deptno=10				

```
SCOTT          L3AD029\RBNatan  EMP_ACCESS      9/19/2007 12:15:46 AM  EMP
select * from v_emp where deptno=20
```

FGA knows that the object accessed was EMP so the value in OBJECT\_NAME is EMP, but the SQL is recorded correctly as the access to the view. Of course, this is a very simplistic policy—the real value of FGA is in the granular policies you will learn about in the next HOWTO.

## 11.1 HOWTO Define FGA Policies

You create FGA policies using the DBMS\_FGA.ADD\_POLICY procedure. The full specification for this procedure is as follows:

```
DBMS_FGA.ADD_POLICY(
  object_schema        VARCHAR2,
  object_name          VARCHAR2,
  policy_name          VARCHAR2,
  audit_condition      VARCHAR2,
  audit_column         VARCHAR2,
  handler_schema       VARCHAR2,
  handler_module       VARCHAR2,
  enable               BOOLEAN,
  statement_types       VARCHAR2,
  audit_trail          BINARY_INTEGER IN DEFAULT,
  audit_column_OPTS    BINARY_INTEGER IN DEFAULT);
```

You already saw the use of object\_schema, object\_name, and policy\_name. Audit\_column allows you to specify a set of columns to be audited. You use audit\_column when you want to specify that an audit record should be created only when certain sensitive columns are accessed. For example, if only the SAL and the COMM columns are sensitive in the EMP table and you want to audit any changes or reads of salary or commission data, execute:

```
SQL> begin
 2 dbms_fga.add_policy(
 3 object_schema=>'SCOTT',
 4 object_name=>'EMP',
 5 policy_name=>'EMP_ACCESS',
 6 audit_column=>'SAL,COMM');
 7 end;
 8 /
```

PL/SQL procedure successfully completed.

At this point only access to the sensitive columns is audited, for example:

```
SQL> select db_user,os_user,policy_name,timestamp,object_name,sql_text from
dba_fga_audit_trail where policy_name='EMP_ACCESS';
no rows selected

SQL> connect scott
Enter password:
Connected.
```

```

SQL> select ENAME,JOB,MGR from emp where EMPNO=7369;
ENAME J      OB          MGR
-----
SMITH CLER    K          7902

SQL> select ENAME,JOB,MGR,SAL from emp where EMPNO=7369;
ENAME J      OB          MGR          SAL
-----
SMITH CLER    K          7902        800

select db_user,os_user,policy_name,timestamp,object_name,sql_text from
dba_fga_audit_trail where policy_name='EMP_ACCESS'

DB_USER OS_   USER     POLICY_NAME      TIMESTAMP      OBJECT_NAME      SQL_TEXT
-----  -----  -----  -----  -----  -----
SCOTT    oracle11  EMP_ACCESS  11/17/2007    EMP          select
ENAME,JOB,MGR,SAL from emp where EMPNO=7369

1 record(s) selected [Fetch MetaData: 16/ms] [Fetch Data: 0/ms]

```

As you can see, only the second SELECT caused an audit record to be written.

When you have more than one sensitive column you can control whether you want to audit access to any of the sensitive columns or only access that involves all of these columns. The default is to audit any access that involves any of these columns. To explicitly set the behavior use one of:

```

SQL> begin
 2  dbms_fga.add_policy(
 3  object_schema=>'SCOTT',
 4  object_name=>'EMP',
 5  policy_name=>'EMP_ACCESS',
 6  audit_column=>'SAL,COMM',
 7  audit_column_opts=>DBMS_FGA.ANY_COLUMNS);
 8 end;
 9 /

```

PL/SQL procedure successfully completed.

Or

```

SQL> begin
 2  dbms_fga.add_policy(
 3  object_schema=>'SCOTT',
 4  object_name=>'EMP',
 5  policy_name=>'EMP_ACCESS',
 6  audit_column=>'SAL,COMM',
 7  audit_column_opts=>DBMS_FGA.ALL_COLUMNS);
 8 end;
 9 /

```

PL/SQL procedure successfully completed.

You can control which statements this policy applies to. For example, if you want to specify that only inserts and updates are audited you can set:

```

begin
dbms_fga.add_policy(
object_schema=>'SCOTT',
object_name=>'EMP',
policy_name=>'EMP_ACCESS',
audit_column=>'SAL,COMM',
statement_types=>'INSERT, UPDATE',
audit_column_opts=>DBMS_FGA.ANY_COLUMNS);
end;
/

```

Contrary to standard auditing where you set the location of the audit trail up front, you specify the location of the audit trail when you define the policy. Use the AUDIT\_TRAIL parameter to control where the audit records are written. For example, if you want the audit records for this policy to be written to FGA\_LOG\$ and you want the audit trail to include the SQL text and the bind values, use:

```

begin
dbms_fga.add_policy(
object_schema=>'SCOTT',
object_name=>'EMP',
policy_name=>'EMP_ACCESS',
audit_column=>'SAL,COMM',
statement_types=>'INSERT, UPDATE',
audit_column_opts=>DBMS_FGA.ANY_COLUMNS,
audit trail=>DBMS_FGA.DB + DBMS_FGA.EXTENDED;
end;
/

```

If you want the audit records to be written to XML files and to include the SQL text and bind values:

```

begin
dbms_fga.add_policy(
object_schema=>'SCOTT',
object_name=>'EMP',
policy_name=>'EMP_ACCESS',
audit_column=>'SAL,COMM',
statement_types=>'INSERT, UPDATE',
audit_column_opts=>DBMS_FGA.ANY_COLUMNS,
audit trail=>DBMS_FGA.XML + DBMS_FGA.EXTENDED;
end;
/

```

In the later case, if you update the table:

```
SQL> update emp set sal=8000 where EMPNO=7369;
1 row updated.
```

You get the following audit record:

```
[oracle11@o11g adump]$ cat ora_7513.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Audit xmlns="http://xmlns.oracle.com/oracleas/schema/dbserver_audittrail-11_1.xsd"
      x:mln:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```

x si:schemaLocation="http://xmlns.oracle.com/oracleas/schema/dbserver_audittrail-
11_1.xsd">
  < Version>11.1</Version>
<AuditRecord><Audit_Type>2</Audit_Type><Session_Id>545805</Session_Id><StatementId>9</Sta-
tementId><EntryId>2</EntryId><Extended_Timestamp>2007-11-
17T03:11:39.998501</Extended_Timestamp><DB_User>SCOTT</DB_User><OS_User>oracle11</OS_User
><Userhost>o11g.guardium.com</Userhost><OS_Process>7513</OS_Process><Instance_Number>0</I-
nstance_Number><Object_Schema>SCOTT</Object_Schema><Object_Name>EMP</Object_Name><Policy_
Name>EMP_ACCESS</Policy_Name><Stmt_Type>4</Stmt_Type><TransactionId>05002100E70B0000
</TransactionId><Scn>3619381</Scn>
<Sql_Text>
update emp set sal=8000 where EMPNO=7369
</Sql_Text>
</AuditRecord>
</Audit>
```

Notice the Audit\_Type is 2 here; in Chapter 9 you saw a similar audit record for standard auditing where the type was 1.

## FGA Conditions

You've already seen some qualifiers that give you more granular control over what gets audited using FGA—for example, at a column level. The most powerful qualifier that allows you to fine-tune the auditing policy at a row level is the audit\_condition qualifier. This is especially powerful for SELECT statements and this is what FGA was born to do (back in 9i). For example, if you want to audit all SELECT statements that access data involving highly paid individuals:

```

begin
dbms_fga.add_policy(
object_schema=>'SCOTT',
object_name=>'EMP',
policy_name=>'EMP_ACCESS',
audit_column=>'SAL',
audit_condition=>'SAL>9999',
statement_types=>'SELECT',
audit_column_opts=>DBMS_FGA.ANY_COLUMNS,
audit_trail=>DBMS_FGA.DB + DBMS_FGA.EXTENDED);
end;
/
```

If the table contains this data:

7369	SMITH	MANAGER	7902	17-DEC-80	11000	1500	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20

And you perform the following two SELECT statements:

```
SQL> select * from emp where EMPNO=7369;
EMPNO ENAME      JOB          MGR HIREDATE      SAL     COMM    DEPTNO
----- -----
 7369 SMITH      MANAGER      7902 17-DEC-80   11000   1500     20
SQL> select * from emp where EMPNO=7499;
EMPNO ENAME      JOB          MGR HIREDATE      SAL     COMM    DEPTNO
----- -----
 7499 ALLEN      SALESMAN    7698 20-FEB-81   1600    300      30
```

You only get one audit record:

```
SQL> select SQL_TEXT from dba_fga_audit_trail;
SQL_TEXT
-----
select * from emp where EMPNO=7369
```

FGA is very flexible; the audit condition is a PL/SQL expression which allows you to implement pretty much any audit requirement on DML and SELECT at a row level. A NULL as the audit\_condition is interpreted as a null condition and will match every row. Do not use a condition such as 1=1 and do not use an empty string as a condition.

## Handlers

The final two qualifiers of an FGA policy are HANDLER\_SCHEMA and HANDLER\_MODULE. These qualifiers allow you to send a real time alert when an action that does not comply with your policy happens. A handler is a procedure that gets called when the policy fires. An audit record is still written to the audit trail but in addition the handler is fired.

Let's look at an example. You'll send an alert through e-mail every time someone accesses a record in the EMP table with a salary over 9999. First, create the handler that uses the UTL\_SMTP package to send out the e-mail. The e-mail includes the details of the policy violation along with who violated the policy and what the SQL statement was

```
create or replace procedure rt_alert(
  p_o      object_schema varchar2,
  p_o      object_name  varchar2,
  p_      policy_name  varchar2)
as
  v      _email_text varchar2(10000);
  v_email_host varchar2(100) := 'smtpout.secureserver.net';
  v_e      mail_connection utl_smtp.connection;
  v_from varchar2(100)  := 'fga@dbasecurity.com';
  v_to  varchar2(100)  := 'ron@dbasecurity.com';
begin
  v_email_text := 'Policy violation for ' ||
                 p_policy_name ||
                 ': User=' ||
```

```

US          ER ||
' Schema=' ||
p_obj      ject_schema ||
' Object=' ||
p_obj      ject_name ||
' Date=' ||
SY          SDATE ||
' SQL=' ||
sys_
           context('userenv', 'current_sql');
v_email_connection := utl_smtp.open_connection(v_email_host, 25);
u    tl_smtp.helo(v_email_connection, v_email_host);
u    tl_smtp.mail(v_email_connection, v_from);
u    tl_smtp.rcpt(v_email_connection, v_to);
utl_smtp.data(v_email_connection, utl_tcp.crlf ||
'Subject: Policy Violation' || utl_tcp.crlf ||
'To: ' || v_to || utl_tcp.crlf ||
v_email_text);
u    tl_smtp.quit(v_email_connection);
end;

```

Then, create the FGA policy using the handler qualifiers:

```

begin
db ms_fga.add_policy(
object_schema=>'SCOTT',
object_name=>'EMP',
policy_name=>'EMP_ACCESS',
statement_types=>'SELECT',
audit_condition=>'SAL>9999',
audit_trail=>DBMS_FGA.DB + DBMS_FGA.EXTENDED,
handler_schema=>'SECADM',
handler_module=>'RT_ALERT');
end;
/

```

### The e Things to Remember about Defining FGA Policies

1. FGA does not require a database restart to be enabled—you simply add a policy for auditing to start.
2. FGA gives you granular control over what to audit—at a statement level, at a column level, and at a row level.
3. A policy dynamically specifies where audit records should be written—to FGA\_LOG\$ or to operating system files.

## 11.2 HOWTO Manage FGA Policies

When you first add a policy it starts auditing based on your specification. You can temporarily disable the policy without dropping it using

```

SQL> begin
2 dbms_fga.disable_policy(
3 object_schema=>'SCOTT',

```

```
4 object_name=>'EMP',
5 policy_name=>'EMP_ACCESS
6 );
7 end;
8 /
```

To enable the policy and reenable auditing;

```
SQL> begin
 2 dbms_fga.enable_policy(
 3 object_schema=>'SCOTT',
 4 object_name=>'EMP',
 5 policy_name=>'EMP_ACCESS
 6 );
 7 end;
 8 /
```

To drop a policy completely:

```
SQL> begin
 2  dbms_fga.drop_policy(
 3  object_schema=>'SCOTT',
 4  object_name=>'EMP',
 5  policy_name=>'EMP_ACCESS'
 6  );
 7 end;
 8 /
```

## 11.3 HOWTO Read FGA Tables and Views

To see which policies are defined and enabled use DBA\_AUDIT\_POLICIES:

The POLICY\_TEXT column records the SQL text for the policy condition and the POLICY\_COLUMN records the columns in which the policy is defined. PF\_PACKAGE and PF\_FUNCTION record the handler schema and the handler module.

FGA records go into SYS.FGA\_LOG\$ or into operating system files. You saw a sample XML file and sample records earlier in the chapter. When using FGA\_LOG\$ you use the DBA\_FGA\_AUDIT\_TRAIL view to look at audit records. The structure of this view is very similar to the structure of DBA\_AUDIT\_TRAIL and it has pretty much the same columns as described in HOWTO 9.7. You can also use DBA\_COMMON\_AUDIT\_TRAIL (see Figure 9.1) to view all audit records including FGA audit records in the database and FGA audit records in XML files.

#### **Four Things to Remember about Reading the FGA Audit Trail**

1. FGA audit records can be stored in a data dictionary table or in XML operating system files.
2. Use DBA\_FGA\_AUDIT\_TRAIL to view audit records stored using DBMS\_FGA.DB.
3. Use V\$XML\_AUDIT\_TRAIL to view audit records stored in XML operating system files.
4. Use DBA\_COMMON\_AUDIT\_TRAIL to view audit records regardless of where they are stored (FGA records or standard auditing records are stored in the data dictionary or in XML files).

## **11.4 Discussion: FGA Performance**

As any topic related to auditing, the performance impact of FGA is usually the issue most people think about first. This is even more appropriate for FGA because when it first came out in Oracle 9i it required the use of the Cost Based Optimizer (CBO) rather than the Rule Based Optimizer (RBO) and you would actually get faulty records if you used the RBO. In Oracle 9i the CBO was not as mature as it is in 10g and 11g and there could be severe performance penalties with the CBO. This ended up giving FGA a very bad name, even though the real “culprit” was the CBO. In Oracle 10g and 11g, this is no longer an issue so you can assess the impact of FGA on performance in an isolated manner.

The directives for how to use FGA for auditing are similar to those discussed at the end of Chapter 9. Like standard auditing, FGA becomes an issue in terms of performance if you audit a lot. The same resources quoted at the end of Chapter 9 apply here. For example, Pete Finnigan’s presentation at the Oracle UK User Group in 2006 ([www.insight.co.uk/files/presentations/Does%20VPD,%20FGA%20or%20Audit%20Cause%20Performance%20Issues.pdf](http://www.insight.co.uk/files/presentations/Does%20VPD,%20FGA%20or%20Audit%20Cause%20Performance%20Issues.pdf)) quotes the impact of FGA as being between 3 and 300 percent. Nice sized range—what it means is that “it all depends”; it all depends on what you audit. That’s the whole reason for FGA, it was created so you can create a very granular definition to cut down and limit what you audit. That’s precisely the reason why it was first introduced for SELECT auditing and later enhanced with DML auditing. It is not used for DDL. It is used for exactly those statements that occur often. Remember, performance impact depends on the number of audit records per second you produce with your policies.

Dave Moore’s presentation presented at the Twin Cities Oracle user Group ([www.tcoug.org/Archive/Winter2007/Auditing\\_Options.ppt](http://www.tcoug.org/Archive/Winter2007/Auditing_Options.ppt)), because it uses a very simple example, conveys this point even better. In his test 50,000 inserts with no FGA policy took 17 seconds. 50,000 inserts with an FGA policy using a condition that did not fire on any of these inserts also took 17 seconds.

50,000 inserts with a condition that does apply took 120 seconds to complete when the audit trail was written to the FGA\_LOG\$ and 37 seconds when it was written to XML files.

Another test, using a Sun E6500 with 28 CPUS and 28 GB of memory running Solaris 9, ran 100 concurrent connections each doing real application-level transactions and sleeping for 100 ms between each such transaction. Without auditing the throughput was 19,000 per minute and the load average on the machine was 7.2. With FGA auditing of a fourth of the statements the throughput went down to 14,000 transactions per minute (a 20 percent drop) and the load average on the server was 11 (~40 percent increase).

As discussed in Chapter 9, none of these numbers are very scientific and unfortunately no one can tell you how it will affect your application. However, it does seem that FGA gives you better performance than standard auditing. More importantly (and the “main event” here) is that FGA allows you to define the precise condition when an audit record is produced and thus cut down on the impact and the number of records to review. Besides that, the main points to remember are the same as in Chapter 9, namely

- Understand your requirements and make sure you audit only what you need. Audit less by defining the appropriate FGA policies and conditions. This will also make your reviewing process less tedious, will require less disk space, etc.
- If your requirements or implementation changes and you change your audit policy in a substantial way, you must go through an entire change management process and comprehensive testing to avoid surprises in production.
- If your requirements for SELECT and DML auditing are still extreme and FGA conditions are still not going to cut the volume down considerably you either have to accept the performance impact or consider an auditing solution that is not based on the database doing more I/O. You’ll learn about such Database Activity Monitoring (DAM) solutions in Chapter 14.
- Remember that the performance impact is not only in the generation of the audit trail itself. The audit trail needs to be moved. You also need a process that keeps reading these records, copying them elsewhere and deleting them. That will add to the impact on performance.



## *Chapter 12*

---

# Auditing Before/After Values and Monitoring Selected Data

---

In Chapters 9 through 11 you saw various auditing options. All these options focus on showing the “who,” “what,” “when,” “where” based on a set of criteria. The criteria can be a command, an object, a privilege, any combination thereof, or even an arbitrary highly granular condition. What is common to all these techniques is that they are user-focused—i.e., they log what the user did. They give you the Structured Query Language (SQL) text that was performed by the user. They don’t give you which data was selected by the user or what data was modified by the user (in terms of what the value was before the change and what it is after the change). You can usually infer the “after value” from the SQL statement but getting the “before value” is a whole different ballgame. This chapter outlines the various options you have for getting before and after values and selected data as part of the audit trail.

The question of whether before and after values are really a valid and practical requirement is one that is open to debate and the jury is still out on whether this is truly required. Section 12.5 discusses this and presents both arguments. Regardless of what you believe, you will be happy to know that Oracle has mature support for getting before and after values in case you do need this as part of your audit trails.

### **12.1 HOWTO Use Triggers for Capturing Before/After Values**

A trigger is a named program unit that is stored in the database and fired in response to a specific event that occurs in the database. Triggers can be used to implement auditing, especially auditing of Data Manipulation Language (DML) on tables (and Oracle has specific Data Definition Language (DDL)-based system triggers). When you use DML triggers, the event that the trigger fires on is associated with a table or view. For example, if you want to audit DML activity on a certain table you can define an audit trigger that fires on INSERT, UPDATE, and DELETE on a table and, when fired, inserts an audit record into an audit trail table. This would not be AUD\$ or FGA\_LOG\$; it would be your custom audit trail table.

Triggers are often heavily used for application logic, sometimes overly used. As such, they are often viewed as negatively impacting the application, adding to the complexity of the applications,

potentially affecting performance, and are generally disliked by DBAs. Still, they are a method by which you can get very detailed information and you should at least be aware that this method exists and sometimes you will find that for very specific purposes and a very selective requirements triggers are simple to use and quite effective. In addition, triggers have two properties that can make them attractive when auditing is concerned:

1. Trigger-based auditing respects transactional boundaries. If you run a command that updates a table and then rollback the transaction, the audit trail will not respect the transactional boundary but the trigger will. For example, suppose you perform:

```
update emp set sal = sal * 1.05;
rollback;
```

If you are using standard auditing or fine-grained auditing an audit record is created even though nothing really has changed. On the other hand, if you are using triggers to insert an audit record into your audit table, the record will not be inserted because the trigger that is fired is part of the transaction. In security terms this means that audit records generated with triggers do not generate false positives. In the next HOWTO you'll see another technique which respects transactional boundaries but does not have the deficiencies that triggers have.

2. Standard and fine-grained auditing do not give you the before and after values (unless you use the System Change Number (SCN) and flashback queries, see HOWTO 12.3), triggers do this very easily.

Let's look at a simple example in which a trigger is used to record changes made to data in the SCOTT.EMP table. For each DML operation an audit record is produced specifying all the values in the operation, when the change was made and who did it. In addition, because the salary information is considered financially affecting, it may have to be rolled back to previous values if you determine that an unauthorized change was made—so you'll want to keep both the old and the new values in the audit table. Follow the sequence in this example:

Step 1: Create the audit table:

```
SQL> create table emp_audit(
  2   EMPNO NUMBER(4),
  3   ENAME VARCHAR2(10),
  4   JOB VARCHAR2(9),
  5   MGR NUMBER(4),
  6   HIREDATE DATE,
  7   SAL NUMBER(7,2), -- this will be the old value
  8   AFTER_SAL NUMBER(7,2),
  9   COMM NUMBER(7,2),
 10  DEPTNO NUMBER(2),
 11  user_name VARCHAR2(30),
 12  action VARCHAR2(30),
 13  upd_date DATE);
Table created.
```

The audit table contains columns for all data elements. It includes two columns for the salary: to contain the salary before the change and the salary after the change. It also includes a column for logging which database user made the change, when the change was made, and whether the action was an INSERT, an UPDATE, or a DELETE.

Step 2: Create the trigger:

```

SQL> CREATE OR REPLACE TRIGGER audit_emp
  2      AFTER INSERT OR
  3          DELETE OR
  4          UPDATE
  5      ON SCOTT.EMP
  6      FOR EACH ROW
  7      BEGIN
  8          IF INSERTING THEN
  9              INSERT INTO emp_audit VALUES (
10                  :new.empno,
11                  :new.ename,
12                  :new.job,
13                  :new.mgr,
14                  :new.hiredate,
15                  NULL,
16                  :new.sal,
17                  :new.comm,
18                  :new.deptno,
19                  User,
20                  'INSERT',
21                  Sysdate
22                  );
23      ELSE IF DELETING THEN
24          INSERT INTO emp_audit VALUES (
25              :old.empno,
26              :old.ename,
27              :old.job,
28              :old.mgr,
29              :old.hiredate,
30              NULL,
31              :old.sal,
32              :old.comm,
33              :old.deptno,
34              User,
35              'DELETE',
36              Sysdate
37              );
38      ELSE IF UPDATING THEN
39          INSERT INTO emp_audit VALUES (
40              :new.empno,
41              :new.ename,
42              :new.job,
43              :new.mgr,
44              :new.hiredate,
45              :old.sal,
46              :new.sal,
47              :new.comm,
48              :new.deptno,
49              User,
50              'UPDATE',
51              Sysdate
52              );
53      END IF;
54  END;
55  /
Trigger created.
```

This is a very simple and naïve AFTER trigger. It fires after an INSERT, UPDATE, or DELETE and for each such event will add a line to EMP\_AUDIT. If the action is an INSERT or an UPDATE it adds the new values and for an UPDATE it also adds the before value of the salary column. If the action is a DELETE then the row that was deleted is recorded (the old values).

Step 3: Make a few changes.

The values inside the EMP table before making the changes are:

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	8000		20
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2		20
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	40000		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	6050		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	1900		30
7902	FORD	ANALYST	7566	03-DEC-81	800		20
7934	MILLER	CLERK	7782	23-JAN-82	1573		10

11 rows selected.

Now make the changes, some as SCOTT and some as SYSTEM:

```
SQL> delete from emp where EMPNO=7844;
1 row deleted.
SQL> update emp set sal=sal+1000 where sal<1000;
2 rows updated.
SQL> connect system
Enter password:
Connected.
SQL> update scott.emp set sal=3000 where EMPNO=7521;
1 row updated.
```

Step 4: Look at the audit records:

```
SQL> select EMPNO,ENAME,JOB,SAL,AFTER_SAL,USER_NAME,ACTION,UPD_DATE from emp_audit;
```

EMPNO	ENAME	JOB	SAL	AFTER_SAL	USER_NAME	ACTION	UP_DATE
7844	TURNER	SALESMAN		1500	SCOTT	DELETE	09-FEB-08
7566	JONES	MANAGER	2	1002	SCOTT	UPDATE	09-FEB-08
7902	FORD	ANALYST	800	1800	SCOTT	UPDATE	09-FEB-08
7521	WARD	SALESMAN	1250	3000	SYSTEM	UPDATE	09-FEB-08

As you can see, this is quite simple. For those that are dead set against triggers, the next HOWTO describes a better scheme using Oracle Streams.

## 12.2 HOWTO Use Oracle Streams for Capturing Before/After Values

Triggers have been around forever, for better and for worse. Triggers are simple to use and are a “known factor” but many prefer shying away from them as an infrastructure for auditing before and after values. Luckily, Oracle has other infrastructure components for allowing you to capture and propagate such changes. These changes are based on the data that Oracle maintains within the redo logs. Whenever you make a change to data this change and the relevant values are recorded in the redo logs. One way to get at this data is to use LOG\_MINER. You can build a solution around LOG\_MINER, but there are better options, Oracle Streams is the way to go.

Oracle Streams allow you to capture and propagate changes to data as data streams either within a database or even between databases. It allows you to specify rules that affect what data goes into a stream, how that data is propagated among queues, and how to apply that data to another table. When you use streams you typically specify

- Events upon which you want to capture data—e.g., DML or DDL for certain tables.
- Where to stage this data and how to propagate this data between staging areas (queues).
- What data is captured per event and what data needs to be added to the data that was changed.
- How to apply that data from the staging area to the target table (which may be the same or different from the original table where the event occurred).

Oracle Streams are broadly used within many solutions. For example, they are used for Change Data Capture (CDC) and other data management needs. They were built as a single unified solution for addressing all data sharing and data propagation requirements, and auditing is one of them. In fact, as you’ll see in Chapter 13, Oracle Audit Vault uses Oracle Streams for the REDO collector that is used for capturing before and after values. The technique you’ll see in this HOWTO is precisely what Oracle Audit Vault does (apart from the fact that for simplicity, the example in the HOWTO logs it to a table within the same database and in Oracle Audit Vault the logging is done into a table in another database over a database link).

Let’s proceed with a detailed example. In this example you’ll use Oracle Streams to gather change records based on data changed in the course of INSERT, UPDATE, and DELETEs on the SCOTT.EMP table. You’ll build an audit table called EMP\_AUDIT where you will store these records. Because SAL is information that you consider critical and may financially affect your company, you would like to have the option to roll back any change made to salaries. For each change made to any records in EMP you will keep all the values but for the SAL column you will keep both the value before the change was made and the value after the change was made. The setup involves a queue (the staging area), the audit table (EMP\_AUDIT), a capture process that inserts change data into the queue, a handler procedure that is called to massage the data, and an apply process to put the data from the queue into EMP\_AUDIT. Follow this sequence to fully implement this example:

Step 1: Create a streams user and assign it privileges.

Create a user which you’ll use for the Streams:

```
SQL> create user gdmstreams identified by "0okmnhy6";
User created.
```

Grant the user privileges and grant specific Streams privileges. You can refine the privileges assigned to this user based on your environment:

```
SQL> grant dba to gdmstreams;
Grant succeeded.

SQL> begin
 2  DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE(
 3  grantee=>'gdmstreams',
 4  grant_privileges => true);
 5  end;
 6  /
PL/SQL procedure successfully completed.
```

**Step 2:** Create the audit table and grant privileges to the audit table and the EMP table.

Create the table where the audit records will reside:

```
SQL> create table emp_audit(
 2  EMPNO NUMBER(4),
 3  ENAME VARCHAR2(10),
 4  JOB VARCHAR2(9),
 5  MGR NUMBER(4),
 6  HIREDATE DATE,
 7  SAL NUMBER(7,2), -- this will be the old value
 8  AFTER_SAL NUMBER(7,2),
 9  COMM NUMBER(7,2),
10  DEPTNO NUMBER(2),
11  user_name VARCHAR2(30),
12  action VARCHAR2(30),
13  upd_date DATE);
Table created.
```

The table has three types of columns. It has a column for every column in EMP. Each record will have all the values affected by the DML operation. In addition, the AFTER\_SAL column will include the salary value after the DML operation and the SAL columns will include the salary value before the DML operation. Finally, because these are audit records, you must know who made the change, when the change was made, and whether the command was an INSERT, an UPDATE, or a DELETE. Usually this table would reside in an auditing schema and certainly not in the SCOTT schema itself. To keep this example simple the table is placed in the SCOTT schema. Grant permissions to the streams user both to access the EMP and the AUDIT\_EMP tables:

```
SQL> grant all on scott.emp_audit to gdmstreams;
Grant succeeded.

SQL> grant all on scott.emp to gdmstreams;
Grant succeeded.
```

**Step 3:** Create the queue:

```
SQL> begin
 2  DBMS_STREAMS_ADM.SET_UP_QUEUE(
 3  queue_table=>'gdmstreams.streams_queue_table',
 4  queue_name=>'gdmstreams.streams_queue');
 5  end;
 6  /
PL/SQL procedure successfully completed.
```

Step 4: Set up the capture process.

Streams support both DML and DDL changes. In this example you'll only be collecting DML changes:

```
SQL> begin
 2  DBMS_STREAMS_ADMIN.ADD_TABLE_RULES(
 3  table_name=>'scott.emp',
 4  streams_type=>'capture',
 5  streams_name=>'capture_emp',
 6  queue_name=>'gdmstreams.streams_queue',
 7  include_dml=>true,
 8  include_ddl=>false,
 9  inclusion_rule=>true);
10 end;
11 /
PL/SQL procedure successfully completed.
```

This will capture all the changed data. But you also want to capture “who did it” so you have to tell the capture process to include extra data:

```
SQL> begin
 2  DBMS_CAPTURE_ADMIN.INCLUDE_EXTRA_ATTRIBUTE(
 3  capture_name=>'capture_emp',
 4  attribute_name=>'username',
 5  include=>true);
 6 end;
 7 /
PL/SQL procedure successfully completed.
```

Tell Oracle to start the capture from the current system change number in your database. Insert your name for the database name below:

```
SQL> declare
 2  v_scn NUMBER;
 3  begin
 4  v_scn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER();
 5  DBMS_APPLY_ADMIN.SET_TABLE_INSTANTIATION_SCN(
 6  source_object_name=>'scott.emp',
 7  source_database_name=>'on0satur',
 8  instantiation_scn=>v_scn);
 9 end;
10 /
PL/SQL procedure successfully completed.
```

Step 5: Build a handler to massage your data. Probably the main use of Streams and CDC is data replication, propagation, and real-time extracts/loads. In these scenarios the data that needs to be copied to the other side has the same structure in the source and in the target. Not so with auditing—when you use Streams for auditing the audit records are similar to the original changed data but not identical. When you built the EMP\_AUDIT table you added a few extra columns such as the username and an extra column so you can store both the before and the after values for the salary data. When you get the logical change record (LCR) from the capture process it has the structure of the source table, EMP. Before you insert it into EMP\_AUDIT using the apply process you need to modify the LCR to have the structure of EMP\_AUDIT.

The EMP\_DML\_HANDLER procedure is shown in Listing 12.1. In line 8 you cast the LCR from the ANYDATA and in line 11 get the command from the LCR. The command will either be an INSERT, a DELETE, or an UPDATE. Regardless of what the original action is, an audit record needs to be inserted. Therefore, in line 14 you change the command type to INSERT to ensure that an audit record is created for every change. This really has nothing to do with the audit itself.

```

1 CREATE OR REPLACE PROCEDURE gdmstreams.emp_dml_handler(in_any IN ANYDATA) IS
2   lcr          SYS.LCR$_ROW_RECORD;
3   rc           PLS_INTEGER;
4   command      VARCHAR2(30);
5   new_sal      ANYDATA;
6   old_values   SYS.LCR$_ROW_LIST;
7 BEGIN
8   -- Access the LCR
9   rc := in_any.GETOBJECT(lcr);
10  -- Get the object command type
11  command := lcr.GET_COMMAND_TYPE();
12
13  -- Set the command_type in the row LCR to INSERT
14  lcr.SET_COMMAND_TYPE('INSERT');
15  -- Set the object_name in the row LCR to EMP_DEL
16  lcr.SET_OBJECT_NAME('EMP_AUDIT');
17
18  -- Set the new values to the old values for update and delete
19  IF command IN ('DELETE', 'UPDATE') THEN
20    IF command IN ('UPDATE') THEN
21      -- Get the new salary
22      new_sal := lcr.GET_VALUE('new','SAL');
23    END IF;
24    -- Get the old values in the row LCR
25    old_values := lcr.GET_VALUES('old');
26    -- Set the old values in the row LCR to the new values in the row LCR
27    lcr.SET_VALUES('new', old_values);
28    -- Set the old values in the row LCR to NULL
29    lcr.SET_VALUES('old', NULL);
30  END IF;
31
32  -- Add a SYSDATE for upd_date
33  lcr.ADD_COLUMN('new', 'UPD_DATE', ANYDATA.ConvertDate(SYSDATE));
34  -- Add a user column
35  lcr.ADD_COLUMN('new', 'user_name',
36                  lcr.GET_EXTRA_ATTRIBUTE('USERNAME') );
37  -- Add an action column
38  lcr.ADD_COLUMN('new', 'ACTION', ANYDATA.ConvertVarChar2(command));
39
40  -- Add the new salary information
41  lcr.ADD_COLUMN('new', 'AFTER_SAL', new_sal);
42
43  -- Make the changes
44  lcr.EXECUTE(true);
45  commit;
46 END;

```

---

**Listing 12.1 EMP\_DML\_HANDLER procedure.**

do with the INSERT that may have been made on EMP—this is the insert into the audit table. Because the audit records go into EMP\_AUDIT you set the object name in line 16. Next you need to take care of the values. If this is an INSERT statement then all the values are in the right place. But if this is a DELETE or an UPDATE value then all the values are the “old values” and they need to be copied to the new values in the LCR. This is done in lines 24 through 29. In addition, if this is an UPDATE statement, you also want to keep the salary information from before the change, this happens at line 22. And finally, lines 32 through 41 add the additional columns to the LCR.

#### Step 6: Create the DML handlers.

Tell Oracle to apply the handler for any INSERT, UPDATE, or DELETE made on the table:

```
SQL> begin
 2  DBMS_APPLY_ADMIN.SET_DML_HANDLER(
 3  object_name=>'scott.emp',
 4  object_type=>'TABLE',
 5  operation_name=>'INSERT',
 6  error_handler=>false,
 7  user_procedure=>'gdmstreams.emp_dml_handler',
 8  apply_database_link=>NULL,
 9  apply_name=>NULL);
10 end;
11 /
PL/SQL procedure successfully completed.

SQL> begin
 2  DBMS_APPLY_ADMIN.SET_DML_HANDLER(
 3  object_name=>'scott.emp',
 4  object_type=>'TABLE',
 5  operation_name=>'UPDATE',
 6  error_handler=>false,
 7  user_procedure=>'gdmstreams.emp_dml_handler',
 8  apply_database_link=>NULL,
 9  apply_name=>NULL);
10 end;
11 /
PL/SQL procedure successfully completed.

SQL> begin.
 2  DBMS_APPLY_ADMIN.SET_DML_HANDLER(
 3  object_name=>'scott.emp',
 4  object_type=>'TABLE',
 5  operation_name=>'DELETE',
 6  error_handler=>false,
 7  user_procedure=>'gdmstreams.emp_dml_handler',
 8  apply_database_link=>NULL,
 9  apply_name=>NULL);
10 end;
11 /
PL/SQL procedure successfully completed.
```

#### Step 7: Create the apply rule.

The handler procedure makes sure that your LCR has the right structure. You still need to create an apply rule to tell the Streams to take the LCRs from the queue and apply them:

```

SQL> declare
 2   emp_rule_name_dml VARCHAR2(30);
 3   emp_rule_name_ddl VARCHAR2(30);
 4   begin
 5     DBMS_STREAMS_ADMIN.ADD_TABLE_RULES(
 6       table_name=>'scott.emp',
 7       streams_type=>'apply',
 8       streams_name=>'apply_emp',
 9       queue_name=>'gdmstreams.streams_queue',
10       include_dml=>true,
11       include_ddl=>false,
12       source_database=>'on0satur',
13       dml_rule_name=>emp_rule_name_dml,
14       ddl_rule_name=>emp_rule_name_ddl);
15     DBMS_APPLY_ADMIN.SET_ENQUEUE_DESTINATION(
16       rule_name=>emp_rule_name_dml,
17       destination_queue_name=>'gdmstreams.streams_queue');
18   end;
19 /
PL/SQL procedure successfully completed.

```

You can define error handlers, but right now let's just continue even if there is an error (which is typical for auditing):

```

SQL> begin
 2   DBMS_APPLY_ADMIN.SET_PARAMETER(
 3     apply_name=>'apply_emp',
 4     parameter=>'disable_on_error',
 5     value=>'N');
 6   end;
 7 /
PL/SQL procedure successfully completed.

```

Step 8: Start the capture and apply processes:

```

SQL> begin
 2   DBMS_APPLY_ADMIN.START_APPLY(
 3     apply_name=>'apply_emp');
 4   end;
 5 /
PL/SQL procedure successfully completed.

SQL> begin
 2   DBMS_CAPTURE_ADMIN.START_CAPTURE(
 3     capture_name=>'capture_emp');
 4   end;
 5 /
PL/SQL procedure successfully completed.

```

Now let's test and see how it works. Connect as SCOTT and select from the table to see the data (data is shown below so you can see the before and after values):

```

SQL> connect scott
Enter password:
Connected.
SQL> select * from emp;

```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	1100		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2964.5		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	6051		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950	30	
7902	FORD	ANALYST	7566	03-DEC-81	30000		20
7934	MILLER	CLERK	7782	23-JAN-82	1573		10

13 rows selected.

Now make a few changes—some as SCOTT and some as SYSTEM:

```
SQL> update emp set sal=80 where EMPNO=7369;
1 row updated.
SQL> commit;
Commit complete.
SQL> connect system
Enter password:
Connected.

SQL> update scott.emp set SAL=300 where EMPNO=7902;
1 row updated.
SQL> commit;
Commit complete.
```

The resulting EMP\_AUDIT records are shown in Figure 12.1.

The screenshot shows the Oracle SQL Developer interface with a query window open. The query is:

```
select * from emp_audit
```

The results show two rows of audit data:

1	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	AFTER_SAL	COMM	DEPTNO	USER_NAME	ACTION	UPD_DATE
2	7902	(null)	(null)	(null)	(null)	30000	300	(null)	(null)	SYSTEM	UPDATE	2/6/2008 4:04:50 PM
3	7369	(null)	(null)	(null)	(null)	1100	80	(null)	(null)	SCOTT	UPDATE	2/6/2008 4:01:17 PM

Below the table, the message "2 record(s) selected [Fetch Metadata: 0/ms] [Fetch Data: 0/ms]" is displayed.

Figure 12.1 Sample audit records with before and after values.

### The e Things to Remember about Using Oracle Streams for Before/After Values

1. Typically you would create an audit table per table you want to audit for before/after values. For every column in the source table for which you want before/after values the audit table will have a column for the before value and a column for the after value. The audit table will also have a column for the extra information, such as which user performed the action.
2. The capture process makes sure that each DML is recorded and placed on the queue. The apply process takes the modified LCR and applies it to the audit table. This always means an insert into the audit table.
3. The handler is responsible for converting the LCR from the source table's structure to the audit table's structure.

## 12.3 HOWTO Use the SCN and Flashback Queries

Oracle has a set of features that are collectively known as Flashback and that allow you to view the past state of database objects. Flashback relies on the undo data of the transactions and uses the undo data to query past data. Most of the functions in Flashback are used as a data recovery mechanism such as recovering a table to its state at a previous point in time, recover a dropped table, return the database to an earlier point in time, etc. However, Flashback can be a useful tool for auditing—both for knowing a value before a certain transaction took place as well as knowing what data a user selected.

In Chapters 9 through 11 you saw that when you turn on both standard auditing and fine-grained auditing, the audit records include a SCN. The SCN is required when you want to use Flashback features because it serves as a pointer to the state of the database when the action was performed. Using the `SELECT.. AS OF SCN..` command you can look at the state of the table when the action with the relevant SCN was performed. In addition to using an SCN, Flashback lets you look at the past state of a database object based on time using the `SELECT.. AS OF TIMESTAMP..` command. These queries are called Flashback queries and they are useful in two main scenarios (relevant to security and auditing):

1. For extracting the before and after values for a certain activity.
2. For knowing what data a certain user saw when performing a certain command.

### 12.3.1 Notification Laws

The first scenario is obvious as you're already familiar with other methods for extracting the before and after values. The second scenario is important for investigations and when you have to comply with notification regulations. Notification regulations require companies to notify people if they suspect that personal or confidential information may have been accessed by unauthorized parties. Such notification requirements are very common these days, partly because of the huge cost of identity theft to the worldwide economy. In January 2007 the Home Office of the United Kingdom calculated the cost of identity theft to the British economy at \$3.2 billion. Estimates of the cost to the Australian economy are around \$3 billion per year.

The Federal Trade Commission estimates the impact to the U.S. economy to be \$50 billion per year. These are staggering costs to any economy. As a result, a lot is being done to try to drop the number of identity theft incidents. There is a lot of focus in trying to lessen the impact that a data breach incident has on identity theft. The real damage caused by identity thieves takes time. Loan applications, credit card applications, all take time. Therefore, if you can quickly identify that someone's identity has been stolen and there are signs that a criminal is starting to use that identity, you can take action immediately and dramatically reduce the damage that can be done. The key to this is often notification. If a company has a data breach and knows that your personal information may have been compromised, it is important for that company to tell you of this breach. This will allow you to closely track your credit reports, your statements, etc. If you do this, you will identify if someone is indeed launching an identity theft attack and you will be able to stop it. The real damage will be avoided. On the other hand, if the company keeps it all very quiet, you most likely will not discover anything until it is much too late and a lot of damage has already been done.

The key to limiting the damage is early warning. This is why there are so many notification clauses within regulations. As an example, SB 1386 (later became AB 1950) is a California law that requires notification of any resident of California whose unencrypted personal information was, or is reasonably believed to have been, acquired by an unauthorized person.

Notification regulations are hard to implement. They imply that if you suspect that you had a data breach you need to notify your customers, employees, patients; whomever's data has been compromised. There are two problems with notification. The first is that it damages your brand; people may not want to do business with you any more. The second is the cost of the notification itself. Gartner Research estimates that organizations are being required to spend at least \$90 for each personal electronic record affected by a data security breach. The Ponemon Institute claims the cost is even higher, reporting that organizations spend as much as \$140 per lost customer record. These figures include

- Communication with affected individuals
- Resulting customer service expenditures (call centers, credit protection, etc.)
- Identity theft monitoring services
- Professional fees and legal expenses
- After-the-fact cleanup costs and security improvements

Because the cost is per lost customer record, it is very valuable to know how many of them are there. For example, if you identify that a rogue connection was used to access personal data and you have 800,000 customers in your database it would be very valuable to know if all 800,000 records were extracted from the database or only 80. If you have a way to know that only 80 records were selected and you know which 80 records were selected, your notification costs will be very small. If you don't have a way to know you may end up having to notify all 800,000 customers. The damage to your brand and the total cost will be high.

### 12.3.2 Using Flashback Queries: An Example

Flashback queries coupled with appropriate auditing can help you know what data may have been selected by a rogue connection. They can also be used for extracting before and after values, so let's look at a simple example:

Step 1: First let's see what the EMP table has as contents so you can see if it matches the before values in the Flashback query:

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	8000		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
14 rows selected.
```

Step 2: Enable standard auditing (to the database for example) and audit all (just for this example):

```
SQL> alter system set audit_trail='db_extended' scope=spfile;
System altered.
```

```
SQL> shutdown
Database closed.
Database dismounted.
ORACLE instance shut down.
```

```
SQL> star3tup
ORACLE instance started.
Total System Global Area 201326592 bytes
Fixed Size          1218484 bytes
Variable Size       71305292 bytes
Database Buffers   121634816 bytes
Redo Buffers        7168000 bytes
Database mounted.
Database opened.
```

```
SQL> connect sys as sysdba
Enter password:
Connected.
SQL> audit all by access;
Audit succeeded.
```

Step 3: Connect as SCOTT. SELECT some data and update some data:

```
SQL> connect scott/tiger
Enter password:
Connected.

SQL> select sal from emp where EMPNO=7900;
```

```

SA      L
-----
95      0

SQL> update emp set sal=9500 where EMPNO=7900;
1 row updated.

```

Step 4: Connect as SYSTEM and look at the audit trail. Find the two SCN numbers for the SELECT and the update:

```

SQL> select USERNAME,OBJ_NAME,ACTION_NAME,SCN,SQL_TEXT from DBA_AUDIT_TRAIL;

USERNAME    OBJ_NAME   ACTION_NAME   SCN          SQL_TEXT
-----      -----      -----      -----
SCOTT       EMP        SELECT      15529316  select sal from emp where EMPNO=7900
SCOTT       EMP        UPDATE      15529324  update emp set sal=9500 where EMPNO=7900

```

Step 5: Run your Flashback queries. First, let's see what data SCOTT extracted from the database. From the audit trail you know exactly which SQL statement the user issues. From the audit trail you know the SCN. Together, you can see exactly what data the user saw:

```

SQL> select sal from scott.emp as of scn 15529316 where EMPNO=7900;

EMPNO ENAME   JOB          MGR HIREDATE      SAL     COMM      DEPTNO
----- -----   -----      -----  -----      -----
7900 JAMES    CLERK      7698 03-DEC-81    950

```

To see the state of the EMP table just before the update was made, use the SCN in the UPDATE audit record. This gives you the before values for the table:

```

SQL> select * from scott.emp as of scn 15529324;

EMPNO ENAME   JOB          MGR HIREDATE      SAL     COMM      DEPTNO
----- -----   -----      -----  -----      -----
7369 SMITH    CLERK      7902 17-DEC-80    8000
7499 ALLEN    SALESMAN   7698 20-FEB-81    1600    300      30
7521 WARD     SALESMAN   7698 22-FEB-81    1250    500      30
7566 JONES    MANAGER   7839 02-APR-81    2975
7654 MARTIN   SALESMAN   7698 28-SEP-81    1250    1400      30
7698 BLAKE    MANAGER   7839 01-MAY-81    2850
7782 CLARK    MANAGER   7839 09-JUN-81    2450
7788 SCOTT    ANALYST   7566 19-APR-87    3000
7839 KING     PRESIDENT 17-NOV-81    5000
7844 TURNER   SALESMAN   7698 08-SEP-81    1500
7876 ADAMS    CLERK     7788 23-MAY-87    1100
7900 JAMES    CLERK     7698 03-DEC-81    950
7902 FORD     ANALYST   7566 03-DEC-81    3000
7934 MILLER   CLERK     7782 23-JAN-82    1300

```

14 rows selected.

You can also use Flashback queries with dates. As an example, if you know when a change occurred (e.g., using the audit trail) you can get the before and after values from a Flashback query:

```
SQL> select * from scott.emp as of timestamp to_timestamp('2008-01-26 22:37:48', 'YYYY-MM-DD HH24:MI:SS') where empno=7900;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7900	JAMES	CLERK		7698 03-DEC-81	9500		30

```
SQL> select * from scott.emp as of timestamp to_timestamp('2008-01-26 22:37:47', 'YYYY-MM-DD HH24:MI:SS') where empno=7900;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7900	JAMES	CLERK		7698 03-DEC-81	950		30

You can use ORA\_ROWSCN on any row to get at the SCN for the most recent change to a given row. This returns the SCN for the last commit per row. You can also use the function SCN\_TO\_TIMESTAMP to convert an SCN to the appropriate time stamp. For example:

```
SQL> update emp set sal=9999 where EMPNO=7876;
1 row updated.
SQL> commit;
Commit complete.
SQL> select empno, sal, ora_rowscn, scn_to_timestamp(ora_rowscn) from emp where
EMPNO=7876;
```

EMPNO	SAL	ORA_ROW	SCN	SCN_TO_TIMESTAMP(ORA_ROWSCN)
7876	9999	15531736	09-FEB-08	04.2936.00000000 AM

Note that there is a three second granularity for SCN\_TO\_TIMESTAMP.

### 12.3.3 Getting Versions Using Flashback

You can even get all versions of the data through the changes. For example:

```
SQL> update emp set sal=sal*1.1;
14 rows updated.
SQL> commit;
Commit complete.

SQL> update emp set sal=sal*1.05;
14 rows updated.
SQL> commit;
Commit complete.

SQL> select empno,ename,sal from emp
  2 versions between scn 15531736 and 15533304
  3 where empno=7876;

E      MPNO ENAME          SAL
----- 
787      6 ADAMS        11548.85
7     876 ADAMS        10998.9
7     876 ADAMS         9999
```

There are a few pseudo-columns that you can use when using the flashback versions query:

- VERTICALS\_STARTSCN: Returns the SCN when the row was first created.
- VERTICALS\_STARTTIME: Returns the time stamp when the row was first created.
- VERTICALS\_ENDSCN: Returns the SCN when the row expired.
- VERTICALS\_ENDTIME: Returns the time stamp when the row expired.
- VERTICALS\_OPERATION: Returns “I,” “D,” or “U” as an encoding of the DML operation type that was performed on the row.

Flashback queries are just a small piece of the Flashback features. In addition to the SELECT ... AS OF functionality that you saw previously, the DBMS\_FLASHBACK package acts as a “time machine” that allows you to turn back the clock, perform any query “in the past,” and then return to the present. You can reuse existing queries and PL/SQL code—you do not need to add the AS OF clause anywhere. To do that use DBMS\_FLASHBACK.ENABLE\_AT\_TIME or DBMS\_FLASHBACK.ENABLE\_AT\_SYSTEM\_CHANGE\_NUMBER to start the Flashback session and then use any of your PL/SQL code. You can use only queries—don’t try to perform DDL or DML. Once you’re done, call DBMS\_FLASHBACK.DISABLE to revert back to the present.

### 12.3.4 Prerequisites for Flashback

Before you can use Flashback features you need to ensure that your database is in ARCHIVELOG mode. You can see whether it is or not using one of the following queries (showing a database not ready for flashback yet):

```
SQL> select log_mode from v$database;
LOG_MODE
-----
NOARCHIVELOG
SQL> connect sys as sysdba
Enter password:
Connected.

SQL> archive log list
Database log mode          No Archive Mode
Automatic archival        Disabled
Archive destination        USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence 96
Current log sequence       98
```

Once you have ARCHIVELOG mode you need to create an undo tablespace with enough space to keep the required data for Flashback operations. The longer you want the data available the more of an issue this becomes. This is the main issue with Flashback as a security tool—for most practical audit and security needs this would require too much storage because you may need a long retention period. You may have to wait until the next HOWTO and use flashback archives. If you can allocate enough space, you also have to enable automatic undo management. This involves setting UNDO\_MANAGEMENT, UNDO\_TABLESPACE, UNDO\_RETENTION, and set the RETENTION GUARANTEE clause for the undo tablespace. Grant flashback privileges to the users or application that will require the

Flashback functionality. Grant EXECUTE privileges to DBMS\_FLASHBACK, grant FLASHBACK, and SELECT privileges on the appropriate objects and grant FLASHBACK ANY TABLE privileges.

The initialization parameter UNDO\_RETENTION allows you to specify the length of time Oracle maintains undo information in the undo segments. The default is 900 seconds (15 minutes)—this is definitely not enough for a security implementation. Syntactically you can set this number to any value up to  $2^{32}$  second. You should use the Oracle Undo Advisor to get approximate undo parameter values as well as suggestions regarding the sizing of your undo tablespace to successfully support flashback for a specified time. Setting UNDO\_RETENTION does not guarantee the retention period. You need to use the RETENTION GUARANTEE clause when you create your undo segment:

```
SQL> create undo tablespace undo_1 datafile '/home/oracle10/oradata/on0r4231/undo_1.dbf'
  2 size 100M autoextend on RETENTION GUARANTEE;
Tablespace created.
```

The amount of time for which undo data is retained for undo tablespaces is available from the V\$UNDOSTAT dynamic performance view:

```
SQL> select distinct TUNED_UNDORETENTION from v$undostat;
TUNED_UNDORETENTION
-----
900
```

### Two Things to Remember about Flashback Queries

1. Using Flashback queries based on the SCN from the audit trails can give you both before/after values as well a view into what data was selected by the user.
2. Flashback queries are limited by the retention period and the size of the undo segments. It usually means that this feature is not available.

## 12.4 HOWTO Use Flashback Data Archive

Oracle 9i introduced Flashback query. Oracle 10g extended Flashback to provide recovery at the database, table, row, and transaction levels. Oracle 11g introduced Flashback Data Archives which can be used to automatically track and maintain historical changes to data. Flashback Data Archive tracks every change in a separate archive of historical data. The captured historical data can be retained as long as you need it and is accessible using Flashback queries (see the previous HOWTO). Enabling Flashback Data Archive can be done per table or for an entire application. From a security perspective, Flashback Data Archive provides access to the data but in a way that is immutable to all users, access is read only. Oracle Flashback Data Archive is part of the Oracle Total Recall Option.

The main problem that Flashback Data Archive comes to solve is that undo data is cleared when the database is recycled or when Oracle needs to make room for new undo segments. Undo data is typically available for a short amount of time only and is thus not appropriate for auditing where the data needs to be guaranteed. Flashback Data Archive does not rely on the transient storage of undo segments—it records changes in the Flashback Recovery area—a new, persistent location.

To enable Flashback Data Archive follow these steps:

Step 1: Create a Flashback Data Archive:

```
SQL> create flashback archive audit_data tablespace flash_arch_1
  2  retention 1 month;
Flashback archive created.
```

Step 2: Start recording changes in the archive. For example, if you want all changes to the emp table to be recorded then

```
SQL> alter table scott.emp flashback archive audit_data;
Table altered.
```

The table has been altered and is now in Flashback archive mode. All changes to the table are now tracked up to the retention period. You set the retention period to one month, so that data will be kept there for that period after which when new data comes in it is purged.

#### **Two Things to Remember about Flashback Data Archive**

1. Flashback Data Archive is a new product in 11g that makes use of Flashback queries for security and auditing practical and useful.
2. Once setup, implementing flashback for a table is a simple ALTER TABLE command.

## **12.5 Discussion: Do You Really Need the Before Values?**

Security and auditing is focused on knowing what a user did. This is why the various Oracle auditing mechanisms show you what action was performed (along with the SQL text is required) but not the before and after values. One view to this issue is that before and after values are not what auditors focus on and need. They certainly need the after values as part of the SQL text but why would they need the before values? Before values may be needed to restore the state to a previous value for recovery sake, but what has that got to do with audit? One view is that you should have a baseline value recorded at an initial point in time after which if you record all SQL text that alters the value, you can recover the data to the value at any point in time (as a roll-forward process).

The other view is that the before value is also important, even if it is less important than the after value. Oracle does give you the SCN number within the audit records and does provide you with the ability to get at the values based on the SCN number. Moreover, as you'll see in Chapter 13 Oracle Audit Vault has a special collector for automating the collection of before and after values for you.

At the end of the day which view you adopt depends on you and on your auditors (internal and external). However, you should remember a few important points that may help you in keeping your world sane:

- Before and after values are the hardest to collect as part of an audit both from the perspective of how it is done and more importantly from a performance point of view. Logging before and after values is much slower any way you do it as compared with standard and fine-grained auditing.

- Comprehensively collecting before and after values is completely impractical and unnecessary. If you have a table with 10,000 records and perform an update that affects all rows, the audit record is a single SQL statement but the before/after values will be up to 10,000 rows. It is impractical to start keeping all the snapshots of all your data over time where the alternative can be so much simpler, efficient, and low cost.
- Before values are not as important for audit as they are for remediation. Often the real reason for maintaining the before values is that if a certain value (for example) is financially affecting your company you may want to be able to roll it back to a previous value very quickly once you determine that a mistake of an unauthorized change has occurred.

This topic is at the heart of a very serious debate which has not run its course. However, there are a few things that are clear by now:

1. If you do decide to keep before and after values, be very selective. There is no justification for collecting these values widely. Pick only the tables and columns that truly justify the investment and analyze the frequency with which these values change to ensure it is practical.
2. Understand the difference between before and after values as compared with just the SQL text. Understand the requirement that is being used to prompt an auditor to ask for before and after values. This is very important. Auditors are reasonable people and you can “negotiate” with them. You will be surprised to learn that there are very few regulations or audit requirements that truly require the before values, sometimes it is just thrown into the mix.

## *Chapter 13*

---

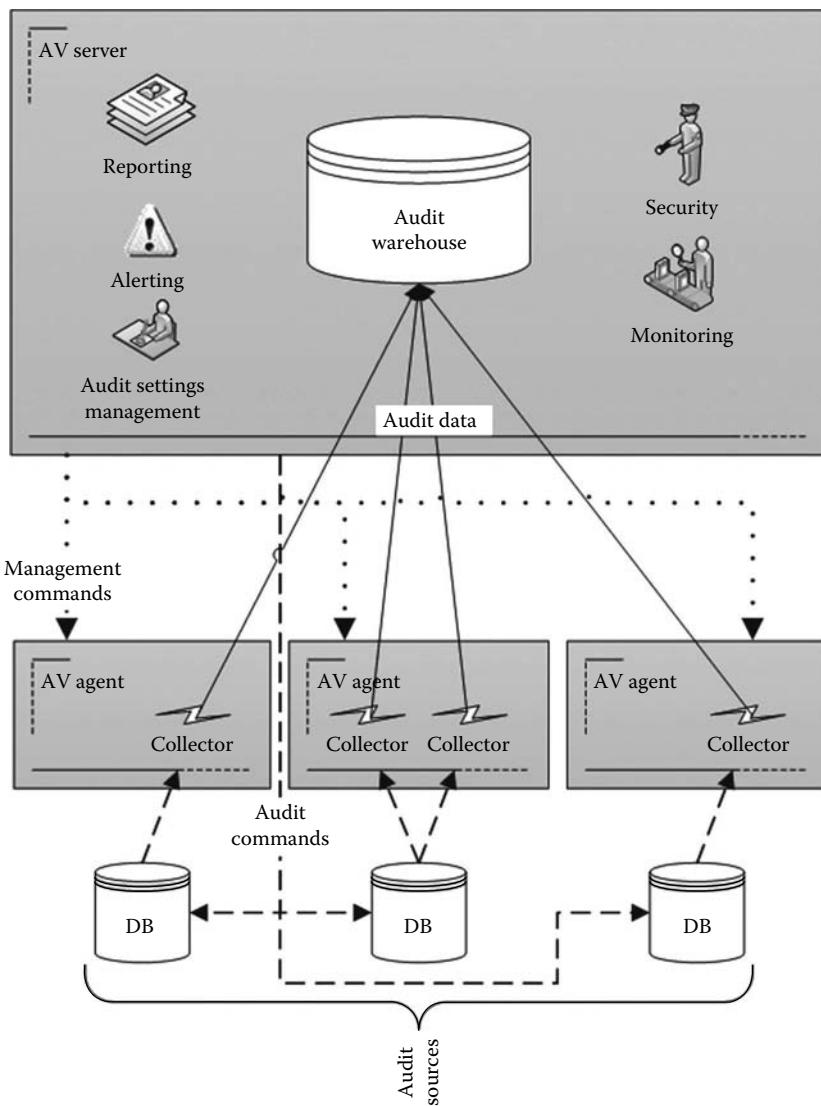
# Oracle Audit Vault

---

In the previous chapters, you learned about auditing capabilities within the Oracle database. All of these chapters dealt with how to define and control what gets audited—but the focus was always on a single database. There was no discussion on how you can collect this audit data from many databases nor was there a discussion around the important issues relating to where to store the audit data and how to protect it from tampering. This is because database vendors tend to focus on the capabilities that they can build within the database engine, and Oracle has been no exception—until fairly recently. Oracle Audit Vault (AV) is a relatively new product that addresses how to collect audit data, how to manage it, and how to protect it.

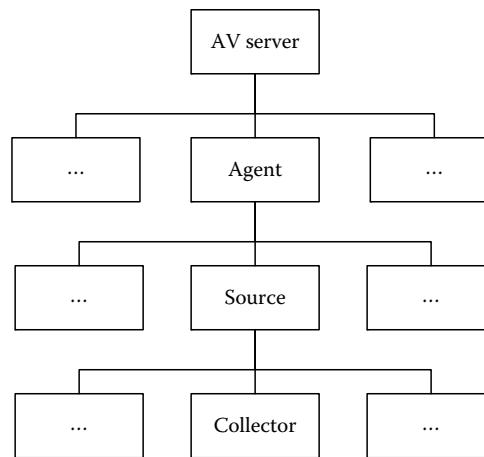
Oracle AV is a separate product and is not linked-in with the database. It is an application that is used to manage audit policies and collect the audit data from Oracle databases. It is different from all other Oracle features and products you will see in this book—it is not a part of the database. It uses an Oracle database for managing audit data and audit policies and it reads audit records from your databases, but it is not an option you add to your databases. The most important point to remember regarding AV is that it does not introduce any new audit capabilities and the auditing on the source databases is done using the native mechanisms that you learned in the previous chapters—there is nothing new on that front. What is new is that the data is collected from these audit trails and placed into a centralized audit warehouse where reporting and alerting applications can be used.

Figure 13.1 shows a high-level overview of the AV architecture. The three most important concepts in AV are the AV server, AV agents, and AV collectors. The AV server is a complex application that should live on its own server. It is a full-blown application—along with a database, an application server, utilities, and application code. The AV server has been built using a lot of Oracle technology—it has an Oracle data warehouse at its core and an Oracle Application Server running the AV applications and console; it uses Oracle Database Vault for protecting the audit data in the database, the Oracle wallet for securing certificates, ASO to encrypt network traffic for databases connections, etc. The audit data lives within the database running on the AV server but the AV server is not extracting the audit data itself. The components that are responsible for getting the audit data from the source database are the collectors. Collectors are processes that run on the same machine as the source database. They are either operating system processes or procedures that run within the database and their job is to feed the audit data into the AV server's



**Figure 13.1** AV architecture.

database. A typical collector will extract audit information from the source database (e.g., by making a database connection and reading AUD\$) and then make a connection directly to the AV server's database to feed the audit data to AV. Agents manage collectors and communicate with the AV server (the application and not the database) to receive control information. You can start and stop agents and collectors from the AV server—either using utilities or using the AV console. The AV server also communicates with the source database—when you want to change an audit policy you do so within the AV console. The rules are used to update the audit definitions within the source database. This is called provisioning—quite simply, the AV console logs into the source database and issues a sequence of AUDIT commands. This in turn changes what gets audited and the next time the collector reads this data, it is reading audit trails based on the new audit policy.



**Figure 13.2 AV association hierarchy.**

The AV hierarchy is shown in Figure 13.2. There is a single AV server—audit data is centralized into one repository so that you don't have to work too hard to produce compliance reports. An AV server monitors and manages multiple agents. Each agent can be managing multiple sources—e.g., when you have multiple databases within a single server. Each source is an audited database and each source can have multiple collectors—e.g., when you need to collect audit records from AUD\$ but also need before/after values for DML statements extracted from the redo log.

Collectors always live on the server where the source database resides. The AV server always lives on a machine by itself. The agents however communicate less with the AV server, with the collectors, and with the source databases. Therefore, they can be deployed using any number of configurations as shown in Figure 13.3. In the first configuration, the agent is colocated with the source database and the collectors it is managing. In the second configuration, it is colocated with the AV server. In the third configuration, it is located on a separate server. All of these will work fine. The most common deployment is the first scheme—unless you have a specific reason to choose a different deployment you should prefer to have the agent running where the collectors are running.

Collectors write data into the AV server's database over an ORACLE\*NET connection (using ASO so the data is encrypted as it goes over the wire). Collectors need to insert data very fast so they write data into a table called AV\$RADS\_FLAT. This is a generic table that has columns such as STR1\_VAL,..., STR50VAL, NUM1\_VAL, ..., NUM30\_VAL, etc. It is certainly not where the data ends up—it is used because it allows the collectors to run quickly. Once the data is in this table, an AV background process called refresh\_warehouse populates a star schema from the data within this table. The main entities of the star schema are shown in Figure 13.4. At its core, AV is a data warehouse and the star schema is the heart of the application.

## AV Collectors

There are three types of AV collectors: DBAUD, OSAUD, and REDO. The DBAUD and the OSAUD collectors are responsible for reading audit records produced by the source database and the REDO log is responsible for extracting before/after values from the redo logs. DBAUD and OSAUD run as operating system processes and REDO runs as Oracle Streams processes within the database.

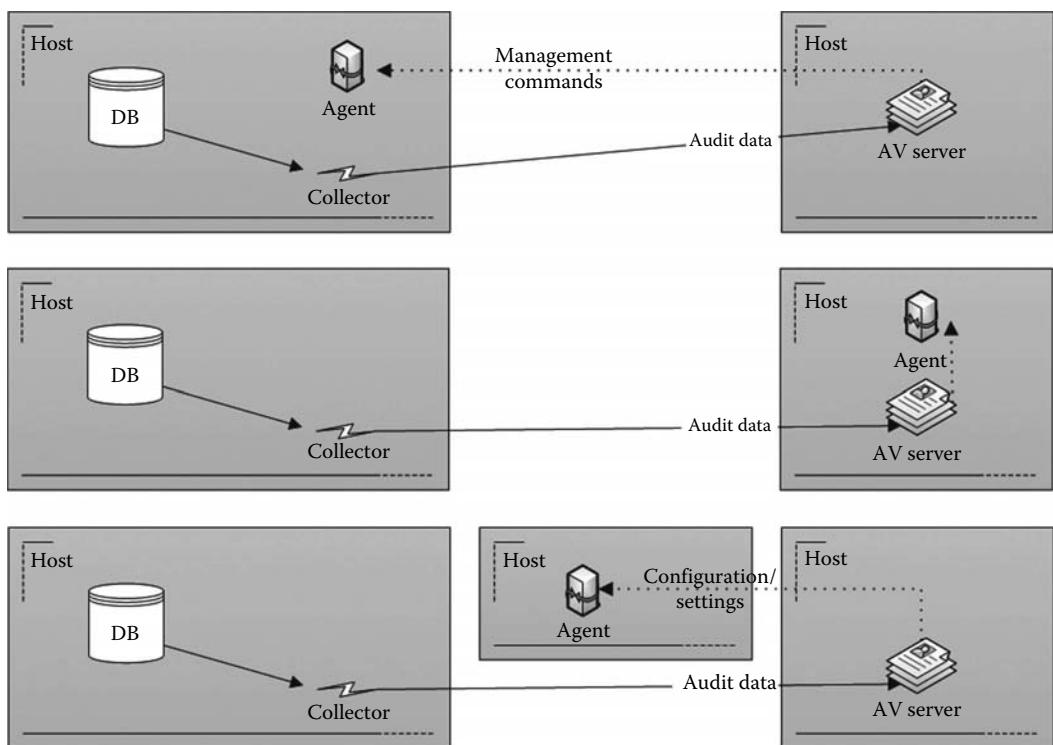


Figure 13.3 Deploying AV agents.

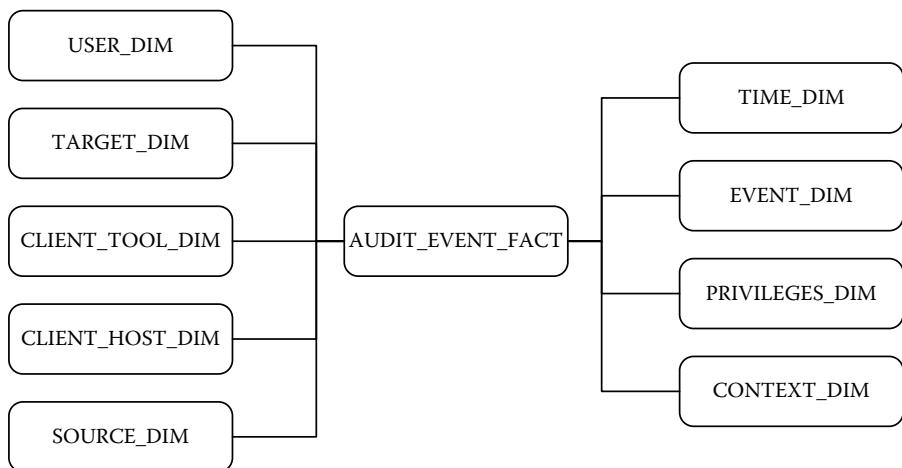
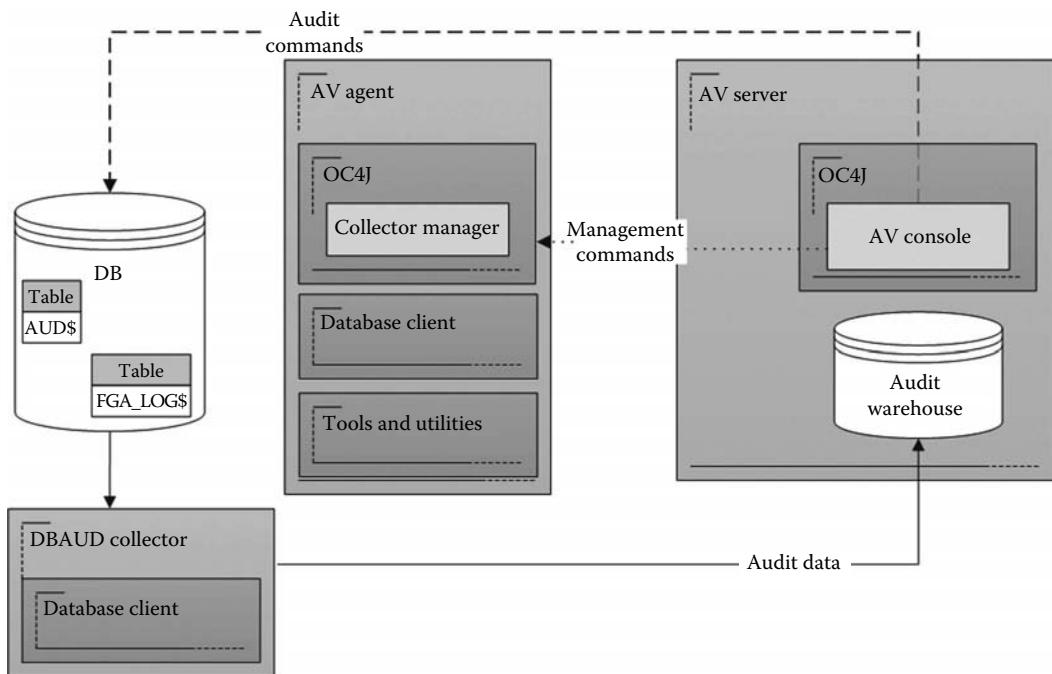


Figure 13.4 AV audit warehouse star schema.



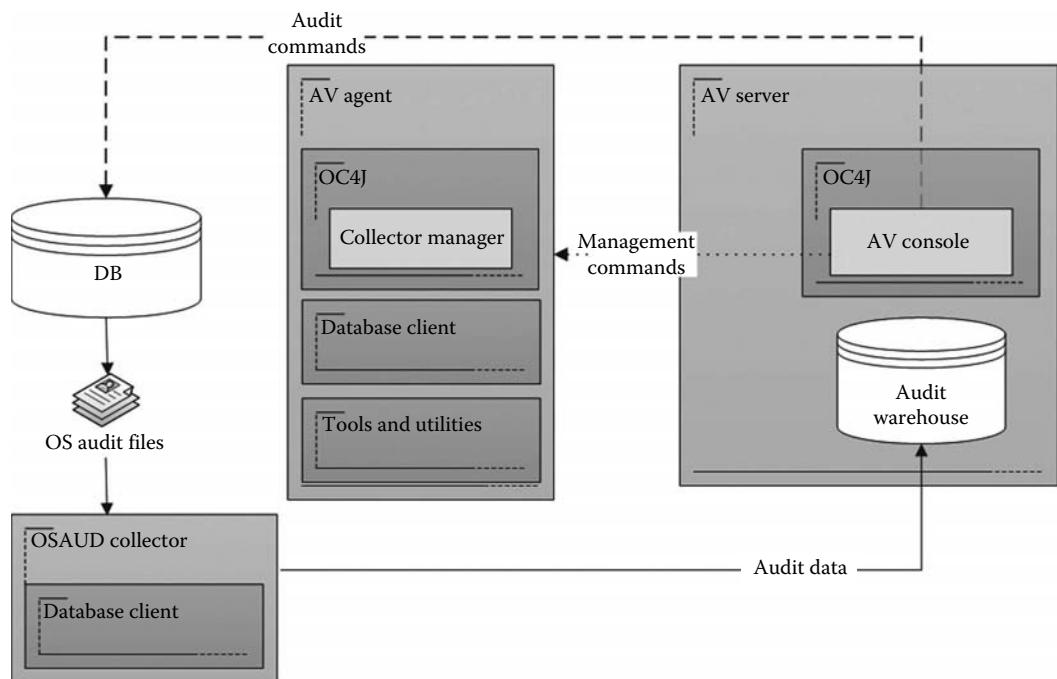
**Figure 13.5 DBAUD collector architecture.**

Figure 13.5 shows the working of the DBAUD collector. As any collector, it is managed by the AV agent. The DBAUD collector uses a polling interval. The DBAUD collector creates a connection to the database when it first starts up. Then, based on a configurable interval, it reads from the AUD\$ and FGA\_LOG\$ tables. It does this over a normal listener connection. DBAUD is also logged on to the AV server's database and inserts the audit records it has just read into the RADS\_FLAT table. The OSAUD collector works in much the same way but the audit records come from the operating system audit files (including the Windows event log for administrator auditing). The OSAUD collector is an operating system process that reads these files and inserts the records into the RADS\_FLAT table over the connection it maintains to the AV server's database (Figure 13.6).

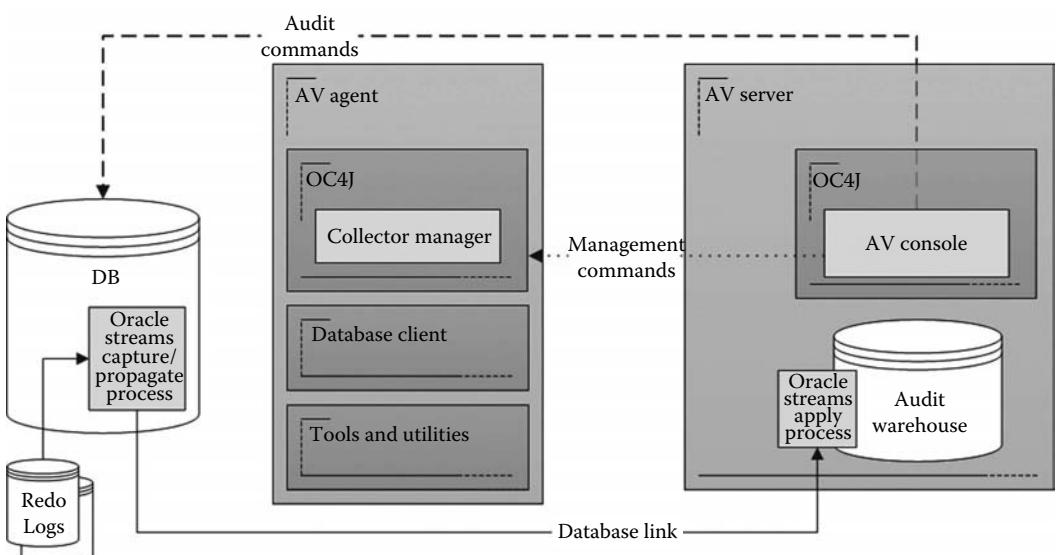
The REDO collector is very different. As Figure 13.7 shows, the REDO collector is based on Oracle Streams. The REDO collector creates a capture process which is based on your audit definitions. It uses a propagate process to push the before/after values over a database link to the AV server's database. On the AV server's database an apply process extracts this audit data.

## AV Roles

Because one of the reasons for AV's creation is to better secure the audit data, AV uses Oracle Database Vault (DV) to secure the database. Therefore (see Chapter 17 for more details), you will not be able to connect to the AV database using/as sysdba:



**Figure 13.6** OSAUD collector architecture.



**Figure 13.7** REDO collector architecture.

```
-bash-2.05b$ sqlplus "/ as sysdba"
SQL*Plus: Release 10.2.0.3.0 - Production on Mon Jan 14 12:22:26 2008
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
ERROR:
ORA-01031: insufficient privileges
```

All roles are part of the comprehensive user and role management layer that AV provides and that makes use of features within DV. Security is managed through a set of predefined AV roles that you should understand. The most important function that these roles provide is separation of duties. Each role has a set of privileges that enforce what users can do. As an example, if you try to log on to the AV console using your administrator account but try selecting the AV\_AUDITOR role, AV will kick you out. The administrator can start and stop collectors, change settings etc., but cannot change the audit definitions or review the audit data. The AV and DV roles defined within AV are:

- DV\_OWNER—Use this role to grant AV roles to users.
- DV\_ACCTMGR—Use this role to manage accounts within the database. SYS and SYSTEM cannot do this in AV.
- AV\_ADMIN—Use this role to administer, configure, and manage AV.
- AV\_AUDITOR—Use this role to define audit policies, view audit reports, set up alerts, etc.
- AV\_AGENT—Agents connect to the AV database using a user granted this role to query configuration information.
- AV\_SOURCE—Collectors connect to the AV database using a user granted this role to feed the audit data to AV.
- AV\_ARCHIVER—Use this role to archive audit data and clean up unused alerts and metadata.

### 13.1 HOWTO Add, Configure, and Manage Agents

Installing an AV agent involves work on both the AV server and the source database. At a high level, the steps you need to take are

1. On the AV server:
  - a. Create a user in the AV server that is used by the AV agent
  - b. Add the agent to AV
2. On the source databases:
 

Install the agent software

Of course, adding the agent is not enough—you'll then have to add sources and collectors, but let's focus right now on the agents only. Let's look at a detailed example for adding an agent:

Step 1: In the AV server, log in as the DV account manager, create a user to represent the AV agent, and then grant it the AV\_AGENT role:

```
-bash-2.05b$ sqlplus administrator@dv/guardium1.
SQL*Plus: Release 10.2.0.3.0 - Production on Fri Jan 11 09:08:45 2008
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
```

```
SQL> create user sys1_agent identified by "0okmju7";
User created.
SQL> grant AV_AGENT to SYS1_AGENT;
Grant succeeded.
```

Step 2: Add the agent to AV using avca (using the user you created in the previous step):

```
-bash-2.05b$ avca add_agent -agentname sys1_agent -agenthost
sys1.dbasecurity.com -agentusr sys1_agent
AVCA started
Adding agent...
Agent added successfully.
```

Next, install the agent on the source database server:

Step 3: Log on to the database server to the account that owns the ORACLE\_HOME environment and cd to the directory with the AV agent installation files. Execute:

```
-bash-2.05b$ ./runInstaller
```

Step 4: This starts up an interactive installer. On the Agent Details page specify the agent name, the directory where you want to install the agent, the account name and password on the AV server that you created previously and the connect string for the AV server in the format <host-name> : <port> : <service name>—for example:

```
avserver.dbasecurity.com:1521:av_server
```

Click Next, check your settings and click Next again. Click Install to start the installation. You now have an agent installed. You can verify that it is installed by looking at the process list.

```
-bash-3.00$ ps -ef | grep oravault
1003      5909      1 0 Jan10 ?          00:00:27
/home/oravault2/oracle/product/10.2.2/av_agent_1/jdk/bin/java -
-Dav.oracle.home=/home/oravault2/oracle/product/10.2.2/av_agent_1 -jar
/home/oravault2/oracle/product/10.2.2/av_agent_1/oc4j/j2ee/home/oc4j.jar -out
/home/oravault2/oracle/product/10.2.2/av_agent_1/av/log/agent.out -err
/home/oravault2/oracle/product/10.2.2/av_agent_1/av/log/agent.err -config
/home/oravault2/oracle/product/10.2.2/av_agent_1/oc4j/j2ee/home/config/server.xml
```

This Java process running within the oc4j container is the agent—it manages the sources and the collectors which you'll add later in this chapter. In addition, you will see the agent on the AV console if you log in as AV\_ADMIN. Click on the Agents tab at the top and you'll see your agent and its status as shown in Figure 13.8.

Normally, the communications between the AV server and AV agents occur over an unencrypted protocol and there is no mutual authentication between the components. This means that it is fairly simple for an attacker to masquerade as the AV server and, as an example, tell the agent to stop or send it commands to stop auditing. Because the agent controls the sources and collectors, you should add security to this communication. AV has the ability to set up a secure communication scheme between the AV server and the AV agents. In this case, the communication is done over SSL and AV uses certificates for authentication before it allows the communication to happen. To turn this feature on, use the avca utility. You need to run this both on the AV server and on the AV agent host. On the AV server run:

## Agents

Select	Agent	Host	Port	HTTPS	Status
<input checked="" type="radio"/>	sys1_agent	sys1.dbasecurity.com	70110	0	up
<input type="radio"/>	sys2_agent	sys2.dbasecurity.com	0	0	down

Figure 13.8 Viewing agent status on the AV console.

```
-bash-2.05b$ avca secure_av -avkeystore $ORACLE_HOME/avkeystore -avkeystorepwd
9ijnhy6 -avtruststore $ORACLE_HOME/avkeystore
AVCA started
Stopping OC4J...
OC4J stopped successfully.
Starting OC4J...
OC4J started successfully.
TZ set to US/EasternOracle Audit Vault 10g Database Control Release 10.2.2.0.0
Copyright (c) 1996, 2005 Oracle Corporation. All rights reserved.
https://avserver.dbasecurity.com:5700/av
Oracle Audit Vault 10g is running.

-----
Logs are generated in directory
/home/oravault/oracle/product/10.2.2/av_1/av/log
```

Note that once you run this you will no longer be able to connect to the AV console using the normal universal resource locator (URL) and the Hypertext Transfer Protocol (HTTP)—instead, all communications with the AV server are encrypted and you need to use the `https://avserver.dbasecurity.com:5700/av` URL as shown in the output of `avca`.

If you run into problems on the agent side you can revert back to unencrypted communications with the agent using

```
-bash-2.05b$ avca secure_av -remove
AVCA started
Stopping OC4J...
OC4J stopped successfully.
Starting OC4J...
OC4J started successfully.
TZ set to US/EasternOracle Audit Vault 10g Database Control Release 10.2.2.0.0
Copyright (c) 1996, 2005 Oracle Corporation. All rights reserved.
http://goose.guardium.com:5700/av
Oracle Audit Vault 10g is running.

-----
Logs are generated in directory
/home/oravault/oracle/product/10.2.2/av_1/av/log
```

Finally, run the `avca` utility on the agent:

```
-bash-3.00$ avca secure_agent -agentname sys1_agent -agentkeystore
$ORACLE_HOME/agentkeystore -agentkeystorepwd 8uhbgt5 -agentdn "CN=sys1_agent,
O=dbasecurity, C=us" -avdn "CN=avserver, O=dbasecurity, C=us"
```

Once the agent is installed and running you can manage it from within the AV console. Log on as the AV\_ADMIN, click on the Management tab at the top right and on the Agents tab on the left. Select the agent from the list and click either the Start or Stop button. To view or modify a configuration entry for the agent click on the Management tab at the top right and the Agent tap on the left. Then select the agent from the list and click View or Edit.

#### The e Things to Remember about AV Agents

1. A user is defined in the AV database per AV agent.
2. An agent is added by running AVCA on the AV server.
3. Agents run within the oc4j container. Secure communication with agents is done over SSL through the oc4j container.

## 13.2 HOWTO Add, Configure, and Manage Sources

Once you have an agent setup it is time to set up a source. The source will be used by AV when it has to provision the audit policy to the source database. To create the source:

Step 1: Log on to the oracle instance account on the source server.

Step 2: Make sure that the source database has a password file set up (see Chapter 6). You need to do this because avorcldb connects to the source database using sysdba privileges and any remote connection that uses sysdba privileges needs to log on with a password. You should also do this because it is a good practice and because you will need it if you also plan on using Database Vault.

Step 3: Create a source user in the source database:

```
[oracle10@rh4test ~]$ sqlplus sys as sysdba
SQL*Plus: Release 10.2.0.3.0 - Production on Fri Jan 11 12:50:35 2008
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, OLAP and Data Mining options
SQL> create user sys1_src identified by "7ygvfr4";
User created.
```

Step 4: Grant permissions to the source user. Because the source user will be used by AV to set all sorts of audit policies within the database, you need to give it certain privileges within your database. These privileges are listed within an AV script called zarsspriv.sql. You run this script by giving the user name (the one you just created) and a mode. The mode, either SETUP or REDO\_COLL. SETUP sets up the permissions required when you use the DBAUD and OSAUD collectors and REDO\_COLL sets up the permissions required when you use the REDO collector. Run the script once or twice depending on which collectors you plan on using:

```
SQL> @$AGENT_HOME/av/scriptsstreams/source/zarsspriv.sql sys1_src SETUP;
PL/SQL procedure successfully completed.
```

```
SQL> @$AGENT_HOME/av/scriptsstreams/source/zarsspriv.sql sys1_src REDO_COLL;
PL/SQL procedure successfully completed.
```

Step 5: On the AV server log on as the Database Vault account manager and create a user for the source:

```
-bash-2.05b$ sqlplus administratordva
SQL*Plus: Release 10.2.0.3.0 - Production on Fri Jan 11 11:57:29 2008
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
SQL> create user sys1_src identified by "6tfcde3";
User created.
```

Step 6: Grant proxy connect privileges to the source user on the AV server through the agent user.

```
SQL> alter user sys1_src grant connect through sys1_agent;
User altered.
```

Step 7: Verify that the source database is compatible with the collector type you plan on running. What you're checking is that the source database will allow the collector to run. Run this from the AV server using the connection string for the source database:

```
-bash-2.05b$ avorcldb verify -src sys1.dbasecurity.com:1521:source
-srcusr sys1_src/7ygvfr4 -colltype ALL
source SYS1.DBASECURITY.COM verified for OS File Audit Collector collector
source SYS1.DBASECURITY.COM verified for Aud$/FGA_LOG$ Audit Collector collector
source SYS1.DBASECURITY.COM verified for REDO Log Audit Collector collector
```

Colltype can use ALL, or one of REDO, DBAUD, OSAUD, EVTLOG to verify compatibility with an individual collector (EVTLOG for Windows systems where records need to be extracted from the Windows event log). If you have problems, they will usually be due to compatibility issues for the REDO collector as there are more parameters that need to be set up correctly, e.g.:

```
-bash-2.05b$ avorcldb verify -src sys2.dbasecurity.com:1521:sys2_ora -srcusr
sys2_src/6tfcvgy8 -colltype ALL
source SYS2.DBASECURITY.COM verified for OS File Audit Collector collector
source SYS2.DBASECURITY.COM verified for Aud$/FGA_LOG$ Audit Collector collector
ERROR: parameter _SPIN_COUNT is not set; recommended value is 5000
ERROR: parameter _JOB_QUEUE_INTERVAL is not set; recommended value range [4 -
ANY_VALUE]
ERROR: parameter UNDO_RETENTION = 900 is not in recommended value range [3600 -
ANY_VALUE]
ERROR: parameter GLOBAL_NAMES = false is not set to recommended value true
ERROR: source database must be in ARCHIVELOG mode to use REDO LOG collector
ERROR: set the above init.ora parameters to recommended/required values
```

Step 8: Still on the AV server, run the avorcldb command to add the source to the AV server database:

```
-bash-2.05b$ avorcldb add_source -src sys1.dbasecurity.com:1521:source -srcusr
sys1_src/7ygvfr4 -avsrcusr sys1_src -agentname sys1_agent
```

```

Adding source...
Source added successfully.
source successfully added to Audit Vault
remember the following information for use in avctl
Source name (srcname): SYS1.DBASECURITY.COM
map_source_to_agent
map_source_to_agent

```

### The e Things to Remember about AV Sources

1. A source is used to provision the audit policy to the source database.
2. A database user is created on the source database for the source. This is called the source user.
3. The source database must have a password file because avorcldb connects remotely using sysdba privileges.

## 13.3 HOWTO Add, Configure, and Manage Collectors

With the source in hand, you can add your collectors from the AV server. All collectors are added in the same way using the avorcldb utility—what changes is the colltype parameter value.

To add a DBAUD collector that will collect data out of AUD\$ and FGA\_LOG\$ run:

```

-bash-2.05b$ avorcldb add_collector -srcname SYS1.DBASECURITY.COM -srcusr
sys1_src/7ygvfr4 -agentname sys1_agent -colltype DBAUD
source SYS1.DBASECURITY.COM verified for Aud$/FGA_LOG$ Audit Collector collector
Adding collector...
Collector added successfully.
collector successfully added to Audit Vault

remember the following information for use in avctl
Collector name (collname): DBAUD_Collector

```

To add an OSAUD collector that will collect data from the operating system files audit trail run:

```

-bash-2.05b$ avorcldb add_collector -srcname SYS1.DBASECURITY.COM -srcusr
sys1_src/7ygvfr4 -agentname sys1_agent -colltype OSAUD
source SYS1.DBASECURITY.COM verified for OS File Audit Collector collector
Adding collector...
Collector added successfully.
collector successfully added to Audit Vault

remember the following information for use in avctl
Collector name (collname): OSAUD_Collector

```

To add a REDO collector that will capture before and after values for data changes run:

```

-bash-2.05b$ avorcldb add_collector -srcname SYS1.DBASECURITY.COM -
srcusr
sys1_src/7ygvfr4 -agentname sys1_agent -colltype REDO -avsrcusr
sys1_src/6tfcde3 -av avserver.dbasecurity.com:1521:av_server

```

```

source SYS1.DBASECURITY.COM verified for REDO Log Audit Collector collector
Adding collector...
Collector added successfully.
collector successfully added to Audit Vault

remember the following information for use in avctl
Collector name (collname): REDO_Collector
initializing REDO Collector
setting up APPLY process on Audit Vault server
setting up CAPTURE process on source database

```

If a collector cannot be added, for example because you never fixed the parameters after validation, you will get an error as follows:

```

-bash-2.05b$ avorcldb add_collector -srcname SYS2.DBASECURITY.COM -srcusr
sys2_src/7ygvfr4 -agentname sys2_agent -colltype REDO -avsrcusr
sys2_src/6tfcde3 -av avserver.dbasecurity.com:1521:av_server
ERROR: parameter _SPIN_COUNT is not set; recommended value is 5000
ERROR: parameter _JOB_QUEUE_INTERVAL is not set; recommended value range [4 -
ANY_VALUE]
ERROR: parameter UNDO_RETENTION = 900 is not in recommended value range [3600 -
ANY_VALUE]
ERROR: parameter GLOBAL_NAMES = false is not set to recommended value true
ERROR: source database must be in ARCHIVELOG mode to use REDO LOG collector
ERROR: set the above init.ora parameters to recommended/required values

```

The last step is running avorcldb on the agent instance:

```

-bash-2.05b$ avorcldb setup -srcname SYS1.DBASECURITY.COM -srcusr
sys1_src/7ygvfr4 -wpwd dbasecurity
unrecognized argument -verbose was ignored
updated tnsnames.ora with alias [SRCDB21] to source database
adding credentials for user sys1_src for connection [SRCDB21]
Storing user credentials in wallet...
Create credential oracle.security.client.connect_string2
done.
verifying SRCDB21 connection using wallet

```

You're done. If you log onto the host where the collectors are running you will see a separate process per collector (in this example the sys2 server is running both DBAUD and OSAUD):

```

-bash-3.00# ps -ef | grep coll
1003      5977      1 2 Jan10 ?          00:39:26
/home/oravault2/oracle/product/10.2.2/av_agent_1/bin/avaudcoll
hostname=sys2.dbasecurity.com sourcename=sys2.dbasecurity.com
collectorname=DBAUD_Collector avname=AV command=START
1003      6012      1 0 Jan10 ?          00:00:06
/home/oravault2/oracle/product/10.2.2/av_agent_1/bin/avoscoll hostname=
sys2.dbasecurity.com sourcename= sys2.dbasecurity.com
collectorname=OSAUD_Collector avname=AV command=START

```

You can review and manage configurations and status for your agents and collectors. Reviewing and controlling collectors can be done either from the AV console or using the avctl utility. From the AV console, log on with the AV\_ADMIN role. The first page will show you the status of your collectors as shown in Figure 13.9. You can review the status of each collector as well as

## Collectors

Select	Collector	Agent	Audit Source	Status	Records Per Second	Bytes Per Second
<input checked="" type="radio"/>	DBAUD_Collector	sys1_agent	sys1.dbasecurity.com	↑ 0.0000	0.0000	0.0000
<input type="radio"/>	OSAUD_Collector	sys1_agent	sys1.dbasecurity.com	↑ 0.0000	0.0000	0.0000
<input type="radio"/>	REDO_Collector	sys1_agent	sys1.dbasecurity.com	↓ 0	0	0

Figure 13.9 View start and stop collectors.

manage them. If you click on the Configuration tab on the top right and then on the Audit Source tab on the left, you can select a source and edit its configuration.

### Encrypting Collector Traffic

The data that is sent from the collector to the AV server can be highly sensitive—it includes data as part of the queries performed as well as before/after values. Therefore, you may need to encrypt this traffic to ensure that you are not creating a vulnerability through your AV implementation. Because the AV collectors are processes that read out of the source database (locally) and then write to the AV database as a regular ORACLE\*NET application, you can use any of the facilities for network encryption that you saw in Chapter 7 to secure AV collector traffic. You can even use a non-Oracle scheme to encrypt the traffic—e.g., use IPSEC to encrypt network traffic between the source server and the AV server. If you choose to use ASO to encrypt these communications follow the instructions in Section 7.2—there is nothing special about AV collector communications. Remember that setting up ASO network communications between the source host as a client and the AV server will only encrypt the collector traffic. The agent traffic is encrypted (and more importantly—authenticated) using Secure Sockets Layer (SSL) as discussed previously in HOWTO 13.1.

### Using AVCTL to Manage Agents and Collectors

You can manage the collectors using avctl on the AV server. First, you need to make sure that the agent is started

```
-bash-2.05b$ avctl show_agent_status -agentname sys1_agent
AVCTL started
Getting agent metrics...
-----
Agent is running
-----
Metrics retrieved successfully.
```

Note that the agent name is case sensitive—make sure you stay consistent. For example:

```
-bash-2.05b$ avctl show_agent_status -agentname SYS1_AGENT
AVCTL started
Getting agent metrics...
Error executing task show_agent_status: Unknown Agent SYS1_AGENT
```

To check the status of a collector:

```
-bash-2.05b$ avctl show_collector_status -collname DBAUD_Collector -srcname
sys1.dbasecurity.com
AVCTL started
Getting collector metrics...
-----
Collector is running
-----
```

Use the avctl stop\_agent command to stop an agent. You can't stop an agent when it has running collectors; if you try to stop, you get the following error:

```
-bash-2.05b$ avctl stop_agent -agentname sys1_agent
AVCTL started
Stopping agent...
Error executing task stop_agent: Cannot stop agent sys1_agent; Agent has running
collectors sys1.dbasecurity.com:OSAUD_Collector
```

So you'll first have to stop the collectors:

```
-bash-2.05b$ avctl stop_collector -collname DBAUD_Collector -srcname sys1.dbasecurity.com
AVCTL started
Stopping collector...
Collector stopped successfully.
```

Once all collectors have been stopped you can stop the agent:

```
-bash-2.05b$ avctl stop_agent -agentname sys1_agent
AVCTL started
Stopping agent...
Agent stopped successfully.
```

To start the agent:

```
-bash-2.05b$ avctl start_agent -agentname sys1_agent
AVCTL started
Starting agent...
Agent started successfully.
```

To start a collector (don't be alarmed if it takes a long time to start):

```
-bash-2.05b$ avctl start_collector -collname DBAUD_Collector -srcname
sys1.dbasecurity.com
AVCTL started
Starting collector...
Collector started successfully.
```

### Two Things to Remember about AV Collectors

1. Collectors are added using the avorcldb utility on the AV server and then on the agent.
2. Collectors can use ASO native network encryption to secure the network communications with the AV server's database.

## 13.4 HOWTO Configure Audit Rules

Setting up the agents, sources, and collectors provides the infrastructure to audit it. It allows you to control what gets audited on the servers and allows the collectors to feed in data to the AV server. Now it's time to define what statements, privileges, and objects you want to audit. You do not need to define your audit policy on the source databases themselves—you can set up the rules from the AV console.

To set up audit rules you log on to the AV console as a user with the AV\_AUDITOR role. If you log on with the AV\_ADMIN role you will not have access to this application. Once you log on as an auditor click on the Audit Policy tab at the top right. This brings you to the Audit Settings page. Find your audit source—if you don't see it on the list you may have to put in a partial name into the search field and click Go. Select your audit source and click on the Retrieve from Source button. This connects to the source and retrieves the latest audit settings from the source database. You can see the audit definitions even if you manually use the AUDIT command directly within the source database. This view provides a high-level overview on the number of audit rules in use, problems and the status of the source. The number under In Use is the number of active settings on the source database. The number under Needed is the number of audit settings that you set up in AV but are not active on the source database yet.

Click on the audit source link to drill down and manage the audit settings for your source. On this page you manage and provision the audit policy for a source. The page is composed of internal tabs as shown in Figure 13.10. The Overview tab on this page is where you provision the audit policy to the source. Each of the individual secondary tabs (Statement, Object, Privilege, FGA, and Capture Rule) is used to specify the audit rules based on what you want to audit. Statement,

Select Audit Settings Type	In Use	Needed	Problem
<input checked="" type="checkbox"/> STATEMENTS	66	0	0
<input checked="" type="checkbox"/> OBJECTS	1007	0	1007
<input checked="" type="checkbox"/> PRIVILEGES	2	0	2
<input checked="" type="checkbox"/> FGA	0	0	0
<input checked="" type="checkbox"/> CAPTURE RULE	0	0	0

From C%

\* Audit Source User Name    
 \* Audit Source Password

**Copy Audit Settings from Another Source**

You can quickly replicate audit settings from one database to another to seed it with common audit settings. You can either use settings that are already in use on the database or settings that you have set up in Audit Vault but not yet applied to that database.

Copy  Actual(In Use)  Needed(Not Yet In Use) Audit Settings

Figure 13.10 Managing the audit policy for a source.

object, and privilege tabs use Oracle standard auditing as you saw in Chapter 9—you can define audit rules for auditing by statement, by object or by privilege used. The FGA tab allows you to define FGA rules as you learned in Chapter 11. On the capture rule page you define how the REDO collector will work and what capture rules will be used to get before/after values.

After you finish building audit rules go back to the overview page. Click Save All Audit Settings to save your work. Saving the policy does not push it to the source database—you have to explicitly provision this policy to the source. First, you should verify that your policy will install correctly on the source by clicking on the Verify button. Then, provision the policy to the source by entering the source user name and password and clicking on the Provision button. AV will connect to the source database (using the credentials you enter) and send a set of NOAUDIT and AUDIT commands to the source database.

You can also export the rules as SQL from this page or copy an audit policy from another source. This is very useful if you are managing hundreds of servers and need to apply a consistent audit policy across them. It is not feasible to have to build these policies per source—so you can copy an audit policy from one source to another. Better yet, you can use the exported SQL and run it on your databases. The export file is a simple set of NOAUDIT and AUDIT instructions. Listing 13.1 shows a partial example of exporting the rules shown in Figure 13.11.

To build audit rules click on the appropriate tab. For example, if you want to audit based on a statement click the Statement tab. AV shows you all the statement-based audit rules that are currently being audited as shown in Figure 13.11. Each one of these lines corresponds to an audit

```

NOAUDIT ROLE BY SCOTT;
NOAUDIT DIMENSION BY SCOTT;
NOAUDIT TYPE BY SCOTT;
NOAUDIT SYSTEM GRANT BY SCOTT;
NOAUDIT SYNONYM BY SCOTT;
NOAUDIT INDEX BY SCOTT;
NOAUDIT DIRECTORY BY SCOTT;
NOAUDIT PROFILE BY SCOTT;
NOAUDIT ALTER JAVA CLASS;
NOAUDIT ALTER JAVA RESOURCE;
NOAUDIT ALTER JAVA SOURCE;
NOAUDIT ALTER SEQUENCE;
NOAUDIT ALTER TABLE;
...
AUDIT ROLE BY SCOTT BY ACCESS;
AUDIT DIMENSION BY SCOTT BY ACCESS;
AUDIT TYPE BY SCOTT BY ACCESS;
AUDIT SYSTEM GRANT BY SCOTT BY ACCESS;
AUDIT SYNONYM BY SCOTT BY ACCESS;
AUDIT INDEX BY SCOTT BY ACCESS;
AUDIT DIRECTORY BY SCOTT BY ACCESS;
AUDIT ALTER JAVA CLASS BY ACCESS;
AUDIT ALTER JAVA RESOURCE BY ACCESS;
AUDIT ALTER JAVA SOURCE BY ACCESS;
AUDIT ALTER SEQUENCE BY ACCESS;
AUDIT ALTER TABLE BY ACCESS;
...

```

---

**Listing 13.1 Export file generated from AV.**

The screenshot shows a table of audit rules. The columns are: Statement, User, Proxy User, Execution Condition, Audit granularity, In Use, and Needed. The rows list various database objects and their audit configurations. Most rows have 'BY ACCESS' in the Audit granularity column and an up arrow in the In Use column. Checkmarks are present in the Needed column for most rows.

Statement	User	Proxy User	Execution Condition	Audit granularity	In Use	Needed
ROLE	SCOTT		BOTH	BY ACCESS	↑	✓
DIMENSION	SCOTT		BOTH	BY ACCESS	↑	✓
TYPE	SCOTT		BOTH	BY ACCESS	↑	✓
SYSTEM GRANT	SCOTT		BOTH	BY ACCESS	↑	✓
SYNONYM	SCOTT		BOTH	BY ACCESS	↑	✓
INDEX	SCOTT		BOTH	BY ACCESS	↑	✓
DIRECTORY	SCOTT		BOTH	BY ACCESS	↑	✓
PROFILE	SCOTT		BOTH	BY ACCESS	↑	✓
NOT EXISTS	SCOTT		BOTH	BY ACCESS	↑	✓
PUBLIC DATABASE LINK	SCOTT		BOTH	BY ACCESS	↑	✓
CLUSTER	SCOTT		BOTH	BY ACCESS	↑	✓
TABLESPACE	SCOTT		BOTH	BY ACCESS	↑	✓
PROCEDURE	SCOTT		BOTH	BY ACCESS	↑	✓
ALTER JAVA CLASS			BOTH	BY ACCESS	↑	✓
ALTER JAVA RESOURCE			BOTH	BY ACCESS	↑	✓
ALTER JAVA SOURCE			BOTH	BY ACCESS	↑	✓
ALTER SEQUENCE			BOTH	BY ACCESS	↑	✓
ALTER TABLE			BOTH	BY ACCESS	↑	✓
CLUSTER			BOTH	BY ACCESS	↑	✓
COMMENT TABLE			BOTH	BY ACCESS	↑	✓
CONTEXT			BOTH	BY ACCESS	↑	✓
CREATE JAVA CLASS			BOTH	BY ACCESS	↑	✓
CREATE JAVA RESOURCE			BOTH	BY ACCESS	↑	✓
CREATE JAVA SOURCE			BOTH	BY ACCESS	↑	✓
DATABASE LINK			BOTH	BY ACCESS	↑	✓

**Figure 13.11 Defining audit rules.**

statement that you saw in Chapter 9. The up arrow in the In Use column shows that this is what the source database is currently doing and the check marks in the Needed column show you that all these rules are required based on the audit policy defined within AV. You can create a new rule by clicking on the Create button in which case you enter which statements you want to audit, by access or by session on success or failure, etc.

Let's look at Figure 13.12. This shows two audit rules that are active in the source database—the In Use shows an up arrow. However, they both have an X-mark in the Needed column. This happens when the audit policy as you define it within AV does not require these audit rules—for example, when you add AUDIT statement directly within the source database and not through AV. These rules will be deleted the next time you provision the policy to the source database.

The screenshot shows a table of privileges. The columns are: Privilege, User, Proxy User, Execution Condition, Audit granularity, In Use, and Needed. There are two rows, both of which have an exclamation mark icon in the Privilege column and 'SCOTT' in the User column. The Audit granularity is 'BY ACCESS' and the In Use column has an up arrow. Both rows have an X-mark in the Needed column.

Privilege	User	Proxy User	Execution Condition	Audit granularity	In Use	Needed
AUDIT SYSTEM	SCOTT		BOTH	BY ACCESS	↑	✗
AUDIT SYSTEM			BOTH	BY ACCESS	↑	✗

**Figure 13.12 What happens when an audit rule is not needed.**

### The e Things to Remember about Configuring Audit Rules

1. Manage your audit rules through the AV administrator application.
2. Audit rules are provisioned by changing the audit definitions in the source databases using the source.
3. Audit rules can be exported and imported and you can compare an audit policy with what is currently running within a source database.

## 13.5 HOWTO Configure and Manage the AV Server and the Warehouse

To start the AV server you should first startup the database used by the AV server. Then, start the AV server (which runs within an oc4j container):

```
-bash-2.05b$ avctl start_av
AVCTL started
Starting OC4J...
OC4J started successfully.
TZ set to US/EasternOracle Audit Vault 10g Database Control Release 10.2.2.0.0
Copyright (c) 1996, 2005 Oracle Corporation. All rights reserved.
http://avserver.dbasecurity.com:5700/av
Oracle Audit Vault 10g is running.

-----
Logs are generated in directory
/home/oravault/oracle/product/10.2.2/av_1/av/log
```

The AV server must be running before you can add agents, collectors, and sources. To stop the AV server run:

```
-bash-2.05b$ avctl stop_av
AVCTL started
Stopping OC4J...
OC4J stopped successfully.
```

To see whether AV is running:

```
-bash-2.05b$ avctl show_av_status
AVCTL started
TZ set to US/EasternOracle Audit Vault 10g Database Control Release 10.2.2.0.0
Copyright (c) 1996, 2005 Oracle Corporation. All rights reserved.
http://avserver.dbasecurity.com:5700/av
Oracle Audit Vault 10g is running.

-----
Logs are generated in directory
/home/oravault/oracle/product/10.2.2/av_1/av/log
```

Collectors feed data into the AV server's database. You view this data through the AV reports. The collectors however do not directly populate the data into tables that the reports use.

The collectors feed the data into a flat table in the AV Raw Audit Data Store (RADS) and there is a separate loading process which populates the audit warehouse from this table. You can run this process either from the command line or from the AV console. From the command line you have two options—you can populate the warehouse in the background or you can wait for the warehouse to populate. To wait for the process run:

```
-bash-2.05b$ avctl refresh_warehouse -wait
AVCTL started
Refreshing warehouse...
Waiting for refresh to complete...
done.
```

This may take a long time. To run the process in the background:

```
-bash-2.05b$ avctl refresh_warehouse
AVCTL started
Refreshing warehouse...
done.
```

Note that what is “done” here is the avctl run—the process refreshing the warehouse is still running. You cannot have two processes refreshing the warehouse running simultaneously—you’ll get the following error:

```
-bash-2.05b$ avctl refresh_warehouse
AVCTL started
Refreshing warehouse...
done.
-bash-2.05b$ avctl refresh_warehouse -wait
AVCTL started
Refreshing warehouse...
Waiting for refresh to complete...
Error executing task refresh_warehouse: ORA-46623: Message 46623 not found;
product=RDBMS; facility=ORA
; arguments: [OAV-46623: cannot execute warehouse operation; another operation
is currently running
ORA-06512: at "AVSYS.DBMS_AUDIT_VAULT", line 6
ORA-06512: at "AVSYS.AV$DW", line 823
ORA-06512: at "AVSYS.AV$DW", line 1754
ORA-06512: at line 1
ORA-06512: at "AVSYS.DBMS_AUDIT_VAULT", line 836
]
ORA-06512: at "SYS.DBMS_ISCHED", line 150
ORA-06512: at "SYS.DBMS_SCHEDULER", line 441
ORA-06512: at "AVSYS.DBMS_AUDIT_VAULT", line 864
ORA-06512: at line 1
```

You can also refresh the warehouse from the AV console. Log onto the AV console as an AV\_ADMIN, click on the Management tab at the top right and on the Warehouse tab on the left. You can see all the refresh jobs that were run before and you can click on the Refresh Now button as shown in Figure 13.13. What you usually do is set up a refresh schedule. On the AV console click on the Configuration tab at the top right and on the Warehouse tab on the left. On the Warehouse

## Warehouse Load History

[History of Refreshing](#) [History of Loading](#) [History of Purging](#)

Refresh Now						
Scheduled	Start	Duration (Minutes)	CPU Used	Error Number	Message	Status
2008-01-12 00:29:54	2008-01-12 00:29:54	0 0:7:3.0	0 0:1:14.500000000	0		SUCCEEDED
2008-01-11 08:14:55	2008-01-11 08:14:54	0 0:1:1.0	0 0:0:11.670000000	0		SUCCEEDED
2008-01-11 00:29:54	2008-01-11 00:29:54	0 6:37:38.0	0 0:39:45.760000000	0		SUCCEEDED
2008-01-10 00:29:54	2008-01-10 00:29:54	0 0:0:30.0	0 0:0:0.2.710000000	0		SUCCEEDED
2008-01-09 00:29:54	2008-01-09 00:29:54	0 0:0:25.0	0 0:0:0.2.820000000	0		SUCCEEDED
2008-01-08 00:29:54	2008-01-08 00:29:54	0 0:1:57.0	0 0:0:34.390000000	0		SUCCEEDED
2007-10-23 00:29:54	2008-01-07 12:48:49	0 0:0:41.0	0 0:0:2.860000000	8103	ORA-08103: object no longer exists	FAILED

Figure 13.13 Refreshing the warehouse from the AV console.

## Warehouse Settings

The Audit Vault Warehouse is a moving window against the incoming audit data stream. You can run reports against the audit data visible through this window.

**Schedule to Send New Data**  
 Specify the frequency with which new audit data will be moved to the warehouse.

Schedule Type  Use Pre-defined Schedule  Standard

\* Frequency Type    
 \* Interval (Hours)    
 \* Start Date     
(example: Oct 5, 2006)   
 \* Start Time    AM  PM

**Retention Time**  
 Specify the size of the warehouse window.

Retention Time  \* Year  \* Months

Figure 13.14 Setting a refresh schedule for populating the warehouse.

Settings page select the Standard radio button as shown in Figure 13.14. Define the period you want for the warehouse refresh and click Apply.

When you install AV you will find that you can't log onto the database using sysdba privileges. This is caused by the fact that AV uses Database Vault under the covers for securing the audit data. However, it means that many Oracle utilities will not work. As an example, you will not be able to use RMAN to backup your database. Therefore, one of the things you may need to do is re-enable sysdba connections. To do that, run the orapwd utility with nosysdba = n—this will generate a password file allowing sysdba connections and now you can log onto the database as sysdba when using a password. For more information see HOWTO 17.5.

### Two Things to Remember about the AV Data Warehouse

1. Collectors or write into a flat table; it is the ETL process that populates the audit warehouse from the flat table.
2. The end-to-end cycle in AV comprises of audit records being written using standard or fine grained auditing, collectors that read these records and write it into the flat table, and an ETL batch that populates the data warehouse.

## 13.6 HOWTO View Audit Data within the AV Console

To view audit data log onto the AV console using an AV\_AUDITOR role. Click on the Audit Reports tab at the top right and on the activity Reports tab on the left. Expand the Triangular icons on the left to access a specific report as shown in Figure 13.15. Click on any one of the links to go to that report. Note that in Figure 13.15 the Delete column is empty for all the reports. The only reports configured in that sample system are the AV predefined reports—you can't delete these reports. You can only delete reports that you have added yourself.

Each of the Activity Reports shows you audit information for a certain event category. The Activity Overview Report (shown in Figure 13.16) shows you data across the different event categories. For each audit record shown you can click on the Detail icon on the right to drill down to a detailed view of that audit event.

For each report you can specify filtering criteria as shown in Figure 13.17 (this is the filter for the account management activity report). Most of the search criteria will be the same for all reports—what changes is the event type because each report is based on one event category. Enter a range of dates and other filtering criteria and click on Go to retrieve data from the warehouse.

You can create customized reports based on the predefined reports. As an example, let's create a PCI related report. You may want to track a certain table that has card holder information. Go to the data access activity report and define a filter where the Object field has your card holder table.

### Activity Reports

Use this screen to run reports about activity on the various audit sources.

Type	Report	Description	Delete
▼	Activity Overview Report	Activity Overview	
▼	Activity Reports		
	Account Management Activity	This report displays account management activity on the various audited sources.	
	Application Management Activity	This report displays application management activity on the various audited sources.	
	Audit Command Activity	This report displays the use of audit commands on the various audited sources.	
	Data Access Activity	This report displays data manipulation activity on the various audited sources.	
	Exception Activity	This report displays errors and exceptions on the various audited sources.	
	Invalid Audit Record Activity	This report displays events that could not be understood by Audit Vault; there may be some corruption of the audit record.	
	Object Management Activity	This report displays object management activity on the various audited sources.	
	Peer Association Activity	This report displays peer association activity on the various audited sources.	
	Role and Privilege Management Activity	This report displays role and privilege management activity on the various audited sources.	
	Service and Application Access Activity	This report displays application access activity on the various audited sources.	
	System Management Activity	This report displays system management activity on the various audited sources.	
	Uncategorized Activity	This report displays activity on the various audited sources that has not been categorized.	
	User Session Activity	This report displays user session activity on the various audited sources.	

Figure 13.15 AV audit reports.

Audit Source	User	Audit Event Category	Audit Event	Object	Client Host	Client Tool	Privilege	Audit Event Status	Time	Detail
SRCCUSR1	DATA ACCESS	SELECT	SYS.GV\$_SESSION	sys4.dbasecurity.com			0	Jan 12, 2008 9:15:06 AM EST		
SRCCUSR1	DATA ACCESS	SELECT	SYS.GV\$SESSION	sys4.dbasecurity.com			0	Jan 12, 2008 9:15:06 AM EST		
SRCCUSR1	DATA ACCESS	SELECT	SYS.X\$KSLED	sys4.dbasecurity.com			0	Jan 12, 2008 9:15:06 AM EST		
SRCCUSR1	DATA ACCESS	SELECT	SYS.X\$KSUSE	sys4.dbasecurity.com			0	Jan 12, 2008 9:15:06 AM EST		
SRCCUSR1	DATA ACCESS	SELECT	SYS.AUDS	sys4.dbasecurity.com			0	Jan 12, 2008 9:15:06 AM EST		
SRCCUSR1	DATA ACCESS	SELECT	SYS.FGA_LOG\$	sys4.dbasecurity.com			0	Jan 12, 2008 9:15:06 AM EST		
SRCCUSR1	DATA ACCESS	SELECT	SYS.AUD\$	sys4.dbasecurity.com			0	Jan 12, 2008 9:15:05 AM EST		
SRCCUSR1	DATA ACCESS	SELECT	SYS.FGA_LOG\$	sys4.dbasecurity.com			0	Jan 12, 2008 9:15:05 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.DUAL	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:34 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.USER_POLICIES	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:34 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.DBA_POLICIES	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:34 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.RLSS	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:34 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.OBJ\$	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:34 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.USERS\$	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:34 AM EST		
SCOTT	APPLICATION MANAGEMENT	DROP TRIGGER	SCOTT_AUDIT_DB_OBJECT	sys3.dbasecurity.com			4080	Jan 12, 2008 8:57:34 AM EST		
SCOTT	SERVICE AND APPLICATION UTILIZATION	EXECUTE PROCEDURE	SYS.DICTIONARY_OBJ_TYPE	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:34 AM EST		
SCOTT	SERVICE AND APPLICATION UTILIZATION	EXECUTE PROCEDURE	SYS.DBMS_STANDARD	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:34 AM EST		
SCOTT	SERVICE AND APPLICATION UTILIZATION	EXECUTE PROCEDURE	SYS.DBMS_STANDARD	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:34 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.X\$KZSRO	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:33 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.V\$ENABLEDPRIVS	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:33 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.GV\$ENABLEDPRIVS	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:33 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.X\$KZSPR	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:33 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.DBA_POLICIES	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:33 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.RLSS	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:33 AM EST		
SCOTT	DATA ACCESS	SELECT	SYS.OBJ\$	sys3.dbasecurity.com			0	Jan 12, 2008 8:57:33 AM EST		

**Figure 13.16** Activity overview report.

### Account Management Activity

This report displays account management activity on the various audited sources.

Audit Source	<input style="width: 100%;" type="text" value="%"/>		
User	<input style="width: 100%;" type="text" value="%"/>		
Account Management Audit Event	<input style="width: 100%;" type="text" value="%"/>		
Object	<input style="width: 100%;" type="text" value="%"/>		
Audit Event Status <input type="radio"/> Success <input type="radio"/> Failure <input checked="" type="radio"/> Both			
Audit Event Time <input type="radio"/> Last 24 Hours <input checked="" type="radio"/> Last One Week <input type="radio"/> Last One Month <input type="radio"/> The Period From <input style="width: 20px;" type="text" value="0"/> : <input style="width: 20px;" type="text" value="0"/> on <input style="width: 100px;" type="text" value="Jan 1, 2007"/> To <input style="width: 20px;" type="text" value="0"/> : <input style="width: 20px;" type="text" value="0"/> on <input style="width: 100px;" type="text" value="Jan 1, 2008"/> <input type="button" value="Go"/>			

**Figure 13.17** Filtering for audit events.

Click on the Save Definition button on the top right. Fill in the values as shown in Figure 13.18 and click on the Create Report button. If you navigate back to the Activity Reports tab at the top left you'll see that you have a new type (PCI) and that this report does have a Delete button as shown in Figure 13.19.

## Report Generation

---

Audit Event Category	DATA ACCESS
Audit Event Status	Both
Audit Event Time	Last 24 Hours
Compliance Category	<input type="text" value="PCI"/> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> Data access to card holder information </div>
Compliance Category Description	
Report Title	<input type="text" value="PCI - Data access to card holder information"/>
Short Report Description	<input type="text" value="Data access to card holder information"/>
Detail Report Description	<input type="text" value="Data access to card holder information"/>

Figure 13.18 Generate a customized report.

## Activity Reports

---

Use this screen to run reports about activity on the various audit sources.

Type	Report	Description	Delete
▼ /			
▼ Activity Overview Report	Activity Overview Report	Activity Overview	
► Activity Reports			
▼ PCI	PCI - Data access to card holder information	Data access to card holder information	

Figure 13.19 Activity reports with predefined and custom reports.

## 13.7 HOWTO Configure Alerts

In addition to reports, you can get AV to fire an alert when a certain audit event occurs. You configure alerts based on audit events. For example, you can define an audit rule not only to record when a user performed some activity but also to fire an alert and notify you of the violation.

Let's look at an example. You'll define an alert that fires when someone drops a table. In this case the audit event is based on a DROP TABLE statement—you don't care what privileges are used to perform the drop and you don't care what object is dropped. Follow these steps to create the alert:

1. Log onto the AV console with the AV\_AUDITOR role.
2. Click the Audit Policy tab at the top right and the Alerts tab on the left.
3. Click on the Create button.

**Create Alert Rule**

Please provide the data for all required fields

* Alert	Alert on DROP TABLE	<input type="button" value="Cancel"/>	<input type="button" value="OK"/>
Description	Alert when a table is dropped	<input type="button" value=""/>	
* Alert Severity	Warning	<input type="button" value=""/>	
Audit Source Type	ORCLDB	<input type="button" value=""/>	
Audit Source	test_source	<input type="button" value=""/>	
Audit Event Category	OBJECT MANAGEMENT	<input type="button" value=""/>	
Specify additional alert conditions in <input checked="" type="radio"/> Basic <input type="radio"/> Advanced			
<b>Basic Alert Condition</b>			
Specify when an alert should be raised.			
User	<input type="text"/>	<input type="button" value=""/>	<input type="button" value=""/>
Table	<input type="text"/>	<input type="button" value=""/>	<input type="button" value=""/>
Audit Event	DROP TABLE	<input type="button" value=""/>	
Audit Event Status	<input type="radio"/> Success <input type="radio"/> Failure <input checked="" type="radio"/> Both	<input type="button" value=""/>	

---

**Figure 13.20 Creating an alert on DROP TABLE.**

4. Fill in the values as shown in Figure 13.20. Select the audit source for your database and select the OBJECT MANAGEMENT event category. In the Basic Alert Condition area leave User and Table empty because you want to alert on any table that is dropped and for any user. In the Audit Event pull down, select DROP TABLE from the command list.

5. Click on the OK button.

The alert you created above fires when a DROP TABLE occurs in one source database. If you want the alert to fire whenever a table is dropped in any of your databases leave the audit source empty—any table that will be dropped on any database that is being collected by AV will fire the alert. Note that you cannot edit an alert—if you want to change an alert after it has been created you should remove it and re-create it.

If you want to fire an alert on a complex condition select Advanced in the Specify additional alert condition radio button. You can then create a boolean expression as the Advanced Alert Condition as shown in Figure 13.21. Use the two pull downs at the bottom of the page to compose the condition as an AND of conditions. Every time you select from the bottom two pull-downs a new atomic condition is added based on the AV codes. As an example (Figure 13.21), follow these steps:

Create Alert Rule

Please provide the data for all required fields

* Alert <input type="text" value="Sample advanced alert"/> Description <input type="text" value="Sample advanced alert"/> * Alert Severity <input type="button" value="Critical"/> Audit Source Type <input type="button" value="ORCLDB"/> Audit Source <input type="text"/> Audit Event Category <input type="button" value="OBJECT MANAGEMENT"/>	<input type="button" value="Cancel"/> <input type="button" value="OK"/>
---	---

Specify additional alert conditions in  Basic  Advanced

**Advanced Alert Condition**

Enter a valid Boolean condition under which an alert should be raised. You may use any of the constructs below. Please ensure that the condition is syntactically correct, that it contains only the attributes listed below, and that all values entered are valid.

\* Condition:

```
SOURCE_EVENTID = '12' AND USERNAME = 'SCOTT' AND
OSUSER_NAME=RON'
```

Select an event to insert it in the condition

Select an attribute to insert it in the condition

---

Figure 13.21 Building an alert with an advanced alert condition.

1. Create a new rule and select OBJECT MANAGEMENT in the Audit Event Category pull-down.
2. Select CREATE TABLE in Select an event to insert it in the condition pull-down. This creates the first condition as SOURCE\_EVENTID = '12'. Add an AND at the end of the condition.
3. Suppose that you want to fire an alert only when the user SCOTT drops the table. Select USERNAME from the last pull down. This adds USERNAME to the condition and you can tack on = 'SCOTT' so now your condition looks like SOURCE\_EVENTID = '12' AND USERNAME = 'SCOTT'.
4. Finally, suppose you also want to limit this by the OS user that made the connection. Select OSUSER\_NAME from the same pull down. Keep adding conditions to refine when an alert should fire based on your requirements.

For a full list of AV codes and actions with which you can build advanced alerts see Section B.2 in the AV Auditor's Guide.

Alerts fire when collectors insert data into the RADS flat table. They are not fired at real time (because collectors have a polling interval) but they are not dependent on the refreshing

of the warehouse. Because the refresh warehouse batch usually runs infrequently, having the alerts fire when the data first arrives into AV is important. Alerts that are fired are visible in the AV console as reports and in the dashboard. Additionally, the alert is also placed in an output queue within the AV database and you can use Oracle (or other) facilities to send e-mails, SMTP traps, write to syslog, or any other action you need to perform. This follows a publish/subscribe pattern—see AVSYS. AQ\$\_AV\_ALERT\_QT\_S and AVSYS. AV\_ALERT\_QT for more details.

#### **Two Things to Remember about Alerts**

1. Alerts fire when the collectors write into the flat table. Alerts are not real time because the collector itself has a polling interval but they are near real time.
2. Disable alerts when you restore data so they won't fire on old data.

## **13.8 HOWTO Understand Performance and Storage Impact**

AV is an application that uses a centralized audit warehouse within an Oracle database. The OSAUD and DAAUD collectors read data from files and from database tables and write them directly into the AV database. The REDO collector writes into the AV database over a database link. When you look into the performance and storage requirements for AV there are therefore numerous things you should look at—some on the AV database and some on the source database.

### **Performance**

In terms of performance, there is less concern regarding the load that the AV server creates and you should deploy AV on its own server and assume that it will take up the entire machine because the AV server does do a lot of work. The most important set of numbers in terms of performance is the load that the AV collectors create on the source database. The collectors read data that is generated by the native auditing mechanism and write this data to the AV server's database. Because the collectors run on the source host, this additional work will impact the load on your database machines. Oracle stated numbers for this additional overhead in a scenario of 100 audit records per second is approximately 2 percent CPU overhead for the DBAUD collector, approximately 3 percent CPU overhead for the OSUAD collector and approximately 6 percent CPU overhead for the REDO collector. Observed values have been higher but this may be an issue of tuning. This load will depend on the number of audit records you require per second—the more you audit, the more the collector has to read from the source and write to the AV server's database and as load increases there is an almost linear ramp-up. Observed values for rates of 1000 audit records per second on a Sun E6500 with 28 CPUs have been on average 18 percent CPU overhead when one collector type is running, so be careful what you audit. Also, remember that all these numbers are the additional load placed by the collectors and do not include the load generated by standard auditing or FGA (already discussed in Chapters 9 and 11).

### **Storage**

Storage requirements per AV are fairly easy to measure and define. Audit records have a fixed format and their size can easily be measured. Collectors write into the RADS flat table and the refresh\_warehouse process moved these records into the warehouse. Both Oracle stated numbers

**Table 13.1 Average Storage Requirements per Month Based on Audit Records per Second**

<i>Audit Records per Second</i>	<i>Audit Records per Day (in Millions)</i>	<i>Storage Requirement per Month (in TB)</i>
1,000	86	2
10,000	864	19
20,000	1,728	39
30,000	2,592	58
40,000	3,456	78
50,000	4,320	97
60,000	5,184	117
70,000	6,048	136
80,000	6,912	156
90,000	7,776	175

and observed values show that 1 million audit records take anywhere between 600–900 MB of disk space. On average, this means 750 bytes per audit record which is what you can expect if you look at the table structures.

If you plan on auditing very large servers or many servers and you need comprehensive auditing (e.g., for privacy initiatives) you will need to invest a lot in storage. You may also decide to keep very little data online. The reason is that at very high rates, a lot of storage will be used up very quickly. For example, Table 13.1 shows that if you have 50,000 audit records per second (which can easily be generated by a few large servers or from many medium-sized servers) and you thought of keeping data in the warehouse for a year (which is the default retention policy) you would need to allocate ~1 PB of storage. This is a number that very few companies would be willing to allocate for database auditing. Practically speaking, if you plan of collecting a lot of audit data you have to have a VERY short retention period maybe as short as one week. You have to get your auditors to approve this and you have to generate reports on short intervals and find a way to store the results in some other database. Note that even with a low number such as 1000 audit records per second you need a nontrivial investment in storage, 2 TB per year. Of course, the key to sustainability is compression.

## 13.9 Miscellaneous Discussion—Auditing AV

Section 1 of the Audit Vault Administrator’s Guide (Introduction to Audit Vault) states: “Oracle Audit Vault solves these security and audit problems by

- Consolidating audit information from multiple systems across the enterprise
- Detecting data changes associated with regular and privileged users
- Protecting audit data from modification and tampering”

Let's focus on the third item in this list. Before AV, you had all sorts of native auditing capabilities. The audit data was stored either within database tables or as OS files—but they were always on the host where the DBA often has full control. Storing the audit trails in OS files is more secure than storing it in database tables, but the instance owner usually has access to these files. This problem was (and still is) one of the major sources of criticism for any auditing scheme that is based on native database auditing, for Oracle and all other database platforms.

AV improves this situation. The audit data is not kept on the source database for long (where the audited DBAs have control) and is moved to the AV server's database. AV supports separation of duties in this way—to tamper with audit data you would need cooperation from the owner of the AV server. In addition, AV is built on top of Database Vault and this greatly increases the security of the audit data.

Having said that, you should understand the limitation of the AV model. The audit data is indeed stored in a secure database and the owner of the audit data is not the owner of the audited source (or—more precisely—you must pay attention that they aren't the same person/team). However, the data still is stored within an Oracle database and as such it can still be modified, can be tampered with, and can be deleted. In the next chapter, you'll see some additional audit architectures in which the data is stored on a hardened appliance where the data truly is tamper-resistant. In AV, the data sits in a normal database (installed with Database Vault) and without correct controls, it can be modified—don't forget that.

One of the questions you may ask is whether AV itself needs to be audited? Perhaps even by AV? If it is possible for the audit data to be modified, should there be audit rules placed on the AV server's database for DML on the audit data? This turns out to be impractical. The rates at which audit records are inserted and deleted is too great for AV to have an audit trail of this nature. The only practical thing you can audit are UPDATE statements (which should not occur often) and Data Definition Language (DDL). In addition, AV cannot be a source of itself (through a collector that audits changes to the tables housing the audit trail and then feeds that into the same database)—this could create an infinite loop unless you are very careful. Therefore, you should consider auditing only as it pertains to permissions, administration, and such activities that are being made on the AV server's database and not changes that are made to the audit trail itself. Trust separation of duties and make every effort to put in the right controls around the AV server's database.



## *Chapter 14*

---

# Database Activity Monitoring

---

Database activity monitoring (DAM) is a technology for monitoring and analyzing database activity that operates independently of the database and does not rely on any form of native auditing. Database activity monitoring and prevention (DAMP) is an extension to DAM that also prevents activities from happening even if these activities are allowed according to privileges defined in the database. DAM preserves the most important characteristic of DAM—the independence from the database management system (DBMS).

DAM systems emerged when companies faced many requirements that put a focus on the need to provide more visibility into what activity occurs within production databases. DAM systems are very often used as the solution of choice to implement database auditing. But DAM systems do much more than generate audit trails—the focus of this chapter is mostly on the additional functions DAM systems can perform in addition to auditing.

When monitoring requirements first started emerging, database administrators (DBAs) tried to address them with the most appropriate tool that they had—auditing. By using AUDIT statements (and by using fine-grained auditing [FGA] policies), you can write out what the database is doing and then export these records to another system for analysis, alerting, etc. However, these implementations were lacking in two ways. The first is that the overhead of these schemes made them very expensive to implement. Requirements for activity monitoring can be very diverse and can include monitoring activities that occur very frequently. Although one very important use case for DAM is privileged user monitoring, there are other use cases including application monitoring, sensitive data monitoring, and even comprehensive monitoring. When you monitor (and audit) privileged users only, the burden in terms of the size of the audit trail will not have a huge impact on the database. But as you start monitoring and auditing more and more you need to find access anomalies in Data Manipulation Language (DML) or SELECT access, using native auditing implies a performance hit that most people cannot tolerate. This is the first type of overhead that DAM systems help resolve.

The second is the overhead associated with change management. When an auditor changes requirements, you have to modify the AUDIT statements. You cannot just apply these AUDIT statements to your production systems—you have to make sure this will not have any impact to the application. You need to try it on your test systems first and you may need to test it for a lengthy period of time to ensure that there are no issues in terms of performance and storage.

If you have to pass a yearly audit and every year the requirements change (which is not uncommon) this is a huge cost to bear. Because DAM systems do not rely on the database for monitoring and do not affect the production systems, you can apply changes to monitoring and audit policies without fearing that this will impact the applications.

Beyond the overhead, DAM systems emerged because in addition to monitoring requirements, many architectural and organizational requirements emerged. The most important of them are separation of duties and the need for the information security officers to take ownership of database security and auditing. As long as it is the database that does the auditing and as long as audit trails are written to a location where the DBA or instance owner can modify audit records, and as long as the DBA can modify the audit definitions, the implementation will not pass most audits. Additionally, because these audit trails and the definitions of the policies should be owned by the infosec group (and not by the database group), the tools used to manage the definitions and the data should not require database expertise.

Another important architectural requirement that helped DAM emerge is the fact that there are very few companies that use only Oracle. Most companies have a heterogeneous environment that may include Oracle databases but also includes other databases such as SQL Server, DB2, Sybase, Informix, MySQL, etc. Each database has its own implementation of native auditing. The functionality and the usage is different in each platform. For companies that have such heterogeneous environments DAM provides an easy way out—all functions implemented by DAM systems are equivalent no matter what the monitored database type.

DAM systems also implement real-time alerting. Good DAM systems can alert you of inappropriate access at the moment that it occurs. This allows you to launch a remediation process immediately and for most requirements this is enough. Even if someone grabs sensitive data that they are not entitled to have, if you know about it soon enough and if you have enough information about who it was and where the access was made from, then you can usually contain the issue. DAM is considered to be a very effective technology for combating data breaches (using real-time alerting and precise information about the access). However, it is still a reactive technology—the breach already occurred and the data has already leaked. DAMP is a technology that evaluates policies before letting the queries and transactions reach the database. Rather than alerting when inappropriate access occurs, it simply prevents that access from happening. Both DAM real-time alerting and DAMP prevention are important data-security technologies that are at the core of the DAM value proposition.

DAM systems are different from Security Incident Managers (SIEM) systems. SIEM systems read the audit records produced by the database's audit trails and populate an event repository that includes data from other logs such as firewall logs, routers, operating system logs, etc. The biggest differences between DAM and SIEM are in nonintrusiveness and in monitoring functions:

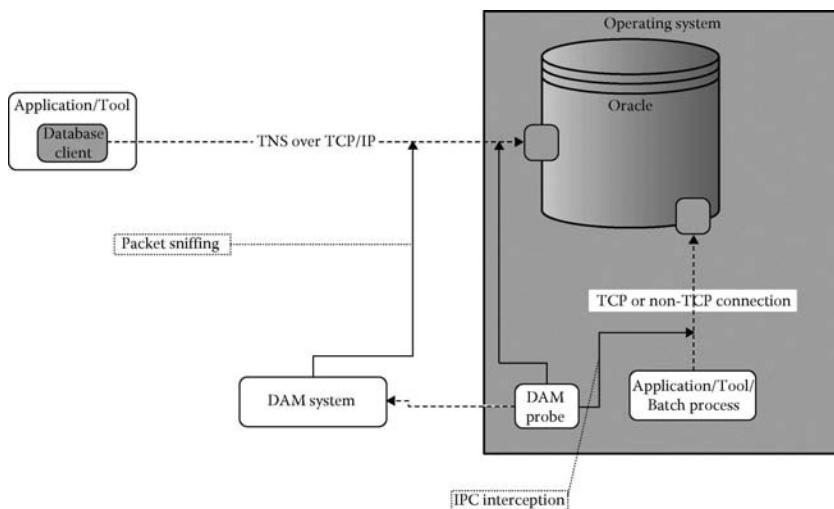
- DAM systems usually produce the monitoring data independently and nonintrusively (to the database), whereas SIEM systems rely on database auditing.
- DAM systems provide far more advanced functions in terms of database monitoring whereas SIEM systems are “generalists” in that database events are merely another type of event.
- SIEM systems can correlate and analyze events from multiple sources whereas DAM systems usually focus on database access (including application access).

Oracle Audit Vault is closer to a SIEM system than to a DAM system both in terms of architecture and in terms of functionality, but it is still an early-stage SIEM in that it can only get logs from Oracle databases and SQL server databases and as of version 10.2.3.1 DB2 UDB and Sybase as well.

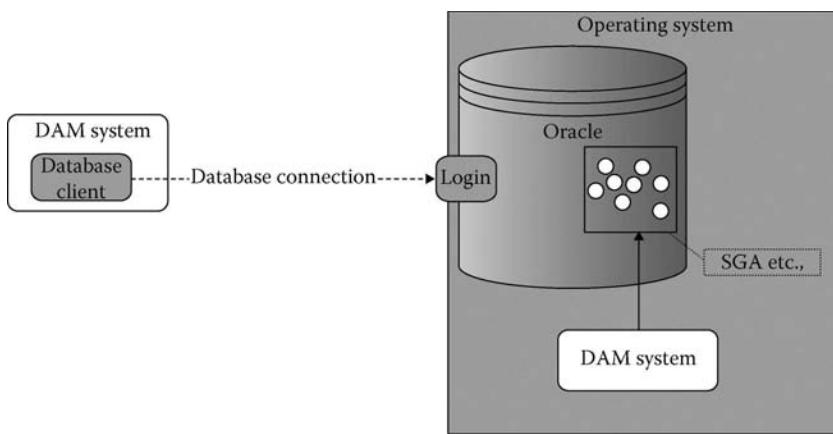
### Common DAM and DAMP Architectures—How They Work?

Although there are many functions that a DAM system provides, the most important function is that of being able to show you what Structured Query Language (SQL) statements were executed on the database. There are three main architectures that DAM systems used:

1. Interception-based architectures (also called inspection-based architectures): Most modern DAM systems collect what the database is doing by being able to “see” the communications between the database client and the databases server. Any database session involves a client that connects to the database server. The client and the server can reside on the same host or can be on different hosts. If the client is running on a remote host the session is a Transparent Network Substrate (TNS) session that usually runs over Transmission Control Protocol/Internet Protocol (TCP/IP). If the client resides on the same host as the server then the session can be of a variety of types—local TCP/IP session or non-TCP/IP sessions (such as those that occur when a client connects using a Bequeath protocol). In any case, there is always a client/server relationship. What DAM systems do is find places where they can view this communication stream and get the requests and responses without requiring participation from the database. The interception itself can be done at multiple points such as the network itself (using a network TAP or a SPAN port—if the communication is not encrypted using advanced security option [ASO]), at the operating system level, or even at the level of the database libraries. As Figure 14.1 shows, if there is unencrypted network traffic then packet sniffing can be used. The advantage is that nothing is done on the host and thus there is no impact on performance whatsoever. To capture local access a probe runs on the host. This probe intercepts all local access and can also intercept all networked access in case you do not want to use network gear or in case the database communications are encrypted. The probe does not do all the processing—it



**Figure 14.1** Inspection-based DAM architecture.

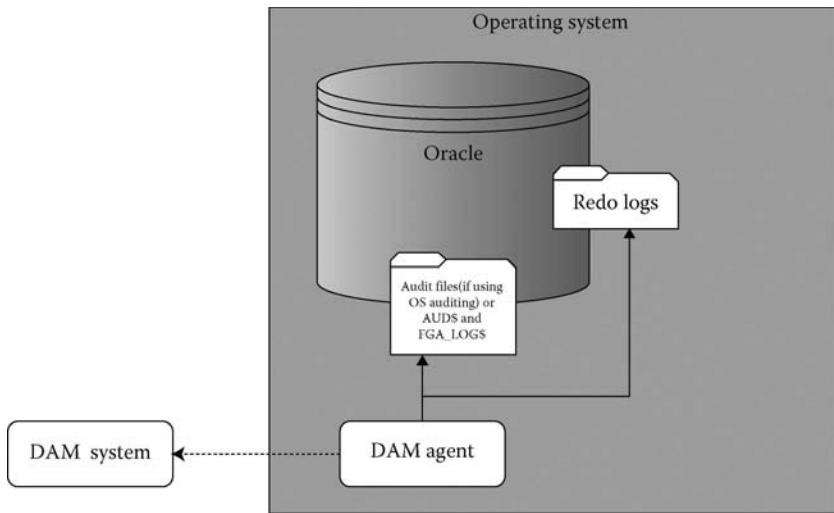


**Figure 14.2** Query-based DAM architecture.

relays the data to the DAM server where all the processing occurs. This ensures that the impact on the database is negligible.

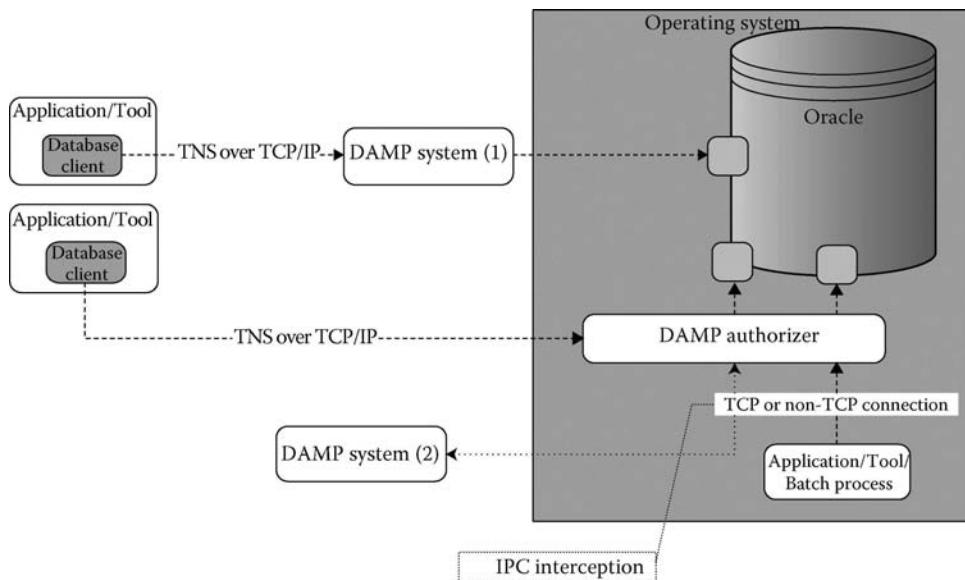
2. **Query-based architectures:** Some DAM systems connect to Oracle and continuously poll the system global area (SGA) to collect what SQL statements are being performed as shown in Figure 14.2. This is a carryover architecture from some of the performance products that used the SGA and other shared data structure. There are two variants of this approach—one that makes a real connection to Oracle and makes valid queries to the database and one that runs on the host and attaches to the process at the OS level to inspect private data structures. This architecture is no longer being used in the mainstream—at least not for DAM and auditing. The main problem with this architecture is that if you don't poll the SGA fast enough you can easily miss many of the statements run against the database and if you poll too often you will impact the performance of the database. Another problem with this architecture is that the DAM system needs a high-privilege connection into each of the database systems it monitors. This architecture has all but gone away and most vendors that started with this architecture have re-architected their product to use an interception-based architecture.
3. **Log-based architectures:** Some DAM systems analyze and extract the information from the transaction logs (e.g., the redo logs). These systems use the fact that much of the data is stored within the redo logs and they scrape these logs. Unfortunately, not all the information that is required is in the redo logs. For example, SELECT statements are not and so these systems will augment the data that they gather from the redo logs with data that they collect from the native audit trails as shown in Figure 14.3. These systems are in many respects a hybrid between a true DAM system (that is fully independent from the DBMS) and a SIEM which relies on data generated by the database. These architectures usually imply more overhead on the database server.

There are multiple architectures for DAM systems but there is only one architecture for DAMP systems. All DAMP systems have an architecture resembling the interception-based DAM architecture. The reason is that in both the query-based and the log-based architectures the DAM system knows that something has occurred only after it has occurred—it needs the database to process the request for it to be available. Therefore, it can never prevent anything from happening.



**Figure 14.3** Log-based DAM architecture.

The interception-based architecture on the other hand has access to the request before it gets to the database and it can evaluate whether to allow it through or not. DAMP systems therefore have an interception-based architecture as shown in Figure 14.4. As in the DAM architectures, interception and prevention can occur on the network (by deploying the DAMP system between the database server and the network switch as shown in Figure 14.4 by system 1) or using a probe



**Figure 14.4** DAMP architecture.

that can decide whether or not to let the request reach the database using a policy defined by the DAMP system (as shown in Figure 14.4 by system 2). The latter architecture has two important advantages—it is far easier to deploy because it does not require network changes and it allows prevention for local connections as well as remote connections.

### Functions Provided by DAM Systems

The term DAM is broad because it covers any type of access to the database and covers both logging/auditing as well as monitoring and real-time alerting. However, a few use cases that are very common and usually prompt people to adopt DAM technologies are

- Privilege user monitoring: DBAs, ISAs, developers, and other privileged users need to be monitored. This has become a best practice and is mandated by most regulations. Monitoring privileged user activity includes auditing their activities, identifying anomalous privileged activity and reconciling privileged activities with change requests. A very common example of this use case is a SOX implementation.
- Application activity monitoring: Application activity differs from users who connect directly to the database. Application activity tends to be very intensive but also highly repetitive. There are two forms of application activity monitoring. One usage of DAM monitors all application activity and generates a normative baseline. Once approved, this baseline can be used to identify anomalous application activity. The main goal in such a scenario is to detect and prevent an attack on the data from the application layer (possibly using a vulnerability in the application). A second usage type occurs when application users perform administrative activities or access sensitive information through the application and the application uses connection pooling to connect to the database. Audit trails are needed showing which transactions and queries were performed by which application user. The difficulty is that the credentials are at the application level whereas the activity is at a database level; this is what this form of application activity monitoring resolves. DAM can help you to identify the real user—more on that in Section 14.7.
- Access to sensitive data: Many regulations mandate the tracking of access to sensitive data. Sensitive data differs for different companies—sometimes it is personally identifiable information (PII), sometimes it is patient information, sometimes it is financial data, and other times it is intellectual property. The use cases around sensitive data sometimes require only the changes to be monitored (i.e., DML). In other cases you may be required to also monitor SELECT statements. When SELECT statements need to be monitored, the result sets (or subsets thereof) may also need to be monitored. In both cases the need to monitor this activity may be limited to some connections or may encompass all connections irrespective of the originator. Two common initiatives that fall in this use case are privacy initiatives and payment card industry (PCI) (see more in Appendix A).
- Access to encryption keys: As you saw in Chapter 8, you can use DBMS\_CRYPTO to encrypt data or you can use TDE. TDE only encrypts data at the storage level and is not used as an access control mechanism. If you decide to use DBMS\_CRYPTO then you are responsible for managing the encryption keys yourself. Normally, these keys are kept inside a database table so that application code and stored procedures can use them. The main issue then becomes whether or not you have enough controls in place to ensure that an unauthorized user cannot use the keys. One of the DAM use cases is to monitor and alert on unauthorized access to the encryption keys. Although this is really a subset of the sensitive data monitoring use case, it is common enough and specific enough to warrant special mention.

- Anomaly detection and intrusion detection: Because DAM systems can monitor all activity in the database without requiring auditing to be turned on, they can monitor for anomalous behavior of authenticated users and application and help identify an attack that is launched on the database. Because DAM systems inherently understand the SQL commands and the result sets, they act as an intrusion detection system (IDS) for databases.
- Support for notification laws: Most countries have some form of notification laws. Notification laws help combat identity theft. They specify that if you have PII of a person and have had a data breach in which that PII was compromised, you must notify that person that the breach occurred and usually you also have to compensate that individual (e.g., with free credit reports). The reason that these notification laws have emerged is that companies, left to their own devices, prefer to hide the data breach rather than make it public and notify the people whose information was stolen. Letting the public know about a data breach can damage a company's brand, cause the company to lose customers, and more. However, these data breaches often lead to identity theft and to damage that can be avoided if the individuals knew their data was compromised and that they are at risk. Simple monitoring of bank accounts, credit card accounts, credit reports, etc., is a very effective measure. Hence the notification laws—they ensure that companies do the responsible thing and tell the people whose data has been compromised of the breach. What DAM systems can help with (besides help identify the breach), is limit the number of people to whom a company needs to send this letter (and limit the number of people that need to receive remediation provisions). By monitoring access to PII and result sets, a DAM system can record precisely which records were accessed by an offending connection. If this offending connection extracted 2000 records from a table that has 2 million records, and if you monitor this access, you may be able to notify only 2000 people—much simpler and much cheaper.

Given these use cases, the main capabilities that DAM systems must have:

- Ability to monitor activity which is both local and remote, and cover all types of connections such as TCP, BEQ, IPC, etc.
- Ability to monitor activity no matter how SQL\*Net is configured. Ability to monitor the activity even when connections are encrypted.
- Ability to set policies that determine what to audit, what to monitor and at which granularity to audit. Policy rules must be sensitive to users, IPs, source programs, commands, objects, etc.
- Ability to extract and report on all attributes such as the user name, the program, the OS user, the client host, the client OS, the SQL statement run, etc. There are typically over a 100 attributes in a modern DAM system that can be used in a report.
- Ability to manage an independent audit trail that cannot be modified. Ability to prove this.
- Ability to support full identification and accountability. For example, when the connection is made using the Oracle instance account using “/ as sysdba”, show who is logged into the instance account (more on this in Section 14.7).
- Ability to monitor and record information about the requests and the responses. Ability to not only show the SQL statements but also data about the result sets and error conditions returned by the database.
- Ability to send real-time alerts.
- Ability to manage large quantities of data efficiently without huge storage costs. Ability to archive data securely and efficiently and restore data quickly when needed.
- Ability to support and prove separation of duties without incurring additional staffing costs.

## 14.1 HOWTO Protect against SQL Injection

SQL injection is a technique for exploiting bad coding practices in applications that use relational databases. The attacker uses the application to send a SQL statement that is composed from an application statement concatenated with an additional statement that the attacker introduces. Many application developers compose SQL statements by concatenating strings and do not use prepared statement; in this case the application is susceptible to a SQL injection attack. The technique transforms an application SQL statement from an innocent SQL call to a malicious call that can cause unauthorized access, deletion of data, or theft of information.

SQL injection has received a lot of press and is usually considered to be related to Web applications. This is not true. SQL injection can be present in any application architecture. The focus on Web applications is however justified because Web applications cater to a broad range of users—internal as well as external—so the chance of an attacker trying to exploit the application is much higher.

Let's start with the classic example of application authentication bypass using SQL injection. Suppose that you have a Web form that has two fields that need to be entered by a user when they want to login to the system as shown in Figure 14.5. The application receives a user id and a password and needs to authenticate the user by checking the existence of the user in the USER table and matching the password with the data in the PWD column in that table. If the query produces a result set then the user is logged on. If not, an error message is shown. Assume (and this is the really important assumption) that the application is not doing any validation of what the user types into these two fields and that the SQL statement is created by doing string concatenation. Let's look at what happens if you maliciously type in the following user ID and password:

User ID: '**OR “=’**

Password: '**OR “=’**

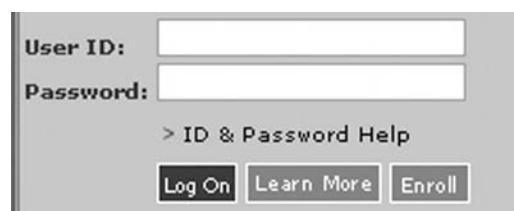
In this case the SQL string that would be used to create the result set would be

```
select USERID from USER where USERID = " OR "=" and PWD = " OR "="
```

This will surely return a result set and the attacker will be logged onto the application (usually as the first user in the table).

Another very popular SQL injection technique involves the use of UNION ALL SELECT to grab data from any table in the system. The syntax for this SELECT option is:

```
SELECT ...
UNION [ALL | DISTINCT]
SELECT ...
[UNION [ALL | DISTINCT]
 SELECT ...]
```



**Figure 14.5 Sample application login form.**

UNION is used to combine the result from many SELECT statements into one result set. If you don't use the keyword ALL for the UNION, all returned rows will be unique, as if you had done a DISTINCT for the total result set. If you specify ALL, you will get all rows from all the used SELECT statements. Therefore, most SQL injection attacks make use of UNION ALL.

Attackers use UNIONs to "piggy back" additional queries onto existing ones. Lists that are displayed following a conditional search issue a select and display the contents of a result set on the page. For example, suppose that you can look up all flights to a certain city by entering the city name to get a list of flights. Each line in the list shows you the airline, flight number, and departure time. Assume that the application is vulnerable to SQL injection—i.e., it uses string concatenation and does not do any validation on what you type into the city input field which is used in the WHERE clause. The normal SELECT issued by such an application may be

```
select airline, flightNum, departure from flights where city='ORD'
```

Suppose that instead of entering ORD (for Chicago) into the search input field you inject the following string:

```
ORD' union all select userid, 'dummy1',sysdate from USER where '1'='1
```

In this case the resulting select statement will be

```
select airline, flightNum, departure from flights where city='ORD' union all
select userid, 'dummy1',sysdate from USER where '1'='1'
```

The result set you will get will include all user names in the application as shown in Figure 14.6. The attacker is using SQL injection to get information that they should not have access to and that may be used to further launch an attack.

Airline	Flight#	Departure Date/Time
Delta	2362	8/20/2004 17:00
AA	62	8/20/2007 19:00
JetBlue	51	8/20/2007 16:30
Continental	144	8/20/2007 19:40
Delta	414	8/20/2007 22:00
United	2314	8/20/2007 20:00
RONB	dummy1	20-AUG-2007
JANE	dummy1	20-AUG-2007
JOHN	dummy1	20-AUG-2007
ALEX	dummy1	20-AUG-2007
URSULA	dummy1	20-AUG-2007
MITCH	dummy1	20-AUG-2007
SYSADM	dummy1	20-AUG-2007
SYSTEM	dummy1	20-AUG-2007

---

Figure 14.6 User names exposed in a UNION-based SQL injection attack.

RECENT POSTS			
Subject	Author	Date/Time (ET)	
<a href="#">Dr. Who is the Liberal Candidate</a>	taie_naf_n	11:29am, Aug 30	
<a href="#">Stronger dollar earnings of companies</a>	trending01	11:29am, Aug 30	
<a href="#">Stronger dollar earnings of companies</a>	trending01	11:29am, Aug 30	
<a href="#">"SECRET COURT POSES CHALLENGES"</a>	samlebowC	11:29am, Aug 30	
<a href="#">Dr. Apple using chip from IBM</a>	AUREOCUND	11:23am, Aug 30	
<a href="#">AFLAC VS IBM BROOKED @ MTV</a>	monoronic...	11:21am, Aug 30	
<a href="#">MONKEY DUCK AND COVERIN</a>	monoronic...	11:19am, Aug 30	
<a href="#">Dr. Apple using chip from IBM</a>	AUREOCUND	11:16am, Aug 30	
<a href="#">Dr. Who is the Fairy Candidate</a>	want_fries	11:14am, Aug 30	
<a href="#">Dr. Apple using chip from IBM</a>	AUREOCUND	11:13am, Aug 30	
<a href="#">Who is the Liberal Candidate</a>	terrym07	11:04am, Aug 30	
<a href="#">Japan is showing IBM Japan big love for</a>	deflation...	11:01am, Aug 30	
<a href="#">Dr. IBM &amp; JPMC contract</a>	northseach	10:57am, Aug 30	
<a href="#">Google money</a>	m273_15c	10:53am, Aug 30	
<a href="#">LET ME TELL YOU SOMETHING</a>	n244_23h	10:51am, Aug 30	
<a href="#">IBM to lose \$1 billion JPMC contract</a>	newswall099	10:50am, Aug 30	
<a href="#">Dr. Another IBM/Martha Bay/ OIL --</a>	want_fries	10:46am, Aug 30	
<a href="#">Dr. Another IBM/Martha Bay/ OIL --</a>	defibrillate	10:21am, Aug 30	
<a href="#">Dr. mamma bush</a>	taie_naf_n	10:18am, Aug 30	
<a href="#">Dr. See whom name</a>	m273_15c	10:18am, Aug 30	

Figure 14.7 Posting a message to a message board.

Finally, let's quickly look at another SQL injection pattern—one involving insert selects. This method makes use of the fact that SELECT subqueries can be used within an INSERT request. As an example, suppose that you have a screen that allows you to add a message to a message board as shown in Figure 14.7. The application functionality may be as simple as inserting this message to a MESSAGE table and allowing all members to review messages posted to the board as shown in Figure 14.8 (blurred to protect the innocent).

Type message subject  
  
 Type message  
  
 Preview   Post Message   Cancel

Figure 14.8 Viewing messages on the message board.

Building a message board can use a table in the database called MESSAGES. The application can do a SELECT on this table, and the posting function can do an INSERT into this table. For simplicity, assume that the columns in the MESSAGES table are called SUBJECT, AUTHOR, TEXT, and TIMESTAMP and that the timestamp is auto generated. In this case the application code for posting a message may simply do

```
INSERT into MESSAGES (SUBJECT, AUTHOR, TEXT) values (<whatever you type
in the subject field>, <your login name in the application>, <whatever
you type in the message text area>)
```

This simple function is vulnerable to an injection attack using an insert select command. If you type in the following into the appropriate fields (with the proper escape characters omitted here for the sake of clarity):

Subject field: start', 'start', 'start'); insert into messages (subject, author, text) select  
owner, object\_name, object\_type from all\_objects; insert into messages values ('end

Author field: end

Text field: end

The following SQL statements will be sent to Oracle:

```
INSERT into MESSAGES (SUBJECT, AUTHOR, TEXT) values ('start', 'start', 'start')
insert into messages (subject, author, text) select owner, object_name,
object_type from all_objects
insert into messages values ('end', 'end', 'end')
```

In this case you will be able to see all the table object names listed on the message board.

## Combating SQL Injection

There are a number of things you can do to combat SQL injection, including limiting application vulnerabilities, discovering SQL injection vulnerabilities and requiring that they be fixed, and protect your database by using DAM. As you've seen, SQL injection is not really a vulnerability of the database. It is a vulnerability in the application code that exposes the database and the data.

The first implementation option is to remove the application vulnerabilities. This is normally the responsibility of the application owner but sometimes it is appropriate for you as the database owner to be involved. By now there are some very good SQL injection guidelines for application developers; guidelines such as:

- All data entered by users needs to be sanitized of any characters or strings that should not be part of the input expression. All input fields must be validated.
- SQL used to access the database from application code should never be formed using string concatenation.
- Strongly typed parameters (usually in combination with stored procedures) should be used wherever possible.
- Prepared statements, parameter collections, and parameterized stored procedures should be used wherever possible.
- Application login should be implemented as a stored procedure.

These guidelines are for developers. If you have some leverage—use it. Make developers adhere to these guidelines. If you are fortunate you can even require a code review in which you participate. If you do, stress the use of prepared statements. When you use prepared statements as opposed to string concatenation the SQL strings are distinct from the values that you get from the user and thus there is no mixing of SQL and parameters. This is therefore one of the simplest ways to combat SQL injection. Beyond better security, they also can imply better performance if used correctly.

Monitoring and tracking whether or not prepared statements are used is one of the simplest things uses of a DAM system—you can see the difference in the SQL that travels on the network when using prepared statements and you can easily look at all the SQL traffic generated by an application to make sure that only prepared statements are used. With prepared statements, the SQL looks like:

```
update test set a = :1
```

The value would be communicated in an adjoining packet. Without prepared statements, the SQL looks like:

```
update test set a = 'ABC'
```

By monitoring this access and producing a report per application you can work towards more widely used prepared statements and a more secure environment.

In addition to code and design reviews, you can also make use of SQL injection detection tools. Tools can help you simulate a SQL injection attack to test your applications. These tools should be used by the developers themselves but in case you are the last bastion of hope for the data, you might want to explore the use of these tools yourself. Note that although these tools are effective, they are not comprehensive and are not always easy to use. The good news is that these tools are usually free of charge. As an example, SQL injector is a tool offered as part of the SPI Toolkit by SPI Dynamics (now HP) ([http://www.spidynamics.com/products/Comp\\_Audit/toolkit/SQLinjector.html](http://www.spidynamics.com/products/Comp_Audit/toolkit/SQLinjector.html)). This tool conducts automatic SQL injection attacks against applications making use of Oracle and test if they are vulnerable to SQL injection. The tool only supports two of the common SQL injection attacks—but even this limited test can be useful.

If you already have code deployed, or don't have the authority or energy to conduct code reviews you can use a DAM systems to identify and prevent SQL injection by monitoring the application activity and generating a baseline. A baseline of application access is a set of SQL structures that the application uses in normal operation. When a SQL injection attack occurs, these structures change. For example, in the login example a new OR condition suddenly appears. In the message board example, new INSERT statements appear, and in the flights example a UNION command suddenly appears. Because an application is generating these SQL statements and not a user, there should not be any changes to the SQL structures that the application is sending to the database. A DAM system generates a baseline of "normal behavior" and is able to identify an attack by seeing that there is a divergence from normal SQL structures and normal sequences. This is the only method that is effective on all types of SQL injection attacks and that does not introduce many false positives.

Before moving on to the next subject, be aware that there are IDSs that have SQL injection signatures. Unfortunately, signatures are not an effective tool in an Oracle environment simply because SQL and PL/SQL are very rich and any attack can be carried out in any number of ways. Looking for an attack based on a signature is not effective because of the many permutations that can occur. Signatures try to identify certain patterns as an indicator of an attack. The signatures that IDS use are usually the commonly used techniques of SQL injection. For example, you can

look for signatures such as 1=1 or UNION SELECT. The problem with this approach is that there are too many ways to carry out such an attack. For example, think how many different predicates you can think up that compute to an always true value. It may be ‘1’=‘1’, or ‘a’=‘a’ or ‘my dog’=‘my dog’ or ‘ron was here’=‘ron was here’ or ‘ron’ LIKE ‘ro%’ or 1<2 or ...—an infinite number of ways. The second problem is that some of these signatures may actually be used in real systems—it is not unheard of for people to use UNION ALL—this is why SQL supports the function. So your IDS may alert you on completely legal SQL and DAM is your best choice for SQL injection.

### The 7 Things to Remember about SQL Injection

1. SQL injection has many variants and can happen in any application architecture. It is a vulnerability at the application level that occurs when user input is not validated and is used blindly to construct SQL statements that are sent to the database.
2. Use prepared statements or check user input at the application level and you will not have SQL injection vulnerabilities.
3. Use DAM systems with baselining features by identifying when a SQL statement has been modified from its normal structure.

## 14.2 HOWTO Categorize and Identify Misuse and Intrusions

Because the database performs many activities on behalf of many different users, it is not trivial to identify a n i ntrusion o r m isuse o f t he d ata. F or e xample, t he u se o f D BMS\_CRYPT O o r UTL\_HTTP may be completely legitimate for some users and some applications and may be a sign that something malicious is going on in other cases. Identifying intrusions needs to be based on context as well as content.

Because DAM monitors all database activity and has full visibility into what is being asked of the database, it is necessary for identifying misuse and intrusions. It is necessary because without it you only see what you have setup in your audit trails. DAM is a necessary technology but not all DAM systems can identify intrusions. It is not enough that the DAM system can see everything—it also needs to have analysis capabilities. Looking for intrusion or misuse is like looking for a needle in a haystack.

DAM systems that can differentiate between normal behavior and between intrusions look at content, context, and historical information. Content includes the use of signatures (or the signature equivalent in the database world), the use of profiles and the use of data. For example, the use of UTL\_FILE or UTL\_SMTP will trigger more scrutiny because of possible exposure. Context includes analysis of environmental parameters as well as the relationships between different SQLs. For example, a session in which 50,000 identity numbers are selected in sequence needs to be handled differently versus a session that does the normal sequence of application behavior which includes two selects, a n update, a nother select, a nd a n insert. I ntrusion a nd anomaly detection for Oracle requires at least the following categories of analysis:

- Baselines and behavioral divergence: There are two forms of database activity—there are DBAs, developers, and power users that connect to the database and issue SQL statements and there are users that navigate through application modules and in the process cause SQL statements to be

sent to the database. For the first class of users it is hard to identify patterns because such users can potentially perform any SQL statement. However, these tend to have low volume and it is easier to monitor these connections closely. It is much harder to monitor application activity because of the volume. However, because the applications have a fixed set of SQL statements that they may call (it is fixed code after all), it is not too difficult to create a baseline of all observed application activity. This is called an application baseline and it encodes the state machine of the application as it accesses the database. Once such a baseline has been created it can be used for detecting intrusions. For example, if someone tries to launch a SQL injection attack then the SQL statement will have a different structure (e.g., different number of conditions) and the intrusion can be identified.

- Sequence monitoring: Baselines are usually constructed as points in a multidimensional space comprising of all the defining attributes of a SQL access. These dimensions include the objects, the statement, the structure of the SQL, the user, the application, the time, etc. However, you can get an even more granular definition of application activity if you account for sequences of statements. This allows you to identify intrusions not only based on a single activity but on a pattern of activity, frequency of activity, and the order between different activities.
- Errors and exceptions: In addition to monitoring SQL statements it is important to monitor ALL errors generated by Oracle and what SQL statements (or other activity) caused such an error to occur. An attack on a database does not take one second—it takes time and usually requires trial and error (especially if it is an attack launched through an application as opposed to an attack performed by a trusted insider). As an example, the attack may involve a UNION-based SQL injection attempt in which the types of columns may be wrong. It may involve trying to get data and causing privilege errors or errors when a column or table does not exist. All these errors are of utmost importance where intrusion detection is concerned and your DAM system must be able to monitor these events.
- Data extrusion: Another form of detection involves inspecting what data is returned on which session, how much data is returned, etc.
- Signatures of statements and packages: There are many Oracle procedures and packages that are either vulnerable (if you don't have the latest critical patch updates (CPUs) installed) or that are useful when performing an attack. You can monitor usage of these procedures to identify an attack—especially if you can profile under what conditions they are used normally.
- White lists and black lists: Almost all detection of intrusions involves some form of white list and black lists. You can have lists of users, lists of statements, list of objects, list of applications, lists of IPs, etc. White lists enumerate the elements from which you expect certain behavior and through which you approve certain behavior. For example, you can create a white list of users and a white lists of IPs for the use of UTL\_SMTP. If you see any use of UTL\_SMTP from a user or an IP not within the relevant white list you can classify this access as an intrusion. A black list is an enumeration of elements that you do not allow. For example, you can list a set of errors that you do not allow for any session—any session that will generate such an alert will be immediately flagged as an intrusion or as misuse.

Finally, note that to identify misuse you need to be able to analyze across multiple sessions. It is not enough to just monitor each session separately—sometimes understanding what is happening requires you to look at multiple connections and even multiple databases. As an example, a single user credential that is concurrently being used from different IPs is at least a misuse of credentials and sometimes an intrusion.

### Two Things to Remember about Identifying Database Intrusions

1. Identifying user intrusions requires a combination of technologies that inspect the SQL requests, error states, and the responses for the database.
2. Detecting intrusions by users accessing the database directly uses different techniques than detecting intrusions that take advantage of applications. In the first you can use combinations of signatures and error messages while in the second using a baseline along with error detection is more effective.

## 14.3 HOWTO Understand the Compliance Landscape

Much of the investment in database security is driven by compliance. When using the term compliance the first question is “compliance with what?” Usually the answer is that your company needs to comply with a certain regulation or with an internal policy. There are thousands of regulations today that affect database security; yes—thousands. This certainly does not mean that all of them affect you—most regulations are geographical, some are related to only certain industries, and some to the size and type of the company. As an example, Table 14.1 provides a very partial list from a few national regulations relevant in certain countries (partial both in terms of countries and the regulations within each country). As you can see, different countries have different mandates. Also, Table 14.1 only lists national regulations. There are also separate regulations which are enforced through local government, for example, in the United States there are 38 separate state-level regulations that affect privacy of PII. There are regulations that are enforced by industry leaders. For example, the Payment Card Industry (PCI) Data Security Standard (DSS) is not a national regulation—it is an industry regulation which has been created by the credit card companies. There are even regulations that have been adopted from one country to another. For example, J-SOX is a Japanese regulation which has been created based on the American Sarbanes–Oxley regulation. Euro-SOX is similar within the European Union. Finally, there are internal controls and policies that are usually defined by internal audit or the internal risk management group that define requirements that you need to abide by in terms of database security, database auditing, and database risk management.

Table 14.1 can be frightening. No one has time to even understand all these requirements, let alone implement for them. The good news is that there aren’t many variants in terms of the requirements posed by regulations on what you need to do in a database environment as inferred by the columns of Table 14.1. As the table shows, different regulations imply different categories within database security/monitoring/auditing that you should invest in, but there aren’t hundreds of different permutations.

At a high level there are two main classes of regulations. There are regulations which are focused on governance and regulations that focus on sensitive data. Regulations that focus on governance are primarily focused on the controls you have in place and on risk management. These regulations usually put a very strong emphasis on DBAs and privileged users and almost always the first step in such implementations is to produce comprehensive audit trails for DBAs and define controls around changes that may affect the applications. The 900 lb gorilla of this category is SOX.

The second set of regulations focus on access to sensitive data. Such regulations also have a set of requirements that deal with DBAs (e.g., making sure that DBAs cannot access PII) but overall the focus of these regulations is largely on sensitive data—i.e., a subset of database objects. Examples include PII in various privacy regulations, PCI DSS where the focus is on credit holder information, credit card information, etc.

**Table 14.1** Partial Mapping of National Regulations to Database Security and Monitoring Categories

Antiterrorism Act (C-36)	✓	✓	✓	✓	✓	✓	✓	✓
Canadian Security Intelligence Service Act	✓	✓	✓	✓			✓	
CSA 52-313	✓		✓	✓	✓			
Federal Access to Information Act	✓		✓		✓			
Federal Privacy Act	✓		✓		✓			
PIPEDA CAN/CSA-Q830-96	✓		✓		✓			
Telecommunications Act	✓		✓		✓		✓	
<b>European Union (EU)</b>								
EU 8th Directive/ FDICIA-EU Corporate Governance	✓			✓				
EU Directive 95/46	✓		✓	✓	✓	✓	✓	
EU Directive 97/66	✓		✓	✓	✓	✓	✓	
EU Directive 2002/58	✓		✓	✓	✓	✓	✓	
EU Directive on the Transfer of PII	✓				✓	✓	✓	
MiFID	✓			✓				
<b>Germany</b>								
4th Financial Market Support Law	✓					✓		
AktG	✓					✓	✓	

(continued)

**Table 14.1(continued) Partial Mapping of National Regulations to Database Security and Monitoring Categories**

Regulatory Requirement	Access Control and Data Access Management/Control	Change Management	Data Integrity and Availability	Monitoring, Auditing, and Reporting	Risk Management and Process Management	Segregation of Duties	Controls Management and Oversight
Cromme Commission Report	✓	✓	✓	✓	✓	✓	
Datenschutz Law	✓	✓	✓	✓	✓	✓	
Federal Data Protection Act	✓	✓	✓	✓	✓	✓	
Telecommunications Carriers Data Protection Ordinance	✓	✓	✓	✓	✓	✓	
TransPuG	✓	✓	✓	✓	✓	✓	
<b>Italy</b>							
Constitution	✓	✓	✓	✓	✓	✓	✓
Employee Data Protection Provisions	✓	✓	✓	✓	✓	✓	✓
Italian Data Protection Act	✓	✓	✓	✓	✓	✓	✓
Telecommunications Privacy Directive							
<b>Japan</b>							
Article 21 of the Constitution	✓	✓	✓	✓	✓	✓	✓
Act on Disclosure of Information by Public Agencies	✓	✓	✓	✓	✓	✓	✓

Act for the Protection of Computer Processed Personal Data held by Administrative Orgs.	✓	✓	✓	✓	✓	✓	✓	✓	✓
MIC	✓	✓	✓	✓	✓	✓	✓	✓	✓
Personal Information Protection Act	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>Russia</b>									
Communications Act	✓	✓	✓	✓	✓	✓	✓	✓	✓
Freedom of Information	✓	✓	✓	✓	✓	✓	✓	✓	✓
Information Protection Act	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>South Africa</b>									
Access to Information Act	✓	✓	✓	✓	✓	✓	✓	✓	✓
Electronic Communications and Transactions Bill	✓			✓					
Interception and Monitoring Act	✓	✓	✓	✓	✓	✓	✓	✓	✓
King II (Corporate Governance)	✓	✓	✓	✓	✓	✓	✓	✓	✓
<b>United Kingdom</b>									
Data Protection Act	✓			✓			✓	✓	✓
Employee Privacy Legislation	✓	✓					✓	✓	✓
Freedom of Information Act	✓	✓					✓	✓	✓
Health and Social Care Bill	✓							✓	✓
<b>United States</b>									
Bank Protection Act	✓	✓	✓	✓	✓	✓	✓	✓	✓

(continued)

**Table 14.1(continued) Partial Mapping of National Regulations to Database Security and Monitoring Categories**

<i>Regulatory Requirement</i>	<i>Access Control and Data Access Management/Control</i>	<i>Change Management</i>	<i>Data Integrity and Availability</i>	<i>Monitoring, Auditing, and Reporting</i>	<i>Risk Management and Process Management</i>	<i>Segregation of Duties</i>	<i>Controls Management and Oversight</i>
CFR 332	✓	✓	✓	✓	✓	✓	✓
CJCSI (multiple)	✓				✓	✓	✓
CJCSM (multiple)	✓	✓			✓	✓	✓
Computer Fraud and Abuse Act (CFAA)	✓	✓	✓	✓	✓	✓	
Computer Security Act	✓	✓	✓	✓	✓	✓	✓
Computer Security Enhancement Act (2001)	✓	✓	✓				
Critical Infrastructure Protection	✓						
DoD Directives (multiple)	✓	✓	✓	✓	✓	✓	✓
DoD Instruction (multiple)	✓	✓	✓	✓	✓	✓	✓
DoDD (multiple)	✓	✓	✓	✓	✓	✓	✓
Electronic Communications Privacy Act	✓	✓		✓	✓	✓	
Energy Policy Act	✓					✓	✓
Federal Reserve Act 12 (multiple)	✓	✓	✓	✓	✓	✓	✓
FIL-124-97	✓	✓	✓	✓	✓	✓	✓

FII-14-97	✓	✓	✓	✓	✓	✓	✓	✓
GISRA	✓	✓	✓	✓	✓	✓	✓	✓
Government Network Security Act (2003)	✓	✓	✓	✓	✓	✓	✓	✓
Gramm-Leach-Bliley Act of 1999	✓	✓	✓	✓	✓	✓	✓	✓
Health Insurance Portability and Accountability Act (HIPAA)	✓	✓	✓	✓	✓	✓	✓	✓
IGES (multiple)	✓	✓	✓	✓	✓	✓	✓	✓
NIST 800 (multiple)	✓	✓	✓	✓	✓	✓	✓	✓
NSTISSP	✓	✓	✓	✓	✓	✓	✓	✓
OMB (multiple)	✓	✓	✓	✓	✓	✓	✓	✓
Patriot Act of 2001	✓	✓	✓	✓	✓	✓	✓	✓
PCI DSS	✓	✓	✓	✓	✓	✓	✓	✓
PPA	✓	✓	✓	✓	✓	✓	✓	✓
Privacy Act	✓	✓	✓	✓	✓	✓	✓	✓
Sarbanes Oxley Public Company Accounting Reform and Investor Protection Act of 2002	✓	✓	✓	✓	✓	✓	✓	✓
TCPA	✓	✓	✓	✓	✓	✓	✓	✓
Telecommunications Act of 1996	✓	✓	✓	✓	✓	✓	✓	✓
USC 1030	✓	✓	✓	✓	✓	✓	✓	✓
USC 5311	✓	✓	✓	✓	✓	✓	✓	✓
Various State Notification Laws	✓	✓	✓	✓	✓	✓	✓	✓

The good news is that if you go through a fairly comprehensive implementation of database security you will likely cover multiple regulations that you may have to comply with. Certainly if you implement one of the regulations you will be covering all regulations within that category, but if you also adhere to the best practices rather than looking at the regulation then you will likely cover many more requirements. In fact, you should consider implementing security best practices rather than addressing a compliance checklist. If you implement elements of what this book covers (e.g., assessments, monitoring, auditing, change tracking, encryption—even if you implement each partially), and if you put a process in place that enforces best practices, then you will almost certainly have a secure environment and you will be in compliance with the relevant regulations. If you invest in complying with a very specific checklist you might find that you have not achieved better security, that you have not brought yourself closer to compliance with other requirements, and you may even be at risk with your main compliance driver because the technical implementation is an interpretation of the regulation. The bottom line is that good security implies compliance (and compliance with multiple regulations) whereas compliance with a certain requirement does not necessarily lead to better security and does not imply compliance with multiple regulations.

#### **Two Things to Remember about the Compliance Landscape and Mapping Compliance Requirements to a Technical Implementation**

1. Secure your database environment well following the guidelines and tools presented in this book—this will guarantee that you are compliant with any regulation your database environment needs to comply with.
2. There is a high degree of commonality between many regulations—don't let the sheer number of regulations overwhelm you. Identify which regulations focus on governance and risk management and which regulations focus on data access and data privacy and implement generic controls and policies that will cover all relevant regulations.

## **14.4 HOWTO Determine Whether You Need DAM or DAMP**

Technically, it seems very simple to know when you need DAMP. You need DAMP when you need to prevent activities from happening versus the use of monitoring and real-time alerting and using these as prevention-through-deterrence. At a business-level, it is harder to know when DAM (and the right process) is enough versus when DAMP is truly required. You should make the distinction because the cost associated with DAMP is usually higher than the cost associated with DAM and because DAM is a truly nonintrusive technology whereas DAMP is not. Here are some important use cases for DAMP deployments:

1. Privileged user access to sensitive data such as PII: DBAs and other privileged users with system privileges can access any data in any schema. There is no need to explicitly grant privileges to objects containing PII data and there is no way (in Oracle or any other database) to limit such access. Preventing such access (mostly preventing SELECTs) is required by multiple regulations and in many geographies. This is usually solved through monitoring and real-time alerting but many are moving to prevention as a way to reduce cost (of review). Monitoring is also viewed as a compensating control whereas preven-

tion is the control that compliance requires. Note that although SELECT statements are the most common, there are many activities that may need to be prevented—e.g., creation of views, creating data pump tables, dropping tables, etc. This use case covers many scenarios that differ only in which rules are defined—based on user accounts, based on which IP or subnet the connection comes from, based on which application is making the request, based on the time of day, based on which schema and database objects are accessed, etc.

2. Outsourced DBAs and cross-boundary laws: Companies that outsource DBA positions are extremely sensitive to access by nonemployees. This use case is a subset of the previous one but gets more visibility and priority. This use case also gets special attention due to the possibility of changes to business-affecting data. In the previous use case the focus is often on access to sensitive data and is often related to PII. However, there is also a similar use case involving business-affecting data. In this use case the focus is usually not only on access to data but also on the ability that a DBA has to change data (i.e., DML versus SELECT). Although this scenario is as valid for internal DBAs as it is to outsourced work, addressing this issue when internal employees are involved is often based on monitoring and auditing whereas outsourced scenario often require better preventive controls.
3. Segmentation of applications and their related data: Because multiple applications often live in the same instance or on the same operating system and because users with system privileges can access data across these schema and database boundaries, there is a need to enforce policies in which privileged users of one application cannot connect to the schema of another application, even when running on the same instance or machine.
4. Separation of duties: There are often requirements for more than one person to collectively perform a highly privileged activity. This is hard or impossible to implement when users have all privileges. Prevention based on multiple factors is used to implement such schemes (metaphorically similar to two people turning a key to launch a missile). This use case includes handling of patch installations, upgrades, backups, replication changes, etc. This use case is also relevant when database links are involved and where data from one application is exposed through another, and the linked application cannot rely on the security rules built into the linking application.
5. Implementation of encryption schemes: When implementing data encryption using DBMS\_CRYPTO (versus TDE), the encryption keys are often stored in the database within tables. Prevention is used to ensure that DBAs do not access the encryption keys. This is another special case of use case #1.
6. External policy definition for legacy applications with limited access control: Applications (especially legacy application and database schemas that are part of a purchased application) often have very limited built-in schema access control. It is not possible to change these applications and yet access control is mandated by regulations. To maintain compliance and not have to replace applications, external prevention is used.
7. Data leak prevention: Vulnerabilities in application and lax access control definitions increase the risk of leakage of sensitive data. For example, a vulnerable script or application can be changed to extract ALL records from the database. Prevention can be based on extraction of data and not on access. Prevention in this case is coupled with reporting to support notification—e.g., rather than have all 2 million customer records extracted, terminate the connection as soon as the breach is identified (e.g., after access to 250 records), report on which 250 records were extracted and supply the records to allow notifying only these 250 customers.

8. Rogue application prevention: Application data in the database should only be accessed and changed through the application. However, access control is almost always based on user name. Compliance requirements often specify that changes to data (and sometimes also access to data) should only be made through the application where application-level security is enforced. Any connection from another application, script or tool should be prevented. An example is the use of Excel to connect to the database, extract data and even change data using VBA.
9. Enforcement of change request process: Most companies manage DBA tasks in a change request/ticketing system. When DBAs need to perform work they should have a valid ticket number that references at least the instance. Most companies rely on auditing and reconciliation of activities produced by the auditing system with the ticket description. A more advanced approach is to require ticket information to be entered as part of the session and prevent certain activities if the ticket number is invalid or is not related to the instance. This can be used for all DBA work or, more likely, for security-affecting work such as changes to authentication attributes, changes to encryption options, changes to backup options, etc.

It is interesting to note that DAMP, like DAM, is very much driven by compliance and risk management.

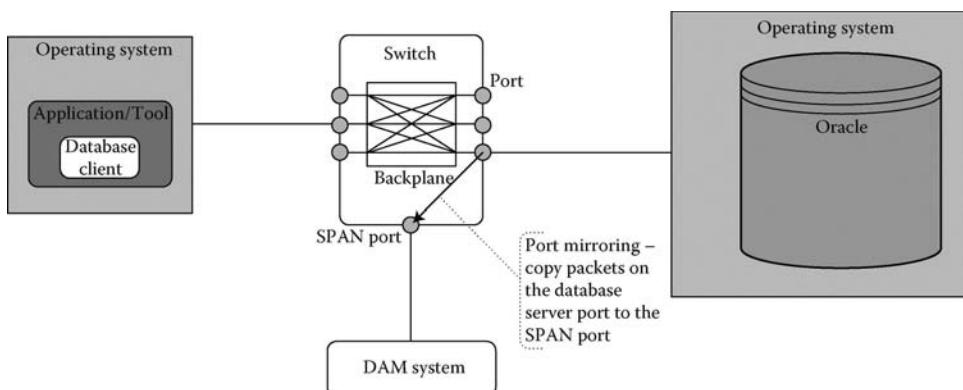
#### **The e Things to Remember about Using DAMP versus DAM**

1. Most DAMP implementations are still driven by compliance, but they have a very strong security orientation because they create an external access control overlay.
2. DAMP implementations are simple because they are rule based and can easily support any requirement for access control based on any number of factors—all without modifications to Oracle.
3. The main use cases for DAMP are controls around users with system privileges and breaches that occur and extract data from the database.

## **14.5 HOWTO Analyze Impact on Performance**

DAM systems (especially those with interception architectures) have less impact on performance than the alternative options. However, even within this DAM architecture there are different attributes that can affect impact on the database server.

Looking back at Figure 14.1, a DAM system can intercept database communications on the database server itself or by using network gear. The only impact that a DAM system can have is when the probe is running on the database server; if packets are inspected using a switch port mirror (e.g., a SPAN port) or using a network tap then the impact on performance is zero. SPAN ports and network taps create a copy of the packets as they are being placed on the port on their way to the database server, and placing this copy on the port mirror (see Figure 14.9). They do so while guaranteeing that there is absolutely no impact on performance—there is no added latency to the real packets on their way to the database, there is no impact on the throughput, etc. Therefore, when you inspect through network gear you don't even have to do any testing.



**Figure 14.9 Using a switch to get a mirror stream of database activity.**

Technically, doing packet sniffing is an optimal solution (at least as far as performance is concerned). However, there are a number of issues with packet sniffing including the fact that local traffic cannot be analyzed, the fact that encrypted traffic cannot be inspected, and the fact that for an environment with a large number of servers it may be impractical to intercept at the switch. Therefore, it is important to understand how these probes work so you can better analyze their impact on performance.

As shown in Figure 14.1, probes do two things—they create a copy of the database activity at real time by inspecting the interprocess communication that the database is involved with, and they send this data to the DAM server. The fact that the probe does so little is key to the ability of a DAM system to remain nonintrusive and not impact the database performance. However, the probe is running on the database server and is consuming resources. You should understand what attributes such probes can have and how this can affect performance:

1. The impact of a DAM probe on performance is directly related to the amount of database activity that it needs to monitor. The probe makes a copy of data packets and writes them—usually on a socket. Writing ten times more information generally means that the probe has ten times more work to do and thus consumes ten times more resources. Therefore, make sure that the DAM system that you’re evaluating allows you to filter out various database activities, for example, based on who the user is.
2. At least half of what the probe does is write the data to a socket. Apart from consuming resources, this also has an impact on network load because this data needs to be sent to the DAM server. A DAM probe that can filter what gets sent to the DAM server not only consumes less resources on the host, it also adds less overhead from a networking perspective. Some probes will even allow you to fine-tune whether or not you want long result sets sent to the DAM server. This gives you even more control on the impact on resources.
3. Encryption is a relatively expensive computational operation. DAM probes have the ability to encrypt the traffic (usually using Secure Sockets Layer, SSL) between the probe and the DAM system. However, think twice about whether or not you need this capability because

it does affect performance. It is not unrealistic to have a probe consume 2 percent CPU without SSL and seeing it go up to 4 to 5 percent when SSL is enabled. As with other operations performed by the probe, the impact on performance is proportional to the amount of data sent to the DAM server—if you need to encrypt ten times more data you will consume ten times more resources. As a rule of thumb, you only need to use encryption between the DAM probe and the DAM server if you are encrypting the data-in-transit between the database clients and the database server.

To summarize, DAM systems have the best audit-to-performance ratio and have the least impact on server performance. Using network inspection has the potential to reduce the impact to zero. Using host probes will have some impact but can be very small especially if your probe allows you to control just how much you write to the network.

#### The Three Things to Remember about Analyzing DAM's Affect on Performance

1. Impact on performance is directly correlated to the amount of data you are inspecting for all but network-based DAM inspection.
2. Use filters to define what you are collecting and what you are auditing—there is little sense in auditing what you do not need.
3. If you are using a host probe only, remember that encrypting DAM communication is relatively time consuming. Do this only if your database traffic is encrypted in the first place.

## 14.6 HOWTO Analyze Impact on Storage

DAM systems potentially collect a lot of information. For example, when a vendor tells you that it can audit at a rate of 2500 statements per second, you can do some simple math to see what the impact on storage can potentially be. For example, if you assume that an audit record takes 200 bytes then a simple calculation implies that the audit records from a single day will consume over 40 GB of disk space. If this data is archived it can be compressed, but will still take a lot of disk space. More importantly, you cannot always compress this data and move it to secondary storage immediately—you may need to keep it online for a certain period of time for compliance reporting purposes. As an example, if you need to keep it online for a period of 60 days, then just that one system may require over 2.5 TB of disk—this is a very expensive proposition. If the average audit record consumes only 20 bytes then this becomes a more manageable 250 GB of disk space. If the average audit record is 800 bytes then this number becomes an unmanageable 10 TB for a period of 60 days. The most important number is the average size of an audit record. Other attributes that will affect the amount of storage you will require for your implementation are:

- Appliance packaging or software-only deployments: Many DAM systems are packaged as a security appliance. Beyond the advantages that this packaging provides in terms of the system being hardened by design, these systems include their own disks so that they can collect and report on data without requiring external storage. For these systems you need to validate how

long data can be kept on the appliance's disks. At some point you will need to archive the data to make room for new data but if the DAM system is well designed this goes directly to secondary storage so that you do not have to analyze storage requirements. For software-only DAM solutions you need to size the disks on the server that will be running the DAM system.

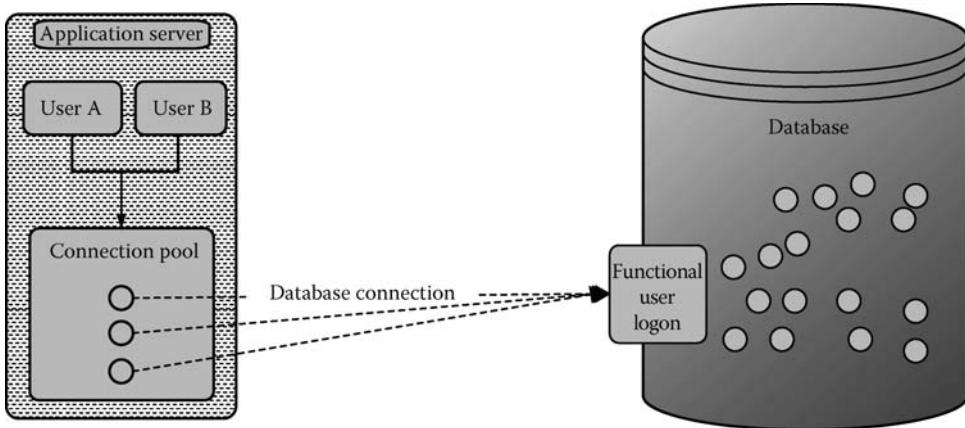
- **Monitoring versus auditing:** Many people do not distinguish between the requirements for DAM monitoring versus DAM auditing. If you audit at a rate of 2500 records per second there is no doubt that no one will review this audit trail. Therefore, auditing at such rates is only valuable for potential future investigations. Much more typical are requirements which include monitoring database activity at very high rates (tens of thousands per second) and auditing a small subset of these activities. From a storage perspective, the monitoring rate is not important; it is just the auditing rate that is important. Therefore, make sure that when you look at DAM systems you have a clear understanding of what your monitoring requirements are, what your auditing requirements are, and that the DAM system is able to define monitoring policy rules and auditing policy rules that may be distinct. As an example, you may need to monitor all database activity to detect misuse and intrusions but will only audit access to sensitive data and statements that are identified as an intrusion.
- **Normalization:** One of the biggest issues with some DAM systems is that they do not normalize the data before they store it. In Chapter 13 you saw that the Audit Vault schema has a FACT table that contains the statement and that all of the attributes that qualify this statement are stored in DIM tables. The data is normalized in that you do not save all of the dimension information multiple times, one per access. This is the convention with DAM systems—most of them store the data in a database in a normalized manner. However, some first-generation DAM systems store the data in flat files and repeat the data per statement. An audit record of that form may look like:

```
14 February 2008 10:32:36, 192.168.1.8,192.168.2.222,3920,1521,
"Risk Management DB-Production","ORACLE","aqua_data_studio",
"user2","user2","client.dbasecurity.com",
"select * from t1","select* from t1","user2", ...
```

Note that the dimension information is written per audit record. This means that if you audit one thousand invocations of "select \* from t1" you are not storing one thousand times 30 bytes (which is the length of the statement plus some pointers to DIM records) but one thousand times 500 bytes! There is no reason to duplicate the session information per audit record.

### The e Things to Remember about Analyzing DAM Storage Requirements

1. Make sure to use a DAM system that normalizes data to avoid expensive duplication of data.
2. Check the average audit record size and your policies and compute the estimated storage requirement for 30–60 days; this is normally the amount of data you need to keep online.
3. Distinguish between what you need to monitor versus what you need to audit to reduce unnecessary storage requirements.



**Figure 14.10 Database connection pooling scenario.**

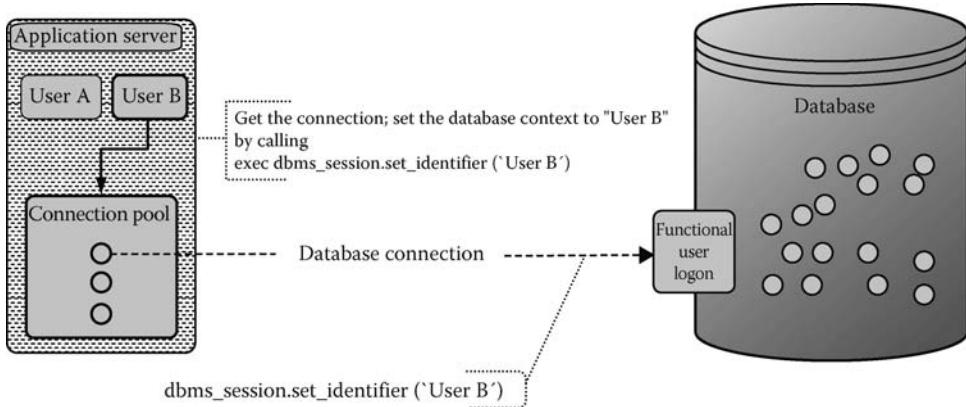
## 14.7 Discussion: Identifying the Real User

One of the hardest problems that DAM solves involves end-user credentials. Most auditors will require full accountability—i.e., every audit record must be associated with a single individual. The same applies to monitoring and security—every action at the database level needs to be mapped to a single user—a unique individual. There are two important scenarios that make it difficult to meet this requirement. The first scenario is that of application servers. As Figure 14.10 shows, application servers manage a pool of connections. These connections all logon to the database when the application server starts up using a single functional ID. Therefore, any audit trail or monitoring system will only be able to mark these activities as being performed by the functional account. Unfortunately, this is not full accountability and is not useful from either a security or an auditing perspective. Instead, you need to be able to tell which database activity was performed on behalf of user A and which database activity was performed on behalf of user B.

The second scenario involves the Oracle instance account. In many environments there are certain activities that you would do from the Oracle instance account. On Unix you might connect to the host using your own operating system account using ssh and from there su into the Oracle instance account and connect using “/ as sysdba”. The problem here is again the lack of full accountability. When you look at the audit trail you will see the database user as SYS and the operating system user as oracle (the instance account). What is really required is an indication that these activities are being done by you—i.e., the account from which the su to the Oracle account occurred.

In both these scenarios inspection-based DAM systems can help you with full accountability. Because these systems live outside the database, they can look at additional information that can be used to augment the information on the database connection. This is a very important advantage afforded by DAM that will make your auditors very happy.

Getting end-user credentials in the application server scenario is done either by instrumentation of the code or by instrumentation of the application server. You've already seen that if you use `dbms_session.set_context` or `dbms_session.set_identifier` as shown in Figure 14.11 then standard auditing will log this user, and DAM systems will also pick that up as the real user. In this case, when the application gets a connection from the connection pool it makes a call to `set_context`



**Figure 14.11** Using `dbms_session` to set the end-user credentials.

and the DAM system knows that all SQL statements on this connection (up until the next call to `set_context`) belong to user B.

DAM systems can do more. DAM systems can observe what is happening on the application server. They can see the requests when they come into the application server and match them with the requests that the application server makes to the database. This matching still requires instrumentation—at the application server. Don't fall for solutions that tell you that they can inspect the requests coming to the application server and the requests the application server makes to the database and correlate them. Time-based correlation and value-based correlation will only give you the right answer sometimes, but unless you can match these requests up with 100 percent accuracy all of the time, you cannot use the solution for monitoring or auditing.

Similarly, in the scenario involving the Oracle instance account, because the DAM probe lives at the operating system it knows that the connection to Oracle has been made from a shell that was initiated by another shell that is owned by the real user. DAM looks wider than just the database level and therefore can help you achieve full accountability.



## *Chapter 15*

---

# Privileges and Authorization

---

Oracle authorization is based on a privilege model through which you can allow or deny access to data, actions, or processing and through which you can apply various limitations to such actions or access. A privilege is a user right—a right to a type of Structured Query Language (SQL) statement, a right to an object, a right to run a package or procedure, etc.

Because Oracle manages many types of objects and commands, there are many types of privileges. Specifically, there are two important types of privileges that you can manage—object privileges and system privileges. Object privileges define which actions can be done by the user on a certain object and system privileges define which administrator tasks can be performed by the user.

Many users are managed within a database. Because there are so many types of privileges, so many potential objects with which privileges are defined, and so many users to which rights are assigned, managing privileges at a user level is impractical. This is where roles come in. Roles group privileges and are assigned to users. Roles support a hierarchical model—you can assign a privilege to a role or you can assign a role A to another role B after which role B has all the privileges assigned to role A. At the end of the day roles are how entitlements are managed within the database.

In this chapter you'll learn how to manage privileges within the database and authorize what users can do. You'll focus on the capabilities available to you for defining the entitlement model for your database. One of the most important principles in any security implementation is that of minimal privileges—i.e., if a user or system does not require a privilege, they should not have that privilege. This is often a lot of work and it's always much easier to have a lax entitlement implementation. Taken to extreme, it's very simple (and very insecure) to give all users all privileges. So in addition to learning about the granting capabilities you'll also learn about how to verify and audit the entitlements defined in your databases.

### **15.1 HOWTO Manage Object and Column Privileges**

An object privilege is the right to execute a SQL statement or to access an object. A schema object privilege is the right to perform a certain action on a specific schema object. Object privileges depend on the object type—for example, a procedure has an execute privilege but a table does not.

Object privileges are assigned to a user or a role (which is later assigned directly or indirectly to a user). Object privileges that can be assigned include

- ALTER—e.g., alter table.
- CONNECT—connect to a database (i.e., create a session).
- DELETE—for tables, views, and materialized views (when updateable)
- EXECUTE—for procedures, functions, packages, etc.
- DEBUG—allows running a debugging session of code and also applies to tables and views (debugging the triggers defined on tables and views).
- FLASHBACK—for tables, views, and materialized views—allows running a flashback session on that object.
- INDEX—allows one user to create an index on another user's table.
- INSERT—for tables, views, and materialized views.
- ON COMMIT REFRESH—allows creating a materialized view that is refreshed as soon as the source table is updated (e.g., a snapshot table in a data warehouse).
- QUERY REWRITE—allows the session to rewrite queries to select from materialized views.
- READ—applies to database objects of type directory.
- REFERENCES—allows creating foreign key constraints on tables without getting SELECT privileges.
- SELECT—for tables, views, materialized views, and sequences.
- UNDER—for subviews and subtypes.
- UPDATE—for tables, views, and materialized views.
- WRITE—for external tables.

You grant object privileges using the GRANT command and revoke privileges using the REVOKE command—e.g.,

```
SQL> grant select on scott.emp to ronb;
Grant succeeded.

SQL> grant update, insert, delete on scott.dept to ronb;
Grant succeeded.

SQL> revoke delete on scott.dept from ronb;
Revoke succeeded.

SQL> select owner,table_name,privilege from dba_tab_privs where grantee='RONB';
OWNER          TABLE_NAME          PRIVILEGE
-----          -----
SCOTT          DEPT              UPDATE
SCOTT          DEPT              INSERT
SCOTT          EMP               SELECT
```

You can also use the shorthand ALL [PRIVILEGES] to grant or revoke all available object privileges for an object. Note that when you revoke privileges (especially if you use REVOKE ALL) you may cause an integrity constraint to be deleted so you should use CASCADE CONSTRAINTS in such cases:

```
SQL> revoke all on scott.dept from ronb cascade constraints;
Revoke succeeded.
```

### 15.1.1 Grant Option

An object privilege can be granted with the GRANT option. This option allows the grantee to propagate the privilege on. For example, if you grant a user privileges to select from a table, then normally it does not mean that they can grant this privilege to another user:

```
SQL> connect scott
Enter password: *****
Connected.

SQL> grant select on emp to ronb;
Grant succeeded.
SQL> connect ronb
Enter password: *****
Connected.

SQL> select count(*) from scott.emp;
CO UNT(*)
-----
16
1 row selected.

SQL> grant select on scott.emp to jones;
grant select on scott.emp to jones
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

But if you do want ronb to be able to grant the privilege on, use the grant option:

```
SQL> connect scott
Enter password: *****
Connected.

SQL> grant select on emp to ronb with grant option;
Grant succeeded.

SQL> connect ronb
Enter password: *****
Connected.

SQL> grant select on scott.emp to jones;
Grant succeeded.
```

### DBA\_TAB\_PRIVS

To view object privileges, use the DBA\_TAB\_PRIVS view—a data dictionary view that holds

- GRANTEE—user or role.
- OWNER—owner of the object on which the privilege is granted.
- TABLE\_NAME—name of table or other object on which the privilege is granted; this is an object name (as opposed to a table name) and you will get entries that are not tables as well.
- GRANTOR—who granted the privilege.

- PRIVILEGE
- GRANTABLE—when using GRANT OPTION, allows the grantee to grant it further to other users.

## Column Privileges

Object privileges are known to almost all Oracle users, but not everyone is aware that privileges can be assigned at a column level as well. This is useful when you want to control access to a certain column as opposed to the entire object. For example, if you want to give Jane privileges to update only one column and not the other:

```
SQL> grant update(hiredate) on emp to jane;
Grant succeeded.

SQL> connect jane
Enter password: ****
Connected.

SQL> update scott.emp set sal=sal+1;
update scott.emp set sal=sal+1
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> update scott.emp set hiredate=hiredate+1;
14 rows updated.
```

Note: This only works for privileges such as INSERT and UPDATE; it does not work for SELECT. To view defined column privileges use the DBA\_COL\_PRIVS view:

```
select grantee, table_name, column_name, owner, grantor, privilege, grantable from
dba_col_privs
```

GRANTEE	TABLE_NAME	COLUMN_NAME	OWNER	GRANTOR	PRIVILEGE	GRANTABLE
JANE	EMP	HIREDATE	SCOTT	SCOTT	UPDATE	NO

1 record(s) selected

## View Privileges

Because a view is derived from data selected from other tables, to create a view you need to have the following privileges:

1. You need to have been granted the CREATE VIEW or the CREATE ANY VIEW system privileges. If you've been granted CREATE VIEW then you can create a view within your schema. If you've been granted the CREATE ANY VIEW privilege then you can create a view in any schema (more on system privileges and system ANY privileges in the next HOWTO).
2. You need to have been granted one of the SELECT, INSERT, UPDATE, or DELETE object privileges on each of the base tables or you need to have been granted one of the SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, or DELETE ANY TABLE system privileges.

The first privilege category is required for creating the object and the second for allowing it to work. The third type of privilege you need is the ability to grant other users access to your view. For that you either need to be granted privileges on the base object with the GRANT OPTION or system privileges with the ADMIN OPTION (more on in the next HOWTO). To understand why, remember that when you grant SELECT on your view to another user you are in effect granting that user SELECT privileges on the base table. Hence, you need to have been granted the SELECT privilege on the base table with GRANT OPTION (as an example). Note that this does not mean that a user that has been given a privilege to the view also needs privileges to the base objects—you will not be using your GRANT OPTION to grant SELECT to the base table. But you need the GRANT OPTION to be able to grant SELECT privileges to your view.

## Procedure Privileges

The only object privilege that a procedure has is EXECUTE. You grant the EXECUTE privilege to users that need to run the procedure or write other procedures that in turn call your procedure. When you use procedures within a package you can grant execute privileges to the entire package or to procedures within the package. For example, let's use the package definition in Listing 15.1. You can download this package definition from [http://www.oracle.com/technology/obe/obe1013jdev/10131/wsfromplsqlpackage/install/EMP\\_PACKAGE.sql](http://www.oracle.com/technology/obe/obe1013jdev/10131/wsfromplsqlpackage/install/EMP_PACKAGE.sql)—it is a very simple package that you can use within the HR schema. For this example we care only about the structure of the package—not what the function does. Once you have this package defined you can grant privileges. To grant privileges to the entire package:

```
SQL> grant execute on EMP_FETCHER to scott;
Grant succeeded.
```

If you want to grant privileges to just a specific procedure or function within a package you should create an encapsulating procedure or function and grant privileges at that level:

```
CREATE OR REPLACE PACKAGE EMP_FETCHER
AS
    FUNCTION get_emp (emp_no IN NUMBER)
        RETURN emp_rec;
END;
/
CREATE OR REPLACE PACKAGE BODY EMP_FETCHER AS
    FUNCTION get_emp (emp_no IN NUMBER)
        RETURN emp_rec
    IS
        emp_found      employees%rowtype;
        emp_rtn       emp_rec;
    BEGIN
        ...
    END;
END;
/
```

---

**Listing 15.1 Simple package body.**

```

create or replace function
  get_emp_wrapper (emp_no IN NUMBER)
  R ETURN emp_rec
IS
BEGIN
  R ETURN EMP_FETCHER.get_emp(emp_no);
END;
/
SQL> grant execute on get_emp_wrapper to scott;
Grant succeeded.

```

The far more complex issue that you need to understand about the procedure privilege model involves the rights to objects that the procedure uses. An Oracle procedure can be defined in two ways—with definer rights and with invoker rights. Definer rights mean that as the procedure is executed, Oracle checks privileges based on the definer of the procedure rather than the user who is executing the procedure. The user who is executing the procedure must have been granted EXECUTE privileges to the procedure but typically will not have been granted privileges to the objects that the procedure is accessing. This allows you to build procedures as an encapsulation of functionality that uses and modifies your schema objects, and allows other users to use this functionality without giving these users privileges to the underlying objects. Definer rights is the default Oracle behavior.

Let's look at an example. Connect as scott and create the following procedure:

```

create or replace procedure
  de      finer_give_raise_to_all(
  ra      ise_amount number)
IS
BEGIN
  update emp set sal=sal+raise_amount;
  co      mmit;
END;
/

```

This procedure is defined with definer rights. When the procedure executes and Oracle needs to determine whether or not the UPDATE of EMP should be allowed, it will use SCOTT's privileges to determine this. Grant EXECUTE privileges on the procedure to another user:

```

SQL> grant execute on definer_give_raise_to_all to ronb;
Grant succeeded.

```

Assume that the contents of the EMP table start out as:

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30

7698	BLAKE	MANAGER	7839	01-MAY-81	2850	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	10
7788	SCOTT	ANALYST	7566	19-APR-87	3000	20
7839	KING	PRESIDENT		17-NOV-81	5000	10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0
7876	ADAMS	CLERK	7788	23-MAY-87	1100	20
7900	JAMES	CLERK	7698	03-DEC-81	950	30
7902	FORD	ANALYST	7566	03-DEC-81	3000	20
7934	MILLER	CLERK	7782	23-JAN-82	1300	10

14 rows selected.

Notice that RONB does not have UPDATE rights to EMP:

```
SQL> update scott.emp set sal=sal+10;
update scott.emp set sal=sal+10
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

And yet, RONB does have EXECUTE privileges and the privilege verification for the operations performed within the procedure is done at runtime using the definer's privileges:

```
SQL> exec scott.definer_give_raise_to_all(10);
PL/SQL procedure successfully completed.
```

After which the table now has:

```
SQL> select * from emp;
-----
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	810		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1610	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1260	500	30
7566	JONES	MANAGER	7839	02-APR-81	2985		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1260	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2860		30
7782	CLARK	MANAGER	7839	09-JUN-81	2460		10
7788	SCOTT	ANALYST	7566	19-APR-87	3010		20
7839	KING	PRESIDENT		17-NOV-81	5010		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1510	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1110		20
7900	JAMES	CLERK	7698	03-DEC-81	960		30
7902	FORD	ANALYST	7566	03-DEC-81	3010		20
7934	MILLER	CLERK	7782	23-JAN-82	1310		10

14 rows selected.

The second mode is invoker rights. In this case, when the procedure runs, the privileges assigned to the user who is executing the procedure are used to determine whether the actions made within the procedure are allowed. For example, let's define a similar stored procedure—but this time using invoker rights. The difference is that this definition uses the AUTHID CURRENT\_USER clause:

```

create or replace procedure
inv      oker_give_raise_to_all(
ra      ise_amount number)
AUTHID CURRENT_USER
IS
BEGIN
    update emp set sal=sal+raise_amount;
co      mmit;
END;
/

```

Grant RONB EXECUTE privileges:

```

SQL> grant execute on invoker_give_raise_to_all to ronb;
Grant succeeded.

```

If RONB now tries to execute this procedure he would get an insufficient privilege error:

```

SQL> exec scott.invoker_give_raise_to_all(10);
BEGIN scott.invoker_give_raise_to_all(10); END;

*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SCOTT.INVOKER_GIVE_RAISE_TO_ALL", line 7
ORA-06512: at line 1

```

This is happening because RONB does not have UPDATE privileges on EMP. If you grant this privilege to RONB then they would be able to execute the procedure:

```

SQL> grant update on emp to ronb;
Grant succeeded.

SQL> connect ronb
Enter password:
Connected.

SQL> exec scott.invoker_give_raise_to_all(10);
PL/SQL procedure successfully completed.

```

Definer right is the default because it is very hard to manage privileges with invoker rights. You have to analyze what the procedure is using and granularly assign privileges to users who need to execute the procedure. Moreover, these are usually not privileges you want to grant—procedures often encapsulate functionality that you want to grant widely without granting direct access to the underlying objects. And yet, for those of you that come from a Unix background, definer rights should be familiar to you—it is just like SETUID. Setuid (short for set user ID upon execution) is a Unix access rights flags that allow users to run an executable with the permissions of the executable's owner. In Unix, setuid is considered to be one of the worst enemies of security best practices—especially when the executable belongs to a highly privileged account.

The same holds true with Oracle definer rights and you need to understand this issue, especially with procedures defined in highly privileged schemas. Let's look at a well-documented example in older versions of Oracle—the EXECSQL procedure in the LTADM package. This package is defined in the SYS schema:

```
SQL> desc sys.ltadm;
PROCEDURE ADDLOCKROWSINFOENTRY
...
PROCEDURE EXECSQL
Argument Name          Type          In/Out Default?
-----
SQLSTR                VARCHAR2      IN
...

```

This procedure accepts a string and executes it. Because it is defined with definer rights it runs with SYS privileges. So if you are granted execute privileges on this package you can in effect execute any code whatsoever. For example, if you grant:

```
SQL> grant execute on sys.ltadm to ronb;
Grant succeeded.
```

You can connect to ronb and grant yourself DBA privileges:

```
SQL> connect ronb
Enter password:
Connected.
SQL> -- no privileges yet...
SQL> alter system set audit_trail=none scope=spfile;
alter system set audit_trail=none scope=spfile
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> -- now grant myself privileges
SQL> exec sys.ltadm.execsql('grant DBA to ronb');
PL/SQL procedure successfully completed.

SQL> - try again -- but need to reconnect first
SQL> connect ronb
Enter password:
Connected.
SQL> alter system set audit_trail=none scope=spfile;
System altered.
```

The problem is that this procedure (in older versions of Oracle) allows arbitrary SQL to be injected to a procedure that is running with elevated privileges due to definer rights. As a rule, NEVER allow arbitrary SQL to be injected as a string into a procedure—procedures should encapsulate only very specific behavior.

### Four Things to Remember about Schema Object Privileges

1. You can assign privileges to specific columns in a table, not only to the table itself.
2. Be wary of using the GRANT OPTION because it can cause a proliferation of privileges.  
Audit all privileges assigned with GRANT OPTION.
3. The creator of a view can only assign privileges to the view if they have been given the ability to grant privileges to the base objects.
4. Dynamic SQL is a dangerous thing in general. Don't write procedures that accept a string to be executed and be wary of such procedures within your database. If you have such procedures check whether they are running with definer rights—those mean that an execute privilege to the procedure is like giving the owner's privileges to the caller.

## 15.2 HOWTO Manage System Privileges

A system privilege is a right to perform an action without a qualification on what the action is applied to. For example, you saw in the previous HOWTO that you can have a privilege to UPDATE a certain table. The UPDATE ANY TABLE is an example of a system privilege that allows the grantee to update any table in any schema. You grant and revoke system privileges using the GRANT and REVOKE commands, e.g.,

```
SQL> grant UPDATE ANY TABLE to ronb;
Grant succeeded.
```

There are around 200 distinct system privileges. Table 15.1 lists system privileges available in Oracle 11g ordered by the database object they operate on. Each system privilege allows a database user to perform a certain class of database operations. Within system privileges there is a further subcategory of privileges called the ANY system privileges. These privileges allow the user to operate on any object and any schema in the database. For example, the CREATE TABLE system privilege allows the grantee to create any table in their schema. The CREATE ANY TABLE privilege allows the grantee to create a table in ANY schema.

System privileges are very powerful and they are the most valuable privileges. Of these, system ANY privileges are a “select subset”—the most valuable of the valuable. They exist because without them it would be almost impossible for a database administrator (DBA) to do their job—it would require ongoing maintenance of entitlements to a point that every task would have an associate prerequisite task for aligning privileges to be able to do the work. However, because they are so powerful, they need to be guarded well. System privileges form the “holy grail” as far as attackers are concerned. In fact, being assigned a system ANY privilege is very often equal to being assigned full access to the database. This statement may surprise you—and it should. After all, there are around 100 different system ANY privileges—so why would all these be equivalent to DBA access? It turns out that although this is not a design goal, there are well-known techniques that usually involve “human engineering” that allow one to broaden their assigned privileges once they are assigned a system ANY privilege. If you assign a system ANY privilege to a user you should expect that that user may later manage to attain DBA privileges or at the very least be able to launch an attack. Obviously, if you grant a user SELECT ANY TABLE privileges then at the very least they have access to all your sensitive data. Less obvious are the following three examples in which a user granted a system ANY privilege can get DBA privileges:

**Table 15.1 System Privileges in Oracle 11g**

Object Type	System Privilege	Authorization
ADVISOR	ADVISOR	Access the advisor framework through PL/SQL packages such as DBMS_ADVISOR and DBMS_SQLTUNE
	ADMINISTER SQL TUNING SET	Create, drop, select, load, and delete a SQL tuning set owned by the grantee through the DBMS_SQLTUNE package
	ADMINISTER ANY SQL TUNING SET	Create, drop, select, load, and delete a SQL tuning set owned by any user through the DBMS_SQLTUNE package
	CREATE ANY SQL PROFILE	Accept a SQL Profile recommended by the SQL Tuning Advisor
	DROP ANY SQL PROFILE	Drop an existing SQL Profile
	ALTER ANY SQL PROFILE	Alter the attributes of an existing SQL Profile
CLUSTER	CREATE CLUSTER	Create clusters in the grantee's schema
	CREATE ANY CLUSTER	Create a cluster in any schema
	ALTER ANY CLUSTER	Alter clusters in any schema
	DROP ANY CLUSTER	Drop clusters in any schema
CONTEXT	CREATE ANY CONTEXT	Create any context namespace
	DROP ANY CONTEXT	Drop any context namespace
DATABASE	ALTER DATABASE	Alter a database
	ALTER SYSTEM	Alter system parameters
	AUDIT SYSTEM	Alter audit definitions
DATABASE LINK	CREATE DATABASE LINK	Create private database links in the grantee's schema
	CREATE PUBLIC DATABASE LINK	Create public database links
	DROP PUBLIC DATABASE LINK	Drop public database links
DEBUG	DEBUG CONNECT SESSION	Connect the current session to a debugger
	DEBUG ANY PROCEDURE	Debug all PL/SQL and Java code in any database object
DICTIONARY	ANALYZE ANY DICTIONARY	Analyze any data dictionary object
DIMENSION	CREATE DIMENSION	Create dimensions in the grantee's schema
	CREATE ANY DIMENSION	Create dimensions in any schema
	ALTER ANY DIMENSION	Alter dimensions in any schema
	DROP ANY DIMENSION	Drop dimensions in any schema

(continued)

**Table 15.1 (continued) System Privileges in Oracle 11g**

<i>Object Type</i>	<i>System Privilege</i>	<i>Authorization</i>
DIRECTORY	CREATE ANY DIRECTORY DROP ANY DIRECTORY	Create directory database objects Drop directory database objects
FLASHBACK ARCHIVE	FLASHBACK ARCHIVE ADMINISTER	Create, alter, or drop any flashback data archive
INDEXTYPE	CREATE INDEXTYPE CREATE ANY INDEXTYPE	Create an indextype in the grantee's schema Create an indextype in any schema and create a comment on an indextype in any schema
INDEX	ALTER ANY INDEXTYPE DROP ANY INDEXTYPE EXECUTE ANY INDEXTYPE CREATE ANY INDEX ALTER ANY INDEX DROP ANY INDEX	Modify indextypes in any schema Drop an indextype in any schema Reference an indextype in any schema Create in any schema a domain index or an index on any table in any schema Alter indexes in any schema Drop indexes in any schema
JOB	CREATE JOB CREATE ANY JOB CREATE EXTERNAL JOB	Create jobs, schedules, or programs in the grantee's schema Create, alter, or drop jobs, schedules, or programs in any schema Create in the grantee's schema an executable scheduler job that runs on the operating system
LIBRARY	EXECUTE ANY PROGRAM EXECUTE ANY CLASS MANAGE SCHEDULER CREATE LIBRARY CREATE ANY LIBRARY DROP ANY LIBRARY	Use any program in a job in the grantee's schema Specify any job class in a job in the grantee's schema Create, alter, or drop any job class, window, or window group Create external procedure or function libraries in the grantee's schema Create external procedure or function libraries in any schema Drop external procedure or function libraries in any schema
MATERIALIZED VIEW	CREATE MATERIALIZED VIEW CREATE ANY MATERIALIZED VIEW ALTER ANY MATERIALIZED VIEW DROP ANY MATERIALIZED VIEW	Create a materialized view in the grantee's schema Create materialized views in any schema Alter materialized views in any schema Drop materialized views in any schema

	GLOBAL QUERY REWRITE	Enable rewrite using a materialized view when that materialized view references tables or views in any schema
	ON COMMIT REFRESH	Create a refresh-on-commit materialized view on any table in the database
	FLASHBACK ANY TABLE	Issue a SQL Flashback Query on any table, view, or materialized view in any schema
MINING MODEL	CREATE MINING MODEL	Create mining models in the grantee's schema using the DBMS_DATA_MINING.CREATE_MODEL procedure
	CREATE ANY MINING MODEL	Create mining models in any schema using the DBMS_DATA_MINING.CREATE_MODEL procedure
	ALTER ANY MINING MODEL	Change the mining model name or the associated cost matrix of any model in any schema by using the applicable DBMS_DATA_MINING procedures
	DROP ANY MINING MODEL	Drop any mining model in any schema by using the DBMS_DATA_MINING.DROP_MODEL procedure
	SELECT ANY MINING MODEL	Score or view any model in any schema
	COMMENT ANY MINING MODEL	Create a comment on any model in any schema using the SQL COMMENT statement
CUBE	CREATE CUBE	Create an online analytical processing (OLAP) cube in the grantee's schema
	CREATE ANY CUBE	Create an OLAP cube in any schema
	ALTER ANY CUBE	Alter an OLAP cube in any schema
	DROP ANY CUBE	Drop any OLAP cube in any schema
	SELECT ANY CUBE	Query or view any OLAP cube in any schema
	UPDATE ANY CUBE	Update any cube in any schema
MEASURE	CREATE MEASURE FOLDER	Create an OLAP measure folder in the grantee's schema
	CREATE ANY MEASURE FOLDER	Create an OLAP measure folder in any schema
	DELETE ANY MEASURE FOLDER	Delete from any OLAP measure folder in any schema
	DROP ANY MEASURE FOLDER	Drop any measure folder in any schema
	INSERT ANY MEASURE FOLDER	Insert a measure into any measure folder in any schema
CUBE DIMENSION	CREATE CUBE DIMENSION	Create an OLAP cube dimension in the grantee's schema
	CREATE ANY CUBE DIMENSION	Create an OLAP cube dimension in any schema

(continued)

**Table 15.1 (continued) System Privileges in Oracle 11g**

<i>Object Type</i>	<i>System Privilege</i>	<i>Authorization</i>
CUBE BUILD PROCESS	ALTER ANY CUBE DIMENSION	Alter an OLAP cube dimension in any schema
	DELETE ANY CUBE DIMENSION	Delete from an OLAP cube dimension in any schema
	DROP ANY CUBE DIMENSION	Drop an OLAP cube dimension in any schema
	INSERT ANY CUBE DIMENSION	Insert into an OLAP cube dimension in any schema
	SELECT ANY CUBE DIMENSION	View or query an OLAP cube dimension in any schema
	UPDATE ANY CUBE DIMENSION	Update an OLAP cube dimension in any schema
	CREATE CUBE BUILD PROCESS	Create an OLAP cube build process in the grantee's schema
	CREATE ANY CUBE BUILD PROCESS	Create an OLAP cube build process in any schema
	DROP ANY CUBE BUILD PROCESS	Drop an OLAP cube build process in any schema
	UPDATE ANY CUBE BUILD PROCESS	Update an OLAP cube build process in any schema
OPERATOR	CREATE OPERATOR	Create an operator and its bindings in the grantee's schema
	CREATE ANY OPERATOR	Create an operator and its bindings in any schema and create a comment on an operator in any schema
	ALTER ANY OPERATOR	Modify an operator in any schema
	DROP ANY OPERATOR	Drop an operator in any schema
OUTLINE	EXECUTE ANY OPERATOR	Reference an operator in any schema
	CREATE ANY OUTLINE	Create public outlines that can be used in any schema that uses outlines
	ALTER ANY OUTLINE	Modify outlines
	DROP ANY OUTLINE	Drop outlines
PROCEDURE	CREATE PROCEDURE	Create stored procedures, functions, and packages in the grantee's schema
	CREATE ANY PROCEDURE	Create stored procedures, functions, and packages in any schema
	ALTER ANY PROCEDURE	Alter stored procedures, functions, or packages in any schema
	DROP ANY PROCEDURE	Drop stored procedures, functions, or packages in any schema
PROFILE	EXECUTE ANY PROCEDURE	Execute procedures or functions, either standalone or packaged
	CREATE PROFILE	Create profiles
	ALTER PROFILE	Alter profiles
	DROP PROFILE	Drop profiles

ROLE	CREATE ROLE	Create roles
	ALTER ANY ROLE	Alter any role in the database
	DROP ANY ROLE	Drop roles
	GRANT ANY ROLE	Grant any role in the database
ROLLBACK SEGMENT	CREATE ROLLBACK SEGMENT	Create rollback segments
	ALTER ROLLBACK SEGMENT	Alter rollback segments
	DROP ROLLBACK SEGMENT	Drop rollback segments
	CREATE SEQUENCE	Create sequences in the grantee's schema
SEQUENCE	CREATE ANY SEQUENCE	Create sequences in any schema
	ALTER ANY SEQUENCE	Alter any sequence in the database
	DROP ANY SEQUENCE	Drop sequences in any schema
	SELECT ANY SEQUENCE	Reference sequences in any schema
SESSION	CREATE SESSION	Connect to the database
	ALTER RESOURCE COST	Set costs for session resources
	ALTER SESSION	Enable and disable the SQL trace facility
	RESTRICTED SESSION	Log-on after the instance is started using the SQL*Plus STARTUP RESTRICT statement
SYNONYM	CREATE SYNONYM	Create synonyms in the grantee's schema
	CREATE ANY SYNONYM	Create private synonyms in any schema
	CREATE PUBLIC SYNONYM	Create public synonyms
	DROP ANY SYNONYM	Drop private synonyms in any schema
TABLE	DROP PUBLIC SYNONYM	Drop public synonyms
	CREATE TABLE	Create a table in the grantee's schema
	CREATE ANY TABLE	Create a table in any schema
	ALTER ANY TABLE	Alter any table or view in any schema
	BACKUP ANY TABLE	Use the export utility to incrementally export objects from the schema of other users
	DELETE ANY TABLE	Delete rows from tables, table partitions, or views in any schema
	DROP ANY TABLE	Drop or truncate tables or table partitions in any schema
	INSERT ANY TABLE	Insert rows into tables and views in any schema
	LOCK ANY TABLE	Lock tables and views in any schema

(continued)

**Table 15.1 (continued) System Privileges in Oracle 11g**

Object Type	System Privilege	Authorization
	SELECT ANY TABLE FLASHBACK ANY TABLE	Query tables, views, or materialized views in any schema Issue a SQL Flashback Query on any table, view, or materialized view in any schema
	UPDATE ANY TABLE	Update rows in tables and views in any schema
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE MANAGE TABLESPACE	Create tablespaces Alter tablespaces Drop tablespaces Take tablespaces off-line and online and begin and end tablespace backups
	UNLIMITED TABLESPACE	Use an unlimited amount of any tablespace. This privilege overrides any specific quotas assigned. If you revoke this privilege from a user, then the user's schema objects remain but further tablespace allocation is denied unless authorized by specific tablespace quotas. You cannot grant this system privilege to roles.
TRIGGER	CREATE TRIGGER CREATE ANY TRIGGER ALTER ANY TRIGGER DROP ANY TRIGGER	Create a database trigger in the grantee's schema Create database triggers in any schema Enable, disable, or compile database triggers in any schema Drop database triggers in any schema
	ADMINISTER DATABASE TRIGGER	Create a trigger on DATABASE. You must also have the CREATE TRIGGER or CREATE ANY TRIGGER system privilege.
TYPE	CREATE TYPE CREATE ANY TYPE ALTER ANY TYPE DROP ANY TYPE EXECUTE ANY TYPE	Create object types and object type bodies in the grantee's schema Create object types and object type bodies in any schema Alter object types in any schema Drop object types and object type bodies in any schema Use and reference object types and collection types in any schema, and invoke methods of an object type in any schema if you make the grant to a specific user. If you grant EXECUTE ANY TYPE to a role, then users holding the enabled role will not be able to invoke methods of an object type in any schema.
	UNDER ANY TYPE	Create subtypes under any nonfinal object types

USER	CREATE USER	Create users, assign quotas, set default and temporary tablespace and assign profiles
	ALTER USER	Alter any user, change passwords and authentication method, assign quotas, set default and temporary tablespace, and assigns profiles
	DROP USER	Drop users
	CREATE VIEW	Create views in the grantee's schema
VIEW	CREATE ANY VIEW	Create views in any schema
	DROP ANY VIEW	Drop views in any schema
	UNDER ANY VIEW	Create subviews under any object views
	FLASHBACK ANY TABLE	Issue a SQL Flashback Query on any table, view, or materialized view in any schema
	MERGE ANY VIEW	If a user has been granted the MERGE ANY VIEW privilege, then for any query issued by that user, the optimizer can use view merging to improve query performance without performing the checks that would otherwise be performed to ensure that view merging does not violate any security intentions of the view creator
	ANALYZE ANY	Analyze any table, cluster, or index in any schema
	AUDIT ANY	Audit any object in any schema using AUDIT schema_objects statements
Others	BECOME USER	Allows users of the data pump import utility (impdp) to assume the identity of another user to perform operations that cannot be directly performed by a third party
	CHANGE NOTIFICATION	Create a registration on queries and receive database change notifications in response to DML or Data Definition Language (DDL) changes to the objects associated with the registered queries
	COMMENT ANY TABLE	Comment on any table, view, or column in any schema
	EXEMPT ACCESS POLICY	Bypass fine-grained access control (see Chapter 16)
	FORCE ANY TRANSACTION	Force the commit or rollback of any in-doubt distributed transaction in the local database
	FORCE TRANSACTION	Force the commit or rollback of the grantee's in-doubt distributed transactions in the local database

(continued)

**Table 15.1 (continued) System Privileges in Oracle 11g**

Object Type	System Privilege	Authorization
	GRANT ANY OBJECT PRIVILEGE	Grant any object privilege that the object owner is permitted to grant and revoke any object privilege that was granted by the object owner or by some other user with this system privilege
GRANT ANY PRIVILEGE		Grant any system privilege
RESUMABLE		Enable resumable space allocation
SELECT ANY DICTIONARY		Query any data dictionary object in the SYS schema
SELECT ANY TRANSACTION		Query the contents of the FLASHBACK_TRANSACTION_QUERY view
SYSDBA		Perform STARTUP and SHUTDOWN operations
	ALTER DATABASE: open, mount, back up, or change character set	
	CREATE DATABASE	
	ARCHIVELOG and RECOVERY	
	CREATE SPFILE	
SYSOPER		Includes the RESTRICTED SESSION privilege
		Perform STARTUP and SHUTDOWN operations
	ALTER DATABASE: open, mount, or back up	
	ARCHIVELOG and RECOVERY	
	CREATE SPFILE	
		Includes the RESTRICTED SESSION privilege

1. CREATE ANY VIEW or ALTER ANY VIEW: Suppose that the DBA grants you the ability to create any view in any schema. You can then create a procedure that grants you DBA privileges. Define the procedure so that it will run with invoker rights and use your CREATE ANY VIEW privilege to create a view in the SYSTEM schema that runs the procedure, or better yet, change an existing view to also run the procedure. Now convince the DBA to take a quick look at this view and you're a DBA.
2. CREATE ANY TRIGGER and ALTER ANY TRIGGER: This follows a similar pattern. Create a procedure as in the previous case. Find a table in a DBA's schema to which PUBLIC has some data manipulation language (DML) privilege. Create a trigger on this table that calls your procedure. Now use the PUBLIC privilege to change some data—you're now a DBA because the trigger runs as the schema owner.
3. EXECUTE ANY PROCEDURE: Depending on the Oracle version, there are procedures in the SYS schema that run with definer rights and that can be used to run dynamic SQL statements. If you're given EXECUTE ANY PROCEDURE use them to run a GRANT statement and grant yourself DBA privileges, e.g.

```
exec sys.dbms_repcat_sql_util.do_sql('grant dba to ronb', true)
```

Hopefully these examples convinced you how sensitive system privileges are. This is true for all system privileges. System ANY privileges are especially dangerous, but you should manage all system privileges with care and audit their assignment.

## ADMIN OPTION

When you grant a privilege you can grant it with the ADMIN option; this is the equivalent in the system privilege world to the GRANT OPTION in the object privilege world. Granting a privilege with the ADMIN option means that the user can grant that privilege onwards to other users. The ADMIN option says not only that the privilege is available to the user but also that the administration of that privilege (granting of that option) is available to the user. As an example, if you just grant a CREATE TABLE privilege to a user (assuming they also have quota) then the user can create tables, but cannot grant that privilege onwards. But if you grant the privilege with the ADMIN option then they can propagate this privilege. First let's see what happens without the ADMIN option:

```
SQL> connect system
Enter password: *****
Connected.

SQL> grant create any trigger to ronb;
Grant succeeded.

SQL> connect ronb
Enter password: *****
Connected.

SQL> grant create any trigger to scott;
grant create any trigger to scott
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

Now try with ADMIN option:

```
SQL> connect system
Enter password: *****
Connected.

SQL> grant create any trigger to ronb with admin option;
Grant succeeded.

SQL> connect ronb
Enter password: *****
Connected.

SQL> grant create any trigger to scott;
Grant succeeded.
```

Of course, a user who has been granted a privilege with the ADMIN option can grant the privilege also with the ADMIN option, so once you start with the ADMIN option you risk loosing control over who else will end up with the privilege:

```
SQL> connect ronb
Enter password: *****
Connected.

SQL> grant create any trigger to scott with admin option;
Grant succeeded.
```

## O7\_DICTIONARY\_ACCESSIBILITY

When you grant a system ANY privilege, you're granting access to any schema. As an example, if you grant the SELECT ANY TABLE privilege to a user, that user can select from any table in any schema. Well, almost any schema—there is an exception. Even with this privilege the user is not able to select dictionary objects. This is intentional and due to the high sensitivity of the data dictionary. If you want users to have such permissions you can always grant them explicit privileges (or connect as the SYS user or with SYSDBA privileges). The exception made here ensures that you don't grant these privileges by mistake while granting broad system privileges.

This default behavior is the one you should always employ; limiting access to the data dictionary is important. There is a way to change this default behavior using the O7\_DICTIONARY\_ACCESSIBILITY initialization parameter. For example, to set the value to FALSE:

```
SQL> alter system set O7_DICTIONARY_ACCESSIBILITY=FALSE scope=spfile;
System altered.
```

The default for this value is FALSE in which case the mentioned restrictions on system privileges apply (for all system ANY privileges). If you set this parameter to TRUE then access to the SYS schema is allowed. This was the behavior in Oracle 7—hence the name of the parameter.

When you want to assign privileges to the data dictionary, use the three built-in roles SELECT\_CATALOG\_ROLE (for data dictionary views), EXECUTE\_CATALOG\_ROLE (for packages and procedures in the data dictionary), and DELETE\_CATALOG\_ROLE (for deleting from the audit trail). These roles group together privileges to objects in the SYS schema. They should

all be assigned with care but they give you a good way to control access to the data dictionary—O7\_DICTIONARY\_ACCESSIBILITY does not. In addition, the SELECT ANY DICTIONARY system privilege allows query access to any object in the SYS schema (not only to views). This is even a more sensitive privilege and as such it can only be assigned explicitly and is not included in GRANT ALL PRIVILEGES.

### **Three Things to Remember about System Privileges**

1. Understand the risk and limit the use of system ANY privileges. Many system ANY privileges can be used to attain more privileges.
2. Be wary of using the ADMIN OPTION because it can cause a proliferation of privileges.  
Audit all privileges assigned with ADMIN OPTION.
3. Don't enable O7\_DICTIONARY\_ACCESSIBILITY.

## **15.3 HOWTO Use Roles to Manage Privileges**

Roles are named groups of privileges. You grant privileges to roles and then grant roles to users or to other roles. When you grant a role you in effect grant a group of privileges to users or to other roles. Roles allow you to build a recursive privilege structure—you can grant a role to any other role (except itself) as long as the created structure is a noncircular graph. Obviously, you cannot create a recursive loop within the privilege structure:

```
SQL> grant role_b to role_a;
Grant succeeded.
SQL> grant role_c to role_b;
Grant succeeded.
SQL> grant role_a to role_c;
grant role_a to role_c
*
ERROR at line 1:
ORA-01934: circular role grant detected
```

It is much easier to maintain a privilege model using roles than using individual privileges—especially when you have many users, applications, and privileges. Roles reduce the burden of privilege administration and allow for dynamic assignment to users. You can use roles in any way you see fit, as long as you remember what the recursive structure implies. This is not always trivial to do and in HOWTO 15.7 you'll see a few queries that help you audit the full entitlement model implied by the role hierarchy.

Sometimes a complex role hierarchy is mandated—especially when you have many application suites over which you have less control over. From a best-practice perspective larger and more complex hierarchies increase risk. Entitlement audits are the main tool with which you address this issue but you can also use MAX\_ENABLED\_ROLES. MAX\_ENABLED\_ROLES is an initialization parameter that limits the number of database roles that can be enabled for a user. This includes recursive containment of roles through other roles. The default value for Oracle 10g is 150 roles and the default value for Oracle 11g is 30 roles.

A user may be assigned any number of roles. For example, you can create two separate roles as two groups of privileges and assign both to a user:

```
SQL> create role hr_manager;
Role created.
SQL> grant select, insert, update, delete on emp to hr_manager;
Grant succeeded.
SQL> grant select, insert, update, delete on dept to hr_manager;
Grant succeeded.
SQL> grant hr_manager to ronb;
Grant succeeded.
SQL> create role hr_clerk;
Role created.
SQL> grant select on emp to hr_clerk;
Grant succeeded.
SQL> grant hr_clerk to ronb;
Grant succeeded.
```

At this point ronb has two roles—which you can see by querying dba\_role\_privs:

```
SQL> select * from dba_role_privs where grantee='RONB';
GRANTEE GR          ANTENDED_ROLE          ADM DEF
-----  -----          -----  -----
RONB H             R_CLERK                NO   YES
RONB H             R_MANAGER              NO   YES
2 rows selected.
```

To revoke the privileges assigned through a role use the REVOKE command. Additionally, if you DROP a role then the appropriate privileges are immediately removed from the users and roles which were granted the privileges through the dropped role—directly or indirectly.

## Default Roles

In the previous example both roles pass their privileges to the user so RONB can perform anything that is allowed through the privileges assigned to either HR\_CLERK or HR\_MANAGER. However, each role has its own unique security realm and Oracle can determine whether an operation is allowed or not using the currently enabled role. When you assign a role to a user you can use the default clause to specify the default role that the user assumes. In this case the privileges assigned to other roles assigned to the user are only available after the user performs a SET ROLE command. For example, instead of assigning the HR\_CLERK and HR\_MANAGER to RONB as before, let's change the definition of the user to have a default role:

```
SQL> alter user ronb default role hr_clerk;
User altered.
```

At this point RONB still has both roles, but only one of them is assumed by default:

```
SQL> select * from dba_role_privs where grantee='RONB';
```

GRANTEE GR	ANTED_ROLE	ADM	DEF
RONB H	R_CLERK	NO	YES
RONB H	R_MANAGER	NO	NO

2 rows selected.

If you connect using RONB you will initially just have the HR\_CLERK role:

```
SQL> connect ronb
Enter password: ****
Connected.
```

```
SQL> select * from scott.emp where ename='RON';
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
4444	RON	DEV	7902	02-NOV-02	1171.6	100	20

1 row selected.

```
SQL> update scott.emp set sal=2000 where ename='RON';
update scott.emp set sal=2000 where ename='RON'
```

```
*
```

```
ERROR at line 1:
ORA-01031: insufficient privileges
```

If you want to have the privileges associated with the HR\_MANAGER role you have to formally assume the role:

```
SQL> set role hr_manager;
Role set.

SQL> update scott.emp set sal=2000 where ename='RON';
1 row updated.
```

To see what roles you have within the session, use the SESSION\_ROLE table:

```
SQL> select * from session_roles;
ROLE
-----
HR_MANAGER

1 row selected.
```

You can set multiple roles within the session even when there is a default role:

```
SQL> set role hr_manager, hr_clerk;
Role set.
```

```
SQL> select * from session_roles;
ROLE
-----
HR_MANAGER
HR_CLERK
2 rows selected.
```

You can also set passwords so that a user has to supply a password for the SET ROLE command to be successful:

```
SQL> create role HR_MANAGER identified by "0okmj7";
Role created.

SQL> grant HR_MANAGER to scott;
Grant succeeded.

SQL> connect scott
Enter password:
Connected.
SQL> set role HR_MANAGER;
set role HR_MANAGER
*
ERROR at line 1:
ORA-01979: missing or invalid password for role 'HR_MANAGER'

SQL> set role HR_MANAGER identified by "0okmj7";
Role set.
```

This is an example where you authorize a role using the database. This can be useful in scenarios where a database user may be mapped to potentially more than one end user and end users should not have the same privileges. This is the main purpose of secure application roles in which application code authorizes the role—as described in the next HOWTO.

### Three Things to Remember about Roles and Privileges

1. Roles group privileges and are very effective in administering privileges. Users can be granted multiple roles and roles can be granted to other roles, forming a recursive privilege hierarchy.
2. Users can have a default role when they are granted multiple roles. A user can manually set a role to be used within the session when they want to use privileges granted to this nondefault role.
3. Roles can be protected using a password. A user who desires to set the role for the session needs to provide the password.

## 15.4 HOWTO Use Secure Application Roles

You saw in the previous HOWTO that you can control what privileges a user session has through the setting of roles within the session. This is a very useful feature because it allows you to

dynamically control privileges by having different roles active for the session at different points in time. The SET ROLE command is the mechanism for assuming privileges and passwords protect this process so that a user will not immediately enable all their roles and then run with maximal privileges. In this HOWTO you'll learn about another mechanism called secure application roles.

A secure application role is a role that can only be enabled by a specific PL/SQL package or procedure. The collection of privileges forming the role is granted by the application code to the user session through a call to a procedure. The package that "authorizes" the privileges acts as the security policy for the application privileges. Although a secure application role is a very well-defined entity (it is a role that can only be enabled by a package), secure application roles are really a design pattern that allows you to dynamically control privileges through application logic.

When you use secure application roles you set up a set of roles that are tied to a package or a procedure, you assign privileges to these roles, and you implement the procedure that sets the roles. The roles cannot be enabled within a session directly—they can only be enabled by the procedure. The calls to the procedure that enable the roles are usually made from within a system log-on trigger (if the roles need to be set for the entire user session) or by the application code when a single session can dynamically assume or release privileges. Using secure application roles involves

- Defining a secure application role (a role that can only be set by a procedure) and assigning privileges to the role.
- Implementing the procedure which sets the role through a call to DBMS\_SESSION.SET\_ROLE.
- Making calls to the assigning procedure from application code, manually (by granting privileges to execute the procedure) or through a system log-on trigger.

Let's look at a simple example. In this example you'll provide access to the SCOTT.EMP table for users whose name is in ENAME and who are managers. A user who is not a manager should not have the ability to look at this data. What you'll do is define the secure application role and the controlling procedure. You'll assign the privileges to the role and implement the procedure. The procedure will use a view on the EMP table to lookup the user name in the session to determine whether the user is a manager and if so, set the role. Follow these steps to implement this example:

Step 1. Define the view and grant privileges. Any user involved in the application should have SELECT privileges on the view because all logons need to lookup their name in the table:

```
SQL> connect scott
Enter password:
Connected.

SQL> create view emp_job as (select ename, job from emp);
View created.

SQL> grant select on emp_job to blake;
Grant succeeded.

SQL> grant select on emp_job to james;
Grant succeeded.
```

Make sure that a user only has privileges to the view but not the table:

```

SQL> connect ronb
Enter password:
Connected.
SQL> select count(*) from scott.emp_job;
CO UNT(*)
-----
14

SQL> select * from scott.emp;
select * from scott.emp
*
ERROR at line 1:
ORA-01031: insufficient privileges

```

Step 2. Create the secure application role and grant it the appropriate object privileges:

```

SQL> create role manager_role identified using access_control_policy;
Role created.

SQL> grant select on scott.emp to manager_role;
Grant succeeded.

```

Note that you're not granting the role to any user—the package will do that. If a user tries to set the role they will fail—only the procedure will be able to set it:

```

SQL> set role manager_role;
set role manager_role
*
ERROR at line 1:
ORA-28201: Not enough privileges to enable application role 'MANAGER_ROLE'

```

Step 3. Build the procedure that assigns the role. The procedure looks up the currently logged on user and looks it up in the emp\_job view. If it is a manager it sets the role thereby granting privileges to access the table:

```

create or replace PROCEDURE access_control_policy
authid current_user
AS
  v_us er varchar2(50);
  v_j ob varchar2(50);

BE GIN
  -- get the user from the context
  v_us er := lower( (sys_context ('userenv','session_user')));
  -- get the job description
  select job
  in to v_job
  f rom scott.emp_job
  where lower(ename) = v_user;
  -- if we're a manager then set the role
  if
    v_j ob = 'MANAGER'
  then

```

```

D  BMS_SESSION.SET_ROLE('manager_role');
else null;
end if;
END access_control_policy;
/

```

Step 4. Grant execute privileges to the appropriate users, e.g.,

```
SQL> grant execute on access_control_policy to blake;
Grant succeeded.
```

```
SQL> grant execute on access_control_policy to james;
Grant succeeded.
```

Step 5. Test manually by invoking the procedure from your session. Blake should be able to access EMP while James should not, due to the contents of emp\_job:

```

SQL> select * from emp_job;
ENAME J      OB
-----
...
BLAKE M      ANAGER
...
JAMES CLER    K
...
SQL> connect blake
Enter password:
Connected.

SQL> exec scott.access_control_policy;
PL/SQL procedure successfully completed.

SQL> select * from session_roles;

ROLE
-----
MANAGER_ROLE

SQL> - I am a manager - I can see salaries
SQL> set linesize 100
SQL> select max(sal) from scott.emp;

MAX (SAL)
-----
5030

SQL> connect james
Enter password:
Connected.

SQL> exec scott.access_control_policy;
PL/SQL procedure successfully completed.

SQL> select * from session_roles;

ROLE
-----
CONNECT

```

```
SQL> -- I am just a lowly clerk..
SQL> select max(sal) from scott.emp;
select max(sal) from scott.emp
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

Finally, you can now create a log-on trigger that will execute the procedure and thus set the role as soon as the user logs in.

To view which package/procedure controls the secure application role:

```
SQL> select * from dba_application_roles;
ROLE S          CHEMA          PACKAGE
-----
MANAGER_ROLE S      COTT          ACCESS_CONTROL_POLICY
```

### Three Things to Remember about Secure Application Roles

- Secure application roles are created and associated with a package or procedure. Only that procedure can set the role. This allows you to add application code that determines whether or not a user can assume the role and therefore the privileges.
- Secure application roles are very powerful and useful. They allow you to build a very dynamic application-logic-based privilege model using database privileges and roles.
- You can use secure application roles statically, setting the roles when the user logs on (using a log-on trigger) or dynamically within a session. You can even have the same session (potentially used by multiple users) set to different roles at different times.

## 15.5 HOWTO Manage the PUBLIC Role

PUBLIC is one of security's worst enemies. PUBLIC is not a user and not a role—it is a user group. It can be puzzling because when you try to select from dba\_roles it does not appear:

```
SQL> select * from dba_roles where role='PUBLIC';
no rows selected
```

And yet, you can't create a role by the name of public:

```
SQL> create role public;
create role public
*
ERROR at line 1:
ORA-01921: role name 'PUBLIC' conflicts with another user or role name
```

The reason is that it is hidden by the dba\_roles view:

```
create or replace view DBA_ROLES (ROLE, PASSWORD_REQUIRED)
```

```

as
select name, decode(password, null, 'NO', 'EXTERNAL', 'EXTERNAL', 'GLOBAL', 'GLOBAL',
'YES')
from user$
where type# = 0 and name not in ('PUBLIC', '_NEXT_USER');

```

It is marked as a role in user\$ (type#=0) but really it is a user group:

```

SQL> select user#, name, type# from sys.user$;

USER#  NAME          TYPE#
-----  -----
0      SYS           1
1      PUBLIC        0
2      CONNECT       0
3      RESOURCE      0
4      DBA           0
5      SYSTEM        1
...

```

What's special about PUBLIC, and what makes it dangerous, is that any user is automatically assigned all privileges granted to PUBLIC—it is like having a role that is granted right after you execute a CREATE USER command. Therefore, you must be very careful when you grant to PUBLIC and you must periodically audit which privileges are granted to PUBLIC as you'll see in HOWTO 15.7. This is also why there is so much emphasis on PUBLIC in all database hardening guidelines. One annoyance is that installing a critical patch update (CPU) or a patchset will often regrant many privileges to PUBLIC that you may have painstakingly revoked using such hardening guidelines.

You must also be careful when you revoke privileges from PUBLIC. Any privilege related to a DML operation that is revoked from PUBLIC will cause all procedures in the database to become invalid and they will have to be reauthorized. If you revoke a privilege on an object from PUBLIC then all objects that depend on it become invalid. If you revoke a system privilege from PUBLIC (which PUBLIC should never have) then all objects in the database become invalid.

#### Four Things to Remember about Privileges and the PUBLIC User Group

1. Be careful about what you grant to PUBLIC—all users will be getting these privileges.
2. Be careful about what you revoke from PUBLIC—you may invalidate much of your application code.
3. Always audit which privileges are assigned to PUBLIC.
4. Installation of CPUs and patchsets may add privileges to PUBLIC that you have revoked as part of a hardening process; you may have to redo this hardening phase.

## 15.6 HOWTO Use Access Control Lists (ACLs) to Limit Access to Database Network Services

The Oracle database has a number of packages that allow you to send messages and data from the database to networked servers, including:

- UTL\_TCP allows you to open Transmission Control Protocol/Internet Protocol (TCP/IP) connections in PL/SQL. Your code acts as a client and uses a TCP/IP socket to a networked service. The package includes procedures such as utl\_tcp.open\_connection, utl\_tcp.write\_line and utl\_tcp\_close\_connection.
- UTL\_SMTP allows you to send e-mail messages over the Simple Mail Transfer Protocol (SMTP). Your code generates SMTP messages and sends them over a TCP/IP connection. The package gives you the same primitives used by e-mail clients and it is up to you to construct the right sequence. For example, the package gives you procedures such as utl\_smtp.helo, utl\_smtp.mail, utl\_smtp.rcpt, utl\_smtp.open\_data, utl\_smtp.write\_data, utl\_smtp.quit, etc.
- UTL\_HTTP allows you to write PL/SQL code that makes Hypertext Transfer Protocol (HTTP) requests over a TCP/IP connection. The package gives you primitives for managing requests, responses, sessions, and even cookies. The package also supports Hypertext Transfer Protocol Secure (HTTPS) requests running over Secure Socket Layer (SSL). The package includes procedures such as utl\_http.begin\_request, utl\_http.set\_header, utl\_http.get\_response, utl\_http.read\_line, etc.

All of these utility packages give you access to network services and all have EXECUTE privileges assigned to PUBLIC (as is UTL\_INADDR which allows you to resolve hosts and IP addresses). This is too risky. These packages can be used by an attacker in two ways. First, an attacker that gains access to any account can use these packages to launch network-based attacks on other services from within the database (that is, almost always running on an internal and trusted node). More importantly, these packages can be used to extract data and send it out of the database. An attacker that manages to access sensitive data can grab some of this data and be done with it. However, a far more lucrative attack is one where this access can be used over a lengthy period of time. Rather than risk having to connect and use the access method many times, it is far less risky for the attacker to plant some process that will periodically use the method to get the sensitive data and then use one of these packages to send that data to the attacker's networked service. This way the attacker does not have to continuously connect to the database—the database will send the data every day or every month to the attacker's service and no one is the wiser.

Given these two reasons, it is not surprising the all hardening guidelines recommend that you revoke access to these utilities from PUBLIC and audit who does have privileges to these packages. In addition, Oracle 11g adds fine-grained access control to these networking packages—access control that allows you to define precisely which hosts you can connect to from the database, which ports, etc. These ACLs allow you to limit connections based on what your application requirements are and reduce the risk of attacks—the attacker will not be able to send data to their service running on their host.

If you upgrade from 10g to 11g and your application uses these packages, you will have to define these ACLs for your application to continue working correctly—it is no longer enough that your application has execute privileges on these packages. If you don't, you will get the following error:

```
ORA-24247: network access denied by access control list (ACL)
```

## **Creating an ACL**

ACLs are stored inside eXtensible Markup Language (XML) Database (DB)—so the first step is to make sure that XML DB is installed. To check whether it is installed, use dba\_registry:

```
SQL> select comp_name, status from dba_registry where comp_name like '%XML%';
COMP_NAME
-----
-----
STATUS
-----
Oracle XML Database
VALID
```

If XML DB is not installed you can install it by:

```
-bash-3.00$ sqlplus /nolog
SQL*Plus: Release 11.1.0.6.0 - Production on Thu Mar 20 12:17:57 2008
Copyright (c) 1982, 2007, Oracle. All rights reserved.
SQL> connect / as sysdba
Connected.
SQL> @?/rdbms/admin/catqm.sql
```

ACLs are XML documents and you can use XML DB directly to create the ACLs. More typically, you will use the DBMS\_NETWORK\_ACL\_ADMIN and DBMS\_NETWORK\_ACL\_UNUTILITY packages to programmatically manage ACLs from PL/SQL. Let's look at a simple example in which you'll allow access to alerts that can either be sent over an HTTP service running on port 8080 on one node or as e-mails sent to an SMTP server that runs on port 25 on a second node. Any other connection attempt will be refused.

Step 1: Create the ACL:

```
begin
  dbms_network_acl_admin.create_acl(
    acl => 'application_alerting.xml',
    description => 'Network permissions limited to sending alerts',
    principal => 'APP_ALERTER',
    is_grant => TRUE,
    privilege => 'connect');
end;
/
```

The first parameter, acl, is the name of the ACL. An XML document by this name is created within XML DB. The principal is a user or role to which the ACL is assigned. You can assign it to additional users and roles—but you must assign it to at least one principal. Is\_grant can be TRUE or FALSE—false meaning that you're denying the access. This allows you to define access using either white lists or black lists or any combination. Privilege is either “connect” or “resolve”—connect is the more common option and affects all utilities that create a connection whereas resolve affects your attempts to resolve a host/IP address.

## **Adding Privileges**

You can add the ACL to additional users or roles:

```
begin
  dbms_network_acl_admin.add_privilege(
    acl => 'application_alerting.xml',
```

```

p_rinciple => 'SCOTT',
i_s_grant => FALSE,
p_rivilege => 'connect',
position => null);
end;
/

```

The position parameter is important when you are adding multiple users and roles and some users may belong to some roles. Because you can either grant or deny permissions, Oracle uses the position value to determine which of the possibly conflicting definitions takes precedence. Evaluation is done top-down—so if you have two conflicting specifications the first one counts. For example, if you want to make sure that SCOTT cannot access the network services even though SCOTT was granted the APP\_ALERTER role, you should execute:

```

begin
d_bms_network_acl_admin.add_privilege(
  a_cl => 'application_alerting.xml',
  p_rinciple => 'SCOTT',
  i_s_grant => FALSE,
  p_rivilege => 'connect',
  p osition => 1);
end;
/

```

## Assigning an ACL

The next step is to say what network services are controlled by the ACL. This is called assigning the ACL to a network service (by specifying hosts and ports):

```

begin
d_bms_network_acl_admin.assign_acl(
  a_cl => 'application_alerting.xml',
  h_ost => 'capricorn.dbasecurity.com',
  l_ower_port => '8080',
  up_per_port => '8080');
end;
/
begin
d_bms_network_acl_admin.assign_acl(
  a_cl => 'application_alerting.xml',
  h_ost => 'smtp.dbasecurity.com',
  l_ower_port => '25',
  up_per_port => '25');
end;
/

```

You can use port ranges when assigning the ACL and you can use wildcards when specifying hosts. Wildcards can be used in both hostnames and IPs—for example, you can assign an ACL to \*.dbasecurity.com or to 192.168.2.\*. When using wildcards an ACL assigned to a more specific network definition takes precedence over an ACL assigned using a general network specification. For example, if you have one ACL assigned to smtp.dbasecurity.com and one ACL assigned to \*.dbasecurity.com then the first ACL takes precedence.

To view the XML document stored within XML DB on your behalf:

```
SQL> SELECT a.OBJECT_VALUE
  2  FROM RESOURCE_VIEW r, XDB.XDB$ACL a
  3 WHERE ref(a) = extractValue(r.RES, '/Resource/XMLRef')
  4 AND equals_path(r.RES, '/sys/acls/application_alerting.xml') = 1;
OBJECT_VALUE
-----
<a:acl description="Network permissions limited to sending alerts" xmlns:a="http://xmlns.oracle.com/xdb/acl.xsd" xmlns:plsql="http://xmlns.oracle.com/plsql" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd http://xmlns.oracle.com/xdb/acl.xsd" shared="true">
  <a:security-class>plsql:network</a:security-class>
  <a:ace>
    <a:grant>true</a:grant>
    <a:principal>APP_ALERTER</a:principal>
    <a:privilege>
      <p lsql:connect xmlns:plsql="http://xmlns.oracle.com/plsql"/>
    </a:privilege>
  </a:ace>
  <a:ace xmlns:a="http://xmlns.oracle.com/xdb/acl.xsd">
    <a:grant>true</a:grant>
    <a:principal>SCOTT</a:principal>
    <a:privilege>
      <p lsql:connect xmlns:plsql="http://xmlns.oracle.com/plsql"/>
    </a:privilege>
  </a:ace>
</a:acl>
```

## Checking Privileges

Use the dba\_network\_acl\_privileges view to display ACLs and their granting to users and roles:

```
select acl, principal, privilege, is_grant, invert from dba_network_acl_privileges
ACL P          RINCIPAL      PRIVILEGE     IS _ GRANT   INVERT
-----  -----  -----  -----
/sys/acls/application_alerting.xml A PP _ ALERTER connect      true      false
/sys/acls/application_alerting.xml S COTT        connect      true      false
2 record(s) selected [Fetch MetaData: 32/ms] [Fetch Data: 46/ms]
```

Use the dba\_network\_acls view to see the assignment of ACLs:

```
select acl, host, lower_port, upper_port from dba_network_acls
ACL HOS          T          LOWER_ PORT      UPPER_ PORT
-----  -----  -----  -----
/sys/acls/application_alerting.xml c apricorn.dbasecurity.com 8080      8080
/sys/acls/application_alerting.xml s smtp.dbasecurity.com   25       25
2 record(s) selected [Fetch MetaData: 0/ms] [Fetch Data: 0/ms]
```

To check programmatically if a privilege has been granted to a user:

```
SQL> select dbms_network_acl_admin.check_privilege_aclid(
  2  '48E276A2AD569002E043C0A802A39002','SCOTT','connect')
  3 from dual;
DBMS_NETWORK_ACL_ADMIN.CHECK_PRIVILEGE_ACLID('48E276A2AD569002E043C0A802A39002',
-----
1
```

### Three Things to Remember about Fine-Grained Access Control to Network Services

1. Utility packages such as UTL\_TCP, UTL\_SMTP, and UTL\_HTTP are risky because they allow an attacker to launch an attack from the database and because an attacker can use these packages to send sensitive data from the database to an attacker's program running on some node.
2. You can build ACLs that control which user or role has the ability to connect to which hosts (IPs) and which ports. You can define this at various granularity levels.
3. ACLs are XML documents that are stored using XML DB.

## 15.7 HOWTO Generate Entitlement Audit Reports

Oracle has a very sophisticated privilege model. This means that you can configure almost anything in terms of privileges and that you can be very granular in your definitions. Privilege and role assignments are recursive—you can grant a privilege to a role, grant that role to another role, and finally that role to a user. Unfortunately, all this means that it is often difficult to know precisely what the current state of privileges are within the database. When you have big applications like Oracle EBS, PeopleSoft, SAP, etc. this can become overwhelming. As a result, auditors will often require you to provide a set of entitlement reports through which privileges can be reviewed. These are not simple—so in this HOWTO you'll see a set of queries that you can use for producing these entitlement reports.

### Privilege Hierarchy

Because of the recursive nature of the privilege model, it is hard to tell which privileges a user has and where they got them from (directly or through a role). The following query produces a result which shows a hierarchical model of privileges:

```
select
  lpad('>', level, '-') ||
  g_ranted_role "User/Role/Privilege"
from (
  select
    null grantee,
    u.username granted_role,
    'N/A' admin_option
  from sys.dba_users
```

```

UNION
SELECT
    grantee,
    granted_role,
    admin_option
FROM sys.dba_role_privs
UNION
SELECT
    grantee,
    privilege granted_role,
    admin_option
FROM sys.dba_sys_privs)
START WITH grantee IS NULL
CONNECT BY grantee = prior granted_role

```

The result produced uses an indentation in which all privileges are shown per user but the indentation is based on the roles granted to users. For example, in the partial result set shown below, the BI user has been assigned the ALTER SESSION privilege but also has the CREATE CLUSTER privilege which it gets through the RESOURCE role:

User/Role/Privilege

```

-----
>ANONYMOUS
->CREATE SESSION
->APEX_PUBLIC_USER
->CREATE SESSION
>BI
->ALTER SESSION
->CREATE CLUSTER
->CREATE DATABASE LINK
->CREATE SEQUENCE
->CREATE SESSION
->CREATE SYNONYM
->CREATE TABLE
->CREATE VIEW
->RESOURCE
->CREATE CLUSTER
->CREATE INDEXTYPE
->CREATE OPERATOR
->CREATE PROCEDURE
->CREATE SEQUENCE
->CREATE TABLE
->CREATE TRIGGER
->CREATE TYPE
->UNLIMITED TABLESPACE
...

```

### System Privileges and Admin Option

Listing 15.2 shows a query for producing all system privileges and admin option values for users and roles. There are two parts of this UNION. The first part gets values assigned through roles and the second gets values assigned directly. The first part of the UNION has an internal UNION that iterates over users and roles. A partial sample result follows:

```

select distinct
  role_grants.grantee,
    user_or_role.u_or_r "User or Role",
    dsp.privilege "System Privilege",
    dsp.admin_option
from
  (select   distinct
    connect_by_root grantee "GRANTEE",
    granted_role
  from    sys.dba_role_privs
    connect by prior granted_role = grantee
    order by 1,2) role_grants,
  (select
    'User' u_or_r,
    username name
  from    sys.dba_users
UNION
  select
    'Role' u_or_r,
    role name
  from  dba_roles) user_or_role,
  sys.dba_sys_privs    dsp
where
  role_grants.grantee = user_or_role.name and
  dsp.grantee = role_grants.granted_role
UNION
select distinct
  du.username grantee,
  'User' u_or_r,
  ds.privilege,
  ds.admin_option
from
  sys.dba_users    du,
  sys.dba_sys_privs    ds
where
  ds.grantee = du.username
UNION
select distinct
  dr.role grantee,
  'Role' u_or_r,
  dsp.privilege,
  dsp.admin_option
from
  sys.dba_sys_privs    dsp,
  sys.dba_roles    dr
where
  dsp.grantee = dr.role
order by 1, 2

```

**Listing 15.2 Report showing all system privilege and admin option for users and roles.**

GRANTEE	User or Role	System Privilege	ADMIN_OPTION
ANONYMOUS	User	CREATE SESSION	NO
APEX_PUBLIC_USER	User	CREATE SESSION	NO

AQ_ADMINISTRATOR_ROLE	Role	CREATE EVALUATION CONTEXT	YES
AQ_ADMINISTRATOR_ROLE	Role	CREATE RULE	YES
AQ_ADMINISTRATOR_ROLE	Role	CREATE RULE SET	YES
AQ_ADMINISTRATOR_ROLE	Role	DEQUEUE ANY QUEUE	YES
AQ_ADMINISTRATOR_ROLE	Role	ENQUEUE ANY QUEUE	YES
AQ_ADMINISTRATOR_ROLE	Role	MANAGE ANY QUEUE	YES
BI	User	ALTER SESSION	NO
BI	User	CREATE CLUSTER	NO
BI	User	CREATE DATABASE LINK	NO
BI	User	CREATE INDEXTYPE	NO
BI	User	CREATE OPERATOR	NO
BI	User	CREATE PROCEDURE	NO
BI	User	CREATE SEQUENCE	NO
BI	User	CREATE SESSION	NO
BI	User	CREATE SYNONYM	NO
BI	User	CREATE TABLE	NO
BI	User	CREATE TRIGGER	NO
BI	User	CREATE TYPE	NO
BI	User	CREATE VIEW	NO
BI	User	UNLIMITED TABLESPACE	NO
CONNECT	Role	CREATE SESSION	NO
CTXSYS	User	ALTER SESSION	NO

....

### Object and Column Privileges Granted WITH GRANT

Similarly to the ADMIN OPTION and system privilege, you also need a query for showing all object privileges granted WITH GRANT, regardless of whether the privilege is on the entire object or on certain columns:

```

select
  grantee,
  privilege,
  table_name,
  owner,
  grantor

from
  sys.dba_tab_privs

where
  grantable = 'YES'
UNION
select
  grantee,
  privilege,
  table_name,
  owner,
  grantor

from
  sys.dba_col_privs

where
  grantable = 'YES'

```

The output shows the object privileges—even if the privilege was defined on a particular column. An example partial output is:

GRANTEE	PRIVILEGE	TABLE_NAME	OWNER	GRANTOR
CTXSYS	SELECT	ARGUMENT\$	SYS	SYS
CTXSYS	SELECT	CCOL\$	SYS	SYS
CTXSYS	SELECT	CDEF\$	SYS	SYS
CTXSYS	SELECT	COL\$	SYS	SYS
CTXSYS	SELECT	CON\$	SYS	SYS
CTXSYS	SELECT	DBA_CONS_COLUMNS	SYS	SYS
CTXSYS	SELECT	DBA_DB_LINKS	SYS	SYS
CTXSYS	SELECT	DBA_INDEXTYPES	SYS	SYS
CTXSYS	SELECT	DBA_OBJECTS	SYS	SYS
CTXSYS	SELECT	DBA_ROLES	SYS	SYS
CTXSYS	SELECT	DBA_ROLE_PRIVS	SYS	SYS
....				

### Object Privileges by Database Account

The following query lists all object privileges by database account. Normally, you would exclude certain grantees and the DBA role as shown below, and you can modify the list as you see fit:

```

select
  tp.grantee,
  t.p.table_name,
  tp.owner,
  tp.privilege
from sys.dba_tab_privs tp
where
  tp.grantee not in (
    'AUDSYS',
    'CTXSYS',
    'DMSYS',
    'LBACSYS',
    'MDSYS',
    'OLAPSYS',
    'PUBLIC',
    'SYS',
    'SYSMAN',
    'SYSTEM',
    'WKSYS',
    'WMSYS')
and tp.grantee not in (
  select role from sys.dba_roles)
order by tp.grantee, tp.owner, tp.privilege

```

A part of an example output follows:

GRANTEE	TABLE_NAME	OWNER	PRIVILEGE
BI	TIMES	SH	SELECT
BI	COSTS	SH	SELECT
BI	FWEEK_PSCAT_SALES_MV	SH	SELECT

DBSNMP	DBMS_MANAGEMENT_PACKS	SYS	EXECUTE
DBSNMP	DBMS_SERVER_ALERT	SYS	EXECUTE
EXFSYS	DBMS_REGISTRY	SYS	EXECUTE
EXFSYS	DBMS_SYS_ERROR	SYS	EXECUTE
EXFSYS	DBMS_JOB	SYS	EXECUTE
EXFSYS	DBMS_CHANGE_NOTIFICATION	SYS	EXECUTE
EXFSYS	EXF\$DBMS_EXPFIL_SYSPACK	SYS	EXECUTE
EXFSYS	DBMS_LOCK	SYS	EXECUTE
EXFSYS	DBMS_SCHEDULER	SYS	EXECUTE
FLWS_030000	CTX_DDL	CTXSYS	EXECUTE
FLWS_030000	CTX_DOC	CTXSYS	EXECUTE
....			

## Object Access by PUBLIC

The following query lists all object privileges granted to PUBLIC (USER# = 1 in USER\$):

```

select
  u.name owner,
  o.name object_name,
  tp.name privilege

from
  sys.obj$ o,
  sys.user$ u,
  sys.objauth$ oa,
  sys.table_privilege_map tpm

where
  oa.obj# = o.obj# and
  u.user# = o.owner# and
  oa.privilege# = tpm.privilege and
  oa.col# is null and
  oa.grantee# = 1

order by 1, 2, 3

```

As an example, if you grant a select privilege on DEPT to PUBLIC:

```
SQL> grant select on scott.dept to PUBLIC;
Grant succeeded.
```

The result will now show this grant as follows:

OWNER	OBJECT_NAME	PRIVILEGE
...		
ORDSYS	yCbCrPositioningType294_T	EXECUTE
ORDSYS	yCbCrSubSamplingType295_T	EXECUTE
SCOTT	DEPT	SELECT
SYS	/1000e323d_DelegateInvocationHa	EXECUTE
SYS	/1000e8d1_LinkedHashMapValueIt	EXECUTE
...		

## Execute Privileges on Procedures Assigned to PUBLIC

Another important query related to PUBLIC is a query showing all execute privileges on PL/SQL procedures in SYS that are assigned to PUBLIC:

```
select distinct
  dtp.table_name "Object name",
  o.object_type "Object type",
  d_tp.grantor "Grantor"
from
  sys.dba_objects o,
  sys.dba_tab_privs dtp
where
  table_name = object_name and
  o.object_type in ('PACKAGE', 'PROCEDURE', 'FUNCTION') and
  dtp.owner = 'SYS' and
  grantee = 'PUBLIC' and
  dtp.privilege = 'EXECUTE'
order by 1, 2, 3
```

A sample output of the query:

Object name	Object type	Grantor
BLASTN_ALIGN	FUNCTION	SYS
BLASTN_COMPRESS	FUNCTION	SYS
BLASTN_MATCH	FUNCTION	SYS
BLASTP_ALIGN	FUNCTION	SYS
BLASTP_MATCH	FUNCTION	SYS
CLIENT_IP_ADDRESS	FUNCTION	SYS
CUBE_TABLE	FUNCTION	SYS
DATABASE_NAME	FUNCTION	SYS
DBJ_SHORT_NAME	FUNCTION	SYS
DBMS_ADDM	PACKAGE	SYS
DBMS_ADVISOR	PACKAGE	SYS

## Accounts Queries

In addition to privilege reports, you will often be asked to produce reports showing accounts that have special privileges. Here are the common reports that auditors will typically ask for

1. Accounts with SYSDBA and SYSOPER privileges—the query:

```
se lect
pw      f.username "Username",
        pwf.sysdba "Is sysdba",
        pwf.sysoper "Is sysoper",
        decode(u.password, 'EXTERNAL', 'TRUE', 'FALSE') "Is external
pa ssword"

fr om
sys      .v_$pwfile_users pwf,
sys      .dba_users u
```

```

where
    pwf.username = u.username
order by 1

```

As an example, create a new user and grant it one of these privileges:

```

SQL> create user ronb identified by "0oknhy&";
Use      r created.

SQL> grant sysoper to ronb;
Gr       ant succeeded.

```

Now run the query; you will see the additional account:

	Username	Is sysdba	Is sysoper	Is external password
RO	NB	FALSE	TRUE	FALSE
S	YS	TRUE	TRUE	FALSE

2. Accounts with BECOME USER privileges—the query:

```

select
gr      antee,
pr      ivilege,
      admin_option
from
sys      .dba_sys_privs
where
      privilege = 'BECOME USER'
order by 1

```

Sample result:

	GRANTEE P	RIVILEGE	ADMIN_OPTION
D	BA BECOME USER	YES	
I	MP_FULL_DATABASE	BECOME USER	NO
S	YS BECOME USER	NO	

3. Accounts with ALTER SYSTEM and ALTER SESSION privileges—the query:

```

select
gr      antee,
pr      ivilege,
      admin_option
from
sys      .dba_sys_privs,
(s      elect
'          User' u_or_x,
u          sername name

```

```

fr      om
       sys.dba_users
UN      ION
se      lect
       'Role' u_or_r,
       role name
fr      om
       sys.dba_roles) user_or_role

where
       user_or_role.name = grantee and
       privilege in ('ALTER SYSTEM', 'ALTER SESSION')
order by 1, 2

```

Sample result:

GRANTEE	PRIVILEGE	ADMIN_OPTION
BI	ALTER SESSION	NO
CTXSYS	ALTER SESSION	NO
DBA	ALTER SESSION	YES
DBA	ALTER SYSTEM	YES
FLows_030000	ALTER SESSION	NO
FLows_030000	ALTER SYSTEM	NO
HR	ALTER SESSION	NO
IX	ALTER SESSION	NO
OWB\$CLIENT	ALTER SESSION	NO
OWBSYS	ALTER SESSION	YES
RECOVERY_CATALOG_OWNER	ALTER SESSION	NO
SH	ALTER SESSION	NO
SYS	ALTER SESSION	NO
SYS	ALTER SYSTEM	NO
SYSMAN	ALTER SESSION	NO
XDB	ALTER SESSION	NO

Finally, the following query lists all roles granted to users or roles:

```

select
  grants.grantee "Grantee",
  user_or_role.u_or_r "User or Role",
  grants.granted_role "Granted Role"

from
  (select distinct
    connect_by_root grantee "GRANTEE",
    granted_role
  from
    sys.dba_role_privs
  connect by prior granted_role = grantee
  order by 1,2) grants,
  (select
    'User' u_or_r,
    username name
  from
    sys.dba_users
  UNION
  select
    'Role' u_or_r,
    role_name name
  from
    sys.dba_roles)

```

```

se lect
  ' Role' u_or_r,
    role

fr om
  sys .dba_roles) user_or_role

where
  grants.grantee = user_or_role.name
order by 1,2,3

```

A sample result is:

Grantee	User or Role	Granted Role
BI	User	RESOURCE
CTXSYS	User	CTXAPP
CTXSYS	User	RESOURCE
DATAPUMP_EXP_FULL_DATABASE	Role	EXECUTE_CATALOG_ROLE
DATAPUMP_EXP_FULL_DATABASE	Role	EXP_FULL_DATABASE
DATAPUMP_EXP_FULL_DATABASE	Role	HS_ADMIN_ROLE
DATAPUMP_IMP_FULL_DATABASE	Role	SELECT_CATALOG_ROLE
DATAPUMP_IMP_FULL_DATABASE	Role	EXECUTE_CATALOG_ROLE
DATAPUMP_IMP_FULL_DATABASE	Role	EXP_FULL_DATABASE
DATAPUMP_IMP_FULL_DATABASE	Role	HS_ADMIN_ROLE
DATAPUMP_IMP_FULL_DATABASE	Role	IMP_FULL_DATABASE
DATAPUMP_IMP_FULL_DATABASE	Role	SELECT_CATALOG_ROLE
DBA	Role	DATAPUMP_EXP_FULL_DATABASE
DBA	Role	DATAPUMP_IMP_FULL_DATABASE
DBA	Role	DELETE_CATALOG_ROLE
DBA	Role	EXECUTE_CATALOG_ROLE
...		

## 15.8 Discussion—SQL92\_SECURITY

In HOWTO 15.2 you learned about how O7\_DICTIONARY\_ACCESSIBILITY controls restrictions on system privileges and in HOWTO 15.3 you learned how MAX\_ENABLED\_ROLES limits the number of roles that a user can enable. Additional initialization parameters that affect the privilege model include

- SQL92\_SECURITY specifies whether or not a user must be granted the SELECT privilege to execute UPDATE or DELETE statements. The default is FALSE and you should not change that.
- OS\_ROLES allows you to configure Oracle to use the operating system for determining which roles and user belongs to. The default is FALSE.
- REMOTE\_OS\_ROLES specifies whether or not OS roles are allowed for remote clients. The default is FALSE and you should not change that.

The issue that SQL92\_SECURITY comes to re solve is the fact that a user with DELETE or UPDATE privileges can determine the value in a row through a fairly simple procedure that involves rollbacks. They can do this even if they have not been granted SELECT privileges.

Let's look at a simple example. Suppose that the EMP table contains these records:

```
SQL> select * from scott.emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	830		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1630	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1280	500	30
7566	JONES	MANAGER	7839	02-APR-81	3005		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1280	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2880		30
7782	CLARK	MANAGER	7839	09-JUN-81	2480		10
7788	SCOTT	ANALYST	7566	19-APR-87	3030		20
7839	KING	PRESIDENT		17-NOV-81	5030		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1530	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1130		20

Suppose also that a user has DELETE privileges but no SELECT privileges. This user can determine the salary value for BLAKE using the following sequence:

```
SQL> delete from scott.emp where ename='BLAKE' and sal > 4000;
0 rows deleted.
SQL> -- therefore, we know it's less than 4000

SQL> delete from scott.emp where ename='BLAKE' and sal > 2000;
1 row deleted.
SQL> -- now we know it's more than 2000 but we need to rollback so as not to affect
anything
SQL> rollback;
Rollback complete.

SQL> delete from scott.emp where ename='BLAKE' and sal > 3000;
0 rows deleted.
SQL> -- so it's between 2000 and 3000

SQL> delete from scott.emp where ename='BLAKE' and sal > 2500;
1 row deleted.
SQL> rollback;
Rollback complete.
SQL> -- so it's between 2500 and 3000
SQL> -- ....
```

Every iteration halves the range in which the salary can be. In less than  $\log_2(N)$  attempts you will zero-in on precisely the value—even though you did not have privileges to know this value. Hence, keep SQL92\_SECURITY set to FALSE.

## *Chapter 16*

---

# Virtual Private Database

---

Oracle Virtual Private Database (VPD) is a security framework that implements Fine-Grained Access Control (FGAC) for tables, views, and synonyms. It was called FGAC when it first came out in Oracle 8i as part of the row-level security (RLS) feature. It was later renamed to VPD because using VPD's RLS each user will see a different virtual data set—only the data that they are entitled to see.

With VPD you attach security policies to database objects that are automatically enforced whenever a user accesses these objects. It is not application specific and is enforced by the database so there is no way to bypass this security layer. Policies are enforced when you read (SELECT) data or write data (INSERT, UPDATE, DELETE). VPD policies can enforce policies both at row-level and at column-level so you can control which rows a user has access to as well as which columns a user has access to. As an example, you can create a policy that ensures that each sales manager and each sales rep can only look at leads within their territory and you can also create a policy that enforces a policy that ensures that the sales reps can see the leads but not the estimated size of the deal.

### **16.1 HOWTO Use VPD Policies to Limit Access to Rows**

VPD allows you to define a granular security policy by which each user gets a result set which is filtered to precisely the rows they are entitled to see. Rather than modifying result sets, VPD is based on query rewriting. What VPD does to ensure that you can only use the data you should is add conditions to WHERE clauses ensuring that you cannot read certain data and cannot write certain data. The security function that is at the heart of VPD (and that you need to implement) returns a predicate that is added to the WHERE clauses. It is your responsibility to structure the predicate such that it correctly limits the rows that a user can access.

Let's look at the simplest example for row filtering. In this example, you'll use the SCOTT.EMP table and the fact that every employee in it has a value for DEPT. Assume for now that each database user has a department to which they belong to and that this value is kept within a context variable that you can retrieve using SYS\_CONTEXT('HR\_CONTEXT','DEPT'). In this case the predicate function for the policy is

```
SQL> CREATE OR REPLACE
  2 FUNCTION rls_dept (obj_owner IN VARCHAR2, obj_name IN VARCHAR2)
  3 RETURN  VARCHAR2
  4 AS
  5 deptno  NUMBER;
  6 predicate  VARCHAR2(200);
  7 BEGIN
  8 deptno  := SYS_CONTEXT('HR_CONTEXT', 'DEPT');
  9 IF deptno IS NULL THEN
10 predicate  := '1=2';
11 ELSE
12 predicate  := 'deptno = '||deptno;
13 END IF;
14 RETURN  (predicate);
15 END rls_dept;
16 /
```

Function created.

The RLS\_DEPT function returns a predicate that is attached to a WHERE clause. The function takes as arguments the schema and the object on which it operates as inputs and must return a VARCHAR2 that is a valid predicate that can be appended to a WHERE clause. Although in the examples you'll see below the predicate function is almost always attached to a single object, this does not have to be the case—which is the reason for the input arguments. If the user session does not have a department within the context then the predicate returned is 1=2. Otherwise, the returned predicate is DEPTNO = <department number in the context>.

Once you have the predicate function you can go ahead and generate the VPD policy:

```
SQL> BEGIN
  2 DBMS_RLS.ADD_POLICY(
  3 object_schema=>'SCOTT',
  4 object_name=>'EMP',
  5 policy_name=>'restrict_dept_policy',
  6 function_schema=>'SCOTT',
  7 policy_function=>'rls_dept'
  8 );
  9 END;
10 /
```

PL/SQL procedure successfully completed.

You need to have been granted privileges to the DBMS\_RLS package to add this policy. The policy, named RESTRICT\_DEPT\_POLICY, is attached to the SCOTT.EMP table and states that the predicate function is SCOTT.RLS\_DEPT. Every time Oracle sees a Data Manipulation Language (DML) statement related to SCOTT.EMP it will invoke the predicate function and rewrite the query using the predicate—in effect limiting the read or write performed by the user. The function can be defined within a package in which case you should use <package name>.<function name> when creating the policy.

The last thing you need to create is a context where the user maintains their department number and a procedure by which the session context is set:

```
SQL> create context HR_CONTEXT using set_hr_context;
Context created.
```

```
SQL> create or replace procedure set_hr_context
  2 (
  3   deptno    in number
  4 )
  5 is
  6 begin
  7   dbms_session.set_context('HR_CONTEXT', 'DEPT', deptno);
  8 end;
  9 /
```

Procedure created.

Let's test the policy. If you log onto the database without any context your queries will all come back empty because they are all rewritten with a 1=2 predicate:

```
SQL> select * from emp;
no rows selected
```

If you set your department context to DEPT 20 then you will be able to see data—but only for that department—VPD is limiting your access for all your DML statements:

```
SQL> exec set_hr_context(20)
PL/SQL procedure successfully completed.

SQL> select * from emp;

EMPNO ENAME   JOB          MGR HIREDATE      SAL       COMM     DEPTNO
----- -----  -----  -----  -----  -----  -----
 7566 JONES    MANAGER    7839 03-APR-81  3599.75
 7788 SCOTT    ANALYST   7566 20-APR-87  3630
 7876 ADAMS    CLERK     7788 24-MAY-87  1331
 7902 FORD     ANALYST   7566 04-DEC-81  3630
 4444 RON      DEV        7902 02-NOV-02  2000      100      20
5 rows selected.

SQL> update emp set sal=sal*1.1 where empno=7934;
0 rows updated.

SQL> update emp set sal=sal*1.1 where empno=4444;
1 row updated.
```

The update to EMPNO=7934 failed because that employee is not within department 20. The employee with EMPNO=4444 is in that department.

## Policy Qualifiers

The ADD\_POLICY procedure has many qualifiers that you can use to refine the way your policy works. An example of a call using more qualifiers follows:

```
SQL> BEGIN
  2   DBMS_RLS.ADD_POLICY(
  3     object_schema=>'SCOTT',
```

```

4   object_name=>'EMP',
5   policy_name=>'restrict_dept_policy',
6   function_schema=>'SCOTT',
7   policy_function=>'rls_dept',
8   statement_types=>'INSERT,UPDATE,DELETE,SELECT',
9   update_check=>TRUE,
10 enable=>TRUE,
11   static_policy=>FALSE,
12   long_predicate=>FALSE
13 );
14 END;
15 /

```

PL/SQL procedure successfully completed.

- STATEMENT\_TYPES allows you to control which statement types the policy will be applied to and therefore which queries can be rewritten. For example, if you only need to control changes to data but want to allow users to SELECT all data you can use INSERT, UPDATE, DELETE as the value for STATEMENT\_TYPES. Possible values for this qualifier are INSERT, UPDATE, DELETE, INDEX (CREATE INDEX and ALTER INDEX), SELECT. The default is all but INDEX. INDEX was added because a user who has index privileges can gain access to the data even without having SELECT privileges. You can create a function-based index on the table using your own user-defined function. The function gets values from the table when the index is created. Oracle will go through all rows and call the function for all rows passing in the (possibly sensitive) data.
- STATIC\_POLICY is used to tell Oracle that the policy predicate is static and does not need to be computed for every query. By default this is set to FALSE. You'll learn more about this qualifier and policy types in HOWTO 16.5.
- LONG\_PREDICATE—if you need your predicate function to return more than 4000 bytes then you should set this qualifier to TRUE. Although there are special cases, if you have such long predicates you should review your policy and see whether you can simplify your rules or use policy groups as you'll see in HOWTO 16.4.
- UPDATE\_CHECK is relevant to INSERTs and UPDATEs. If you want your policy to be checked for both before and after values, set this value to TRUE.

UPDATE\_CHECK deserves an example. If you use the policy definition from the previous section:

```

SQL> begin
2   dbms_rls.add_policy(
3     object_schema=>'SCOTT',
4     object_name=>'EMP',
5     policy_name=>'a',
6     function_schema=>'SCOTT',
7     policy_function=>'rls_dept',
8   );
9 end;
10 /

```

PL/SQL procedure successfully completed.

You will be allowed to insert data into a department to which you do not belong:

```
SQL> exec set_hr_context(20)
PL/SQL procedure successfully completed.
SQL> insert into emp values(5556,'RON1','DEV',7902,'02-NOV-02',2000,100,10)
1 row created.
```

From a security standpoint this is baffling. If you should not be able to view or update data in any department other than 20, how can you be allowed to insert a new employee for that department? The same is true for UPDATE when you consider the after value. If you try to update a record with a department other than 20 you will not be allowed to do so. But if you update an employee in department 20 and move them to belong to department 10 you will be allowed to do so. To set your security policy to check both the before and after conditions:

```
SQL> begin
 2   dbms_rls.add_policy(
 3     object_schema=>'SCOTT',
 4     object_name=>'EMP',
 5     policy_name=>'b',
 6     function_schema=>'SCOTT',
 7     policy_function=>'rls_dept',
 8     update_check=>TRUE
 9   );
10 end;
11 /
PL/SQL procedure successfully completed.
```

You will no longer be able to insert and update records that affect employees which end up in a department other than your own:

```
SQL> exec set_hr_context(20)
PL/SQL procedure successfully completed.
SQL> insert into emp values(5557,'RON1','DEV',7902,'02-NOV-02',2000,100,10);
insert into emp values(5557,'RON1','DEV',7902,'02-NOV-02',2000,100,10)
*
ERROR at line 1:
ORA-28115: policy with check option violation
```

### Four Things to Remember about Limiting Access to Rows

1. VPD implements filtering using query rewriting.
2. You control the predicates used for rewriting the queries using policy functions.
3. A policy is an association between a policy function, a database object and a set of qualifiers that affect how the policy is enforced.
4. Use UPDATE\_CHECK to check whether there is a policy violation using both before and after values in insert and update operations.

## 16.2 HOWTO Use VPD Policies to Limit Access to Sensitive Column Data

In some cases an entire table may be sensitive. In others, just certain columns hold sensitive data. VPD allows you to fine-tune your policies so that the query is rewritten only if you access the sensitive data.

Rather than defining a policy as:

```
SQL> BEGIN
 2   DBMS_RLS.ADD_POLICY(
 3     object_schema=>'SCOTT',
 4     object_name=>'EMP',
 5     policy_name=>'restrict_dept_policy',
 6     function_schema=>'SCOTT',
 7     policy_function=>'rls_dept'
 8   );
 9   END;
10  /
```

Use a qualifier called security-relevant columns:

```
SQL> begin
 2  dbms_rls.drop_policy('scott','emp','restrict_dept_policy');
 3 end;
 4 /
SQL> BEGIN
 2   DBMS_RLS.ADD_POLICY(
 3     object_schema=>'SCOTT',
 4     object_name=>'EMP',
 5     policy_name=>'restrict_dept_policy',
 6     function_schema=>'SCOTT',
 7     policy_function=>'rls_dept',
 8     sec_relevant_cols=>'SAL'
 9   );
10  END;
11  /
```

What you're telling VPD is that it needs to enforce access to the SAL column only. Access to the EMP table that does not use the SAL column will not rewrite the query:

```
SQL> select ename,deptno from scott.emp;
ENAME DEPTNO
-----
Allen 30
Ward 30
Jones 20
Martin 30
Blake 30
Clark 10
Scott 20
King 10
Turner 30
Adams 20
```

```

James 30
Ford 20
Miller 10
Ron 20
14 rows selected.

```

If you try to access the SAL column with no department within your context you will not be able to:

```

SQL> select ename,deptno,sal from scott.emp;
no rows selected

```

This is true also when you use wildcards:

```

SQL> select * from scott.emp;
no rows selected

```

Once you set the context, the query is rewritten to add the department predicate and you can view salary information for your department:

```

SQL> exec set_hr_context(20);
PL/SQL procedure successfully completed.
SQL> select ename,deptno,sal from scott.emp;

ENAME          DEPTNO      SAL
-----  -----  -----
Jones           20    3599.75
Scott           20     3630
Adams           20     1331
Ford            20     3630
Ron             20     2000

5 rows selected.

```

#### Two Things to Remember about Limiting Access to Sensitive Columns

1. Use the sec\_relevant\_cols qualifier to define which columns should trigger the query rewriting process.
2. You can apply column-level VPD policies to tables and views but not to synonyms.

### 16.3 HOWTO Use VPD Policies to Hide Sensitive Column Data

The default interpretation of VPD column-level security is the same as RLS—the query is rewritten to ensure that the result set only includes data that the user is entitled to see. It does so by filtering out complete rows. On the other hand, if you send a query that does not ask for the sensitive columns you will be able to see all rows.

VPD has an alternative way of filtering out the data. Using this alternative, your result sets will always come back with all rows but the data that you are not entitled to see will be “scrubbed out.”

Rather than using a policy such as

```
SQL> BEGIN
 2   DBMS_RLS.ADD_POLICY(
 3     object_schema=>'SCOTT',
 4     object_name=>'EMP',
 5     policy_name=>'restrict_dept_policy',
 6     function_schema=>'SCOTT',
 7     policy_function=>'rls_dept',
 8     sec_relevant_cols=>'SAL'
 9   );
10 END;
11 /
```

Use the SEC\_RELEVANT\_COLS\_OPTS qualifier:

```
SQL> begin
 2 dbms_rls.drop_policy('scott','emp','restrict_dept_policy');
 3 end;
 4 /
SQL> BEGIN
 2   DBMS_RLS.ADD_POLICY(
 3     object_schema=>'SCOTT',
 4     object_name=>'EMP',
 5     policy_name=>'restrict_dept_policy',
 6     function_schema=>'SCOTT',
 7     policy_function=>'rls_dept',
 8     sec_relevant_cols=>'SAL',
 9     sec_relevant_cols_opt => dbms_rls.ALL_ROWS
10   );
11 END;
12 /
```

At this point you will not see sensitive data to which you have no privileges but you will see all rows. For example, if you select from the table before having a department as a context you will see no salary information at all:

```
SQL> select * from emp;
    EMPNO ENAME      JOB              MGR HIREDATE        SAL       COMM     DEPTNO
----- -----  -----
 7499  ALLEN      SALESMAN        7698 21-FEB-81      300
 7521  WARD       SALESMAN        7698 23-FEB-81      500      30
 7566  JONES      MANAGER        7839 03-APR-81
 7788  SCOTT      ANALYST        7566 20-APR-87
 7839  KING       PRESIDENT      18-NOV-81
 7844  TURNER     SALESMAN        7698 09-SEP-81      0       30
 7876  ADAMS      CLERK          7788 24-MAY-87
 7900  JAMES      CLERK          7698 04-DEC-81
```

7902 FORD ANALYST	7566 04-DEC-81	20
7934 MILLER CLERK	7782 24-JAN-82	10
4444 RON DEV	7902 02-NOV-02	100
		20

14 rows selected.

Once you set the context you will see salaries but only salaries that you are entitled to see. All other salaries are sanitized:

```
SQL> exec set_hr_context(20);
PL/SQL procedure successfully completed.
```

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	21-FEB-81	300	30	
7521	WARD	SALESMAN	7698	23-FEB-81	500	30	
7566	JONES	MANAGER	7839	03-APR-81	3599.75		20
7788	SCOTT	ANALYST	7566	20-APR-87	3630		20
7839	KING	PRESIDENT		18-NOV-81			10
7844	TURNER	SALESMAN	7698	09-SEP-81	0	30	
7876	ADAMS	CLERK	7788	24-MAY-87	1331		20
7900	JAMES	CLERK	7698	04-DEC-81			30
7902	FORD	ANALYST	7566	04-DEC-81	3630		20
7934	MILLER	CLERK	7782	24-JAN-82			10
4444	RON	DEV	7902	02-NOV-02	2000	100	20

14 rows selected.

There are a few limitations on this facility. Column-level redaction only works for SELECT statements and does not control the other DML operations. The predicate function must return a predicate based on simple Boolean expressions. However, UPDATE AS SELECT will update only the columns that the user is entitled to see. Finally, remember that what is actually returned within the redacted result set are NULLs so if you have application code that breaks if it can't handle NULLs, stick with the policy from the previous HOWTO.

#### Two Things to Remember about Sanitizing Sensitive Columns

1. Use the sec\_relevant\_cols\_opts qualifier to implement sensitive data redaction, ensuring that users can only see the data that are authorized to see by the VPD policy function.
2. You can use column data masking only for SELECT statements.

## 16.4 HOWTO Use Policy Groups

If you use VPD for granular access control with many applications or many user profiles, you may end up with many policies. Things can get pretty complicated—especially if you have many applications that rely on the same base tables and the security policies governing data access differ from one application to another. In this case you may need to apply different sets of policies to the same

base table depending on the context. You can implement all this logic into the policy predicate function—but that's the wrong way to do it and you'll end up with a huge mess on your hands.

The right way is to use policy groups. A policy group is a set of policies that are managed through an application context. The application context in this case is called the Driving Context. The value of the application context determines which of the policy groups is in effect. Each policy group can have many policies. Rather than apply ALL policies on the base table, the driving context applies only to those policies in one policy group—so you end up with behavior that is specific to the application you are in.

Let's look at an example. In this example you're going to create two policy groups—one for an HR application and one for a finance application. Both groups will have a policy that relates to the EMP table but each one behaves differently. The HR application enforces a policy in which each manager can only see the data for their department. The finance application is used by an analyst who needs to analyze whether salaries and commissions are on par with the market—so they need to be able to see all salary data, need to be able to compute the average salary per department, but should not be able to know which employee makes how much. In the HR application you'll use a policy that filters rows and in the finance application you'll use a policy that redacts columns. You're not going to touch the policy function—you'll use policy groups instead.

Start out by creating two application users:

```
SQL> create user dept_manager identified by "0okmnhy6";
User created.

SQL> grant connect to dept_manager;
Grant succeeded.

SQL> create user fin_analyst identified by "7ygvcde3";
User created.

SQL> grant connect to fin_analyst;
Grant succeeded.
```

Now create two policy groups:

```
begin
  DBMS_RLS.CREATE_POLICY_GROUP(
    object_schema  => 'SCOTT',
    object_name    => 'EMP',
    policy_group   => 'hr_policy_group');
end;
/
begin
  DBMS_RLS.CREATE_POLICY_GROUP(
    object_schema  => 'SCOTT',
    object_name    => 'EMP',
    policy_group   => 'finance_policy_group');
end;
/
```

You will have one policy function. For the HR application you set a department context but for the financial analyst you do not need to. In the later case, the predicate will return 1=2 and the analyst will not see any of the identifying information:

```

CREATE OR REPLACE
FUNCTION rls_dept (obj_owner IN VARCHAR2, obj_name IN VARCHAR2)
RETURN VARCHAR2
AS
deptno NUMBER;
predicate VARCHAR2(200);
BEGIN
deptno := SYS_CONTEXT('APPLICATION_CONTEXT', 'DEPT');
IF deptno IS NULL THEN
predicate := '1=2';
ELSE
predicate := 'deptno = ' || deptno;
END IF;
RETURN (predicate);
END rls_dept;
/

```

Now add the grouped policies using the function on the EMP object:

```

begin
DBMS_RLS.ADD_GROUPED_POLICY(
object_schema => 'SCOTT',
object_name   => 'EMP',
policy_group  => 'hr_policy_group',
policy_name   => 'hr_policy',
function_schema => 'SECADM',
policy_function => 'rls_dept');
end;
/

begin
DBMS_RLS.ADD_GROUPED_POLICY(
object_schema => 'SCOTT',
object_name   => 'EMP',
policy_group  => 'finance_policy_group',
policy_name   => 'finance_policy',
function_schema => 'SECADM',
policy_function => 'rls_dept',
statement_types => 'SELECT',
sec_relevant_cols => 'EMPNO,ENAME,MGR,HIREDATE',
sec_relevant_cols_opt => DBMS_RLS.ALL_ROWS);
end;
/

```

Create the driving context. The driving context is an application context. In the procedure that sets the context you also set the POLICY\_GROUP attribute within the context. This is what VPD uses to decide which of the policy groups to apply:

```

create or replace context application_context using set_app_context;
create or replace procedure set_app_context(
    app in varchar2,
    deptno in number default NULL)
is
begin
dbms_session.set_context('application_context', 'DEPT', deptno);
    if app = 'FINANCE' then

```

```

dbms_session.set_context('application_context','POLICY_GROUP','FINANCE_POLICY_G
ROUP');
else
    -- HR is the default - more protective
dbms_session.set_context('application_context','POLICY_GROUP','HR_POLICY_GROUP'
);
end      if;
end;
/

```

Note that the third parameter (the value for the POLICY\_GROUP) must in uppercase. If it is not, VPD will not match it and you will get an error of the form:

```

SQL> select * from scott.emp;
select * from scott.emp
*
ERROR at line 1:
ORA-28123: Driving context 'APPLICATION_CONTEXT,POLICY_GROUP' contains invalid
group 'hr_policy_group'

```

Now associate the driving context with the table:

```

begin
DBMS_RLS.ADD_POLICY_CONTEXT(
object_schema => 'SCOTT',
object_name => 'EMP',
namespace => 'application_context',
attribute => 'POLICY_GROUP');
end;
/

```

If the driving policy is not defined or if the driving policy is NULL then VPD will run ALL policies—it is taking the most conservative and secure interpretation and applying all the possible filtering. Now grant permissions to the application-setting procedure:

```

SQL> grant execute on set_app_context to dept_manager;
Grant succeeded.

SQL> grant execute on set_app_context to fin_analyst;
Grant succeeded.

```

You're ready to test the policy groups. First, connect as HR department manager, set the department context and see that you can see only your department's data:

```

SQL> connect dept_manager
Enter password:
Connected.
SQL> select * from scott.emp;

no rows selected

```

```
SQL> exec sys.set_app_context('HR', 20);
PL/SQL procedure successfully completed.
SQL> select * from scott.emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	830		20
7566	JONES	MANAGER	7839	02-APR-81	3005		20
7788	SCOTT	ANALYST	7566	19-APR-87	3030		20
7876	ADAMS	CLERK	7788	23-MAY-87	1130		20
7902	FORD	ANALYST	7566	03-DEC-81	3030		20

Connect as a financial analyst. After setting the right application context that enables the finance group policy you will not see identifying information:

```
SQL> connect fin_analyst
Enter password:
Connected.
SQL> select * from scott.emp;

no rows selected

SQL> exec sys.set_app_context('FINANCE');
PL/SQL procedure successfully completed.
```

```
SQL> select * from scott.emp;
```

EMPNO	E JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
	CLERK			830		20
	SALESMAN			1630	300	30
	SALESMAN			1280	500	30
	MANAGER			3005		20
	SALESMAN			1280	1400	30
	MANAGER			2880		30
	MANAGER			2480		10
	ANALYST			3030		20
	PRESIDENT			5030		10
	SALESMAN			1530	0	30
	CLERK			1130		20

EMPNO	E JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
	CLERK			980		30
	ANALYST			3030		20
	CLERK			1330		10

14 rows selected.

There is one special policy group called SYS\_DEFAULT—the default policy group. Policies belonging to SYS\_DEFAULT are always active and they allow you to ensure that some policies are enforced no matter what.

### The e Things to Remember about Policy Groups

1. VPD allows you to define multiple policy groups and control which policy group is in effect using a single driving context. This allows you to manage a multiapplication environment of granular access control rules without complicating your policy functions.
2. There is a default policy group that will be applied always; additionally, policies within a policy group will be applied based on the value within the driving context.
3. Use policy groups when you have multiple applications on the same base objects, when you manage a multitenant or hosting environment, and even when you reuse a connection within an application server and need potentially distinct security behaviors.

## 16.5 HOWTO Choose a Policy Type for Optimal Performance

One of the qualifiers you learned about in HOWTO 16.1 was the STATIC\_POLICY qualifier. In all the policies that you built until now you either did not provide a value for that qualifier or you chose FALSE. A related qualifier is the policy\_type qualifier—which can be dynamic, static, or shared.

The default value for a policy is a DYNAMIC policy. In this case VPD runs the policy function every time the object is accessed. This is useful if your predicate can change dynamically but it is also the slowest because the function has to be called every time and the rewritten query may have to be reparsed. The other option (that you already saw) is a STATIC policy:

```
SQL> BEGIN
 2  DBMS_RLS.ADD_POLICY(
 3  object_schema=>'SCOTT',
 4  object_name=>'EMP',
 5  policy_name=>'restrict_dept_policy',
 6  function_schema=>'SCOTT',
 7  policy_function=>'rls_dept',
 8  statement_types=>'INSERT,UPDATE,DELETE,SELECT',
 9  update_check=>TRUE,
10  enable=>TRUE,
11  policy_type=>dbms_rls.STATIC,
12  long_predicate=>FALSE
13 );
14 END;
15 /
```

PL/SQL procedure successfully completed.

A static policy is one in which the function is not recomputed—the predicate is hashed and reused. This is much more efficient. It may sound useless but if you build your predicates right this

can very often be used to provide the same functionality as dynamic policies. For example, rather than have the policy function look like:

```
SQL> CREATE OR REPLACE
  2  FUNCTION rls_dept (obj_owner IN VARCHAR2, obj_name IN VARCHAR2)
  3  RETURN VARCHAR2
  4  AS
  5  deptno NUMBER;
  6  predicate VARCHAR2(200);
  7  BEGIN
  8  deptno := SYS_CONTEXT('HR_CONTEXT', 'DEPT');
  9  IF deptno IS NULL THEN
10  predicate := '1=2';
11 ELSE
12  predicate := 'deptno = '||deptno;
13 END IF;
14 RETURN (predicate);
15 END rls_dept;
16 /
```

Function created.

You can write a policy function that returns a predicate that does the lookup (assuming the context will always be set):

```
CREATE OR REPLACE
FUNCTION rls_dept (obj_owner IN VARCHAR2, obj_name IN VARCHAR2)
RETURN VARCHAR2
AS
deptno NUMBER;
predicate VARCHAR2(200);
BEGIN
  predicate := 'deptno = SYS_CONTEXT("HR_CONTEXT", "DEPT")';
  RETURN (predicate);
END rls_dept;
/
```

The predicate does not have to be recomputed every time. The fact that the predicate does not change does not mean that the same cursor is not going to produce a different result set—the predicate can filter the data differently based on the context.

If you need the same predicate to be used for multiple database objects use SHARED\_STATIC as the policy type—the fewer policy function calls, the better.

There is a third type of policy type that can be viewed as a middle-of-the-road choice—CONTEXT\_SENSITIVE and SHARED\_CONTEXT\_SENSITIVE. When you use static policies VPD always caches the predicates. When you use a context sensitive policy VPD will also cache the predicate but will know that it has to recompute the predicate if the application context changes. As long as there is no change in the underlying context, the predicate will not be recomputed and the value in the cache will be used. If the context changes, VPD reruns the policy function and recomputes (and caches) the new predicate. Shared context sensitive policies are used when you want the same policy function and predicate to be used for multiple database objects. Table 16.1 provides a summary of policy types.

**Table 16.1 VPD Policy Types—Best Performance First**

<i>Policy Type</i>	<i>When Does the Policy Function Execute</i>	<i>Cached per Object?</i>
SHARED_STATIC	Policy function is executed only once and the predicate is cached in the SGA	No
SHARED	Policy function is executed only once and the predicate is cached in the SGA	Yes
SHARED_CONTEXT_SENSITIVE	Policy function is executed once at parse time and every time the application context changes	No
CONTEXT_SENSITIVE	Policy function is executed the first time one of the objects is referenced in a database session and every time the application context changes. Predicates are stored in the UGA and can be shared among objects	Yes
DYNAMIC	Policy function is evaluated every time the object is accessed	Not cached

### The e Things to Remember about Policy Types

1. Static policies cache a predicate value that is computed only once.
2. Context sensitive policies use a cached predicate which is recomputed every time the underlying application context changes.
3. Dynamic policies, the default, cause the policy function to be called for every access.

## 16.6 HOWTO Review and Debug VPD Policies

VPD is a very powerful and elegant framework and you can do a lot with it. The setup is very simple—the complexity comes in building the policy functions and the predicates. VPD can be thought of as a rules processing system in which the rules are written in free-form PL/SQL.

VPD does not limit you to a single policy per database object. You can assign multiple policies to a single database object and the policy will enforce access by AND-ing together all the predicates. You can attach a policy P1 with a predicate function that returns a set of conditions C1 and a policy P2 with C2 to a table T. If you access data using `SELECT * FROM T WHERE C` the query is rewritten to `SELECT * FROM T WHERE C AND C1 AND C2`. VPD policies specify what users CANNOT see; not what they can see. The exception is policy groups—with policy groups, VPD applied all the policies belonging to one group OR all the policies that belong to another group.

Any rule-based system can become very hard to analyze and debug. This is even more so when there is no set structure to the rules. You should discipline yourself to document your policy functions and use rules that are easy to understand.

To view which policies are defined in the database use one of the dictionary views described in Table 16.2. When debugging policy functions you have two facilities at your disposal.

**Table 16.2 Views Used to Review VPD Policies**

<i>View</i>	<i>Description</i>
ALL_POLICIES	Shows all VPD policies on objects accessible to the current user
ALL_POLICY_CONTEXTS	Shows the contexts used within VPD policies defined for objects accessible to the current user
ALL_POLICY_GROUPS	Shows VPD policy groups defined for objects accessible to the current user
DBA_POLICIES	Shows all VPD policies
DBA_POLICY_CONTEXTS	Shows all contexts used within VPD policies
DBA_POLICY_GROUPS	Shows all VPD policy groups
USER_POLICIES	Shows all VPD policies on objects owned by the current user
USER_POLICY_CONTEXTS	Shows the contexts used within VPD policies defined for objects owned by the current user
USER_POLICY_GROUPS	Shows VPD policy groups defined for objects owned by the current user
V\$VPD_POLICY	Shows all the policies and predicates associated with the cursors currently in the library cache

The first is the V\$VPD\_POLICY view. When a predicate is cached the WHERE clause is added to V\$VPD\_POLICY:

```
SQL> select policy, predicate from v$vpd_policy;
POLICY PREDICATE
-----
RESTRICT_DEPT_POLICY 1=2
RESTRICT_DEPT_POLICY 1=2
RESTRICT_DEPT_POLICY          deptno = 20
RESTRICT_DEPT_POLICY 1=2

4 rows selected.
```

When a policy is dropped, all records for this policy are cleared:

```
SQL> exec dbms_rls.drop_policy('scott','emp','restrict_dept_policy');
PL/SQL procedure successfully completed.
SQL> select policy, predicate from v$vpd_policy;
no rows selected
```

The second facility is to use SQL traces. SQL traces show you a lot of information including the real query that was executed by the database after VPD rewrote it. To start a trace:

```
alter session set sql_trace=true;
SQL> alter session set sql_trace=true;
Session altered.
```

At this point, access data before and after setting the department context:

```
SQL> select * from emp;
no rows selected
SQL> exec set_hr_context(20)
PL/SQL procedure successfully completed.

SQL> select * from emp;

EMPNO ENAME      JOB          MGR HIREDATE      SAL     COMM     DEPTNO
----- -----      ----          --  -----      --      --      -----
7566  JONES       MANAGER      7839 03-APR-81   3599.75    20
7788  SCOTT       ANALYST     7566 20-APR-87   3630      20
7876  ADAMS       CLERK        7788 24-MAY-87   1331      20
7902  FORD         ANALYST     7566 04-DEC-81   3630      20
4444  RON          DEV          7902 02-NOV-02   2000      100      20
5 rows selected.
```

Listing 16.1 includes parts of the resulting trace file. The first query was rewritten with the `(1=2)` predicate and the second query with the `(deptno = 20)` predicate. This debugging information is critical to understand why results are not what you may expect when you start using VPD.

### **Additional Pitfalls to Avoid**

VPD is flexible because it is based on arbitrary PL/SQL code. VPD does not have a validation mechanism in place to check what the function will produce and how it will behave at runtime. Therefore, invest time in reviewing and debugging the policy functions and what they will produce.

One example problem that you should watch out for is recursion. Be careful when selecting from a table or view which has a VPD policy attached to it from within a policy function. This can lead to an infinite loop. The trivial case is when the policy function selects from the table on which it is defined, but a more realistic scenario can occur if a policy function F1 calls a procedure or function P2 which calls P3 which then selects from the table with the VPD policy that uses F1. This will cause the session to go into an infinite loop due to the recursive definition.

For better performance analyze the resulting predicates and ensure that you have appropriate indexes on the tables which have a VPD policy. Because the queries hitting the database will be different than what the users send, you may need to add indexes to help these queries perform faster.

```
=====
PARSING IN CURSOR #4 len=38 dep=1 uid=54 oct=47 lid=54 tim=318108481691
hv=4008088154 ad='25996718'
begin :con := RLS_DEPT(:sn, :on); end;
END OF STMT
PARSE #4:c=0,e=32,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,tim=318108481687
EXEC #4:c=0,e=351,p=0,cr=0,cu=0,mis=0,r=1,dep=1,og=1,tim=318108511629
-----
Logon user      : SCOTT
Table/View     : SCOTT.EMP
Policy name    : RESTRICT_DEPT_POLICY
Policy function: SCOTT.RLS_DEPT
RLS view       :
SELECT "EMPNO", "ENAME", "JOB", "MGR", "HIREDATE", "SAL", "COMM", "DEPTNO" FROM
"SCOTT"."EMP" "EMP" WHERE (1=2)
=====
...
=====
PARSING IN CURSOR #5 len=32 dep=0 uid=54 oct=47 lid=54 tim=318160305193
hv=2188503246 ad='25991a98'
BEGIN set_hr_context(20); END;
END OF STMT
PARSE #5:c=0,e=85,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=318160305184
EXEC #5:c=0,e=216,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=1,tim=318160305564
=====
...
=====
PARSING IN CURSOR #6 len=38 dep=1 uid=54 oct=47 lid=54 tim=318164405511
hv=4008088154 ad='25996718'
begin :con := RLS_DEPT(:sn, :on); end;
END OF STMT
PARSE #6:c=0,e=29,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,tim=318164405507
EXEC #6:c=0,e=105,p=0,cr=0,cu=0,mis=0,r=1,dep=1,og=1,tim=318164405691
-----
Logon user      : SCOTT
Table/View     : SCOTT.EMP
Policy name    : RESTRICT_DEPT_POLICY
Policy function: SCOTT.RLS_DEPT
RLS view       :
SELECT "EMPNO", "ENAME", "JOB", "MGR", "HIREDATE", "SAL", "COMM", "DEPTNO" FROM
"SCOTT"."EMP" "EMP" WHERE (deptno = 20)
=====
```

---

#### **Listing 16.1 Trace information showing resulting query after VPD rewrite.**

### **Exemptions**

Not all users are affected by VPD policies. This is mandated because it would be difficult to ensure that you are backing up all your data if some rows are filtered out. It could also be a problem if you did write a recursive policy function that gets called for all users—potentially not allowing any user to logon.

By default, all users with SYSDBA privileges are exempt from VPD policies. In addition, any user who has been granted the EXEMPT ACCESS POLICY privilege will not be affected by VPD policies. To report upon all users exempt from VPD policies:

```
SQL> grant exempt access policy to secadm;
Grant succeeded.

SQL> select grantee from dba_sys_privs where privilege='EXEMPT ACCESS POLICY';
GRANTEE
-----
SECADM
```

### Six Things to Remember about Debugging VPD Policies

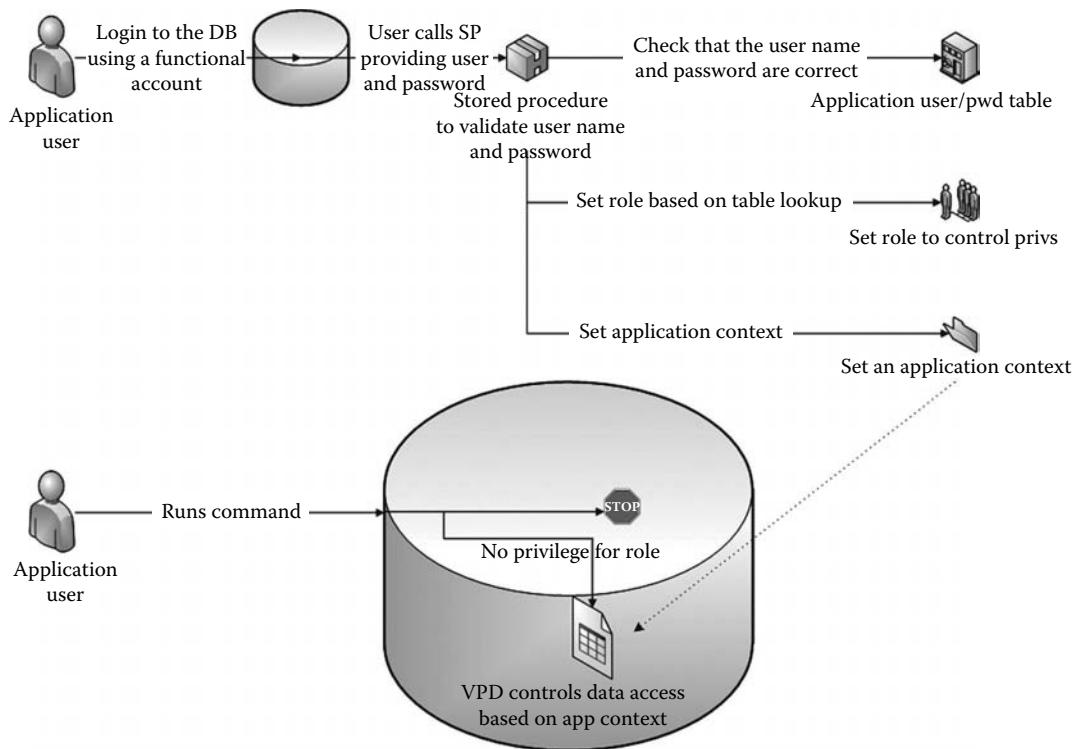
1. Keep VPD policies simple. If you have many policies with complex conditions it will become increasingly harder for you to predict what the resulting behavior is.
2. Use the V\$VPD\_POLICY view to review your resulting predicates.
3. Use traces for detailed debugging but remember to always turn the traces off to avoid impacting the system in production.
4. Don't use recursive definitions within VPD policies where procedures or functions called by a VPD policy function (directly or indirectly) select from the same VPD-enforced table or view.
5. Add indexes to accommodate re-written queries based on what you see in V\$VPD\_POLICY.
6. You should periodically audit which users are exempt from VPD policies.

## 16.7 Discussion—Using Secure Application Roles and VPD

In this section you'll build a small end-to-end example that uses both secure application roles (from the previous chapter) and VPD policies to implement granular access control. This sample can also be implemented completely using VPD but it is illustrative to use both of these techniques together because they are often used in tandem in real life.

This example has three parts. The first part is a set of secure application roles. These roles encapsulate privileges to tables. Through secure application roles you will control who can update data and who can select data. The second part is a set of VPD policies. The VPD policies control which data a user can work on. The third part is a stored procedure through which an application user logs on. This procedure implements the application-level logon and is called by the user after they logon to the database. It can also be used when you have an application server that serves multiple application users. The stored procedure first validates the user's username and password. If it is correct, it looks up which role the user belongs to and sets the secure application role—thereby enforcing the privileges the user should be working with. The procedure also sets an application context so that each access is limited through a VPD policy. The scheme is shown in Figure 16.1.

Before you delve into defining the VPD policy and the secure application role let's make a few assumptions. Let's assume that you have a table with four columns—a username, and password, which department the user belongs to and a role name. This table will be used by the logon procedure to validate the password and look up which role to set. Let's start implementing:



**Figure 16.1 Sample application utilizing VPD and secure application roles.**

Create the package with the procedures that will set the application context values:

```

create package app_context_pkg as
  procedure set_app_role(role_val in varchar2);
  procedure set_app_dept(dept_val in number);
end;
/
create package body app_context_pkg as
  procedure set_app_role(role_val in varchar2) is
begin
  -- set the role
  dbms_session.set_role(role_val);
  -- also set a role into the context
  -- this will allow VPD to mask data for employees
  dbms_session.set_context(
  namespace=>'app_context',
  attribute=>'role',
  value=>role_val);
end;
procedure set_app_dept(dept_val in number) is
begin
  dbms_session.set_context(
  namespace=>'app_context',
  
```

```

attribute=>'dept',
value=>dept_val);
en d;
end;
/

```

Create the application context and associate it with the package:

```
SQL> create context app_context using app_context_pkg;
Context created.
```

Create the secure application roles and grant the appropriate privileges to these roles:

```

SQL> create role manager identified using app_context_pkg;
Role created.

SQL> create role employee identified using app_context_pkg;
Role created.

SQL> create role app_developer identified using app_context_pkg;
Role created.

SQL> grant select on scott.emp to employee;
Grant succeeded.

SQL> grant select, insert, update, delete on scott.emp to manager;
Grant succeeded.

SQL> grant index on scott.emp to app_developer;
Grant succeeded.

```

Create the functional account and grant the roles. When a user logs on using this account by default none of these roles are granted:

```

SQL> create user APPS identified by appserverpwd;
User created.

SQL> grant create session, manager, employee, app_developer to APPS;
Grant succeeded.

SQL> alter user APPS default role none;
User altered.

```

Next you have to write the validate\_login procedure. That procedure needs to do the following:

1. Look up and validate the username and password
2. Get the deptno from the lookup table and call set\_app\_dept
3. Get the role from the lookup table and call set\_app\_role

Build two policy functions. The first function ensures that a manager or employee can only view data within their department and the second ensures that only managers see salary data. The second function returns a NULL predicate for managers—which in effect means no filtering:

```
SQL> CREATE OR REPLACE
2  FUNCTION rls_dept (obj_owner IN VARCHAR2, obj_name IN VARCHAR2)
3  RETURN VARCHAR2
```

```

4   AS
5   deptno  NUMBER;
6  predicate  VARCHAR2(200);
7  BEGIN
8  deptno  := SYS_CONTEXT('app_context','dept');
9  IF deptno IS NULL THEN
10  predicate  := '1=2';
11 ELSE
12  predicate  := 'deptno = '||deptno;
13 END IF;
14 RETURN  (predicate);
15 END rls_dept;
16 /

```

Function created.

```

SQL> CREATE OR REPLACE
2  FUNCTION rls_role (obj_owner IN VARCHAR2, obj_name IN VARCHAR2)
3  RETURN VARCHAR2
4  AS
5  app_role  VARCHAR2(20);
6  predicate  VARCHAR2(200);
7  BEGIN
8  app_role  := SYS_CONTEXT('app_context','role');
9  IF app_role = 'MANAGER' THEN
10  predicate  := NULL;
11 ELSE
12  predicate  := '1=2';
13 END IF;
14 RETURN  (predicate);
15 END rls_role;
16 /

```

Function created.

Finally, set up the VPD policies—one for enforcing access based on department and one for redacting data for an employee (versus a manager):

```

SQL> BEGIN
2  DBMS_RLS.ADD_POLICY(
3  object_schema=>'SCOTT',
4  object_name=>'EMP',
5  policy_name=>'restrict_dept_policy',
6  function_schema=>'SCOTT',
7  policy_function=>'rls_dept'
8 );
9 END;
10 /

```

PL/SQL procedure successfully completed.

```

SQL> BEGIN
2  DBMS_RLS.ADD_POLICY(
3  object_schema=>'SCOTT',
4  object_name=>'EMP',
5  policy_name=>'restrict_sal_policy',
6  function_schema=>'SCOTT',
7  policy_function=>'rls_role',
8  sec_relevant_cols=>'SAL',
9  sec_relevant_cols_opt  => dbms_rls.ALL_ROWS

```

```

10  );
11 END;
12 /
PL/SQL procedure successfully completed.

```

You're done. If you log on as the manager of department 20 you will see only your department (including salaries):

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
5555	RON1	DEV	7902	02-NOV-02	2000	100	20
7566	JONES	MANAGER	7839	03-APR-81	3599.75		20
7788	SCOTT	ANALYST	7566	20-APR-87	3630		20
7876	ADAMS	CLERK	7788	24-MAY-87	1331		20
7902	FORD	ANALYST	7566	04-DEC-81	3630		20
4444	RON	DEV	7902	02-NOV-02	2300	200	20

6 rows selected.

If you logon as an employee of department 20 you will not see salary data:

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
5555	RON1	DEV	7902	02-NOV-02		100	20
7566	JONES	MANAGER	7839	03-APR-81			20
7788	SCOTT	ANALYST	7566	20-APR-87			20
7876	ADAMS	CLERK	7788	24-MAY-87			20
7902	FORD	ANALYST	7566	04-DEC-81			20
4444	RON	DEV	7902	02-NOV-02		200	20

6 rows selected.

If you logon as an application developer, you can't see the object:

```
SQL> select * from scott.emp;
select * from scott.emp
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

The masking of salaries and the partial result set based on department is courtesy of VPD; the privilege error (and the fact that the manager and employee can select from the table but only a manager can update the table) is happening courtesy of secure application roles.

## *Chapter 17*

---

# Oracle Database Vault

---

Oracle Database Vault (DV) was introduced in Oracle 10gR2 and is also available for 11g and Oracle 9iR2. It is a security offering primarily targeting the ability to place better controls on privileged users and the ability to manage security from outside the database. It introduces a comprehensive external security overlay that can be managed by non-database administrators (DBAs) and can enforce even DBA activities. It addresses separation of duties for elevating security. As an example, it introduces a new role that is the only one able to create users and modify passwords for users and it prevent the DBA role from doing such activities.

DV is really a collection of many security-focused mechanisms, applications, and enhancements that work in tandem to make an Oracle database more secure and allows security professionals to better manage the security of the database. At a very high level, the elements making up DV are

- *Hardening.* When you install DV it changes various options in your database to conform to hardening best practices. It is not comprehensive hardening and it is not at a level of what you need to do if you follow hardening guidelines such as the Center for Internet Security (CIS) benchmark or the Security Technical Implementation Guide (STIG) document (see Chapter 2), but it does make your database more secure.
- *Enforcing limits on administrator privileges.* DV tackles what is often the hardest problem of all—that of limiting access of DBAs. After you install DV, DBAs can no longer perform many functions. Even starting the database has to be done using sysoper and you can no longer connect using connect “/ as sysdba.” More importantly, you can build new security rules that define realms of control (e.g., an application schema) where DBAs will no longer be able to select data, use Data Manipulation Language (DML)—all within the control of the realm manager who certainly does not need to be a DBA. The DBA has no way to bypass these controls (well—almost no way—you’ll learn about this too).
- *External security overlay.* DV introduces a full access control mechanism and schema whereby information security administrators can manage security rules that define privileges and permissions—all outside the database, all in addition to normal database authorization, and all without the participation of the DBA.

- *Security reports.* DV brings with it many security reports that can help you to pass your audits and review security configuration.

All of these mechanisms and functions are part of DV. You can use just some of them or all of them, and you can start by using one function and then move on to using others. Regardless of your implementation, the main theme of DV is that of locking down the privileged users, and that is something that happens at installation time regardless of what you decide to use. This all happens because DV uses its own concepts to secure itself. All of DV's definitions that a user builds (e.g., the external security rules) are stored within the database—so normally there is nothing to stop a DBA from modifying those. If that were to happen then DV would be easy to circumvent. To secure itself, DV uses one of its default realms to secure its own data and uses the functionality of realms to ensure that DBAs can't tamper with DV. This is all part of the lockdown that DV comes with out-of-the-box.

## 17.1 HOWTO Use a Realm to Secure Data Access from DBA Access

A very common scenario that DV was built for is that in which you need to prevent a DBA from accessing a schema or a set of tables within a schema. As an example, you may have a table with very sensitive data. Normally, any user granted the DBA role will be able to select from this table. This is because the DBA role has a SELECT ANY system privilege; therefore, no matter what object privileges you define for this table, all DBAs will be able to select from it. That's exactly what DV realms were built to do—allow you to affect the use of system privileges. In the example in this HOWTO you'll create a realm which includes a sensitive table and then grant access to only one privileged user—the application owner. Any other user, even when granted the DBA role will not be able to access this table.

Let's use SCOTT.EMP—it has salary information in it. Before we define a realm, DBAs have access to this table—for example:

```
SQL> connect system
Enter password:
Connected.
SQL> select * from scott.emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30

...

SYSTEM can access this table because it has DBA role and thus has SELECT ANY TABLE system privileges. There is nothing SCOTT can do about it.

What you'll do next is create a protection realm that will include the EMP table. This protection realm will disallow access to any user apart from the users to which you explicitly add as realm participants. The protection realm works on system privileges—so even DBAs will not be able to bypass these definitions.

To create a realm, logon to the DV Administrator and follow these steps:

1. In the Administration tab click the Realms link.
2. Click the Create button at the top right of the screen.
3. Fill in SCOTT\_EMP as the Name for the realm and fill in a description for this realm protection. Leave Status as Enabled and leave Audit on Failure for Audit Options. Click OK.
4. This will create the realm and take you back to the realm list as shown in Figure 17.1. Note that there are no definitions for protected objects or users that are authorized for this realm yet.
5. Select the realm using the radio button and click Edit.
6. In the realm secured objects area, click the Create button.
7. Select the owner as SCOTT, the type as TABLE, and fill in EMP as the object name. Click OK.
8. Click OK on the top right of the screen to return from editing the realm definition. The screen should be similar to Figure 17.2 where now the realm is marked as having protected objects.

If you issue the same query again while logged on as SYSTEM you will no longer be able to access the data:

```
SQL> select * from scott.emp;
select * from scott.emp
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

The screenshot shows the Oracle Database Vault interface. At the top, it says "Logged in as ADMINISTRATORDVO". Below that, the title bar reads "Database Instance: av.guardium.com > Realms". A sub-header "Realms" is visible. The main content area displays a table of realms. The table has columns: "Select", "Name", "Audit Options", "Oracle Defined Realm?", "Objects Protected?", "Users Authorized?", and "Status". There are seven rows in the table, each with a radio button in the "Select" column. The rows are as follows:

Select	Name	Audit Options	Oracle Defined Realm?	Objects Protected?	Users Authorized?	Status
<input checked="" type="radio"/>	AVSYS	Audit Disabled		X	✓	✓
<input type="radio"/>	Database Vault Account Management	Audit On Failure	✓	✓	✓	✓
<input type="radio"/>	Oracle Data Dictionary	Audit On Failure	✓	✓	✓	✓
<input type="radio"/>	Oracle Database Vault	Audit On Failure	✓	✓	✓	✓
<input type="radio"/>	Oracle Enterprise Manager	Audit On Failure	✓	✓	✓	✓
<input type="radio"/>	SCOTT_EMP	Audit On Failure		X	X	✓

At the bottom of the table, there are "Edit" and "Remove" buttons. The footer of the page includes links for "Database", "Help", and "Logout", and a copyright notice: "Copyright © 1996, 2006, Oracle. All rights reserved." and "About Oracle Database Vault Administrator".

Figure 17.1 New realm created before assigning objects/users.

The screenshot shows the Oracle Database Vault Realms interface. At the top, it says "ORACLE Database Vault" and "Database". On the right, there are "Help", "Logout", and "Database" buttons. Below that, it says "Logged in as ADMINISTRATORORDVO". The main area is titled "Realms" and contains the following text: "Database Vault realms provide a capability to classify database schemas and database roles into functional groups in order to provide fine-grained access control of the ability to use system level privileges against these types of database objects." There is a "Create" button at the top right of the table. Below the table, there are "Edit" and "Remove" buttons. The table has columns: "Select", "Name", "Audit Options", "Oracle Defined Realm?", "Objects Protected?", "Users Authorized?", and "Status". The data in the table is as follows:

Select	Name	Audit Options	Oracle Defined Realm?	Objects Protected?	Users Authorized?	Status
<input checked="" type="radio"/>	AVSYS	Audit Disabled		✓	✓	✓
<input type="radio"/>	Database Vault Account Management	Audit On Failure	✓	✓	✓	✓
<input type="radio"/>	Oracle Data Dictionary	Audit On Failure	✓	✓	✓	✓
<input type="radio"/>	Oracle Database Vault	Audit On Failure	✓	✓	✓	✓
<input type="radio"/>	Oracle Enterprise Manager	Audit On Failure	✓	✓	✓	✓
<input type="radio"/>	SCOTT_EMP	Audit On Failure		✓	X	✓

At the bottom, there are "Edit" and "Remove" buttons. The footer includes "Copyright © 1996, 2006, Oracle. All rights reserved.", "About Oracle Database Vault Administrator", "Database", "Help", and "Logout".

---

Figure 17.2 Realm after adding protected objects.

Note that if you connect as SCOTT and issue this query you will have access:

```
SQL> connect scott
Enter password:
Connected.
SQL> select * from scott.emp;
```

EMPNO	ENAME	J	OB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH		CLERK	7902	17-DEC-80	800		20
7499	ALLEN		SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD		SALESMAN	7698	22-FEB-81	1250	500	30

...

Scott can access this data because his access to the table has nothing to do with system privileges—he has direct object privileges. Realms only look at activities if they use system privileges. SYSTEM can no longer access this data because there is now a realm protecting this table from access through a SELECT ANY TABLE privilege.

### More on Realms

In the example you just completed you built a very simple realm—it only included a single table. Usually realms contain a larger set of objects—either a complete schema or a group of objects that form a logical security group with which you want to associate a security policy. In addition, realms can include a group of roles. The authorizations you define for a realm also affect grants and

revokes for the roles that belong to the realm. As an example, if you associate a role with a realm you can ensure that only you can assign this role and that a DBA can't grant this role.

In addition to the realm objects and roles, a realm has a set of realm authorizations. These are the set of database accounts and roles that can manage or access realm-protected objects and roles. One type of realm authorization allows you to define who the realm owners are. Realm owners can use system privileges to grant and revoke realm-protected roles. They are conceptually the “account managers” within the realm. However, they can't add users to the realm—that is only done by users who have the DV\_OWNER or the DV\_ADMIN roles. The second type of realm authorization is a realm participant. Realm participants can use their system privileges to access a realm-protected object.

DV includes a number of prebuilt realms. These realms help form the backbone of DV's own security model—to ensure that DV itself cannot be bypassed by a DBA. The prebuilt DV realms are

1. DV Account Management Realm. This is perhaps the most important realm from the perspective of DV security. It limits who can manage and create database accounts when DV is used within the database. This is the realm that enforces that DBAs can not manage database accounts; as an example, if you try to grant the DV\_ACCTMGR role to a user (even as SYS) you will find that your command violates a realm check:

```
-bash-3.00$ sqlplus sys as sysdba
SQL*Plus: Release 10.2.0.3.0 - Production on Sat Jul 14 00:38:46 2007
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
SQL> grant DV_ACCTMGR to SYS;
grant DV_ACCTMGR to SYS
*
ERROR at line 1:
ORA-00604: error occurred at recursive SQL level 1
ORA-47401: Realm violation for grant role privilege on DV_ACCTMGR.
ORA-06512: at "DVSYS.AUTHORIZE_EVENT", line 55
ORA-06512: at line 30
```

2. DV Realm. This realm protects the DV schemas (DVSYS, DVF, and LBACSYS) so that DBAs cannot modify the rules and definitions that you build with DV and those that come prebuilt within DV.
3. Oracle Data Dictionary Realm. When you install DV, a DV realm is created to protect the catalog, the SYS schema, the SYSTEM schema, etc.
4. Oracle Enterprise Manager (EM) Realm. This realm is created to protect SYSMAN and DBSNMP.

### Two Things to Remember about DV Realms

1. Realm checking is done for the use of system ANY privileges and for GRANT and REVOKE of roles within the realm—they do not affect normal object or user privileges.
2. Use realms when you want to protect a group of objects and a group of roles from DBAs and other privileged users (users with system privileges).

## 17.2 HOWTO Use Command Rules to Secure User Activity

A command rule is a definition that can be used to protect any activity on any object in the database. A command rule can secure any sort of activity, by any user on any object—all based on a security policy phrased within a rule. A command rule can even be defined to prevent a CONNECT event—so you can disallow certain connections to the database even if authentication succeeds. Command rules are different and more comprehensive than realms in that a realm protects an asset only when a system ANY privilege is used on it. Command rules on the other hand are based on the command—any command.

Recall back to Chapter 9 where you learned about standard auditing—you saw that you can define audit rules based on a statement (which command a user performed) or on a privilege (which privilege was used when the user made the call). The difference between realm protection and command rules protection is not too dissimilar—realms control access based on the privilege used (specifically—system privileges) and command rules control access based on the statement that is used.

A command rule is evaluated after the realm is checked and only if the realm check succeeds. A command rule defines more than just the command—it defines which objects it acts upon, which object owners it acts upon, etc. Therefore, a user action can cause multiple command rules to fire—DV checks all relevant command rules and only if they all evaluate to true will the action be allowed. Command rules override regular object privileges. For example, if the user Scott has access to the emp table and you create a command rule that disallows all updates to this table, then Scott will not be able to update emp. Another way to think about this is that although database authorizations provide privileges, DV command rules deny privileges and this denial takes precedence over standard authorizations. Command rules form a complete security overlay for the definitions within the database, all managed through DV.

Let's look at three command rule examples—two for an SQL command (we'll use UPDATE) and one for CONNECT. First, you'll define the simplest command rule. You'll build a command rule that disables the ability to update the SCOTT.EMP table. Think of this as a special control you need for highly important data where you need special authorization for any update. Without affecting the database (directly) and completely within DV, you will create a command rule that does not allow any update. Then, when someone does need to update that table, they need to contact you, get special approval after which you edit command rule, disable it temporarily (or provided temporary access to that user only) and then revert back to the original rule. To do all this follow these steps:

1. Log on to the DV Administrator and in the Administration tab click the Command Rules link.
2. In the Command Rules screen click on the Create button at the top right.
3. In the General area, select UPDATE from the Command pull down and leave Enabled as the Status.
4. In the Applicability area select SCOTT as the Object Owner and EMP as the Object Name.
5. From the Rule Set drop down select Disabled. This is a prebuilt rule that will always return FALSE—i.e., will not allow the update.

Your command rules will now have an additional rule as shown in Figure 17.3 (third from the bottom). At this point, no update to the table is possible—not by SCOTT and not by SYSTEM. Note that SCOTT can insert into this table but can't update it:

The screenshot shows the Oracle Database Vault Command Rules interface. At the top, there are links for Help, Logout, and Database. Below that, it says "Logged in as ADMINISTRATORDVO". The main area is titled "Command Rules" and contains a table of command rules. The table has columns: Select, Command, Object Owner, Object Name, Rule Set Name, and Status. The "Select" column contains radio buttons. The "Command" column lists various DDL commands like ALTER PROFILE, ALTER ROLE, etc. The "Object Owner" and "Object Name" columns often show "%". The "Rule Set Name" column provides context for each rule, such as "Can Maintain Accounts/Profiles" or "AV Role Rule Set". The "Status" column shows checkmarks. Buttons for Create, Edit, and Remove are located at the top right of the table and at the bottom right.

Select	Command	Object Owner	Object Name	Rule Set Name	Status
<input checked="" type="radio"/>	ALTER PROFILE	%	%	Can Maintain Accounts/Profiles	✓
<input type="radio"/>	ALTER ROLE	%	%	AV Role Rule Set	✓
<input type="radio"/>	ALTER SYSTEM	%	%	Allow System Parameters	✓
<input type="radio"/>	ALTER USER	%	%	Can Maintain Own Account	✓
<input type="radio"/>	CREATE PROFILE	%	%	Can Maintain Accounts/Profiles	✓
<input type="radio"/>	CREATE USER	%	%	Can Maintain Accounts/Profiles	✓
<input type="radio"/>	DROP PROFILE	%	%	Can Maintain Accounts/Profiles	✓
<input type="radio"/>	DROP ROLE	%	%	AV Role Rule Set	✓
<input type="radio"/>	DROP USER	%	%	Can Maintain Accounts/Profiles	✓
<input type="radio"/>	GRANT	%	%	AV Grant Rule Set	✓
<input type="radio"/>	REVOKE	%	%	AV Revoke Rule Set	✓
<input type="radio"/>	ALTER USER	AVSYS	%	AV User Rule Set	✓
<input type="radio"/>	DROP USER	AVSYS	%	AV User Rule Set	✓
<input type="radio"/>	UPDATE	SCOTT	EMP	Disabled	✓
<input type="radio"/>	GRANT	SYS	DBMS_RLS	Can Grant VPD Administration	✓
<input type="radio"/>	REVOKE	SYS	DBMS_RLS	Can Grant VPD Administration	✓

Copyright © 1996, 2006, Oracle. All rights reserved.  
About Oracle Database Vault Administrator

Figure 17.3 Adding a command rule to prevent updates on SCOTT.EMP.

```

SQL> connect scott
Enter password:
Connected.
SQL> update emp set sal=sal*1.1;
update emp set sal=sal*1.1
*
ERROR at line 1:
ORA-01031: insufficient privileges
SQL> insert into emp values(8888,'RON','CLERK',7782,'23-JAN-82',9999,99,10);
1 row created.
SQL> connect system
Enter password:
Connected.
SQL> update scott.emp set sal=sal*1.1;
update scott.emp set sal=sal*1.1
*
ERROR at line 1:
ORA-01031: insufficient privileges

```

Let's refine this command rule a bit—let's change the command rule to allow UPDATES only if the connection is made locally over a bequeath (BEQ) session, or more specifically, only if there is no CLIENT\_IP defined on the connection. In the next HOWTO you'll learn about factors; for now, it is enough to know that DV has a factor (you can think of this as a functional application programming interface (API)) that returns the CLIENT\_IP for a connection. To make these changes you will do three things:

1. Create a rule based on this factor that returns TRUE if the connection has no CLIENT\_IP.
2. Create a rule set with this one rule—a rule set that evaluates to TRUE if the connection has no CLIENT\_IP.
3. Add the rule set to the command rule.

Follow these steps to make this change:

1. Logon to the DV Administrator, and in the Administration tab click on the Rule Sets link.
2. Click the Create button at the right hand corner.
3. In the General area enter DISALLOW\_TCP\_ACCESS as the Name and a description. The rule will return false if the client connection has a client IP (BEQ sessions will not have a client IP as an example). Status should be enabled, Evaluation Options should be All True (although it really doesn't matter—you'll only have one rule). Click OK.
4. Select your new rule set from the list of rule sets and click the Edit button.
5. Scroll down to the Rules Associates to the Rule Set area and click Create.
6. Enter a name and the expression for the rule as shown in Figure 17.4. Click OK.
7. Back on the Edit Rule Set page click OK. Now you have a rule set with a single rule.
8. Click on the database instance breadcrumb link at the top left to navigate back to the home page. Click on the Command Rules link, select your command rule, and click the Edit button.
9. In the Rule Set pull down select your new rule set as shown in Figure 17.5 and click OK.

Now log in to the database as SCOTT—first using a BEQ connection and then using a listener connection (i.e., a TCP connection that will have a client IP address even if the connection is on the database host itself)—and let's see whether the command rule works:

```
-bash-2.05b$ sqlplus scott
SQL*Plus: Release 10.2.0.3.0 - Production on Mon Jan 7 19:57:14 2008
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
SQL> update emp set sal=sal*1.1;
14 rows updated.
SQL> quit
Disconnected from Oracle Database 10g Enterprise Edition Release
10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
```

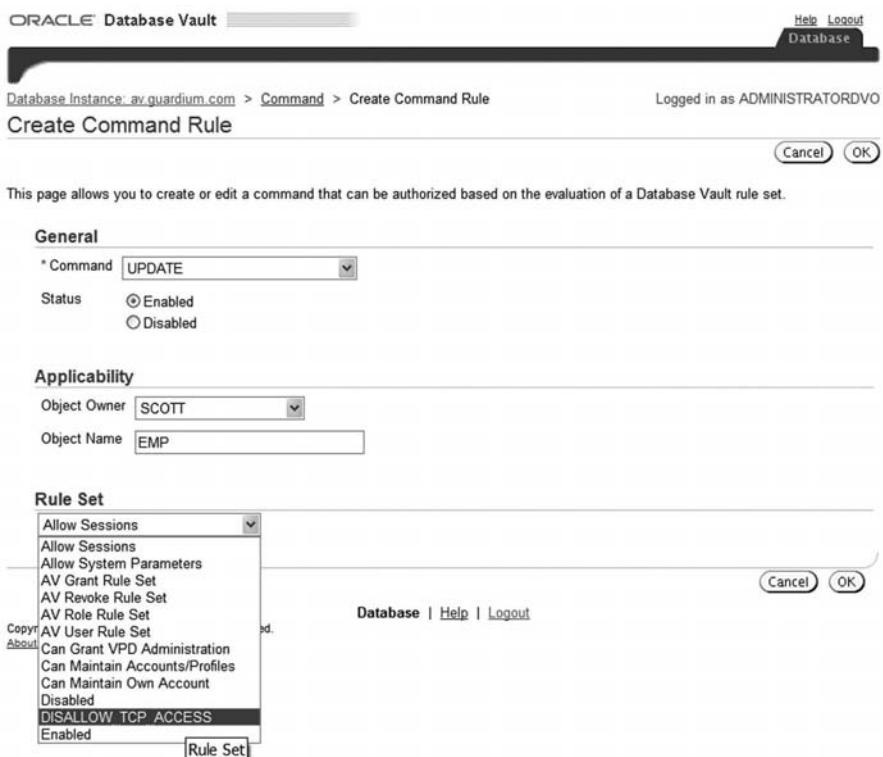
```
-bash-2.05b$ sqlplus scott@dvt
SQL*Plus: Release 10.2.0.3.0 - Production on Mon Jan 7 19:57:53 2008
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
SQL> update emp set sal=sal*1.1;
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

Let's look at a third example, this time with CONNECT. In this example you'll define a command rule that will limit the user SCOTT to connecting to the database only from SQL\*Plus—i.e., any connection from any other application will be disallowed. The steps to build this rule are

1. Build a factor that returns the program name—you'll build this factor in the next HOWTO. Building this factor will generate an API function (see the next HOWTO in the factor section). Connect to the database using SQL\*Plus and select the value to see that the value looks like `sqlplus@<hostname>` (TNS V1-V3).
2. Build a rule that returns TRUE if the program name starts with `sqlplus`.
3. Build a rule set that includes this new rule.
4. Build a command rule that limits CONNECT for SCOTT based on this rule as shown in Figure 17.6. You're done! If you try to connect to Oracle using the SYSTEM user by any other application the connection will be refused.



**Figure 17.4** A new rule returning FALSE if the client IP is not null.

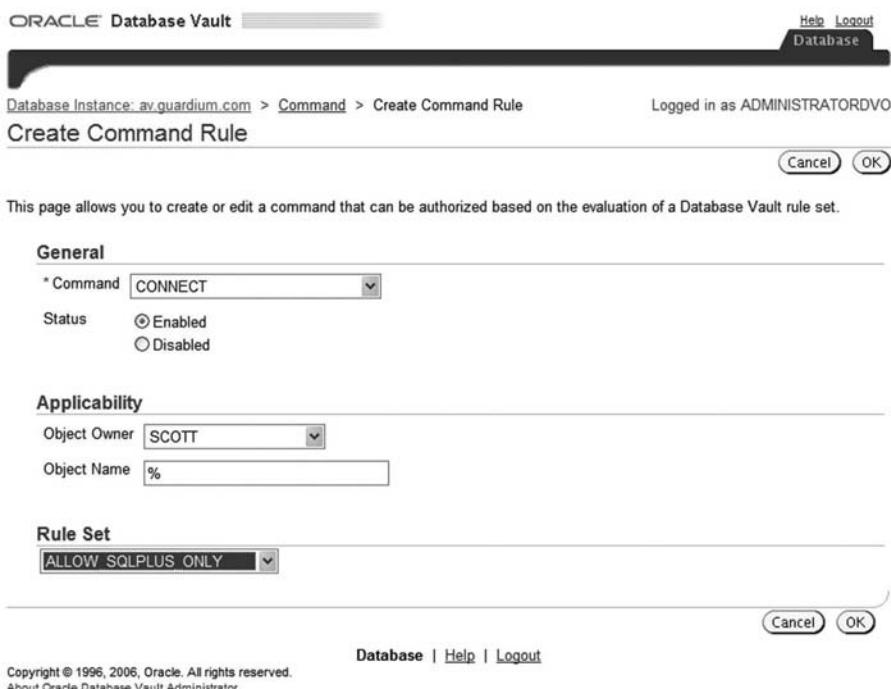


**Figure 17.5 Selecting the rule set for the command rule.**

Finally, just like everything else in the Oracle world, DV comes with a set of PL/SQL procedures that can be used to create these constructs. These are part of the DBMS\_MACADM package within the DVSYS schema. As an example, to create a command rule that disallows dropping the EMP table in the SCOTT schema call:

```
SQL> begin
 2   dvsys.dbms_macadm.CREATE_COMMAND_RULE(
 3     command => 'DROP TABLE',
 4     rule_set_name => 'Disabled',
 5     object_owner => 'SCOTT',
 6     object_name => 'EMP',
 7     enabled => 'Y');
 8   end;
 9 /
```

PL/SQL procedure successfully completed.



**Figure 17.6** Command rule allowing SCOTT connections through SQL\*Plus only.

#### Two Things to Remember about DV Command Rules

1. Command rules are global rules that are checked for every access and for all users. It can be used to secure the use of any statement.
2. Command rules can also be used to limit connections to the database through a command rule based on CONNECT.

### 17.3 HOWTO Use Rule Sets, Factors, and Secure Application Roles

Realms and command rules are the two most important concepts within DV—but they rely on a few others to make DV work. You saw in the two previous HOWTOs that both realms and command rules rely on rule sets. Rule sets allow you to define the conditions for allowing or disallowing activities. Factors are the attributes by which you can make these decisions and they are used within rules. Secure application roles allow you to control role-based permissions

by using DV rules. In this HOWTO you'll review all these concepts and wrap up by looking at the decision process DV makes when it needs to decide whether or not to allow a certain activity to occur.

## Rule Sets

Rule sets are used from within command rules, to determine assignment of factors, to assign DV secure application roles, and as part of realm checks. As the name implies, rule sets are composed of rules. Each rule is a PL/SQL expression that evaluates to true or false. A rule set evaluates to true or false, based on what its member rules evaluate to. A rule set can be of two types—an OR set evaluates to true if any of its member rules evaluates to true and an AND set evaluates to true if all of its member rules evaluate to true.

Rules are PL/SQL expressions that can be evaluated independently to true or false. These expressions can only be 255 characters long but you can call any PL/SQL function or procedure from within them so if you need a long expression wrap it in a PL/SQL function and call the function from the rule. You saw a simple example of a new rule in the previous HOWTO. In addition to building your own functions that you can use within these rules, DV comes with a large set of prebuilt rule functions that you can use. DV also has a few prebuilt rule sets. Some of these rule sets have been supplied as a convenience to you and some have been supplied because almost all implementation will require such rule sets. Note that some will often require you to define the actual rules and conditions—they are really templates which you need to augment based on your specific requirements.

Convenience rule sets include

- Enabled—Use it to allow activity
- Disabled—Use it to prevent activity

Template rule sets include

- Allow Sessions—Use it to control the ability to create a session—e.g., for use within a CONNECT command rule. You have to add your own rules derived from your requirements defining when not to allow connections.
- Can Grant Virtual Private Database (VPD) Administration—Use it to control the ability to GRANT EXECUTE and REVOKE EXECUTE privileges on the DBMS\_RLS package.
- Can Maintain Accounts/Profiles—Use it to control CREATE USER, DROP USER, CREATE PROFILE, ALTER PROFILE, and DROP PROFILE.
- Can Maintain Own Account—Use it to control ALTER USER both for users changing their password and users with the DV\_ACCTMGR role doing account maintenance.
- Check Trigger Init Parameter—Use it to control the ability to set system initialization parameters.

## Factors

Factors are variables that you use within rules. They form the atomic elements within the conditionals of the rules. They allow you to define rules that make a decision based on the IP the connection is coming from, the time of day the connection is made, the user making the connection, the proxy user, and pretty much anything that the Oracle database can be aware of. Factors are extremely flexible—anything that you can express within a PL/SQL function can be turned into a factor. There are many default factors that come prepackaged with DV and you can build your own factors if you need rules that are based on a custom attribute.

Oracle recognized that factors can be very useful not only within DV but also when you use other features. A good example would be VPD. You saw in Chapter 16 that you build policy functions that return a predicate used to rewrite queries. These predicates are usually built in a series of conditionals based on where the request is coming from, who the user is, etc. That's exactly what factors are. To allow you to reuse the work you do within DV, factors were abstracted out as reusable APIs available from the DVF schema. You can use these functions from any of your applications. The factor API functions are shown in Listing 17.1. Some of the useful functions are

1. DVF.F\$CLIENT\_IP—Use it when you need to base a decision on the client IP from which the connection is made. Note that not all connections are TCP/IP connections and for those the client IP will be empty. As an example, you can connect locally through the listener thus creating a TCP/IP session or get your own server process by using BEQ connection. This can be useful if you need to distinguish between listener connections and BEQ connections. In the example below, the first connection has an IP but the second does not:

```
-bash-3.00$ sqlplus sys@dv as sysdba
SQL*Plus: Release 10.2.0.3.0 - Production on Sat Jul 14 01:13:26 2007
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
SQL> select dvf.f$client_ip from dual;
F$CLIENT_IP
-----
192.168.222.30
SQL> quit
Disconnected from Oracle Database 10g Enterprise Edition Release
10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
-bash-3.00$ sqlplus sys as sysdba
SQL*Plus: Release 10.2.0.3.0 - Production on Sat Jul 14 01:13:52 2007
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
SQL> select dvf.f$client_ip from dual;
F$CLIENT_IP
-----
```

In addition to accessing values (called factor identities) through the DVF API functions you can also retrieve factor values using the DVSYS.GET\_FACTOR function, for example:

```
SQL> select get_factor('client_ip') from dual;
GET_FACTOR('CLIENT_IP')
-----
192.168.222.30
```

```
SQL> select owner,object_name from all_objects where object_name like 'F$%';

OWNER OBJECT_NAME
-----
DVF F$AUTHENTICATION_METHOD
DVF F$CLIENT_IP
DVF F$DATABASE_DOMAIN
DVF F$DATABASE_HOSTNAME
DVF F$DATABASE_INSTANCE
DVF F$DATABASE_IP
DVF F$DATABASE_NAME
DVF F$DOMAIN
DVF F$ENTERPRISE_IDENTITY
DVF F$IDENTIFICATION_TYPE
DVF F$LANG
DVF F$LANGUAGE
DVF F$MACHINE
DVF F$NETWORK_PROTOCOL
DVF F$PROXY_ENTERPRISE_IDENTITY
DVF F$PROXY_USER
DVF F$SESSION_USER

17 rows selected.
```

**Listing 17.1 Available factor API functions.**

2. DVF.F\$NETWORK\_PROTOCOL—Use it when you need to make a decision based on the protocol the database client is using to connect to the server. When you connect to the listener (even locally) you will get TCP and when you have a BEQ connection the return value is empty:

```
-bash-3.00$ sqlplus sys@dvh as sysdba
SQL*Plus: Release 10.2.0.3.0 - Production on Sat Jul 14 01:23:30 2007
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
SQL> select dvf.f$network_protocol from dual;
F$NETWORK_PROTOCOL
-----
TCP

SQL> quit
Disconnected from Oracle Database 10g Enterprise Edition Release 10.2.0.3.0
- Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
-bash-3.00$ sqlplus sys as sysdba
SQL*Plus: Release 10.2.0.3.0 - Production on Sat Jul 14 01:23:38 2007
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
```

```
and Oracle Database Vault options
SQL> select dvf.f$network_protocol from dual;
F$NETWORK_PROTOCOL
-----
```

3. DVF.F\$MACHINE—Use this factor when you need to make a decision based on the client hostname:

```
SQL> select dvf.f$machine from dual;
F$MACHINE
-----
rh4u2x32t1.dbasecurity.com
```

4. DVF.F\$ENTERPRISE\_IDENTITY—Use this factor to get the enterprise identity of the logged-on user when you use advanced authentication such as Kerberos, RADIUS, or Oracle Internet Directory (OID) authentication. If you are using standard authentication this value is empty.
5. DVF.F\$PROXY\_ENTERPRISE\_IDENTITY—Use this factor to get the OID Distinguished Name (DN) when the proxy user is an enterprise user.
6. DVF.F\$PROXYUSER—Use this factor to get the proxy user as opposed to the user who opened the connection. This is useful when you have an application where you use proxy users and you want to make decision based on the “real” application user as opposed to the user with which the database session was created.
7. DVF.F\$IDENTIFICATION\_TYPE—Use this factor when you need to base your decision on how the user was identified. This is based on the modifier you use when you create the user—for example, IDENTIFIED EXTERNALLY versus the normal database management of users. As an example, a user defined externally will show a value of EXTERNAL and a user defined and authenticated within the database will show LOCAL:

```
SQL> select dvf.f$identification_type from dual;
F$IDENTIFICATION_TYPE
-----
LOCAL
```

8. DVF.F\$AUTHENTICATION\_METHOD—Use this factor to know how the user was authenticated—for example, PASSWORD for database authentication, KERBEROS, SSL, RADIUS, OS, etc. See Chapter 6 for more details. Note that when you have DV installed you will never get an OS value because that is disabled (in the example below, you get an insufficient privileges error). Even when you connect to the database and your password is checked with a password file the value will come up as PASSWORD:

```
-bash-3.00$ sqlplus "/ as sysdba"
SQL*Plus: Release 10.2.0.3.0 - Production on Sat Jul 14 01:38:18 2007
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
ERROR:
ORA-01031: insufficient privileges
Enter user-name: sys as sysdba
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
```

```
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
SQL> select dvf.f$authentication_method from dual;
F$AUTHENTICATION_METHOD
```

---

**PASSWORD**

You can build your own factors using the DV Administrator. For every new factor you define on the console, DV creates a factor function in the DVF schema for you. This is done automatically.

When you define a new factor you have to define how the value is computed. You can base a factor on a PL/SQL expression, on a constant value (which, for example, you can change using the DV Administrator), or using other factors. This last option is done by mapping values from the child factors to the parent factor—i.e., defining a mapping that tells DV HOWTO compute a factor value for the parent based on the child factor values. Each factor also has an attribute called EVALUATION that defines when it is computed. By-session factors compute the value when a connection is made and by-access factors compute the value when you request the value. The default is by-session and you should be wary of by-access factors because they may cause performance degradation (more on this in HOWTO 17.7).

Factors can be set from the outside—they are not only useful when you need to compute it from the session properties. For example, if you create a factor called DEPARTMENT then a client connected to the database can set the factor value by calling

```
dvsys.set_factor('DEPARTMENT', 'HR');
```

This would seem to limit the value of the factor—how can you trust it if a user can set it arbitrarily? For that, you can assign a rule set to a factor definition. You can define an Assignment Rule Set and select a rule set from a list. This rule set is called when the factor value is set and can check for additional conditions in determining what the value set will be. You can also define a validation method per factor which is called during the factor initialization process; the factor validation method receives the proposed factor value and if the validation method fails or returns FALSE then DV will set the factor value to NULL.

Let's look at an example of a new factor that is very handy (you used it in the previous HOWTO to limit CONNECT). It is used so often that it should probably have been one of the prebuilt factors. You'll build a SOURCE\_PROGRAM factor that returns the name of the program that initiated the connection to the database. You can use it when you want to distinguish between connections made from your application versus connections made through other tools. To create this factor, follow these steps:

1. Log in to the DV Administrator using an account that has the DV\_OWNER role.
2. Select the Factors page.
3. In the factors page click the Create link.
4. In the General area set the Name to PROGRAM and enter a description.
5. Set Factor Type to Application.
6. In the Factor Identification area set the value to By Method because the value of the factor will be computed.
7. In the Evaluation area set Evaluation to By-Session because the value will be computed and applied when the session is created; the source program does not change within a session.
8. In the Factor Labeling area set By Self.
9. Set Retrieval Method to be

```
dvsys.dbms_macadm.get_session_info('program')
```

- This is the interesting part of this factor—we'll discuss it after you finish building the factor.
10. Leave Validation Method and Assignment Rule Set empty.
  11. Set the Audit Options to Always for debugging purposes and after you are satisfied with the result you can set it to Never.

The core of this factor is the call to dvsys.dbms\_macadm.get\_session\_info ('program'). This function is prebuilt within the DVSYS schema and gives you access to information stored in V\$SESSION (which is used often to build factors). Another such function, get\_instance\_info, gives you access to information within V\$INSTANCE. Note that although you have access to these from within factors, you do not have access to them directly. As an example:

```
SQL> select dvsys.dbms_macadm.get_session_info('program') from dual;
select dvsys.dbms_macadm.get_session_info('program') from dual
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

Therefore, you must define the factor rather than try to access these dbms\_macadm functions directly. Note that by building this factor you not only can use it within DV but also get a function within the DVF schema that is accessible to all.

### **Secure Application Roles**

DV introduces a construct called DV Secure Application Roles. These are roles within the database that depend on a rule set. They are similar to the secure application roles discussed in Chapter 15. The difference is that normal secure application roles are set using custom PL/SQL procedures whereas DV secure application roles are enabled based on the outcome of a DV rule set. You use DV secure application roles in much the same way as you use secure application roles. First, you define privileges based on these roles. Then, you can use rule sets to control when a user can become that role and when not. Because rule sets are external to the database, can be managed by an information security professional not familiar with PL/SQL, are extremely flexible, and can span across applications, DV secure application roles allow you to better control the privileges that you assigned to these roles. They are a very nice extension to the functionality provided by regular secure application roles.

### **Putting It All Together—DV Checks**

Now that you've seen all the concepts within DV, let's review the process by which DV checks whether a particular activity is allowed. Figure 17.7 shows the interaction between realms, command rules, and rule sets and how DV determines whether to deny a user command. When a user executes a command, DV first checks whether the command is using a system privilege and thus whether or not it needs to check a realm. If not—i.e., it is not using a system privilege—then DV checks command rules only and the associated rule sets. If the user command does use a system ANY privilege, DV checks whether the statement affects an object or role within a realm; if not—it checks command rules and the associated rule sets. If the object is protected by a realm it checks whether the user is a realm participant or owner based on whether the command is a GRANT or REVOKE or deals with object privileges; if the user is not, the command fails. If the user is an owner or participant, DV checks the rule set assigned with the realm if it exists. If the rule set evaluates to FALSE the command fails. Otherwise, DV goes on to check the command rules and the associated rule sets. In all cases, if DV decides to allow the action, regular Oracle authorization still occurs.

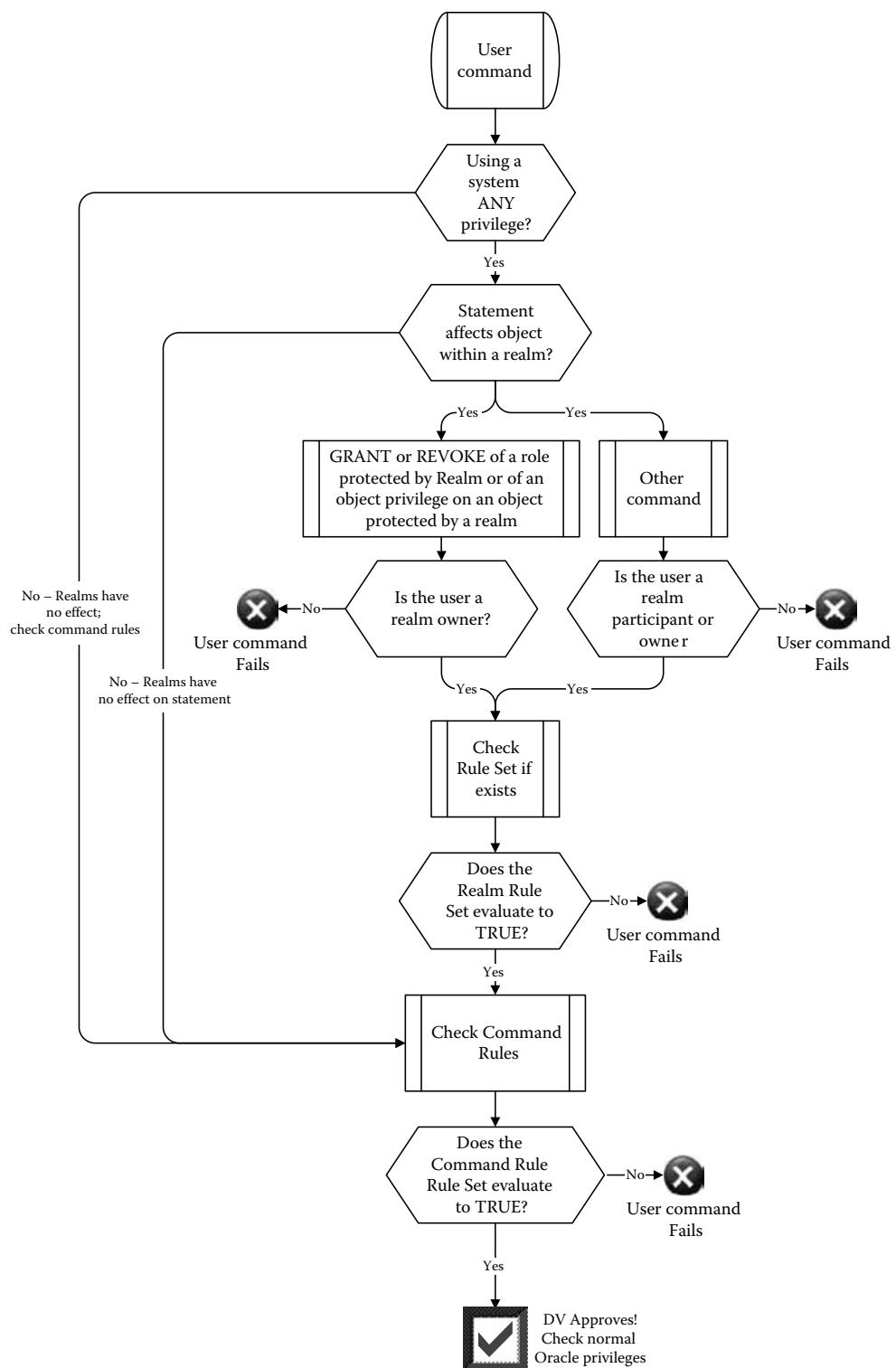


Figure 17.7 DV checks—realms, command rules, and rule sets.

### Five Things to Remember about Rule Sets, Factors, and DV Secure Application Roles

1. Rules are built using arbitrary PL/SQL expressions and thus provide the ultimate flexibility.
2. Rules are composed into rule sets which are used within realms, command rules, and secure application roles—they are the heart of DV.
3. There are useful built-in factors that are available as part of DV—learn HOWTO use them before you build your own factors.
4. Every time you create a factor, DV creates an API method that is available to all from the DVF schema.
5. DV secure application roles are similar to standard secure application roles where a DV rule set determines whether the role is set for a user or not.

## 17.4 HOWTO Use Reports in DV

DV comes prepackaged with two types of reports—DV reports and general security reports. Both are available from the main DV Administrator page on two separate tabs. The DV reports are split into two further types—DV configuration issues and DV auditing reports. The configuration issue reports show information about problems involving command rules, factors, realm authorizations, etc. For example, you will see when rule sets have been disabled in command rules or realms, when a grantee does not exist for realm authorization, etc. The DV auditing reports show information on activities that caused DV audit records to be reported based on your definitions. Every time you create a realm, rule set, etc., you select an auditing option. You can audit on failure or always audit—whether successful or failed. You can also disable the auditing but you should not do so. These audit records are then available within reports. There is an audit report for realms, command rules, factors, DV secure application roles, and more. Use these reports to provide an audit trail on activities within DV. When you use DV to manage security, your auditors will always want to see that you have an audit trail of the policies and their actions.

Far more interesting than the DV reports are the general security reports. Throughout almost every chapter you saw that for good security you need to audit the status of your database in terms of the privileges, entitlements, initialization parameters etc. Every DBA has a set of scripts that they use to periodically extract which users have which roles, who has system privileges, what are the values of the initialization parameters, and more. These scripts are usually home-grown, manual, and involve work to maintain. Being a product that is so focused on the security of the database, DV provides you with a set of audit reports which you can use instead of these home-grown scripts. You can use these reports to satisfy an auditor or to review definitions that affect the security of your database. Note that most of these reports have parameters that you need to select so they are interactive reports. If you want a starting point on a list of deliverables for an audit review this is an excellent list. The list of general security reports available to you (by category) are

1. Object privilege reports
  - a. Object access by PUBLIC—Lists all objects granted to PUBLIC.
  - b. Object access not by PUBLIC—Per grantee, lists object access.
  - c. Direct object privileges—Lists the direct object privileges granted to nonsystem accounts.
  - d. Object dependencies—Lists all dependencies between PL/SQL procedures, packages, functions, triggers, views, etc.
2. Database account system privileges reports

- a. Direct system privileges by database account—Per selected account, lists system privileges directly granted to this account.
  - b. Direct and indirect system privileges by database account—Similar to the previous report but lists all system privileges including those granted through a role.
  - c. Hierarchical system privileges by database account—Also shows the hierarchical structure of where the privileges came from.
  - d. ANY system privileges for database accounts—Lists all system ANY privileges granted to an account.
  - e. System privileges by privilege—Lists the accounts and roles that have been granted a system privilege.
3. Sensitive objects reports
- a. Execute privileges to strong SYS packages—Lists accounts and roles that have execute privileges on powerful system packages (see Chapter 2 for more details). These include UTL\_FILE, UTL\_HTTP, UTL\_TCP, UTL\_SMTP, DBMS\_RANDOM, DBMS\_REPCAT, DBMS\_JOB, DBMS\_PIPE, DBMS\_RLS, DEBUG\_EXTPROC, and many more.
  - b. Access to sensitive objects—Lists accounts and roles that have access to important system tables and views (see Chapter 2 for more details). These include AUD\$, DBA\_ROLE\_PRIVS, DBA\_TAB\_PRIVS, FGA\_LOG\$, PROFILE\$, USER\$, OBJPRIVS\$, etc.
  - c. Public execute privilege to SYS PL/SQL procedures—Lists accounts and role that have execute privileges on packages owned by SYS.
  - d. Accounts with SYSDBA/SYSOPER privilege—Lists accounts that have SYS-privileged connection privileges.
4. Privilege management—summary reports
- a. Privileges distribution by grantee—Lists a count of privileges granted to accounts and roles.
  - b. Privileges distribution by grantee, owner—Lists a count of privileges by grantee/owner combination.
  - c. Privileges distribution by grantee, owner, privilege—Same for combinations of grantee/owner/privilege.
5. Powerful database accounts and roles reports
- a. WITH ADMIN privilege grants—Lists accounts and roles that have been granted privileges WITH ADMIN (see Chapter 14).
  - b. Accounts with DBA role—Lists accounts that have been granted the DBA role.
  - c. Security policy exemption—Lists accounts that have been granted the EXEMPT ACCESS POLICY system privilege and therefore can bypass VPD policies (see Chapter 16).
  - d. BECOME USER privilege—Lists accounts and roles that have the BECOME USER system privilege.
  - e. ALTER SYSTEM or ALTER SESSION privilege grants—Lists accounts and roles that have the ALTER SYSTEM or ALTER SESSION privilege.
  - f. Password history access—Lists database accounts that have access to the USER\_HISTORY\$ table (because it stores hashed passwords).
  - g. WITH GRANT privilege grants—Lists accounts that have been granted privileges with the WITH GRANT clause (see Chapter 14).
  - h. Roles and accounts that have a given role—Per role, lists the accounts and roles that have been granted this role.
  - i. Database accounts with catalog roles—Lists accounts and roles that have been granted one or more of DELETE\_CATALOG\_ROLE, EXECUTE\_CATALOG\_ROLE, RECOVERY\_CATALOG\_ROLE, or SELECT\_CATALOG\_ROLE.

- j. AUDIT privileges—Lists accounts and roles that have the AUDIT ANY or AUDIT SYSTEM privileges (see Chapters 9 and 10).
  - k. OS security vulnerability privileges—Lists accounts and roles that have privileges to export data to the operating system.
6. Initialization parameters and profiles reports
- a. Security-related database parameter—Lists the current security-related database parameters. You've seen most of these in previous chapters, e.g., O7\_DICTIONARY\_ACCESSIBILITY, SQL92\_SECURITY, REMOTE\_OS\_AUTHENT, etc.
  - b. Resource profiles—Lists resource profiles that may help you uncover a DoS attack.
  - c. System resource limits—Lists current system resource consumption by the database.
7. Database account password reports
- a. Database account default password—Lists accounts that have default password (and must be changed).
  - b. Database account status—Lists the existing database accounts.
8. Security audit reports
- a. Core database audit trail—Shows data from AUD\$ (see Chapters 9 and 10).
9. Other security vulnerability reports
- a. Java policy grants—Lists the Java policy permissions stored in the database.
  - b. OS directory objects—Lists directory objects and what their privileges are.
  - c. Objects dependent on dynamic SQL—Lists objects that use dynamic SQL.
  - d. Unwrapped package bodies—Lists PL/SQL procedures that are not wrapped.
  - e. Username or Password tables—Creates a list of tables that potentially contain usernames and passwords.
  - f. Tablespace quotas—Lists accounts that have quotas on one or more tablespaces.
  - g. Nonowner object trigger—Lists triggers that are owned by an account that is different from the account that owns the object on which the trigger is defined.

## 17.5 HOWTO Enable sysdba Connections

By default DV precludes connections using the sysdba role—this is one of the most apparent (and initially disturbing) things that people see:

```
-bash-2.05b$ sqlplus / as sysdba
SQL*Plus: Release 10.2.0.3.0 - Production on Fri Jan 4 15:54:51 2008
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
ERROR:
ORA-01031: insufficient privileges
```

DV does this because there are checks that DV cannot do on a user connected using sysdba privileges. However, in almost any environment you will need to reenable it, even temporarily. An example is when you need to use RMAN; in that case you must reenable sysdba connections before you can use RMAN. The way to reenable sysdba connections is to generate a password file using a new parameter to orapwd called nosysdba. This is an option of orapw that is available in both Oracle 10g and 11g (it was not available in Oracle 9i and previously). To allow connections with a sysdba role follow these steps:

1. Cd to the directory where you keep your password files:

```
-bash-2.05b$ cd $ORACLE_HOME/dbs
```

2. Backup the password file you currently have:

```
-bash-2.05b$ cp orapwonron00 orapwonron00.backup
```

3. Generate a new password file setting the nosysdba parameter to “N”:

```
-bash-2.05b$ orapwd file=./orapwonron00.allowsysdba password=<pwd> nosysdba=n
```

4. Make it the password file:

```
-bash-2.05b$ cp orapwonron00.allowsysdba orapwonron00
```

At this point you can logon as sysdba using your password:

```
-bash-2.05b$ sqlplus sys as sysdba
SQL*Plus: Release 10.2.0.3.0 - Production on Fri Jan 4 15:59:55 2008
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Database Vault options
```

To disable sysdba connections follow the same steps but now use nonsydba=y:

```
-bash-2.05b$ cd $ORACLE_HOME/dbs
-bash-2.05b$ orapwd file=./orapwonron00.nosysdba password=<pwd> nosysdba=y
-bash-2.05b$ cp orapwonron00.nosysdba orapwonron00
-bash-2.05b$ sqlplus sys as sysdba
SQL*Plus: Release 10.2.0.3.0 - Production on Fri Jan 4 15:59:29 2008
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.
Enter password:
ERROR:
ORA-01031: insufficient privileges
```

How do you ensure that DBAs only connect using sysdba when it is appropriate and don't misuse this very simple method of enabling sysdba connections? That's quite simple—by auditing such connections and sending notifications if these connections are made outside of approved maintenance windows.

### The e Things to Remember about Enabling sysdba Connections

1. You can only connect using with a sysdba role by connecting with a password—you cannot connect with “/ as sysdba.”
2. To enable connections with sysdba role you must have a password file. To enable connections as sysdba, create a password file with the nosysdba option set to N. To disable these connections generate a password file with the nosysdba option set to Y.
3. To ensure that enabling sysdba connection is not misused, you must audit sysdba connections and review them. Consider to use real-time alerts when such connections occur.

## 17.6 HOWTO Disable DV and Track Whether It Is Enabled

You may have to temporarily disable DV (COMPLETELY!) to perform maintenance tasks or upgrade the database. Additional scenarios where you may have to disable DV include cases where the DV owner (with role DV\_OWNER) or the DV administrator (with role DV\_ADMIN) accounts are locked out or if the DV account managers (with role DV\_ACCTMGR) forget their password. In all these cases you must disable DV so that you can login as sysdba and perform activities without DV's enforcement.

Because DV enforcement happens directly within the Oracle engine, disabling DV requires you to regenerate the Oracle executable. When you first install DV, DV's installer modifies the Oracle executable to incorporate DV's checks; what you need to do when disabling DV is the inverse—you need to regenerate the Oracle executable to remove DV's checking. If you are running Oracle on Unix follow these steps:

1. Logon as the Oracle instance owner to the operating system
2. Make sure that ORACLE\_SID, ORACLE\_HOME, and PATH are set correctly

```
[oracle10]$ echo $ORACLE_SID
onoracl10
[oracle10]$ echo $ORACLE_HOME
/home/oracle10/oracle/product/10.2.0/db_1
[oracle10]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/oracle10/bin:/home/oracle10/oracle/product/10.2.0/db_1/bin
```

3. Stop dbconsole

```
[oracle10]$ emctl stop dbconsole
```

4. Shut down the instance

```
[oracle10]$ sqlplus "sys / as sysoper"
SQL*Plus: Release 10.2.0.1.0 - Production on Fri Nov 16 02:19:22 2007
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> quit
Disconnected from Oracle Database 10g Enterprise Edition Release
10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
```

5. Disable DV by running make with dv\_off

```
[oracle10]$ cd $ORACLE_HOME/rdbms/lib
[oracle10]$ make -f ins_rdbms.mk dv_off
/usr/bin/ar d
/home/oravault/oracle10/oracle/product/10.2.0/db_1/rdbms/lib/libknlopt.
```

```
a kzvidv.o /usr/bin/ar cr
/home/oravault/oracle10/oracle/product/10.2.0/db_1/rdbms/lib/libknlopt.
a /home/oravault/oracle10/oracle/product/10.2.0/db_1/rdbms/lib/kzvndv.o
```

## 6. Regenerate the Oracle executable

```
[oracle10]$ cd $ORACLE_HOME/bin
[oracle10]$ relink oracle
chmod 755 /home/oracle10/oracle/product/10.2.0/db_1/bin
- Linking Oracle
...
mv -f /home/oracle10/oracle/product/10.2.0/db_1/bin/oracle
/home/oracle10/oracle/product/10.2.0/db_1/bin/oracle0
mv /home/oracle10/oracle/product/10.2.0/db_1/rdbms/lib/oracle
/home/oracle10/oracle/product/10.2.0/db_1/bin/oracle
chmod 6751 /home/oracle10/oracle/product/10.2.0/db_1/bin/oracle
```

## 7. Restart the database

```
[oracle10]$ sqlplus "sys / as sysoper"
SQL*Plus: Release 10.2.0.1.0 - Production on Fri Nov 16 02:23:54 2007
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Enter password:
Connected to an idle instance.
SQL> startup
ORACLE instance started.
Database mounted.
Database opened.
SQL> quit
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
```

## 8. Disable DV using dvca

```
$ORACLE_HOME/bin/dvca
-action disable
-oh $ORACLE_HOME
-service on0r4231
-sys_passwd <sys password>
-owner_account dvowner
-owner_passwd <dvowner password>
-acctmgr_account dvaccount_manager
-acctmgr_passwd <dvaccount_manager password>
```

You're done—DV is disabled and you have normal unfettered access to the databases.

Reenabling DV after you finish whatever maintenance activity you require is a very similar process to the above. First, run dvca as described in step 8 above but use “enable” as the value to the action parameter:

```
$ ORACLE_HOME/bin/dvca
-a      ction enable
-      oh $ORACLE_HOME
```

```
-s      evice on0r4231
      -sys_passwd <sys password>
      -owner_account dvowner
      -owner_passwd <dvowner password>
      -acctmgr_account dvaccount_manager
      -acctmgr_passwd <dvaccount_manager password>
```

Then, perform steps 1 through 7 in the same way with the only difference being that in step 5 you use “dv\_on”:

```
[oracle10]$ cd $ORACLE_HOME/rdbms/lib
[oracle10]$ make -f ins_rdbms.mk dv_on
/usr/bin/ar d
/home/oracle10/oracle/product/10.2.0/db_1/rdbms/lib/libknlopt.a
kzvndv.o /usr/bin/ar cr
/home/oravault/oracle10/oracle/product/10.2.0/db_1/rdbms/lib/libknlopt.a
/home/oravault/oracle10/oracle/product/10.2.0/db_1/rdbms/lib/kzvidv.o
```

At this point you may be questioning this whole scheme. After all, if the Oracle instance owner can disable DV so easily, then how can we say that one of the purposes of DV is to secure against the sysdba. Even more problematic—the instance owner can regenerate the Oracle executable to disable DV and never put it back. DV will not be enforcing anything but will appear to work in terms of the tools and administrative console (if you don’t run the dvca command or run it with enable without relinking the executable). How can you ever trust that DV is actively enforcing your policies?

The answer to this question, like many other things related to security, is defense-in-depth. No single security layer is perfect—and DV is no exception. The ease with which DV can be disabled is a real concern and you need to address it. You can do it in one of three ways (or a combination of these methods):

1. Monitoring V\$OPTION
2. Auditing
3. Change tracking

You can see whether or not DV is active through the V\$OPTION view. For example, when DV is enabled you will get:

```
SQL> select * from v$option where parameter like '%Vault%';
PARAMETER V          ALUE
-----
Oracle Database Vault    TRUE
```

Therefore, if you are using DV and want to make sure that it is not disabled without your knowing, you should continuously monitor this value. You should add it to your security monitoring solution—i.e., continuously extract this value (perhaps every two to three minutes) and if this value changes you should issue an alert and initiate an investigation.

Auditing administrator activity is always useful and always necessary and you’ve seen various options for auditing activities in previous chapters. In the case of the instance owner disabling DV, what you can achieve with auditing depends on how you audit. If you use Oracle administrator auditing or some other audit facility within Oracle then you will not see the activity that disables DV because that is done at the operating system level. All you can audit that is relevant here is

the shutdown and the startup of the database. That in itself is not enough. However, this can be a trigger for checking the value for DV within V\$OPTION. Another option is to use an operating system audit trail. If you (or someone in the infosec group) can set up an audit trail for the Oracle instance account then you will certainly be able to know if the owner disables DV. In any case, don't forget one important thing—auditing at the database level may not give you the audit trail to show when DV was enabled, but it will show you what the sysdba did within the database. Administrator auditing, if reviewed correctly, turns out to be a very effective deterrence-based prevention technique in itself (unrelated to DV).

Finally and perhaps the most appropriate resolution to this problem is change tracking. Disabling and enabling DV changes the Oracle binary. You've already seen in Chapter 2 and other chapters that one of the best tools for elevating Oracle security is a change tracking system that alerts you when changes are made to files or other structures. From a general security perspective, a change to the Oracle binary due to the owner disabling DV is no different than a change to the Oracle binary caused by someone injecting a Trojan into the code, and in both cases change tracking is a powerful tool.

There are a number of change tracking systems; general tools include Tripwire, Symantec ESM, CA eTrust. More appropriate for databases is Guardium's Change Audit System (CAS). A change tracking system allows you to specify which files and other structures you want to monitor for changes. As an example, in Figure 17.8 you can set up a change template that specifies what to

The screenshot shows the 'CAS Template Set Definition' dialog box. At the top, there is a 'Template Set name:' field containing 'Guardium Unix/Oracle Template Set'. Below it are dropdown menus for 'OS Type:' (set to 'Unix') and 'DB Type:' (set to 'Oracle'). To the right of these is a 'Apply' button with a checkmark icon. Underneath these controls is a section titled 'Current item templates:' containing a table. The table has columns for 'Item', 'Type', 'Period', 'Use MD5', and 'Keep Data'. The 'Item' column lists various Oracle executable paths, each preceded by a checkbox. Most items have 'File Pattern' selected as the type, except for 'SORACLE\_HOME/bin/\*' which is 'File Pattern' with a period. The 'Period' column shows intervals like '1m', '10m', '1h', '10m', '1h', etc. The 'Use MD5' column contains checked checkboxes for most items. The 'Keep Data' column contains unchecked checkboxes for all items.

Item	Type	Period	Use MD5	Keep Data
<input type="checkbox"/> SORACLE_HOME/*.sh	File Pattern	1m	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/dbs/*.*	File Pattern	10m	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/inventory/Templates/bin/*.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/jdk/jre/bin/*.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/jre/*bin/*.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/network/admin/*.*	File Pattern	10m	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/oui/bin/*.*so	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/Apache/Apache/bin/*.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/Apache/Apache/fcgi-bin/*.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/Apache/Jsdk/bin/servletrunner	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/Apache/open_ssl/bin/openssl.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/assistants/dbma/mep.cfg	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/bin/*.*	File Pattern	5m	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/ds/bin/*.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/javavm/admin/*.*so	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/jdk/bin/*.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/jdk/jre/bin/*.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/jre/*bin/*.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/ocommon/nls/lbuilder/lbuilder	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/olap/cv.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/soap/bin/*.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/syndication/bin/*.*	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SORACLE_HOME/sysman/admin/OMSRepositoryConstraints.properties	File Pattern	1h	<input type="checkbox"/>	<input type="checkbox"/>

Figure 17.8 A change tracking template for monitoring changes to the Oracle executable.

CAS Change Details								
Start Date: 2007-06-19 00:00:00 End Date: 2007-06-24 00:00:00								
Host Name	Monitored Item	OS Type	DB Type	Instance Name	Type	Owner	Last Modified Time	Sample Time
L3AD029	C:\oracle\product\10.2.0\db_1\BIN\123er.dll	WIN	ORACLE	on0ron00	File		2004-10-26 19:36:00	2007-06-23 16:03:09
L3AD029	C:\oracle\product\10.2.0\db_1\NETWORKADMIN\tnsnames.ora	WIN	ORACLE	on0ron00	File		2007-06-23 17:53:55	2007-06-23 17:54:38
L3AD029	C:\oracle\product\10.2.0\db_1\NETWORKADMIN\tnsnames.ora	WIN	ORACLE	on0ron00	File		2007-06-23 18:10:33	2007-06-23 18:10:58

Figure 17.9 A change report based on the Oracle template.

monitor. In Figure 17.8 all the files under the bin directory are being monitored. Each such file is monitored at a certain frequency (usually one that you can set) and each time the change tracking system computes a signature of the file. These signatures are usually based on one-way hash functions (see HOWTO 8.2 for a bit about cryptographic hash functions). Every cycle, the hash value is compared with the value stored originally. If there is any change to the executable, small as it may be, the hash function will compute a completely different value. This is very effective because even a change of a single byte in the executable will be flagged. Once you set a change tracking template you will see changes in reports such as that shown in Figure 17.9 and can set up alerts in case there are changes made to the Oracle executable. Normally, you would require any disabling of DV to be logged through a change management process and if you see a change to the Oracle executable outside that maintenance window, you can further investigate it with the instance owner.

In this specific example of DV you can even monitor two things; because CAS is a database-specific change tracking system it not only allows you to track files but also allows you to track changes between the output of a query at two different times. You can periodically hash the Oracle executable or you can periodically hash the output of the SELECT \* FROM V\$OPTION query as shown in Figure 17.10. If DV is disabled then the output of this query will change and you will get a notification. Note that other changes that affect V\$OPTION will also trigger this notification—this is usually required also but is unrelated to DV.

#### Four Things to Remember about Disabling DV

1. The Oracle binary can be regenerated by the instance owner (or whoever else has the operating system privileges to do so). If you need to disable DV you can do so by relinking the binary after disabling the dv\_option. You can then reenable DV by following a reverse procedure.
2. Disabling DV is done within the binary itself. No checks will be performed. However, all of the DV applications and utilities will continue to work as if DV is still working. This can create a false sense of security—you may think that DV is enforcing policies but it is not.
3. Use V\$OPTION to check whether DV is enabled and possibly to monitor whether DV has been disabled.
4. Use a change tracking system to alert you upon a change to the Oracle binary—this is a good general security practice and will alert you when DV has been disabled.




---

Figure 17.10 A change item based on querying V\$OPTION.

## 17.7 HOWTO Better Understand DV's Impact on Performance

DV introduces quite a bit of security functionalities and nothing comes for free. As you've seen throughout this chapter, there are many checks that DV makes, depending on what you have enabled. In the first releases of DV there were bugs that caused performance to be horrendous. In fact, you couldn't even call these performance problems—it was simply bugs; you could wait ten seconds for an update statement that was controlled by a very simple command rule. However, these bugs have mostly been addressed and by the time this book will go to press they should all be gone—both for 10g and 11g. You should understand the effect of each of these features on the performance of your database:

- Realms and performance—As a rule of thumb, the performance impact of realms is negligible and can add ~1–3 percent CPU overhead. However, you have to design your realms appropriately for this to be true. You should prefer to define realms around entire schemas rather than specific objects. In any case, you should completely avoid enclosing a realm within another realm as this will adversely affect performance.
- Command groups and performance—Most of the effect command rules have an impact on performance are caused by badly designed rule sets and the procedures that are fired by rule sets. If you make sure that your rule sets are simple, command rule overhead on DML operations can be reduced to as low as 1–5 percent.
- Rule sets and performance—Rule sets are the most dangerous in terms of effect on performance and you should review them very carefully. Rule sets provide you the ultimate flexibility because you can build any security and enforcement policy. With this flexibility

comes the risk. Rule sets have known to cause serious performance degradation. When you start using rules sets, you need to review your design and then test the rule sets well. In terms of design, the more rules you have and the more complex the rules are, the more performance impact you will see. From a performance standpoint, it is better to have one PL/SQL procedure that encodes complex logic rather than have many small rules—there is overhead in the rules processing itself. Even more important is when the rule sets fire—so you must review what triggers the rule set evaluations.

- Factors and performance—There are two evaluation types for factors—by-session and by-access. By-session factors get evaluated when a database session is created. By-access factors get evaluated every time they are accessed. Factors can affect performance mainly in two cases—when you use by-access factors and when you use by-session factors and have an environment with a large number of sessions being created. By-access factors should be avoided if possible and almost all factors used commonly seem to be by-session factors. By-session factors can also affect performance because they are all computed for every session that is created whether you use them or not. As an example, DV comes prepackaged with 17 default factors. Even if you only use Client\_IP and Database\_IP within your rule sets, all 17 will be computed for every session that is created. If you have an environment in which a great many sessions are created this can add up. You should delete all by-session factors that you do not use to limit the impact on performance.
- DV auditing and performance—All auditing affects performance and DV auditing is no exception. Enable only the DV auditing that you are required to collect. You might even consider using an external auditing system that can have less of a performance impact as discussed in Chapter 14.

### Seven Things to Remember about DV Performance

1. Don't define realms within realms.
2. Prefer realms that contain an entire schema over a realm that protects specific objects.
3. Rule sets are the major cause of performance issues when using DV. Design your rule sets carefully and be wary of having many rules or complex logic.
4. From a performance perspective, prefer a rule set with a single PL/SQL procedure that encodes logic that can be otherwise phrased as many rules within a rule set.
5. Avoid by-access factors.
6. Delete by-session factors that you don't use.
7. Limit the use of DV auditing based on your requirements; don't audit too much.

## 17.8 Miscellaneous Discussion—Is Auditing Alone Enough?

You saw two HOWTOs that required that you use auditing as a compensating control in a DV environment. The first was in the case where someone enables sysdba connections and the second was in the case where someone completely disables DV. Auditing is a very effective and practical method, and almost all of us have a lot of experience with it. You can argue that you might as well rely only on auditing and alerting and avoid the complexity and overhead introduced within DV. You can claim that if you're going to have to audit administrators to compensate for these scenarios, why add the prevention capability at all? You can even claim that deterrence is a good enough

prevention mechanism—i.e., if you tell all your DBAs that they cannot do certain things, and you have a full audit trail that cannot be bypassed, then you have a great prevention mechanism (or, that it is as good as any other prevention mechanism because even DV can be bypassed).

This is almost a philosophical discussion and there is no right or wrong answer—it is subjective as are many things in security. This argument is valid and many very good security implementations are based on this principle. Auditing, monitoring, and change tracking are often used within prevention layers. However, when you make this decision, you should remember the following points:

- Systems that prevent usually imply less work on an ongoing basis than systems that monitor and audit. If you have a clear policy and want to prevent a DBA from certain activities it is simpler to define a policy that can be enforced once than to continuously monitor for that activity.
- There is a big difference between the need to monitor or track changes and alert when DV is disabled and the need to monitor and review all DBA activity. You can easily put a change tracking rule to alert when DV is disabled and use rules within DV to enforce a policy—thereby saving you a lot of reviews of DBA activities.
- Real prevention is different from prevention based on deterrence. You may have a requirement that a DBA cannot select from a sensitive table. What happens if a DBA does do it? You can't turn back time—this person has seen the data. What happens if this data is so sensitive that it really cannot be viewed? In a less sensitive environment, what happens if the DBA accessed this data by mistake? Can you really afford to fire this person?
- Auditors and regulators look for preventive controls. Monitoring and auditing is viewed as a compensating control. Although it is usually acceptable to use such compensating controls, you might run into a situation where you are mandated to use preventive controls and cannot use compensating controls. This is not likely to happen soon. The reason is that although Oracle has DV, none of the other database platforms has an equivalent capability. When auditors phrase their requirements they prefer an implementation that is consistent across different database types. Because most organizations use multiple database vendors, auditors are unlikely to require something that can only be implemented in an Oracle environment.
- You need to understand these tradeoffs and make an educated decision. If you do, and as long as you make sure to place all the appropriate controls (and compensating controls), your implementation will be effective and successful.

## *Appendix A*

---

# Payment Card Industry (PCI) Data Security Standard (DSS) Version 1.1: Impact on Oracle Security Implementations

---

The Payment Card Industry Data Security Standard (PCI DSS) was developed by the major credit card companies as a guideline to help organizations that process card payments. Its goal is to prevent losses due to fraud, hacking, and other security threats. The current version of the standard is PCI DSS 1.1 (as of the middle of 2008). A company processing, storing, or transmitting payment card data must be PCI DSS compliant or risk losing their ability to process credit card payments or fined. Merchants and payment card service providers must validate their compliance periodically. This validation is conducted by auditors. Compliance with the PCI DSS has become one of the most visible and demanding requirements in the area of Oracle security.

In this Appendix you'll see what capabilities and functions you should implement in order for your Oracle databases to be in compliance. As you'll see, the PCI DSS is very specific in defining requirements and is very robust. This makes a full-blown implementation a large undertaking. Your take-away from this table should not be that to be compliant with the PCI DSS you need to tackle all of this in one go. An implementation can be spread across a few projects and with time you will not only become fully compliant but also elevate the security of your data.

The table below lists the sections relevant to database security from the PCI DSS. All sections in italics are extracted from the PCI DSS itself. The table shows you what type of function you need to implement to be in compliance with each relevant section. You can use this table as a cookbook of sorts. You can prioritize either based on the PCI DSS chapter or by function. In the later case this table shows you that you can often address multiple sections of the PCI DSS by implementing a single function (e.g., auditing, database activity monitoring, encryption of data at rest, or encryption of data in transit). In this case you can prioritize based on what will give you the biggest and faster return on investment.

<i>Chapter/Section</i>	<i>PCI 1.1 Requirement</i>	<i>Implemented Function</i>
<b>2</b>	<b><i>Do not use vendor-supplied defaults for system passwords and other security parameters</i></b>	
	<i>Hackers (external and internal to a company) often use vendor default passwords and other vendor default settings to compromise systems. These passwords and settings are well known in hacker communities and easily determined via public information.</i>	
2.1	<i>Always change the vendor-supplied defaults <b>before</b> you install // a system on the network (for example, passwords, Simple Network Management Protocol [SNMP] community strings, and elimination of unnecessary accounts).</i>	<p>Use vulnerability assessment and configuration assessment tools. Harden the database (see Chapter 2).</p> <p>Oracle 11g is locked-down upon installation including locking default accounts with their password expired and including checks for default passwords after an upgrade (see Chapter 4).</p>
2.2	<i>Develop configuration standards for all system components. Assure that these standards address all known security vulnerabilities and are consistent with industry-accepted system hardening standards as defined, for example, National Institute of Standards Technology (NIST), and Center for Internet Security (CIS).</i>	<p>Use vulnerability assessment and configuration assessment tools. Harden the database (see Chapter 2).</p> <p>Use change tracking to ensure that a hardened configuration is not altered over time (see Chapter 2).</p>
2.2.3	<i>Configure system security parameters to prevent misuse.</i>	<p>Use vulnerability assessment and configuration assessment tools. Harden the database (see Chapter 2).</p> <p>Use database activity monitoring (DAM) for identifying misuse (see Chapter 14).</p>

2.2.4	<p><i>Remove all unnecessary functionality, such as scripts, drivers, features, subsystems, file systems.</i></p>	<p>Use vulnerability assessment and configuration assessment tools. Harden the database (see Chapter 2).</p>
2.3		
3		
3.3		
3.3	<p><i>Mask PAN when displayed (the first six and last four digits are the maximum number of digits to be displayed).</i></p>	<p>Use a DAM system with extrusion and redaction capabilities to mask certain data (see Chapter 14).</p>

(continued)

Chapter/Section	PCI 1.1 Requirement	Implemented Function
	<p><i>Note: This requirement does not apply to employees and other parties with a specific need to see the full PAN; nor does the requirement supersede stricter requirements in place for displays of cardholder data (for example, for point of sale [POS] receipts).</i></p>	<p>Use virtual private database (VPD) to completely mask certain columns (see Chapter 16).</p> <p>Use access control to disallow access to PAN data using authorization (see Chapter 15), Database Vault (see Chapter 17), or a DAM system with prevention capabilities (see Chapter 14).</p>
3.4	<p><i>Render PAN, at minimum, unreadable anywhere it is stored (including data on portable digital media, backup media, in logs, and data received from or stored by wireless networks) by using any of the following approaches:</i></p> <ul style="list-style-type: none"> <li>• One-way hashes (hashed indexes)</li> <li>• Truncation</li> <li>• Index tokens and pads, (pads must be securely stored)</li> <li>• Strong cryptography with associated key management processes and procedures</li> </ul>	<p>Encrypt data-at-rest. Use either file-system encryption or TDE-based encryption to combat media theft. Alternatively, use DBMS_CRYPTO or a third-party encryption tool to render data unreadable even if a user has access to the tables (but has no access to the keys). In this case use a monitoring or prevention system to protect access to the keys (see Chapter 8).</p>
		<p><i>The MINIMUM account information that needs to be rendered unreadable is the PAN. If for some reason, a company is unable to encrypt cardholder data, refer to Appendix B: "Compensating Controls for Encryption of Stored Data."</i></p>
3.5	<p><i>Protect encryption keys used for encryption of cardholder data against both disclosure and misuse.</i></p>	<p>Use Oracle Wallet or hardware security modules (HSM) for storing keys (see Chapters 5, 7, and 8).</p>
3.5.1	<p><i>Restrict access to keys to the fewest number of custodians necessary</i></p>	<p>Ditto</p>
3.5.2	<p><i>Store keys securely in the fewest possible locations and forms</i></p>	<p>Ditto</p>
3.6.1	<p><i>Generation of strong keys</i></p>	<p>Ditto</p>

3.6.2	<i>Secure key distribution</i>	Ditto
3.6.3	<i>Secure key storage</i>	Ditto
3.6.4	<i>Periodic changing of keys:</i> • As deemed necessary and recommended by the associated application (for example, rekeying); preferably automatically • At least annually	Ditto
3.6.5	<i>Destruction of old keys</i>	Ditto
3.6.6	<i>Split knowledge and establishment of dual control of keys (so that it requires two or three people, each knowing only their part of the key, to reconstruct the whole key)</i>	Ditto
3.6.7	<i>Prevention of unauthorized substitution of keys</i>	Ditto
3.6.8	<i>Replacement of known or suspected compromised keys</i>	Ditto
3.6.9	<i>Revocation of old or invalid keys</i>	Ditto
4	<b><i>Encrypt transmission of cardholder data across open, public networks</i></b>	
	<i>Sensitive information must be encrypted during transmission over networks that are easy and common for a hacker to intercept, modify, and divert data while in transit.</i>	
4.1	<i>Use strong cryptography and security protocols such as SSL/TLS and IPSEC to safeguard sensitive cardholder data during transmission over open, public networks.</i>  <i>Examples of open, public networks that are in scope of the PCI DSS are the Internet, WiFi (IEEE 802.11x), global system for mobile communications (GSM), and general packet radio service (GPRS).</i>	Encrypt data-in-transit (see Chapter 7).

(continued)

Chapter/Section	PCI 1.1 Requirement	Implemented Function
<b>6</b>	<b>Develop and maintain secure systems and applications</b>	
	<p>Unscrupulous individuals use security vulnerabilities to gain privileged access to systems. Many of these vulnerabilities are fixed by vendor-provided security patches. All systems must have the most recently released, appropriate software patches to protect against exploitation by employees, external hackers, and viruses.</p> <p>Note: Appropriate software patches are those patches that have been evaluated and tested sufficiently to determine that the patches do not conflict with existing security configurations. For in-house-developed applications, numerous vulnerabilities can be avoided by using standard system development processes and secure coding techniques.</p>	
6.1	<p>Ensure that all system components and software have the latest vendor-supplied security patches installed. Install relevant security patches within one month of release.</p>	Follow Oracle critical patch updates (CPUs) and install patches as soon as possible within your change management process.
6.2	<p>Establish a process to identify newly discovered security vulnerabilities (for example, subscribe to alert services freely available on the Internet). Update standards to address new vulnerability issues.</p>	Use a vulnerability assessment system to track and verify patch levels or all instances. (see Chapter 2).
6.4	<p>Follow change control procedures for all system and software configuration changes.</p>	Ditto
	<b>Implement Strong Access Control Measures</b>	
<b>7</b>	<b>Restrict access to cardholder data by business need-to-know</b>	
	<p>This requirement ensures critical data can only be accessed by authorized personnel.</p>	
7.1	<p>Limit access to computing resources and cardholder information only to those individuals whose job requires such access.</p>	Classify data based on policies and use this classification as the basis for defining access control.
		Use privileges, VPD, Database Vault, or DAM with a prevention option for access control (see Chapters 14 through 17).

<p><b>7.2</b></p> <p><i>Establish a mechanism for systems with multiple users that restricts access based on a user's need-to-know and is set to "deny all" unless specifically allowed.</i></p>	<p>See above. Access policies should start out with no privileges and add privileges only on a need-to-know basis.</p> <p>This does not cover system privileges. For such users you can use Database Vault or a DAM system with prevention capabilities (see Chapters 14 and 17). Also, periodically audit which users are granted system privileges (see Chapter 15) and produce an audit trail of access to cardholder data by users having system privileges (see Chapters 9 through 14).</p>
<p><b>8</b></p> <p><i>Assign a unique ID to each person with computer access</i></p>	<p><i>Assigning a unique identification (ID) to each person with access ensures that actions taken on critical data and systems are performed by, and can be traced to, known and authorized users.</i></p>
<p><b>8.1</b></p> <p><i>Identify all users with a unique user name before allowing them to access system components or cardholder data.</i></p>	<p>Ensure that users do not share accounts by tracking which users are being used from which client Internet Protocols (IPs) using DAM (see Chapter 14).</p>
<p><b>8.2</b></p> <p><i>In addition to assigning a unique ID, employ at least one of the following methods to authenticate all users:</i></p> <ul style="list-style-type: none"> <li>• Password</li> <li>• Token devices (e.g., SecureID, certificates, or public key)</li> <li>• Biometrics</li> </ul>	<p>Evaluate whether to use database authentication or advanced authentication options supported through advanced security options (see Chapter 6).</p> <p>Use advanced security options for advanced authentication methods including remote authentication and dial-in service (RADIUS)-based SecureID authentication (see Chapter 6).</p>
<p><b>8.3</b></p> <p><i>Implement two-factor authentication for remote access to the network by employees, administrators, and third parties. Use technologies such as RADIUS or terminal access controller access control system (TACACS) with tokens; or VPN (based on SSL/TLS or IPSEC) with individual certificates.</i></p>	<p>Use advanced security options for advanced authentication methods including RADIUS-based two-factor authentication (see Chapter 6).</p>

(continued)

<i>Chapter/Section</i>	<i>PCI 1.1 Requirement</i>	<i>Implemented Function</i>
8.4	<i>Encrypt all passwords during transmission and storage on all system components.</i>	This is the default on Oracle 10g and 11g (see Chapter 4).
8.5	<i>Ensure proper user authentication and password management for nonconsumer users and administrators on all system components as follows:</i>	See below.
8.5.1	<i>Control addition, deletion, and modification of user IDs, credentials, and other identifier objects</i>	<p>Use DAM with prevention, or Database Vault to separate duties for account management (see Chapters 14 and 17).</p> <p>Alternatively, audit all such activity and implement an approval and reconciliation process for such activities (see Chapters 9 through 14 and Appendix B).</p>
8.5.2	<i>Verify user identity before performing password resets</i>	Normal database features + ensure adherence to good practices.
8.5.3	<i>Set first-time passwords to a unique value for each user and change immediately after the first use</i>	Create accounts with the password expired option (see Chapter 4).
8.5.4	<i>Immediately revoke access for any terminated users</i>	Use ASO's enterprise security feature or implement an assessment (as part of a configuration assessment system) that imports user data from Lightweight Directory Access Protocol (LDAP) and compares it with database definitions.
8.5.5	<i>Remove inactive user accounts at least every 90 days</i>	Ditto
8.5.6	<i>Enable accounts used by vendors for remote maintenance only during the time period needed</i>	Use DAM with prevention capabilities, Database Vault, or monitoring/auditing with real-time alerts to control or notify access outside approved time windows (see Chapters 9 through 14 and 17).

		Use behavioral assessment systems to identify when accounts are shared (e.g., from multiple client hosts).
		Use auditing or DAM for the raw data (see Chapter 14).
		Password profiles (see Chapter 4)
8.5.8	<i>Do not use group, shared, or generic accounts and passwords</i>	
8.5.9	<i>Change user passwords at least every 90 days</i>	
8.5.10	<i>Require a minimum password length of at least seven characters</i>	Ditto
8.5.11	<i>Use passwords containing both numeric and alphabetic characters</i>	Ditto
8.5.12	<i>Do not allow an individual to submit a new password that is the same as any of the last four passwords he or she has used</i>	Ditto
8.5.13	<i>Limit repeated access attempts by locking out the user ID after not more than six attempts</i>	Ditto
8.5.14	<i>Set the logout duration to 30 minutes or until administrator enables the user ID</i>	Ditto
8.5.15	<i>If a session has been idle for more than 15 minutes, require the user to reenter the password to reactivate the terminal</i>	Ditto
<b>Regularly Monitor and Test Networks</b>		
<b>10 Track and monitor all access to network resources and cardholder data</b>		
<i>Logging mechanisms and the ability to track user activities are critical. The presence of logs in all environments allows thorough tracking and analysis if something goes wrong. Determining the cause of a compromise is very difficult without system activity logs.</i>		

(continued)

<i>Chapter/Section</i>	<i>PCI 1.1 Requirement</i>	<i>Implemented Function</i>
10.1	<i>Establish a process for linking all access to system components (especially access done with administrative privileges such as root) to each individual user.</i>	Implement identity propagation that fully identify the end-user (even when using connection pooling) (see Chapter 14).
10.2	<i>Implement automated audit trails for all system components to reconstruct the following events:</i>	Use DAM and auditing and utilize the client identifier attribute when you have it (see Chapters 9 through 14).
10.2.1	<i>All individual user accesses to cardholder data</i>	Ditto
10.2.2	<i>All actions taken by any individual with root or administrative privileges</i>	Ditto
10.2.3	<i>Access to all audit trails</i>	Ditto. Put special emphasis on analyzing how the audit is protected, where it resides etc. (see Chapter 14 for an architectural discussion).
10.2.4	<i>Invalid / logical access attempts</i>	Ditto
10.2.5	<i>Use of identification and authentication mechanisms</i>	Use DAM for access tracking and ASO's advances authentication capabilities.
10.2.7	<i>Creation and deletion of system-level objects</i>	Ditto from 10.2
10.3	<i>Record at least the following audit trail entries for all system components for each event:</i>	Ditto from 10.2
10.3.1	<i>User identification</i>	Ditto from 10.2
10.3.2	<i>Type of event</i>	
10.3.3	<i>Date and time</i>	
10.3.4	<i>Success or failure indication</i>	

10.3.5	<i>Origination of event</i>	
10.3.6	<i>Identity or name of affected data, system component, or resource</i>	
10.5	<i>Secure audit trails so they cannot be altered.</i>	Ditto. Put special emphasis on analyzing how the audit trail is protected, where it resides etc. (see Chapter 14 for an architectural discussion on various auditing architectures and Chapter 13 for specifics on Oracle Audit Vault).
10.5.1	<i>Limit viewing of audit trails to those with a job-related need</i>	Ditto
10.5.2	<i>Protect audit trail files from unauthorized modifications</i>	Ditto
10.5.3	<i>Promptly back-up audit trail files to a centralized log server or media that is difficult to alter</i>	Evaluate how audit files are stored with a special emphasis on long-term storage. Consider using Write Once Read Many (WORM) devices. Analyze how long it takes to restore such data for an investigation and how much manual work is required.
10.5.5	<i>Use file integrity monitoring and change detection software on logs to ensure that existing log data cannot be changed without generating alerts (although new data being added should not cause an alert).</i>	Ditto like 10.5
10.6	<i>Review logs for all system components at least daily. Log reviews must include those servers that perform security functions like intrusion detection system (IDS) and authentication, authorization, and accounting protocol (AAA) servers (for example, RADIUS).</i> <i>Note: Log harvesting, parsing, and alerting tools may be used to achieve compliance with Requirement 10.6.</i>	Auditing implementation must support an effective review process. There is no point in auditing if you cannot review the audit data; avoid logging for the sake of logging. Use DAM (Chapter 14) or Oracle Audit Vault (Chapter 13).
10.7	<i>Retain audit trail history for at least one year, with a minimum of three months online availability.</i>	Ditto

(continued)

Chapter/Section	PCI 1.1 Requirement	Implemented Function
	<b>Appendix B: Compensating Controls</b>	
<b>B</b>	<b>Compensating Controls for Requirement 3.4</b>	
	<i>For companies unable to render cardholder data unreadable (for example, by encryption) due to technical constraints or business limitations, compensating controls may be considered. Only companies that have undertaken a risk analysis and have legitimate technological or documented business constraints can consider the use of compensating controls to achieve compliance.</i>	
	<i>Companies that consider compensating controls for rendering cardholder data unreadable must understand the risk to the data posed by maintaining readable cardholder data. Generally, the controls must provide additional protection to mitigate any additional risk posed by maintaining readable cardholder data. The controls considered must be in addition to controls required in the PCI DSS, and must satisfy the "Compensating Controls" definition in the PCI DSS glossary. Compensating controls may consist of either a device or combination of devices, applications, and controls that meet <b>all of the</b> following conditions:</i>	
B.2	<p>Provide ability to restrict access to cardholder data or databases based on the following criteria:</p> <ul style="list-style-type: none"> <li>• IP address/Mac address</li> <li>• Application/service</li> <li>• User accounts/groups</li> <li>• Data type (packet filtering)</li> </ul>	Use DAM with prevention capabilities, Oracle Database Vault VPD, or any combination of the above (see Chapters 14 through 17).
B.3	<p>Restrict logical access to the database</p> <ul style="list-style-type: none"> <li>• Control logical access to the database independent of Active Directory or LDAP</li> </ul>	Ditto
B.4	<p>Prevent/detect common application or database attacks (for example, Structured Query Language [SQL] injection).</p>	Use DAM. See Chapter 14.

## *Appendix B*

---

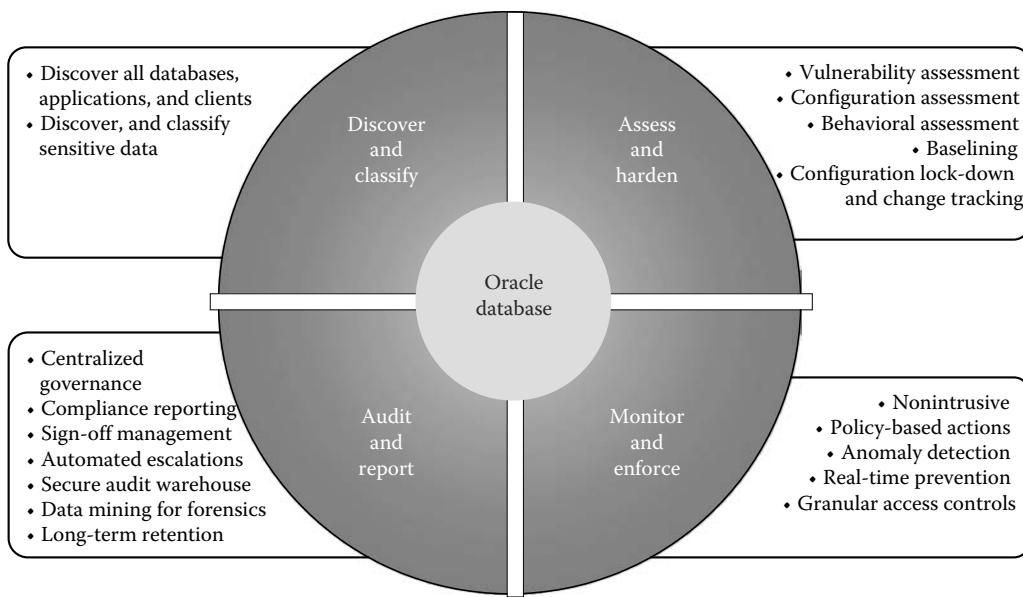
# Using an “All-in-One” Solution: An Example

---

This appendix describes the Guardium system—a suite of applications developed by Guardium Inc. that gives you almost all the functions required for a security implementation involving an Oracle database environment. I have chosen not to mention the Guardium solution in any chapter of the book—I am one of the designers of the Guardium system and did not want this book to be viewed as a way to “sell” the product I helped build. I often read other security books where the product that the authors built is mentioned so often that it taints the technical discussion. I have therefore chosen to discuss all topics without mentioning the Guardium system, so that you can read the material without feeling that I am trying to sell you anything. However, I do fully believe that a new generation of tools such as the Guardium system is needed to address the security of database environments, so I am including a very brief functional overview of the Guardium system here as an appendix.

The functional footprint of the Guardium solution is shown in Figure B.1. The Guardium applications span the entire security lifecycle. Starting from the top-left quadrant, tools are provided to assist discovery—discovering both database instances and sensitive data that resides within these instances. Once you have a good mapping of your databases and where your sensitive data lives, you have tools that help you assess the security-strength of your databases and that help you harden your databases. You use the assessment tool to find weaknesses such as default accounts and passwords, faulty configuration options, patches that have not been installed, and misuse of the database (e.g., sharing of accounts). Once you assess your database you get a report of changes you should consider and recommendations on how to harden your database. Once you follow these steps and create a secure baseline, tools are provided to track all changes that may weaken your environment.

In the lower-right quadrant of Figure B.1 are the monitoring and enforcement tools—starting from the ability to nonintrusively monitor and audit all database activity with practically no impact on performance (even when you need to monitor DML and SELECT activity). When using these tools you do not need to turn on Oracle auditing and you do not need to make any change whatsoever to the database or to the applications. In fact, no one knows you can see their activity—the



**Figure B.1 Elements of the entire security lifecycle.**

system is completely invisible to the database environment. These tools also have the ability to prevent database activities based on a policy—in addition to being able to alert at real time of their occurrence. Most database security projects start off with an implementation of monitoring and auditing but then continue to an implementation that ensures that certain users cannot perform certain activities or view certain data. Finally, in the lower-left hand quadrant are the tools that help you implement security and (especially) compliance projects without having to invest a lot of manpower and without having to add staff. Many of the activities described throughout the book can be time consuming and must be automated in order for them to be sustained over time. These applications automate the generation of reports, manage the sign-off processes for results, and manage archiving and long-term retention of the compliance information.

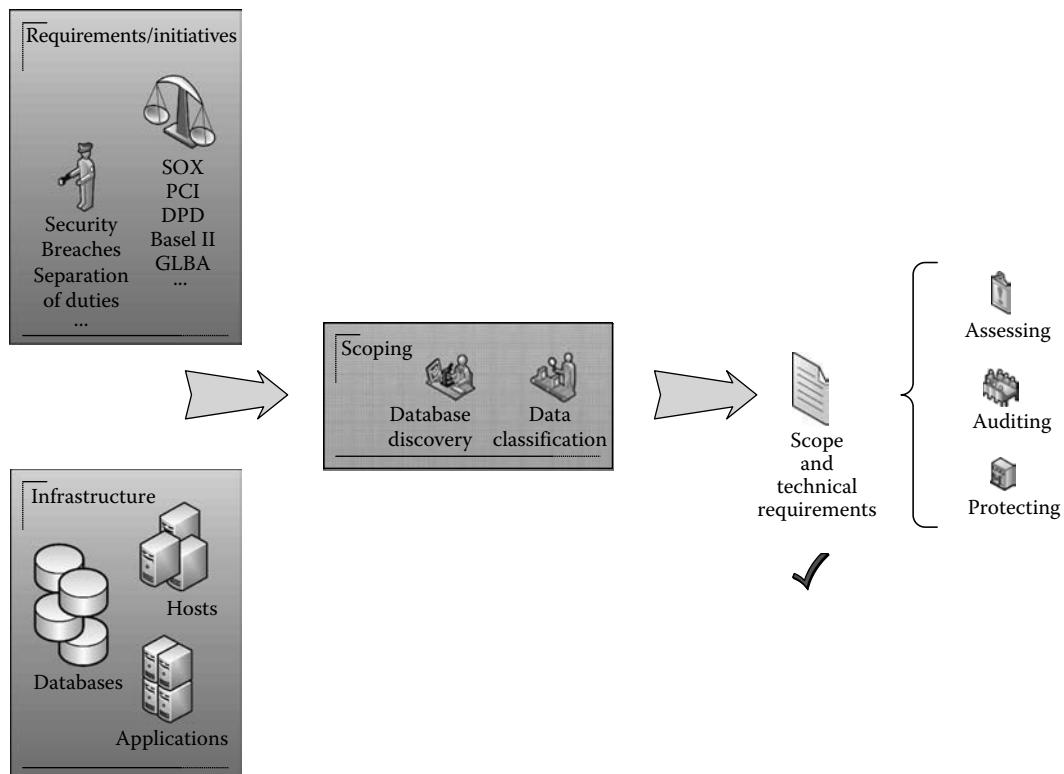
One last important aspect of the Guardium solution is that it is heterogeneous. If you’re reading this book you surely have Oracle databases. However, more often than not, you (or others in the organization) are also managing other databases—SQL Server, DB2, Sybase, Informix, Teradata, MySQL, etc. All of the topics discussed throughout the book are also relevant for these databases. Because the Guardium systems supports all these databases it can help you implement better database security uniformly without having to resort to multiple tools or multiple methods to do the same thing. This not only saves a lot of time but also means that security becomes consistent and compliance becomes predictable.

## B.1 Discovery

When a database security project starts, it must have a scope. The scope not only defines what functionality needs to be implemented and what features must be applied—it also specifies which databases or what data within which schemas needs to be protected or audited. Although some

organizations have good inventory and asset management tools that can help them determine what databases exist, what applications they are used for, etc., sometimes these lists are partial or not fully up-to-date. It is even harder to determine which tables have sensitive data. Defining scope in terms of which databases are important is relatively easy compared with defining which tables may have sensitive data. For example, a large SAP R/3 system can have over 40,000 tables. Obviously, not all of them hold sensitive data and no one would contemplate auditing all 40,000 tables. When defining the audit policy you need to specify which tables need to be audited in terms of database administrator (DBA) access to sensitive data—how do you do that efficiently? Another example is a Payment Card Industry (PCI) implementation involving card holder information and magnetic stripe data. Magnetic stripe data involves the data that is stored in the black magnetic stripe on the back of your credit cards. This data can be used to create forged cards and according to the PCI directives should not be stored anywhere. But how do you know whether it is never stored? How can you tell if all of your thousands of applications, some of which were written a very long time ago and possibly not by you don't store such data somewhere? Discovery is the process that helps you define the scope both in terms of instances and in terms of sensitive data.

As Figure B.2 shows, projects are usually driven by some regulatory requirements or audit finding, or as an infrastructure investment. When the requirements are derived from regulation or a specific business initiative, scoping involves defining which applications/databases need to be covered and sometimes also defining which database objects are in scope. For example, in a



**Figure B.2** Using discovery tools to bootstrap an implementation.

governance-related initiative such as Sarbanes–Oxley the focus is usually on defining which applications are in scope and therefore which databases and which schemas are in scope. In privacy-driven initiatives there is a need to also be able to identify which tables have sensitive data and this becomes an important part of the scoping exercise. When the project is not driven by regulation, but is an infrastructure investment, auditors (either internal or external) will often require the company to implement certain facets of database security across the enterprise. In this case, there still is a scoping stage that involves identifying the databases or data either because priorities and phases need to be defined or because not all databases and tables will have the same audit and security policy rules. In both usage scenarios, the output of the discovery process is data that is used to build up a granular and up-to-date scope and technical requirements document that is the basis for the various implementation projects—be they focused on assessment, monitoring/auditing, or protection.

The first step of a discovery process helps you enumerate the servers and the client connections. This usually involves a static scan on the network (looking for open database ports) and then later monitoring the connections to these instances (usually at the network level). The end result is an access map as shown in Figure B.3 where the servers are enumerated and the client connections mapped.

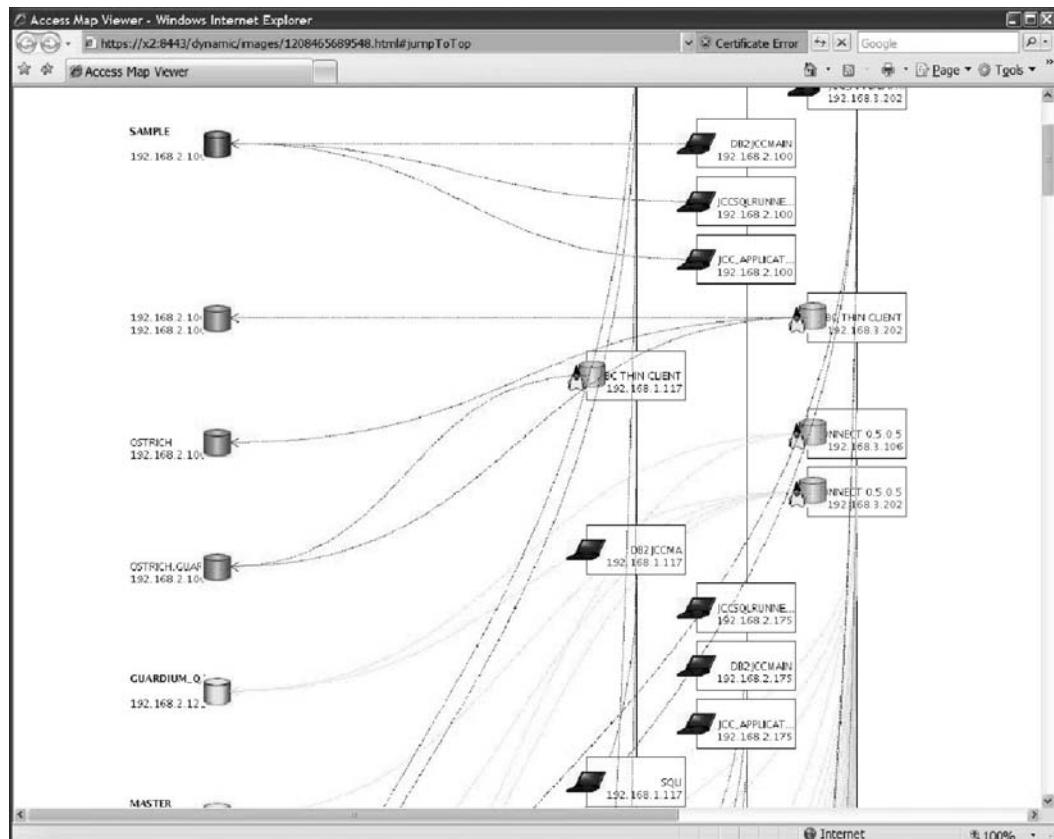


Figure B.3 Access map showing database access over networks.

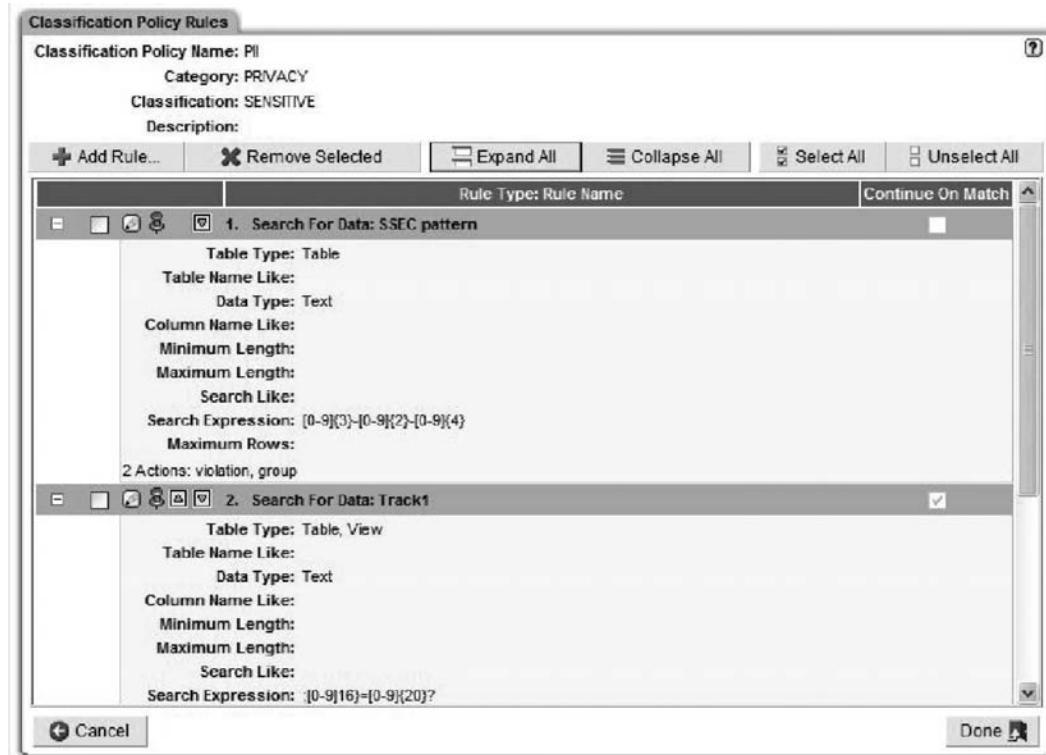


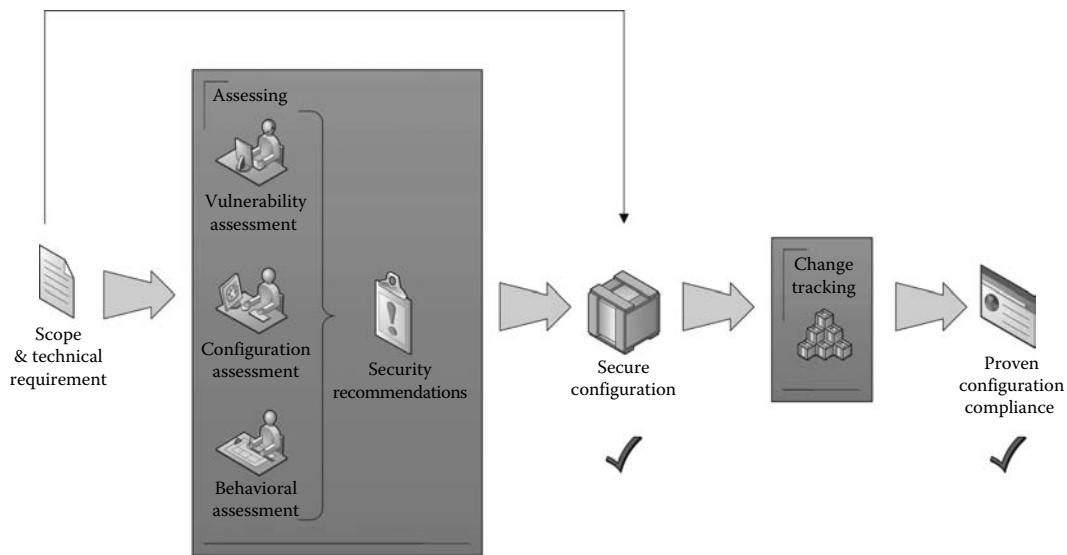
Figure B.4 Defining classification rules.

The next step is data discovery based on a classification policy. Figure B.4 shows two classification rules which are used to scan for sensitive data. The first rule defines a regular expression pattern for U.S. social security numbers. The second rule defines a regular expression for Track 1 magnetic stripe data. In both cases data that matches these patterns will be marked as SENSITIVE and is included in the PRIVACY classification label. These rules are used during data scans which occur by connecting to the database (using a set of credentials which are defined for the discovery engine) and sampling data from the tables. If a match is found then it is added to the classification report which is produced by the discovery process. Additional actions can be triggered including the sending of alerts, automated updating of audit policies, etc.

## B.2 Vulnerability Assessments

Next comes vulnerability assessment tools. As Figure B.5 shows, these tools are used to assess the vulnerability, configuration, and usage of the in-scope databases. You've already seen many samples of things that these tools do—both in Chapter 2 and throughout other chapters, including

- Listener security—e.g., checking for errors, that you have admin restrictions enabled and that listener security is used.



**Figure B.5 Vulnerability assessment and change tracking use case.**

- Account security—e.g., that you use password policies and that you don't use weak and default passwords.
- Authentication checks—e.g., checking that you don't configure your databases to allow for remote password-less authentication.
- Privilege checks and entitlement reports—e.g., checking that you don't grant system privileges too freely.
- Patch installation—e.g., checking that you have the latest CPUs installed on your production servers.

You define two things when using a vulnerability assessment tool. The first thing, shown in Figure B.6, involves defining which tests to run. You select from a set of available tests as shown in the lower pane in Figure B.6. Different databases will have different tests and each test will only be applied for the relevant servers. Next you define which databases you want to run the assessment on. This set is either defined manually or defined based on a discovery process as discussed in the Section B.1. Once you have both defined, the assessment can be run manually or scheduled to be run on a periodic basis. The output of a VA run is shown in Figure B.7. Tests are categorized based on their severity and based on their type. For example, default accounts which have default passwords are classified as CRITICAL and have a category of AUTHENTICATION. These are all parameters that you can tune before you run the assessment. The report shows a summary of how many tests passed, failed, or could not be run (error state) and displays the results by severity and category. The details section lists the tests, the pass/fail/error status and provides a recommendation on how to address the issue. Finally, you can drill down and see more details per result. Such drill-downs can integrate data that is retrieved for other purposes as well. For example, Figure B.8 is an entitlement report that shows a hierarchical view of granted system privileges. This report shows all system privileges granted to users including privileges recursively granted through roles. This report is valuable as part of an audit but it is also used as supporting documentation for a VA report.

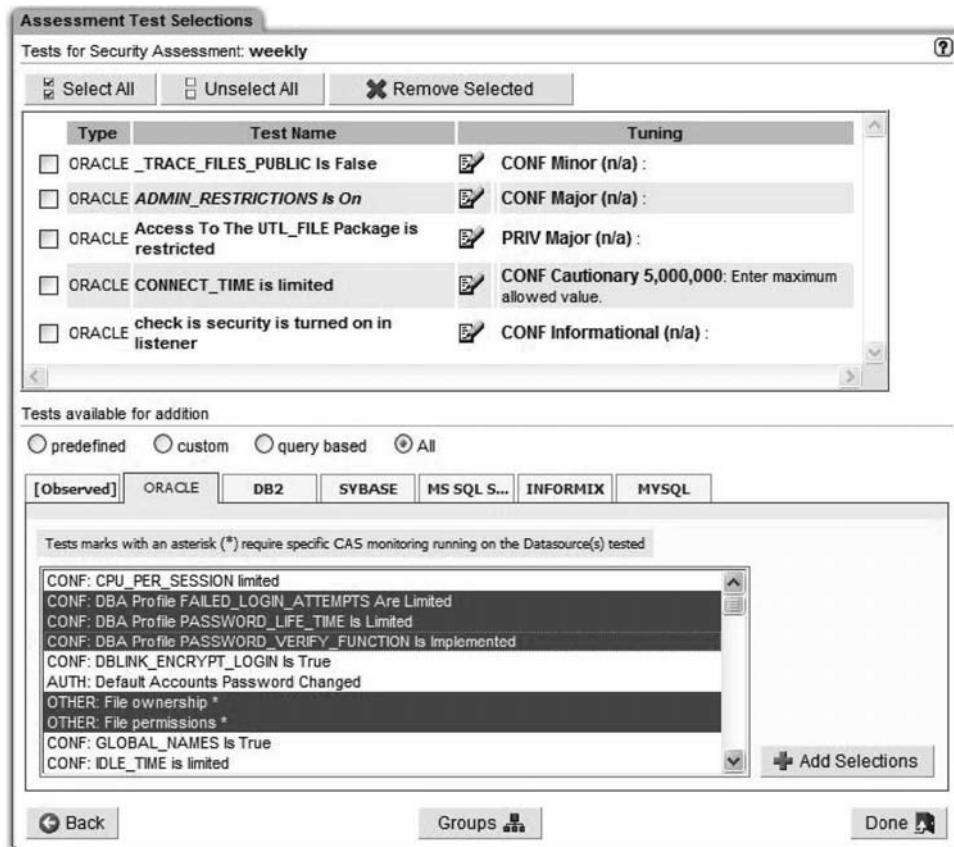


Figure B.6 Defining a security (vulnerability) assessment.

### B.3 Change Tracking

As Figure B.5 shows, the VA report is just the first step in producing a secure baseline configuration. The VA report gives you a set of recommendations. Using these recommendations you can go ahead and harden your database yielding a configuration that you want preserved. Changes can occur at any time—a careless DBA or system administrator (SA) can change file permissions, configuration settings or privileges. All these can affect the security of the database. One method to ensure continuing compliance is to continue periodic assessments. This is something you should do, but it is usually limited to running once a month, sometimes once a week, and very seldom once a day. A better way to ensure continued compliance is to track changes.

Change tracking tells you when a configuration of the database affecting security changes. A change does not necessarily imply a problem—some changes are valid and not every change needs to be tracked. The system gives you a list of attributes, parameters, and values that are part of an Oracle change tracking template (shown in Figure B.9). Such a template includes certain files that need to be tracked for changes (changes to content and/or permissions or ownership—see Figure B.10), parameter values, environment variables, and more. For each such item you specify a polling interval (e.g., one hour or ten minutes) that controls how often the system

The screenshot shows the Guardium Security Assessment Results interface in a Windows Internet Explorer browser. At the top, there's a summary table showing counts for Critical, Major, Minor, Caution, and Info levels across various categories like Privilege, Authentication, Configuration, Version, and Other. To the right, a sidebar displays current filtering applied (Show All for Severity, Scores, and Types, which is set to ORACLE). Below the summary is a detailed table of assessment test results. The table has columns for Category (Cat.), Test Name, Datasource, PF (Passed/Failed), Sev. (Severity), and Reason. A tooltip is visible over a row regarding the 'FAILED\_LOGIN\_ATTEMPTS' parameter. The tooltip states: "The FAILED\_LOGIN\_ATTEMPTS parameter is not set. A high number of failed login attempts can indicate that an unauthorized user is trying to gain unauthorized access to your data. We recommend that you set this parameter in order to limit the number of failed login attempts." The reason column for this row also includes a link to "Filter / Sort Controls".

**Figure B.7** Viewing the results of an assessment.

checks whether there is a change and whether the data is kept. Keeping the data allows you to view a difference between the old value and the new value, so if someone changes sqlnet.ora and thereby disables Advance Security Option (ASO) network encryption you will know what the change was and what the previous file looked like. The benefit of change tracking is continued compliance without adding work. Only security-affecting changes are signaled and as long as nothing comes up in the report or as an alert you know that your baseline configuration has been preserved.

## B.4 Auditing

Because all database security projects require audit trails, the Guardium system allows you to set up any audit rule and manages your audit trails in a secure manner. Setting up an audit rule involves setting conditions and an audit granularity level. The rules are flexible both in that any condition can be phrased using a variety of fields and in that there are multiple granularity levels for audit—once per session, once per time period, once per match, with full details, with masked

User Role Privilege	Datasource Name	SqlGuard Timestamp	Count of ORA Sys Priv Granted
>ANONYMOUS	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE SESSION	ora10-win	2008-03-22 17:39:13.0	1
>APP	ora10-win	2008-03-22 17:39:13.0	1
>>APP_DEVELOPER	ora10-win	2008-03-22 17:39:13.0	1
>>EMPLOYEE	ora10-win	2008-03-22 17:39:13.0	1
>>MANAGER	ora10-win	2008-03-22 17:39:13.0	1
>>APPS	ora10-win	2008-03-22 17:39:13.0	1
>>BI	ora10-win	2008-03-22 17:39:13.0	1
>>ALTER SESSION	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE CLUSTER	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE DATABASE LINK	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE SEQUENCE	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE SYNONYM	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE TABLE	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE VIEW	ora10-win	2008-03-22 17:39:13.0	1
>>RESOURCE	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE CLUSTER	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE INDEXTYPE	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE OPERATOR	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE PROCEDURE	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE SEQUENCE	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE TABLE	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE TRIGGER	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE TYPE	ora10-win	2008-03-22 17:39:13.0	1
>>UNLIMITED TABLESPACE	ora10-win	2008-03-22 17:39:13.0	1
>>CTXSYS	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE PUBLIC SYNONYM	ora10-win	2008-03-22 17:39:13.0	1
>>CTXAPP	ora10-win	2008-03-22 17:39:13.0	1
>>DROP PUBLIC SYNONYM	ora10-win	2008-03-22 17:39:13.0	1
>>DBSNMP	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE PROCEDURE	ora10-win	2008-03-22 17:39:13.0	1
>>OEM_MONITOR	ora10-win	2008-03-22 17:39:13.0	1
>>ADVISOR	ora10-win	2008-03-22 17:39:13.0	1
>>ANALYZE ANY	ora10-win	2008-03-22 17:39:13.0	1
>>ANALYZE ANY DICTIONARY	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE JOB	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE SESSION	ora10-win	2008-03-22 17:39:13.0	1
>>MANAGE ANY QUEUE	ora10-win	2008-03-22 17:39:13.0	1
>>SELECT ANY DICTIONARY	ora10-win	2008-03-22 17:39:13.0	1
>>SELECT ANY DICTIONARY	ora10-win	2008-03-22 17:39:13.0	1
>>DIP	ora10-win	2008-03-22 17:39:13.0	1
>>DMSYS	ora10-win	2008-03-22 17:39:13.0	1
>>ALTER SYSTEM	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE JOB	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE LIBRARY	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE TRIGGER	ora10-win	2008-03-22 17:39:13.0	1
>>CREATE TYPE	ora10-win	2008-03-22 17:39:13.0	1
>>QUERY REWRITE	ora10-win	2008-03-22 17:39:13.0	1
>>EXFSYS	ora10-win	2008-03-22 17:39:13.0	1
>>ADMINISTER DATABASE TRIGGER	ora10-win	2008-03-22 17:39:13.0	1
>>CONNECT	ora10-win	2008-03-22 17:39:13.0	1

Figure B.8 Entitlement report sample.

details, as a representative SQL, etc. This allows you to efficiently match any audit requirement that you may have and all without turning on Oracle auditing and incurring the expense.

An audit policy is a collection of audit rules applied to all databases. You can have an audit rule relevant to all databases, can have an audit rule applied only to Oracle databases, or can apply an

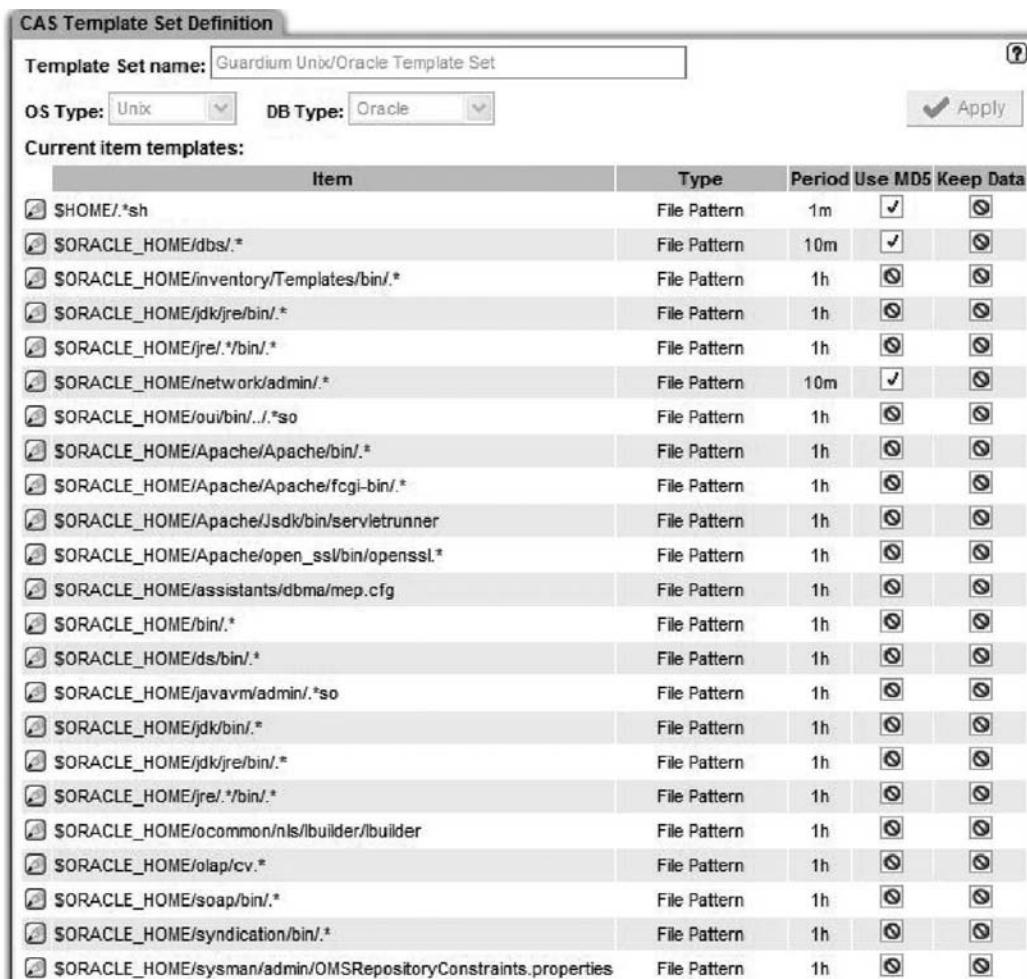


Figure B.9 An Oracle template set.

audit rule to a group of instances or a group of servers. You can easily create policies that track only privileged users or that track application access. Within a single tool you can set rules that provide more flexibility than that provided by standard auditing and fine-grained auditing (FGA) combined, as well as information on the returned data that you can only get using flashback queries natively.

The audit policies determine what data gets stored within the audit warehouse. The audit warehouse uses advanced compression techniques to ensure that you can keep a lot of data for long periods of time online (and archives the data for you based on your schedule). The audit warehouse is secured in a closed database engine that is not open to DBAs or administrators—audit trails cannot be modified or deleted (other than the archive/purge schedule that is managed and audited). The audit trail is independent and never resides in transient storage or on the database server where it can be tampered with.

A set of tools allow you to query the audit trail and generate reports. Figure B.12 shows how a report is built. The tool has three panes. The left pane lists a few hundreds of attributes available

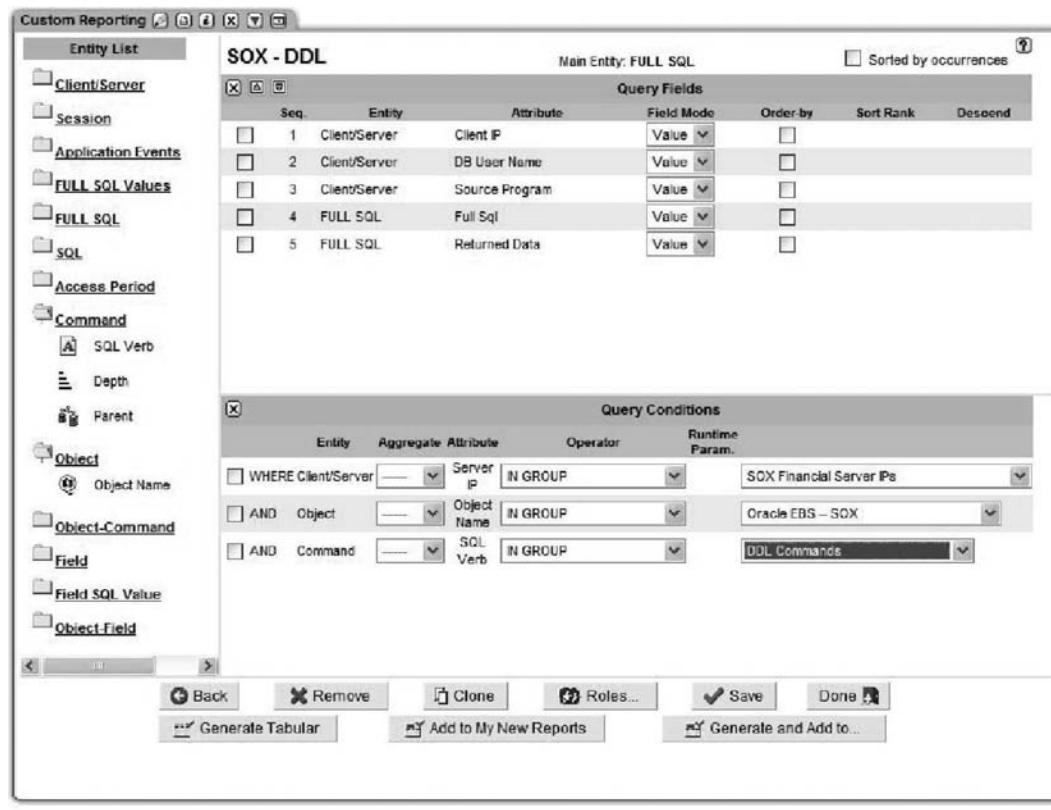


**Figure B.10** Verifying file permissions and ownership compliance.

for reporting. The top-right pane lists the attributes you want to include and see in the report—you simply drag and drop attributes from the left pane onto the top-right pane. As an example, the report shows the client Internet Protocol (IP), the user name, which program was used, the SQL statement, and the result set that was returned to the client. The bottom-right pane lists a set of conditions—these are conditions under which an audit record should appear on the report. For example, Figure B.11 shows a report that includes only data definition language (DDL) commands where the object is part of the Oracle EBS-SOX group and the server IP is within the SOX Financial Server IPs group. Adding conditions involves dragging an attribute from the left pane and dropping it into the bottom-right pane and then specifying the operator and operand for the condition. Figure B.12 shows a sample audit report (in this case including both audit items from Oracle and DB2).

## B.5 Database Activity Monitoring

As Figure B.13 shows, database activity monitoring (DAM) is more than just setting audit policies that create audit trails. DAM encompasses all types of monitoring activities as well as all the operations needed to manage both the monitoring policies and the recorded data. From

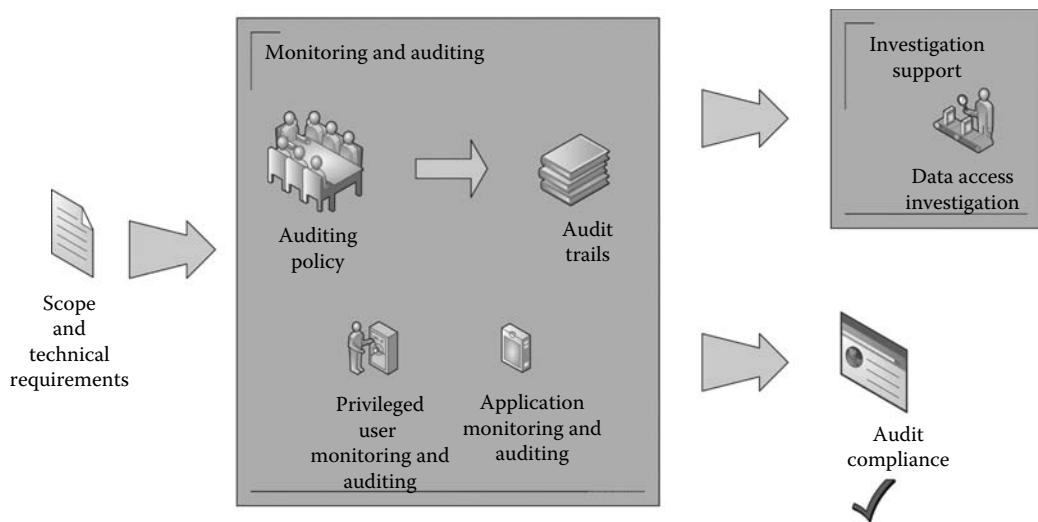


**Figure B.11 Building and customizing reports.**

a monitoring perspective this includes monitoring privileged users as well as application and batch access, deriving baselines of normal behavior, identifying anomalous behavior by both applications and users, and more. From an operational standpoint DAM requires real-time alerting versus the mere collection of audit records and analysis of these audit records at a later stage and also requires the ability to do sophisticated analysis of real-time data across multiple dimensions and across multiple database connections. Finally, DAM includes data management aspects such as archiving of monitored information in a format that can be easily retrieved to perform an investigation and ensuring that such data has valid time stamps and signatures that can be used to prove the authenticity of the records.

Monitoring is managed through a policy. As in the audit policy, a monitoring policy is a collection of rules. Each rule has a set of conditions (defining what to monitor) and an action (e.g., sending a real-time alert or generating a policy violation record). Rules can be applied to queries and transaction (an access rule), to error conditions generated by a database in response to a query/transaction (an exception rule), or to the result set a database returns (an extrusion rule). All three monitoring types are essential for full database monitoring, and the three types cover all requirements that DAM addresses. Additionally, policies can be generated based on learned database activity.

**Figure B.12** Viewing reports.

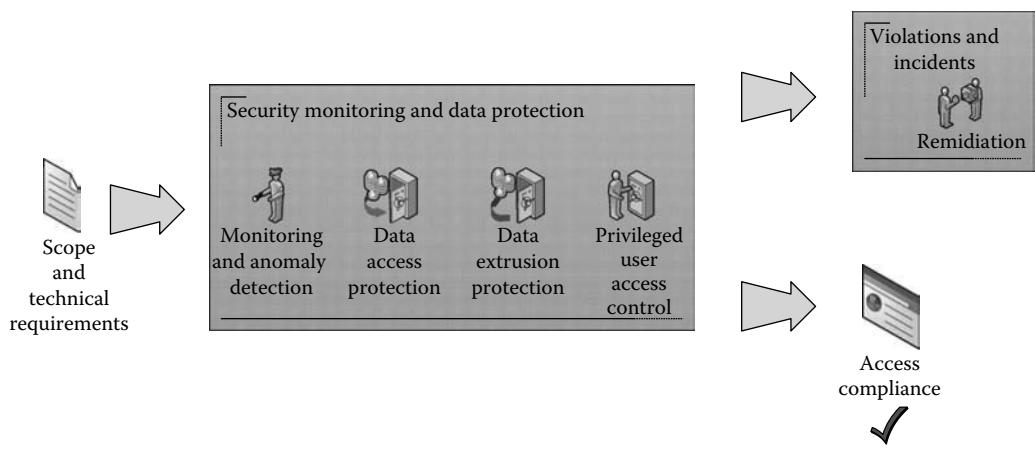


**Figure B.13** Use case for DAM and auditing.

## B.6 Data Access Protection

Monitoring and auditing is an effective prevention method but prevention is through deterrence. If you need to ensure that your DBAs don't look at sensitive information, you can use an auditing policy to know exactly what they are doing and define a real-time alert that will immediately notify you when such access occurs. You can then tell your DBAs that you can see everything they do, that you review everything they do, and if they view sensitive data then you will fire them and press charges. This is usually considered to be a strong-enough scheme to protect the data. Although it is only a compensating control, it is very practical and for many people, good enough.

However, there are more and more scenarios and compliance requirements that require true access control and prevention—both for rogue applications as well as for a privileged user. As Figure B.14 shows, protecting the data includes access control for privileged users (including SYSDBA connections) but also includes using anomaly detection to identify a misbehaving connection and terminating it, and includes analysis of both the requests as well as the result set. As an example, Figure B.15 shows what happens when SYS tries to SELECT application data in an application schema and Figure B.16 shows what happens when an application that is baselined to access sensitive data one record at a time accesses a whole bunch of records. In both cases the connection is terminated by the Guardium S-GATE (a probe that not only can monitor connections but also control what makes it through to the database). In the first case the termination event is triggered because SYS should not be selecting from an application table and in the second case it is terminated because a result set including sensitive data is too large. Note that there are more records in the table than those returned by the SELECT because a threshold is set within the Guardium system. When an application diverges from a threshold-based baseline the result set is limited and so is your liability. Instead of a hacker getting access to a few hundreds of thousands of sensitive record, they get access to a few dozen. Plus—a real-time alert goes out to inform you of the breach.



**Figure B.14** Use cases for external access control overlay.

A screenshot of an Oracle SQL\*Plus session on an Oracle 10g Enterprise Edition database. The session starts with a login as oracle10, followed by a password prompt. The user then runs the command `sqlplus / as sysdba`, which is circled in red. The session continues with the standard Oracle copyright notice and then connects to the database. The user then runs the command `select count(*) from v$parameter;`. The output shows a single row with COUNT(\*) = 258. The user then attempts to run the command `select * from scott.emp;`, which is also circled in red. This is followed by another attempt to run the same command. Both attempts fail with the error message "ORA-03113: end-of-file on communication channel".

```

oracle10@rh4u2x32p:~$ login as: oracle10
oracle10@192.168.222.128's password:
Last login: Fri Mar 21 06:44:53 2008 from 19ad029
[oracle10@rh4u2x32p ~]$ sqlplus / as sysdba
SQL*Plus: Release 10.2.0.1.0 - Production on Sat Mar 22 23:05:38 2008
Copyright (c) 1982, 2005, Oracle. All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> select count(*) from v$parameter;

  COUNT(*)
  -----
        258

SQL> select * from scott.emp;
select * from scott.emp
*
ERROR at line 1:
ORA-03113: end-of-file on communication channel

SQL>

```

**Figure B.15** Preventing SYS access to sensitive data.

## B.7 Compliance

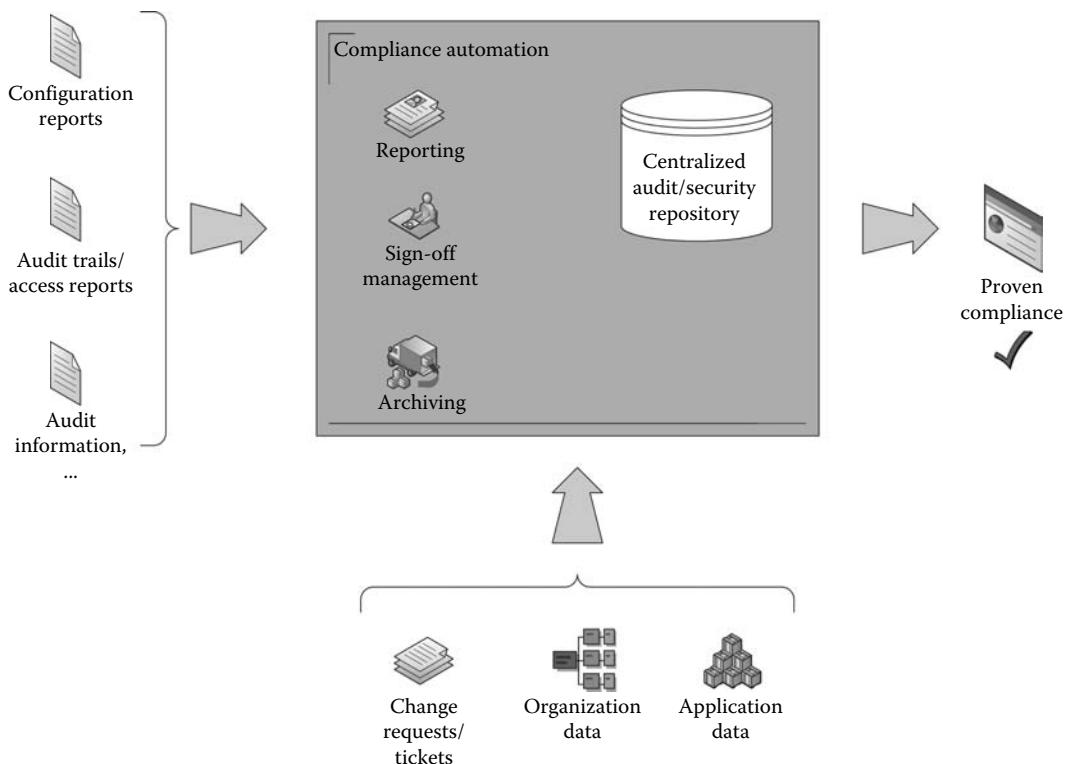
Finally, the last set of tools and modules available within the Guardium suite helps you automate and manage the entire compliance process. These applications are perhaps the most important for two reasons. First, without them you cannot sustain a database security initiative over time without

**Figure B.16** Terminating a connection when many records are accessed.

adding staff. Second, auditors want to see that a process has been defined and implemented. They do not want to see that audit trails, reports, and assessments are generated in an ad hoc manner; they look for repeatable and formalized processes.

Figure B.17 shows how compliance is managed within the Guardium Governance, Risk and Compliance (GRC) suite. Guardium compliance automation does many things. It manages the reports and deliverables—audit reports, entitlement reports, vulnerability reports, etc. It manages scheduling and delivery as well as tracking and archiving. It manages the sign-off and escalation process for these tasks as well as managing the comments and annotations that users add to the reports as part of the workflow. A workflow process is defined per organization, per business unit, or even per application and the system manages the delivery, the signing, the escalations, and the reporting on progress within the process. The system also manages the archiving—both archiving of the raw data and the archiving of the signed results.

The Guardium system also implements a full incident management application. An information security professional can review the current set of violations and derived incidents as shown in Figure B.18. From the dashboard, the user can drill into the individual violations or incidents and track the remediation workflow as shown in Figure B.19. This workflow produces events that are then provided as part of the compliance process thereby adding to the visibility that an auditor has into the process. The end result is a single system which covers every element and every stage of the database security lifecycle.



**Figure B.17 Managing the entire compliance lifecycle.**

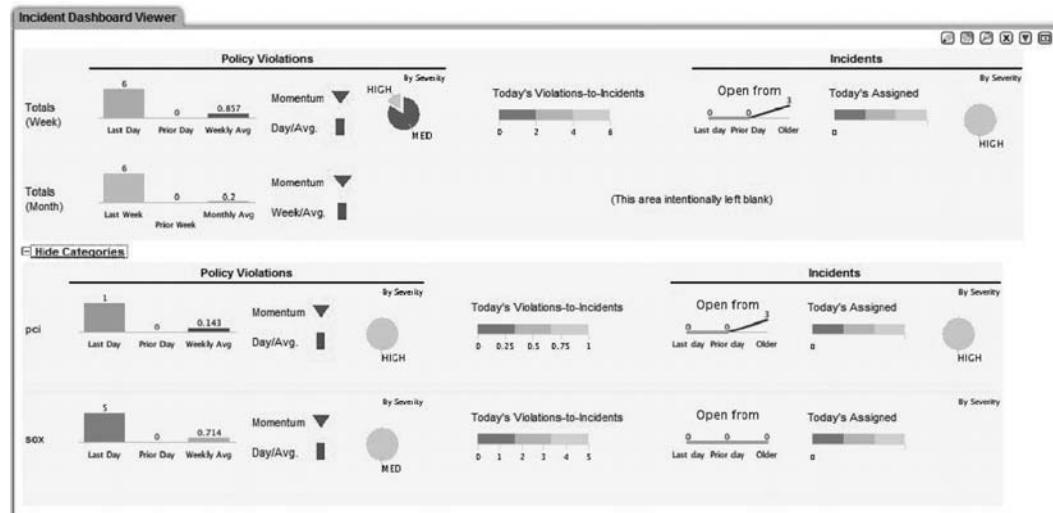


Figure B.18 Incident dashboard.

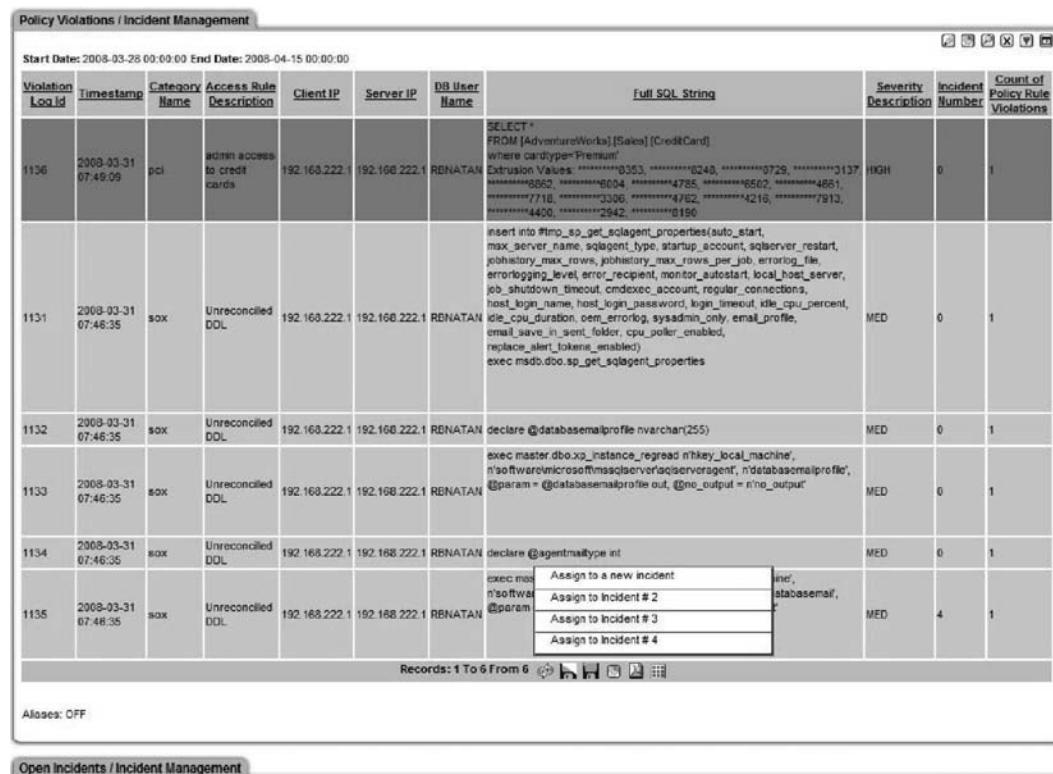


Figure B.19 Incident remediation.

---

# Index

---

## A

- Access control  
  discretionary, 12  
  external policy definition for legacy applications  
    with limited, 307  
  sensitive data, 290  
  users, 9
- Access Control Lists  
  adding privileges to, 345–346  
  assigning to network service, 346–347  
    `dba_network_acls` to view, 347–348  
  creating, 344–345
- Accounts queries  
  `ALTER SYSTEM` and `ALTER SESSION`  
    privileges, 355–357  
  `BECOME USER` privileges, 355  
  `SYSDBA` and `SYSOPER` privileges, 354–355
- ACE/Server  
  authentication to, 122  
  using RADIUS, 123–124
- ACLs, *see* Access Control Lists
- Activity overview report, 277
- Activity reports  
  audit information, 276  
  with predefined and custom reports, 278
- `ADD_POLICY` procedure, 361–362
- Address Resolution Protocol spoofing, 135–136
- Administrator auditing  
  audit trails and, 218  
  DV changes, 407–408  
  to enable, 217
- Administrator authentication, 104–105
- Administrator privileges, enforcing limits on, 383
- `ADMIN_RESTRICTIONS` option, 30
- Advanced Encryption Standard, 78  
  one round of, 76  
  XOR with subkey, 75
- AES, *see* Advanced Encryption Standard
- `ALTER USER` command, 56–57, 128, 130
- Anomaly detection, 291
- API functions, 395–398
- Application activity monitoring, 290
- Application encryption, 154–155
- ARCHIVELOG mode, 251
- ARP spoofing, *see* Address Resolution Protocol spoofing
- ASO  
  data integrity, 141  
  SSL authentication using, 112  
    authentication sequence in, 110–111  
    basic steps in, 113–115  
    handshake between client and server, 112
- Audit commands  
  at column level, 228–229  
  `NOAUDIT` to disable, 209  
  per access or per session, 195–198  
  successful and successful, 193–195  
  user names as qualifiers, 210–211
- `Audit_condition` qualifier, 228
- Audit data log  
  viewing onto AV console, 276–278
- Audit events, filtering for, 277
- Auditing, *see* Standard auditing
- `AUDIT NOT EXIST`, 200
- Audit qualifiers, *see* Audit commands
- Audit records  
  per access, 195  
    `SCOTT.EMP`, 196  
    `SESSION REC`, 197  
  per session, 195, 198  
  tables and views used to view  
    `DBA_AUDIT_TRAIL`, 205–206  
    `DB or DB_EXTENDED`, 204  
    `V$XML_AUDIT_TRAIL` and `DBA_FGA_AUDIT_TRAIL`, 207
- Audit rules, on source databases  
  active, 272  
  defining, 271–272  
  exporting and importing, 271–272  
  managing from AV console, 270  
  setting up, 270–271
- `AUDIT SESSION`, 199–200
- `AUDIT_SYSLOG_LEVEL`, 220

- AUDIT\_SYS\_OPERATIONS, 217
- Audit trails**
- for DBA activity, 188
  - for DB and DB\_EXTENDED, 189–190
  - definition of, 187
  - to enable/disable, 188
  - to OS
    - audit files, 190–192
    - audit record in, 191
    - restarting database for, 190
    - Unix, 190–191
  - to XML, 191–192
- Authentication**
- definition of, 99
  - users, 9
- Auto-login wallet, 107
- AV, *see* Oracle Audit Vault
- AV agents, 256
- and AV server, communications between
    - AV console, 264
    - secure communication scheme, 262–263
    - unencrypted communications, 263
  - deploying, 258
  - installing, 261–262
  - starting, 269
- AV audit reports, 276
- AV audit warehouse star schema, 258
- AV collectors
- audit data from source database, extracting, 255–256
  - managing using avctl on AV server, 268–269
  - reviewing and controlling, 267–268
  - starting, 269
  - types of, 257
- AV console
- refreshing warehouse from, 274–275
  - view audit data within, 276–278
  - viewing agent status on, 263
- AVCTL, managing agents and collectors by, 268–269
- AV server
- collectors and, 273–274
  - components of, 255
  - source database, communication between, 256
  - to start and stop, 273
  - warehouse
    - configuring, 274–275
    - managing, 275
- AV sources, setting up, 264–266
- B**
- Backups
- encryption of
  - RMAN configuration, 181–182
- TRANSPARENT MODE or DUAL MODE, 184
- Baselines, *see* Configuration baselines
- Before and after value, capturing
- problems in, 253–254
  - using flashback queries, 247–250
    - pseudo-columns, 251
    - retention period, long, 251
    - undo segments, size of, 252
  - using triggers
    - AFTER trigger, 237–238
    - audit records, 238
    - audit table, 236
    - changes in data, 238
- Behavioral divergence, intrusion detection, 297–298
- Black lists, 298
- Block ciphers, 78
- C**
- Case-sensitive passwords, 65–66
- CBC, *see* Cipher block chaining
- Certificate Authority (CA), 86
- in browser, 87, 90
- Certificate request
- creating, 95
  - signing process, 96
- Certificates
- for authentication, initiating connection
    - using, 112
  - and digital signatures
    - message authentication, 85
    - public key, 84, 86
  - general information in, 87
  - in IE, 87, 89
  - signing process, 96
  - in wallet, 94–96
- CFB, *see* Cipher Feedback
- CHAIN\_CFB, 156
- Chaining mode, 78
- Change management, overhead associated with, 285
- CHANGE\_PASSWORD, 34, 37
- Change request process, enforcement of, 308
- Change tracking, DV
- due to DBA and SA, 431
  - general tools for, 408
  - for monitoring changes to Oracle executable, 408–409
- Change tracking tools
- creating configuration baseline, 17
  - for monitoring listener.ora, 44
- Cipher block chaining, 79, 156
- Cipher Feedback, 79
- Ciphertext, 83

- CIS Oracle benchmark  
  sections in, 14
- Classification policy, defining, 429
- Code vulnerabilities, 18
- Column encryption key for EMP, 167
- Column privileges  
  granted WITH GRANT option, 351  
  to view, 318
- Command rules, securing user activity with  
  to prevent updates on SCOTT.EMP, 388–389
- UPDATES, BEQ session, 390
- user connection to database, 391–393
- Common Vulnerability Scoring System  
  for security vulnerabilities, 20
- Compliance requirements  
  database security, *see* National regulations,  
    database security and monitoring  
    data security, 7–8
- Computer security standards, 90
- Computerworld, data breach reports, 4–5
- Configuration baselines  
  and behavioral divergence, 297–298
- change tracking tools role in creating, 17–18
- sequence monitoring, 298
- Cost Based Optimizer and FGA, 232
- CPU, *see* Critical Patch Update
- CREATE ANY TABLE privilege, 324
- CREATE LIBRARY, 43–44
- CREATE USER command, 53, 56, 67
- Credit card number, encrypting and decrypting,  
  157, 171
- Critical Patch Update, 15  
  risk matrix, 22  
  security fixes, 21
- Cryptographic accelerators, HSMs as, 176
- Cryptography  
  definition of, 73  
  main elements of, 73–74  
  public-key, *see* Public-key cryptography  
  symmetric-key, 74–75
- Customized report, generating, 278
- CVSS score  
  and Oracle's interpretation of problem, 21  
  for security vulnerabilities, 20
- Cyclic Block Chaining (CHAIN\_CBC), 156
- D**
- DAM, *see* Database activity monitoring
- DAMP, *see* Database activity monitoring  
  and prevention
- DAM systems  
  architectures used by  
    interception-based architectures, 287–288
- query-based and log-based architectures,  
    288–289
- end-user credentials, solving problems of  
  in application servers, 312–313
- Oracle instance account, 312
- using dbms\_session to set, 313
- functions provided by, 290–291
- impact on database communications  
  data packets, 309  
  SPAN port, 308  
  traffic encryption, 309–310
- normal behavior and intrusions, monitoring, 297
- overhead resolved by, 285
- real-time alerting, 286
- storage requirements, 310–311  
  *vs.* SIEM systems, 286
- Data access protection, 438–439
- Data backups, *see* Backups
- Database activity monitoring  
  change management, 285  
  intrusion detection, 8  
  through audit policy, 436  
  use case for, 285
- Database activity monitoring and prevention, 285  
  architecture of, 290  
  important use cases for deploying  
    outsourced DBAs and cross-boundary laws, 307  
    privileged user access to sensitive data, 306–307  
    rogue application prevention, 308
- Database auditing, 13
- Database breaches  
  data targeted in, 7  
  discovery of, 6  
  incidents involving, 3–5  
  involving unknown factors, 5  
  time until discovery of, 6
- Database encryption, 154–155
- Database intrusions, 297–298
- Database security  
  change tracking tools, 17  
  comprehensive implementation of, 306  
  and monitoring, national regulations to, 299–305  
  regulations affecting, 7
- Database security project  
  auditing of  
    audit policy, 433–434  
    audit rules, 432–433  
    audit trails, 432  
    report generation, 434–437
- data discovery, 429
- regulatory requirements, 427–428
- report generation, 434–435
- scope of, 426–427
- servers and client connections, 428
- Database Security Technical Implementation Guide,  
  *see* Database STIGs

- Database STIGs  
 exporting data, 12–13  
 generic section, 12–13  
 Oracle-specific section, 13  
 as unclassified document, 14
- Data breaches, 3, 5
- Data discovery, 429
- Data encryption, 152  
 application, database, and storage, 154–155  
 block ciphers, 77–78  
 and import/export using Oracle data pump utilities, 179–180  
 using DBMS\_CRYPTO, *see* DBMS\_CRYPTO  
 using fixed key, 161  
 using TDE, *see* TDE
- Data extrusion, 298
- Data integrity algorithms, 140
- Data-in-transit, 127  
 encryption, performance impact, 149  
 SSL and NDE for, 137
- Data leak prevention, 307
- Data Masking option, Enterprise Manager  
 test data, sanitizing, 22  
 masking format, 23  
 sensitive data, 24  
 status of masking jobs, 25–26
- Data modification attacks, 140
- Data security, data breaches and  
 noncompliance, 2–3
- DBA\_AUDIT\_POLICIES, 231
- DBA\_AUDIT\_TRAIL, 205–206
- DBA\_COMMON\_AUDIT\_TRAIL, 207
- DBA\_FGA\_AUDIT\_TRAIL, 207
- DBA\_PRIV\_AUDIT\_OPTS and DBA\_OBJ\_AUDIT\_OPTS, 208
- DBA privileges, 323
- DBA\_PROFILES, 69
- DBA\_STMT\_AUDIT\_OPTS, 207–208
- DBA\_TS\_QUOTAS, 70
- DBAUD collectors, 257  
 adding, 266  
 architecture of, 259  
 reviewing and controlling, 267–268
- DBA\_USERS, 70
- DBMS\_CRYPTO  
 cryptographic functions, 158  
 decryption functions, 156  
 encryption functions, 155–156  
 for generating MAC, 158–159  
 hash functions, 159  
 key management within  
   in cleartext on wire, 161–162  
   database table, 160–161  
   file system, 161  
   mapping data to keys, 162  
 in Oracle 10g and 11g, 157
- padding functions, 157  
*vs.* DBMS\_OBFUSCATION\_TOOLKIT, 157–158
- DBMS\_FGA.ADD\_POLICY procedure, 225
- DBMS\_OBFUSCATION\_TOOLKIT, 157
- Deep defense, *see* Defense in depth
- Default accounts and passwords, 64–65
- Defense in depth  
 for information systems, 26  
 military strategy, 26
- Denial-of-service (DoS) attacks, 30
- DES, 77
- 3DES, *see* Triple Data Encryption Standard
- Diffie–Hellman key exchange  
 client and server, 80–81  
 in context of public/private keys, 84  
 math behind, 81–82
- Digital signatures  
 originator and recipient, 86  
 private key and public key, 85
- DML handlers, 243
- DML triggers, 235
- DROP USER command, 57
- DUAL\_MODE, 180
- DV, *see* Oracle Database Vault
- DVF.F\$CLIENT\_IP, 395–396
- DVF.F\$MACHINE, 397
- DVF.F\$NETWORK\_PROTOCOL, 396
- DV realms, 387
- DV secure application roles, 399
- Dynamic policy, 372–373

## E

- Elastic defense, *see* Defense in depth
- Electronic Codebook (ECB)  
 CHAIN\_ECB, 156  
 chaining, 78  
 weak, 79
- Embedding passwords, 109
- EMP\_AUDIT records, 245
- EMP\_DML\_HANDLER procedure, 242
- Encryption, 74  
 CPU-intensive operation, 149  
 of data-at-rest, 152  
 of data-in-transit, *see* Network encryption;  
   Switched networks  
 implementation in DAMP, 307  
 sensitive data, 9
- Encryption accelerators, 147
- Encryption algorithms, *see* Cryptography
- Encryption keys  
 access to, 290  
 column

- for EMP, 167
  - encrypting tablespaces with, 171–172
  - for table columns, 164
  - ENCRYPTION** parameter, 179
  - ENCRYPTION\_PASSWORD** and **ENCRYPTION\_MODE**, 180
  - ENCRYPTION\_WALLET\_LOCATION**, 177
  - Entitlement audit reports, queries for producing accounts queries, 354–357
    - hierarchical model of privileges, 348–349
    - object privileges
      - by database account, 352–353
      - granted to PUBLIC, 353–354
      - granted WITH GRANT, 351–352
    - system privileges and admin option values, 349–351
  - Entitlement audits, 335
  - Entitlement management, 9
  - Ethereal, 132
  - EXECUTE** privilege, 320
    - to appropriate users, 341
    - on procedures assigned to PUBLIC, 354
    - RONB, 321–322
  - External executables, 44
  - External password stores
    - Oracle wallet as, 107–109
    - SSL authentication and, 112
  - External procedures
    - configuring listener for
      - listener.ora entry and connection data, 42–43
      - SID\_DESC section, 40–42
    - securing, 43
      - using principle of minimal privileges, 44–45
  - External security overlay, 383
  - EXTRPROC** program
    - securing, 43–45
    - SID\_DESC sections load, 41–42
    - vulnerabilities, 42–43
- F**
- Factors
    - API functions, 395–398
    - default, 394
    - SOURCE\_PROGRAM**, creating, 398–399
  - FAILED\_LOGIN\_ATTEMPTS**, 61
  - Federal Information Processing Standard-140 (FIPS-140)
    - algorithms, 91
    - levels of security and requirements imposed in, 90
  - FGA**
    - audit policies, *see* FGA audit policies
    - audit trail, reading, 231–232
    - of DML and SELECT, 224
    - of EMP, 225
    - performance of, 232–233
    - PL/SQL expression, 229
    - vs.* standard auditing, 223
- G**
- Guardium system
    - change tracking capabilities of, 431–432
    - compliance process, 439
      - connection termination, 440
      - full incident management application, 441–442
    - database security project auditing in
      - audit policy, 433–434
      - audit rules, 432–433
      - audit trails, 432
      - report generation, 434–437
    - functional footprint of, 425–426
- H**
- Handler qualifiers, 229–230
  - Hardware acceleration, 147
  - Hardware Security Modules, 91
    - secure storage of keys, 176
    - TDE supports to, 177
  - Hash algorithm, 60
  - hex characters, in password hash, 66
  - HSMs, *see* Hardware Security Modules
  - Hubs, 134

**I**

Identity theft, loss due to, 246–247  
**INBOUND\_CONNECTION\_TIMEOUT**\_your listener name, 49  
Initialization parameter, 63–64  
  AUDIT\_SYSLOG\_LEVEL, 190  
  REMOTE\_LOGIN\_PASSWORDFILE, 105  
  SEC\_CASE\_SENSITIVE\_LOGON, 65  
  SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS, 62  
Inspection-based architectures,  
  *see* Interception-based architectures  
Instruction Prevention Systems (IPS), 136  
Integrity, 12  
Interception-based architectures, 287  
Internet Protocol Security  
  capabilities, 143  
  setup on Solaris, 145–147  
  setup on windows, 144–145  
Intrusion detection, 291  
IPSEC, *see* Internet Protocol Security  
IT security implementation, goal of, 26–27

**J**

Java Database Connectivity (JDBC) drivers, configuring network encryption for, 139–140

**K**

Kerberos authentication scheme, 115  
  configuring  
    connection to database, 119  
    creating Service Principal (SP) for, 117–118  
    database users, 118  
    Service Table, 118  
  operating principle of, 116–117  
  security features, 119  
Key exchange SSL, 91  
Key management  
  with TDE  
    encryption key for table columns, 164  
    master key, 165  
KILL command, 58

**L**

Listener, *see* TNS listener  
Listener Control Utility (\$ORACLE\_HOME/bin/  
  lsnrctl), 29  
listener.ora, 114

Local OS Authentication, 33–35  
Log-based architectures, 288–289  
Logon packet, 59–60  
Lsnrctl, 31  
  limiting access to, 37–38  
  in Oracle 10g and 11g  
    change\_password command, 34–35  
    connection and status command, 32  
    securing access to, 33  
    set password, 36  
  STATUS and SERVICES commands, 37–38

**M**

MAC duplication, 136  
MAC flooding, 136  
Malformed packets  
  to listener, 49–50  
  in Oracle 11g, 51  
Mandatory auditing  
  “/ as sysoper,” audit records, 215–216  
  changes to audit trail, 216  
  definition of, 213  
  SYS connections, 216  
  Windows event for database startup, 213–214  
  Windows event for sysdba logon, 213, 215  
Mapping compliance requirements, database security,  
  *see* National regulations, database security and monitoring  
Masking format  
  defining, 23  
  reviewing  
    masking script, 24  
    status of, 25  
    sensitive data, 24  
  “Media eavesdropper,” 151

**N**

n-apply CPU  
  security fixes, 21–22  
National regulations, database security and monitoring  
  Australia and Brazil, 300  
  Canada, 301  
  European Union, 301  
  Germany, 301–302  
  Italy, 302  
  Japan, 302–303  
  Russia, South Africa and United Kingdom, 303  
  United States, 303–305  
NDE, *see* Network Data Encryption  
Network access, 13  
Network attacks, 136

- Network cards  
  encrypting, 147
- Network Data Encryption  
  for data-in-transit, 137  
  on server, 140
- Networked communications, methods  
  for encrypting
- IPSEC, *see* Internet Protocol Security
- tunneling protocol, *see* Tunneling protocol
- using ASO and hardware acceleration, 148
- Network encryption; *see also* Network sniffing  
  database communications, 151  
  data integrity, 140  
  for JDBC drivers, 139–140  
  using ASO, 137–138
- Network Intrusion Detection Systems, 136
- Network protocol analyzer, 132
- Network services  
  assigning ACL to, 346–347  
  utility packages and, 344
- Network sniffers, 132, 133
- Network sniffing  
  and intercepting in switched networks  
    ARP spoofing, 135–136  
    database traffic, 134  
    MAC flooding and MAC duplication, 136  
  network packets between client and server  
    card holder information, 130  
    in Ethereal, 133  
    sensitive data, 131–132
- NIDS, *see* Network Intrusion Detection Systems
- NIST CVSS calculator, 20
- NOAUDIT and NOAUDIT ALL, 209–210
- Notification regulations  
  early warning, 247  
  identity theft, 246  
  support in DAM systems, 291
- O**
- Object auditing, 200  
  by ON DEFAULT, 201–202  
  objects and statements, 201  
  views and base tables, 202
- Object privilege  
  by database account, 352–353  
  granted to PUBLIC, 353  
  GRANT option, 317, 351–352  
  to view, 317–318
- oc4j container, 262, 264
- O3LOGON authentication, 59
- O5LOGON authentication  
  changes in, 99  
  first packets from, 120
- and O3LOGON, 99, 102
- password hash, 119
- server in, 99  
  logon using service names, 100  
  response to AUTH\_SESSKEY and AUTH\_VFR\_DATA, 100–101  
  session key and decryption key, 101–102
- Operating system authentication  
  setting initialization parameters for  
    OPS\$JANE, 104  
    OS\_AUTHENT\_PREFIX, 103  
    REMOTE\_OS\_AUTHENT, 102–103  
  users, 102
- ops\$ default, 103
- OPS\$JANE, 104
- Oracle Audit Vault  
  architecture, 256  
  association hierarchy, 257  
  audit data analysis by, 276–278  
  auditing, 282–283  
  AV agents, *see* AV agents  
  AV collectors, *see* AV collectors  
  AV server, *see* AV server  
  to fire an alert, 278–281  
  performance of, 281  
  source, setting up, 264–266  
  storage requirements per, 281–282
- Oracle Call Interface, Java wrapper for, 139
- Oracle CPUs  
  components of, 18  
  database risk matrix in  
    CVSS score, 20–21  
    vulnerabilities, 19  
  listener, 30
- Oracle database  
  activities to secure and audit, 8–9  
  audit records in, 220  
  authentication in  
    administrator, 104–105  
    Kerberos, *see* Kerberos  
      authentication scheme  
    O5LOGON, 99–102  
    at OS level, 102–104  
    RADIUS, 121, 123–124  
    RSA SecureID, 122  
    SSL authentication using ASO, 112–115  
  capabilities, 8  
  data encryption within, *see* Data encryption  
  data security, importance of investing in, 2–3, 7  
  definition, 1  
  external procedures, *see* External procedures  
  fine-grained auditing, *see* FGA  
  flashback queries, *see* Flashback queries  
  hardening  
    activities and configuration options, 11  
    checklists, 27

- guidelines for, *see* CIS Oracle benchmark;
- Database STIGs
- test data sanitization, 22–26
- VA tools for, *see* Vulnerability assessment tools
- intrusion and anomaly detection for, 297–298
- logon process
  - client and database, 59–60
  - hash algorithm, 60
  - password encryption, 59
  - using sysdba and sysoper privileges, 105–106
- and Oracle client, communications in, 127
- packages to send messages and data from, 343–344
- privileges within, *see* Privileges
- security implementation involving, 425
- security patches, *see* Oracle CPUs
- setting up connections to, *see* TNS listener
- standard auditing, *see* Standard auditing
- user accounts
  - altering, 56–57
  - creating, 53–56
  - deleting, 57–58
  - and password policies, 61–63
  - unlocking, 58
- VPD, *see* VPD
- Oracle Database Vault
  - change tracking
    - general tools for, 408
    - for monitoring changes to Oracle executable, 408–409
    - V\$OPTION, 410
  - checks, 399–400
  - command rules, *see* Command rules, securing user activity with
  - default factors, 394–395
  - disabling
    - for maintenance activity, 405–406
    - V\$OPTION and, 407
  - elements making up, 383–384
  - performance
    - factors and auditing, 411
    - realms and command groups, 410
    - rule sets and, 410–411
  - reenabling after maintenance activity, 406–407
  - secure application roles, 399
  - securing data access from DBA access, 384
    - creating realms for, 385–386
    - prebuilt DV realms for, 387
  - security implementations, 411–412
  - security reports available in, 401–403
  - sysdba connections, 403–404
  - useful functions, 395–398
- Oracle Data Dictionary Realm, 387
- Oracle data pump utilities
  - encrypted data import/export using, 179–180
- Oracle Enterprise Manager (EM) Realm, 387
- Oracle 11g
- initialization parameters, 49
- password profile, 61
- secure hash algorithm-1, 60
- TDE-encrypted data, 169
- \$ORACLE\_HOME/bin/tnslsnr, 29
- \$ORACLE\_HOME/network/admin/listener.ora, 29
- \$ORACLE\_HOME/network/admin/sqlnet.ora, 29
- \$ORACLE\_HOME/network/admin/tnsnames.ora, 29
- OracleNet protocol, 128
- Oracle server
  - network connections, 127
  - network encryption between Oracle clients and
    - enabling NDE, 137
    - parameter values, 137–138
- Oracle streams
  - for capturing before/after values
    - apply rule, 243–244
    - audit tables and queue, 240
    - capture process, 241, 244–245
    - handler procedure, 241–243
    - streams privileges, 240
    - stream users, 239
- Oracle wallet, 73
  - creating
    - trusted certificates and CA, 93–94
    - using orapki or OWM, 92
  - displaying contents of, 92–93
  - as external password store, 107
  - master key, 173
  - opening using OWM, 93
- Oracle Wallet Manager
  - applications of, 92
  - command line utility, 73
  - self-signed certificate in, 95
  - viewing wallet using, 113
- Orapki utility
  - error messages, 98
  - orapki wallet command, 92
- OSAUD collectors, 257
  - adding, 266
  - reviewing and controlling, 267–268
- OS authentication, *see* Operating system authentication
- OS\_AUTHENT\_PREFIX, 103
- Output Feedback (OFB), 79
- OWM, *see* Oracle Wallet Manager

## P

- Packet sniffing, 309
- Partial +, 21
- Password
  - case sensitivity, 65–66
  - check for weak and default, 64–65

- complexity, enforcing, 63–64
- hash
  - hex characters, 66
  - protecting, 124–125
  - within USER\$ table, 67
- in Oracle 11g, 61–62
- policies and user accounts
  - grace time, 62
  - profile and lifetime of, 61
  - UNLIMITED as value within, 63
- using impossible, 67–68
- verification function, 64
- PASSWORD command, 58
- Password files
  - adding passwords to, 106
  - creating, 105
- PASSWORD\_GRACE\_TIME, 62
- PASSWORD\_LIFE\_TIME, 61
- PASSWORD\_LOCK\_TIME, 62
- PASSWORD\_REUSE\_MAX, 61
- PASSWORD\_REUSE\_TIME, 61
- PASSWORD\_VERIFY\_FUNCTION, 62, 64
- Patches for database, 18
- Payment Card Industry Data Security Standard (PCI DSS)
  - card holder information and magnetic stripe data, 427
  - database security from, 413–424
  - encryption within, 153
  - general requirements, 152–153
- PL/SQL procedure, 25, 229
- Policy groups, 367
  - driving context, 369–370
  - for HR application and finance application, 368–369
  - testing of, 370–371
- Privilege auditing, 202–203
- Privileged user, 151
  - access to sensitive data, 306–307
  - monitoring, 290
- Privileges
  - column, *see* Column privileges
  - hierarchical model of, 348–349
  - named groups of, *see* Roles
  - object, *see* Object privilege
  - on PL/SQL procedures, 354
  - procedure, *see* Procedure privileges
  - and PUBLIC user group, 342–343
  - view, *see* View privileges
- Procedure privileges
  - definer rights, 320–321, 323
  - invoker rights, 321
  - simple package body, 319
- Production databases, 22
- Profiles, 68
- Provisioning, 256
- Public certificate, 87
- Public-key cryptography, 80
  - encrypting key in, 82
  - to set up symmetric keys, 83
  - two parties communicating using, 83
  - vs.* symmetric-key cryptography, 82–83
- Public-Key Cryptography Standards, 91
- PUBLIC user group and privileges, 342–343

## Q

- Query-based architectures, 288

## R

- RADIUS, *see* Remote Authentication Dial-In User Service
- Realm
  - authorizations, 387
  - creation
    - after adding protected objects, 386
    - before assigning objects/users, 385
    - set of objects in, 386
  - Real-time alerting, 286
- REDO collector, 257
  - adding, 266–267
  - architecture, 260
  - reviewing and controlling, 267–268
- Remote Authentication Dial-In User Service
  - AAA capabilities, 119
  - applications of, 121
  - using ASO, 123–124
- REMOTE\_LOGIN\_PASSWORDFILE, 105
- Replay attacks, 140
- RESOURCE\_COST, 71
- RMAN encryption
  - backups, 182
  - multiple modes of, 181
  - and restoring data, 183
- Rogue application prevention, 308
- Roles
  - assigned to users, 336
  - default
    - assigned role to user, 336–337
    - authorizing, 338
    - multiple roles within session, 337–338
    - recursive privilege structure, 335
- RONB EXECUTE privileges, 322
- Ron's Code number 4 (RC4), 77
- RSA ACE/Server using RADIUS, 123
- RSA digital signatures, 85
- RSA SecureID, 121
- Rule sets, 394

**S**

- SAP R/3 system, 427  
 SCN and flashback, 246  
 SCN\_TO\_TIMESTAMP, 250  
 SCOTT.EMP table, audit trail on, 223–224  
 SEC\_CASE\_SENSITIVE\_LOGON, 65  
 SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS, 62  
 SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS = *n*, 51  
 SEC\_PROTOCOL\_ERROR\_FURTHER\_  
     ACTION, 51  
 SEC\_PROTOCOL\_ERROR\_TRACE\_ACTION, 51  
 Secure application roles, 339–342  
     and VPD for granular access control  
         access based on department, 381–382  
         application context, 379–380  
         assumptions, 378  
         granting privileges, 380–381  
         sample application utilizing, 379  
 Secure Socket Layer, *see* SSL  
 Security alerts, *see* Critical Patch Update  
 Security Incident Event Managers systems,  
     *see* SIEM systems  
 Security-relevant columns, 364–365  
 Security reports, 384  
 SELECT ANY TABLE privileges, 324, 326, 333  
 SELECT statements  
     audit\_condition qualifier for, 228  
     for table columns, 229  
 Self-synchronizing stream cipher, 79  
 Sequence monitoring, 298  
 SES\_ACTIONS column, 197  
 SIEM systems *vs.* DAM systems, 286  
 SQL92\_SECURITY, 357–358  
 SQL injection  
     application authentication bypass using, 292  
     combating  
         guidelines for application developers, 295–296  
         by tools and signatures, 296–297  
     involving insert selects, 294  
     message board, 295  
     using UNION ALL SELECT option, 292–293  
 SQL injector, 296  
 SQLNET.ENCRYPTION\_SERVER and SQLNET.  
     ENCRYPTION\_CLIENT, connection  
     state based on, 137  
 SQLNET.EXPIRE\_TIME, 49  
 SQLNET.INBOUND\_CONNECTION\_  
     TIMEOUT, 49  
 SSH tunnel, setup for  
     editing file.ssh/authorized\_keys, 143  
     key pair on client, 142  
 SSL, 91  
     authentication using ASO server, 112  
     authentication sequence in, 110–111  
     basic steps in, 113–115  
     handshake between client and server, 112  
     for data-in-transit, 137  
 SSL\_SERVER\_CERT\_DN, 115  
 Standard auditing  
     audit records in, 220  
     audit trail generated in, *see* Audit trails  
     before/after values  
         by flashback queries, 247–250  
         by Oracle streams, 239–246  
         by triggers, 236–238  
     as compensating control in DV, 411  
     database activity, 9  
     object, *see* Object auditing  
     per object, 209  
     privilege, *see* Privilege auditing  
     stages involved in, 188  
     statement, *see* Statement auditing  
     successful and unsuccessful, 193–195  
     tables used to view  
         DBA\_PRIV\_AUDIT\_OPTS and DBA\_OBJ\_  
             AUDIT\_OPTS, 208  
         DBA\_STMT\_AUDIT\_OPTS, 207–208  
     throughout without, 212  
     unexpected errors in network layer, 203–204  
     *vs.* FGA, 223  
 Statement auditing  
     AUDIT SESSION and AUDIT NOT EXIST,  
         199–200  
     CREATE TABLE, 198–199  
 STATEMENT\_TYPES, 362  
 Static policy, 362, 372  
 Storage encryption, 154–155  
 Switched networks  
     environment, 134  
     ethernet frame, 134–135  
     network sniffing and intercepting in  
         ARP spoofing, 135–136  
         database traffic, 134  
         MAC flooding and MAC duplication, 136  
     SPAN ports, 135  
 Symmetric-key cryptography  
     ciphers, 74–75  
     schematic representation of, 75  
 SYS\_DEFAULT, 371  
 Syslog.conf controls  
     sample administrator audit trail, 219  
     sample config file, 218, 220  
 Syslog protocol  
     auditing, 221  
     for forwarding log messages, 218  
 System ANY privilege, 324  
     access to schema, 334–335  
     ADMIN option, 333–334, 349–351  
     and DBA privileges, 332  
     O7\_DICTIONARY\_ACCESSIBILITY,  
         334–335

- System hardening  
     change tracking tools in, 17  
     guideline selection for, 14  
     purpose of, 11
- System privilege; *see also* System ANY privilege  
     in Oracle 11g, 325–332  
     UPDATE ANY TABLE, 324
- T**
- Tablespace encryption  
     in HSMs, 177  
     with TDE, *see* TDE
- TCPS listening port, 114
- TDE  
     encrypting columns with  
         algorithm used within, 166  
         data life cycle, 169  
         empno column and sal column, 165–166  
         enabling, 165  
         in foreign key, 170  
         key management in, 164–165  
         limitation, 168–169  
         and primary keys within indexes, 170–171  
         at SQL layer, 168  
         storage overhead, 167  
         table with encrypted columns, 166  
         transparency of, 163, 167–168  
     encrypting tablespaces with  
         AES128, 172  
         master keys and column encryption keys, 171–172  
     with external tables, 178–179  
     importance of, 184–185  
     master keys, management of, *see* TDE master keys, management of  
     supports to HSM-based master keys, 177
- TDE master keys, management of  
     auto-login wallets, 174  
     risk due to losing keys, 175–176  
     wallet  
         alter system set encryption key command, 173  
         opening for encrypted data, 173–174  
         password, compromised, 174–175
- Template rule sets, 394
- Test databases
- Test data, sanitizing  
     application logic and, 26  
     Data Masking option in Enterprise Manager, 22  
         masking format, 23  
         sensitive data, 24  
         status of masking jobs, 25–26  
     problems in, 22  
     tools for, 26
- TLS, *see* Transport Layer Security
- TNS, *see* Transparent Network Substrate
- TNS listener  
     attacks on, 49–51  
     bounce, 108  
     components of, 29  
     log files  
         error messages, 48–49  
         inspecting, 47  
     process execution at OS level, 29  
         on Linux, 31  
         TNS connection, 30  
     properties, limiting ability to change, 39–40  
     securing  
         attacks considered during, 30–31  
         lsnrctl access, 31–39  
         parameters for, 49  
         steps for, 31  
         security alerts, 51–52  
         security patch, 30  
         trace files, 47  
         valid node checking, 46–47  
     vulnerabilities  
         security alerts and cpus addressing, 49–50  
         tnscmd tool and, 52
- TNS protocol attacks  
     malformed packets, 51  
     parameters securing listeners from, 49
- Transparent Network Substrate, 29
- Transport Layer Security, 91
- Triggers  
     for application logic, 235–236  
     capturing before and after values by, 236–239  
         AFTER trigger, 237–238  
         audit records, 238  
         audit table, 236  
         changes in data, 238  
     for creating audit trails, 235  
     transactional boundaries, 236
- Triple Data Encryption Standard  
     data-in-transit, 137
- EEE and EDE, 77
- Trust hierarchies, 86
- Tunneling protocol  
     for encrypting cleartext Oracle sessions between hosts, 141–142
- Oracle traffic, 141  
     setting up, 142–143
- Two-factor authentication  
     asynchronous, 122–123  
     RSA SecureID, 121–122  
     synchronous, 122
- Type-4 JDBC driver connection, enabling encryption in, 139

**U**

## Unix

- administrator audit records, 218
- audit trail in, 190–191
- mandatory auditing, 213, 215
- pcap library, 132
- UNLIMITED TABLESPACE system privilege, 55
- UPDATE ANY TABLE privilege, 324
- UPDATE\_CHECK, 362–363
- User accounts
  - altering, 56–57
  - changing password, 58
  - creation of
    - ALTER USER format and, 56–57
    - CREATE TABLE privileges, 54–55
    - CREATE USER command for, 53–54
    - PASSWORD EXPIRE option, 54
    - UNLIMITED system privilege, 55–56
  - deleting, 57–58
  - as IDENTIFIED EXTERNALLY., 103
  - limiting system resources used by, 68–69
  - and profiles, viewing
    - DBA\_PROFILES, 69
    - DBA\_TS\_QUOTAS and DBA\_USERS, 70
    - USER\_PASSWORD\_LIMITS and USER\_RESOUCE\_LIMITS, 71
  - unlocking, 58
- USER\_ENCRYPTED\_COLUMNS, 166
- USER\_PASSWORD\_LIMITS, 71
- User qualifiers, 199–200
- USER\_RESOUCE\_LIMITS, 71

**V**

- Valid node checking, 46–47
- View privileges, 318–319
- Virtual Private Database, *see* VPD
- VPD
  - FGAC implementation, 359
  - policy groups, *see* Policy groups
  - security policies, *see* VPD security policies

## VPD security policies, 374

- assigning to database object, 374
- to check before and after conditions, 363
- debugging, 374
  - adding indexes, 376–377
  - recursive definitions and, 376
  - SQL traces for, 376
  - V\$VPD\_POLICY view for, 375
- default value for, 372
- for optimal performance
  - context sensitive and shared context sensitive policies, 373
  - dynamic and static policies, 372–373
- qualifiers for, 361–363
- recursion, 376
- row filtering, 359–361
- sensitive column data
  - hiding, 365–367
  - limiting access to, 364–365
- users exempted from, 377–378
- to view defined, 374–375

## Vulnerabilities

- checking for, 15

## Vulnerability assessment tools

- change tracking, 430
- checks performed by, 15
- defining tests to be run in, 430–431
- listener security, 429
- scheduler, 17
- vulnerabilities and CPUs, 17

## V\$XML\_AUDIT\_TRAIL, 207

**W**

## WALLET\_LOCATION parameter, 165

- White lists, 298

- Windump, 132

- Winpcap, 132

- Wired Equivalent Privacy (WEP), 77

**X**

- XOR data, AES algorithm, 76

# HOWTO Secure and Audit Oracle 10g and 11g

Ron Ben Natan

Foreword by Pete Finnigan



Oracle is the number one database engine in use today. The fact that it is the choice of military organizations and agencies around the world is part of the company's legacy and is evident in the product. Oracle has more security-related functions, products, and tools than almost any other database engine. Unfortunately, the fact that these capabilities exist does not mean that they are used correctly or even used at all. In fact, most users are familiar with less than 20 percent of the security mechanisms within Oracle.

Written by Ron Ben Natan, one of the most respected and knowledgeable database security experts in the world, ***HOWTO Secure and Audit Oracle 10g and 11g*** shows readers how to navigate the options, select the right tools and avoid common pitfalls. The text is structured as *HOWTOs* — addressing each security function in the context of Oracle 11g and Oracle 10g.

Among a long list of *HOWTOs*, readers will learn to —

- Choose configuration settings that make it harder to gain unauthorized access
- Understand when and how to encrypt data-at-rest and data-in-transit and how to implement strong authentication
- Use and manage audit trails, and advanced techniques for auditing
- Assess risks that may exist and determine how to address them
- Make use of advanced tools and options such as Advanced Security Options, Virtual Private Database, Audit Vault, and Database Vault

The text also provides an overview of cryptography, covering encryption and digital signatures and shows readers how Oracle Wallet Manager and orapki can be used to generate and manage certificates and other secrets.

While the book's 17 chapters follow a logical order of implementation, each *HOWTO* can be referenced independently to meet a user's immediate needs. Providing authoritative and succinct instructions highlighted by examples, this ultimate guide to security best practices for Oracle bridges the gap between those who install and configure security features and those who secure and audit them.



CRC Press

Taylor & Francis Group

an Informa business

[www.taylorandfrancisgroup.com](http://www.taylorandfrancisgroup.com)

6000 Broken Sound Parkway, NW  
Suite 300, Boca Raton, FL 33487  
270 Madison Avenue  
New York, NY 10016  
2 Park Square, Milton Park  
Abingdon, Oxon OX14 4RN, UK

AU4127

ISBN: 978-1-4200-8412-2



[www.auerbach-publications.com](http://www.auerbach-publications.com)