Database Security: A Technical Primer

Sixth Edition

February, 2025

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document remain at the sole discretion of Oracle.

Foreword

Having been in the security space for over 30 years, the front seat view has been exhilarating. Twenty-five years ago, mostly governments and financial institutions were interested in security while everybody else trusted the administrators, users, and computing environment to keep their data secure. It was only when browsers opened up new vistas for commerce over the net in the 90s that companies began to understand the vital need for security. This new perspective led to SSL, network firewalls, and strong cryptography.

Fast forward to the present, and just like before, we find ourselves living in a dramatically different world where every piece of data is online and available 24/7. To address this new reality, we see many different security technologies protecting various layers of the IT stack, from the applications down to the chipsets.

Hackers have built sophisticated tools along with a thriving underground market to go after everything we have, whether on mobile devices, laptops, file servers, or databases. For most hackers, the target of choice is not a laptop or a spreadsheet–the target is most often a database with hundreds of millions of records. The hackers may try to break in through attacks on the network, applications, operating systems, and databases. They primarily target the users who have legitimate access to those systems. Sometimes, it's the insiders with deep knowledge of data and defenses who attack the systems for nefarious gains.

Why are organizations so vulnerable to attacks? Many might say they don't know where their sensitive data is, where they are vulnerable, and what the fixes might be. They might also fear that the fixes may break their applications or that the insiders may exploit the trust placed in them. Too many organizations stop at securing the perimeter, not recognizing how easily hackers can bypass the network perimeter, get to the databases, and quietly walk away with their data. It is not surprising that, on average, it takes the victims six months to even know that they have been breached, and it also isn't surprising that they typically learn about the breach from customers or law enforcement.

Many information technology, database, and security leaders now realize that securing databases should be one of their most important goals. After all, in most companies, it is their databases that contain most of the sensitive data assets. They also acknowledge that while they would never be able to block every path hackers might take, protecting databases serves their constituents well since every path eventually leads to one.

During the last twenty-five years, I've seen a significant shift in how hackers go after databases. In response, Oracle has built multiple security technologies for securing data at the source–within the database. We have focused on all pillars of security: evaluating the risk posture, preventing the attacks, and detecting/alerting malicious behavior. Industry analysts and security professionals recognize that the Oracle Database provides the industry's most comprehensive security.

This book, authored by my Database Security Product Management team, explains in simple terms the adversaries of today, how they exploit the weaknesses, and how they access your sensitive data. This book is not meant to be a prescriptive cookbook or a manual but rather a quick study into what every Database or Security Director/ VP should know about the security of Oracle Databases. You will learn about multiple assessment, preventive, and detective security controls for databases so that you can provide high-level guidance to your teams on how to shrink the attack surface and keep your databases secure.

Breaches are coming faster than we can imagine, and we must be prepared! Your data is your asset, but unless you protect it well, it could fall into the wrong hands and become a liability. Let's start by securing the source!

Vipin Samar
Senior Vice President, Oracle Database Security Development
February 2025

From the authors' desk

As security product managers, we often hear from customers grappling with the challenge of managing security risks while keeping their databases running 24 * 7 * 365. Some were tasked to address a specific compliance requirement or implement a specific security control, while others were asked to improve the security of their databases. Despite Oracle's comprehensive security portfolio, what became evident was the lack of a cohesive strategic approach to securing databases. What to protect? How? From whom? Recognizing that adversaries rarely adhere to a fixed attack pattern, we advocate for a "defense in depth" mindset.

Moreover, we observed that the responsibility for database security was dispersed across different roles within organizations. Some entrusted DBAs and application administrators with this duty, while others placed it in the hands of network and system security administrators who might not possess a deep understanding of database architecture or available tools. This book serves as a security roadmap within the context of the Oracle Database, catering to security officers, database owners, DBAs, application administrators, system administrators, and security teams.

Instead of presenting a collection of product highlights, we opted for a threat-and-solution perspective. While this approach may lead to multiple mentions of specific products addressing various threats across chapters, in-depth product features are readily available on our website and in our documentation.

After reading this technical primer, we hope you'll gain insight into how the adversaries exploit the vulnerabilities, what database security controls are available to help you secure your databases, and what risks those controls address. Please note that this ebook is not intended to replace product documentation or offer any regulatory advice.

Since the initial publication of this book, its scope has expanded from sixty pages to approximately 200 pages. This expansion mirrors the evolving threat landscape and regulatory environment, as well as the advancements in database security control and capabilities. We've added an executive summary to help navigate this comprehensive resource – perhaps the longest "executive summary" you've ever seen.

We hope this book broadens your perspective on database security, equips you with actionable insights, and helps you secure your data.

Alan Williams
Angeline Dhanarani
Bettina Schaeumer
Ethan Shmargad
Hakim Loumi
Kajal Singh
Manoj Shringarpure
Michael Mesaros
Nazia Zaidi
Pedro Lopes
Peter Wahl
Rich Evans
Roger Wigenstam
Russ Lowenthal

Chapter Summaries

Chapter One: Protecting Data

Protecting data is what this book is all about. Your Oracle Databases hold a significant amount of data, much of it sensitive – intellectual property, personal data, financial information – the list goes on. Protecting that data may be your direct responsibility (perhaps you are the data owner, security administrator, or database administrator), or you may simply be interested in how the data SHOULD be protected. This chapter discusses approaches to a successful database security program at a high level, including program goals and recommendations for dealing with tasks that do not add value to those goals.

Chapter Two: Why you should read this book

Three topics tend to drive most security projects - regulatory compliance requirements, concern about malicious activity, and the need for business agility. It's a rare organization that does not have multiple regulatory mandates they need to comply with. While regulatory compliance drives many security projects, concerns over malicious activity, especially ransomware, takes top-place in most organizations, reaching all the way up into the board room. As with regulatory compliance concerns, there are very few organizations that have not been impacted by the increase in cybercrime – if nothing else, you're likely to have seen one of your competitors victimized. In this chapter we'll set the stage for everything that comes after in the book, explaining common regulatory compliance goals before diving into what's necessary to protect your databases.

Chapter Three: Assessing database security

Today's systems are complex, with many configuration settings that impact security. As recent data breaches have demonstrated, the bad guys will exploit any weakness they can find, and configuration weaknesses are a very popular target. Human errors could leave your database open to everyone, or an attacker could maliciously exploit configuration mistakes to gain unauthorized access to sensitive data. It is critical to regularly assess database security posture, considering recommendations from different regulations, security frameworks, and vendor best practices.

Failing to implement basic security controls may risk exposing personal data, including names, addresses, birth dates, account information, etc. This can have a devastating impact on both your reputation and bottom line. Therefore, you should regularly harden and scan your databases, remediating deviations from security best practices.

Many regulations, such as EU GDPR, PCI DSS, Sarbanes-Oxley, and various breach notification laws, promote regular security assessments on critical systems like databases to reduce IT risks. Multiple organizations, such as the Center for Internet Security (CIS) and the U.S. Department of Defense, have recommendations for security configuration best practices. This chapter describes how Oracle Database security solutions can help evaluate your database security posture quickly, categorize the findings, and recommend suitable action to develop a strategy to keep your databases secure.

Chapter Four: Discovering sensitive data

Before we can protect sensitive data, we have to know where it is. An important step to protect sensitive data is understanding what kind and how much sensitive data a database has and where it is located. This knowledge can be used to implement appropriate security controls to protect data.

This chapter introduces the essential elements of sensitive data discovery and gives you an overview of the Oracle technologies that can be used to discover sensitive data.

Chapter Five: Authenticating database users

A fundamental step in securing a database system is validating the identity of the users accessing the database (authentication).

This chapter discusses how your authentication strategy helps protect the users of databases and the data within from attackers. It also explains how to manage the user accounts, whether locally within the database or with centralized external services, such as a directory service or a cloud identity provider.

Chapter Six: Controlling database access

In addition to validating the identity of users accessing the database (authentication) discussed in the prior chapter, another fundamental step to secure the database is controlling which operations the users can perform (authorization)

This chapter discusses how a robust authorization strategy helps protect the database from mistakes, misuse, and attackers. It also explains how to manage user account authorizations, whether locally within the database or with centralized directory or identity services.

Chapter Seven: Enforcing separation of duties

Cybersecurity and regulatory concerns drive the use of strong security controls for accounts used by insiders and privileged administrative users. Stealing sensitive data using compromised privileged user accounts is the most common attack vector for database breaches, and rogue users can cause more damage than anyone else because of their knowledge of internal systems and processes.

This chapter discusses how you can minimize the losses from compromised accounts and rogue users by implementing separation of duties and least privilege. While a single administrator may want to perform multiple functions for convenience, the ability to divide these duties among multiple users and understand exactly which privileges are in use can dramatically improve security.

Chapter Eight: Minimizing risk from SQL injection

SQL injection is one of the oldest and most frequently encountered database attack methods. Despite years of education and training developers to solve the problem, SQL injection continues to plague data-driven web applications.

This chapter discusses the two main approaches to mitigate the risk of SQL injection in the Oracle Database after the application layer: Network-based Database Firewall or built-in SQL Firewall. We will review the differences between the two approaches and suggest strategies for selecting the approach that best fits your needs.

Chapter Nine: Data-driven authorization

There are times when your application needs to control access to individual rows of data. Building those controls into the application can be costly because changes to the security model also requires changes to the application code. Centrally enforcing fine-grained data access for application users within the database saves you time and effort and can dramatically increase security. Since the database enforces data access policies centrally, those policies are applied equally to all tools and applications that access the data.

This chapter discusses how applications can handle the problem of fine-grained authorization without coding these rules within the application.

Chapter Ten: Masking sensitive data

Most organizations create copies of their production databases for use in application test and development. These non-production copies also may be used for training or user acceptance testing. Each new copy of a production database increases the risk that data within those databases will be compromised. Data Masking helps mitigate that risk.

Data masking hides sensitive data by replacing the original data with realistic-looking but fake data. Data masking may be "static" – where the stored data is modified, or "dynamic" – where only the presentation of the data to users is altered to hide the original data.

This chapter discusses how static data masking can remove sensitive data from non-production environments like test and development, as well as cases where the database is used for training. It further describes how data redaction prevents the proliferation of sensitive data outside the database through various fine-grained access control mechanisms. These tools are integral to a comprehensive data privacy strategy, helping you effectively meet compliance requirements such as PCI-DSS and EU-GDPR.

Chapter Eleven: Data encryption and key management

Encryption is the best technique for protecting against database bypass attacks where an attacker attempts to steal data without ever logging into the database. This could be by capturing data in motion over the network, accessing the underlying files of the database through the operating system, or stealing database backups or exports.

This chapter explains how encryption protects data in motion and at rest for the database. It also discusses considerations for securely storing and managing the encryption keys that ultimately protect the encrypted data.

Chapter Twelve: Database Activity Monitoring

It's important to monitor database activity to support incident investigation, detect potentially malicious behavior, and fulfill regulatory requirements. This can be done either through database auditing or external monitoring of database commands. This chapter helps you evaluate the pros and cons of these approaches and discusses Oracle's solutions for these approaches.

Chapter Thirteen: Using and Managing Audit Data

Monitoring database activity is important, but it can generate audit trails with large amounts of data. Those audit trails are usually scattered across many different databases, making analysis and reporting on the audit data difficult. If you have more than a few databases, you need to transfer the audit data into a consolidated repository for analysis and reporting, as well as creating associated reports and alerts.

This chapter describes multiple approaches to solve this problem depending upon your requirements: cloud service or a dedicated on-premises solution. In both cases, you get a central audit collection across your target databases, providing a broad list of audit reports with flexible filter capabilities to fulfill your requirements and alert capabilities to notify you of specific user activities or unusual behavior.

Chapter Fourteen: Database Security in the Multicloud World

Most organizations work with multiple cloud providers to avoid vendor lock-in, improve redundancy, optimize performance, cost, and services, and compliance with data sovereignty. By leveraging the strengths of multiple cloud providers, you can tailor your cloud environments to meet your specific requirements and maximize the benefits of cloud computing. Multicloud strategies lead to scenarios where one cloud manages identity, another manages applications, the third has security tools, and yet another contains the data.

This chapter reviews the security and management of Oracle Database in a multicloud environment.

Chapter Fifteen: Ransomware and Zero Trust

Today, most security conversations are driven by one of these topics: Regulatory compliance, data breaches, ransomware, and zero-trust. So far in this book we have focused mainly on data breaches and regulatory compliance.

In this chapter, we'll discuss the last two topics, ransomware and zero-trust, from the standpoint of Oracle Database. We'll open with a tactical approach to dealing with ransomware and close with a strategic approach to implementing zero trust.

Chapter Sixteen: Securing the Autonomous Database

Oracle Autonomous Database provides standardized, hardened security configurations that reduce the time and money spent managing configurations across your databases. Many security functions are performed automatically by the Autonomous Database, but there are still areas that you need to manage yourself. This chapter explains what security functions the Autonomous Database does for you, which security functions remain your responsibility, and how to help manage those responsibilities using the tools provided with the Autonomous Database.

Chapter Seventeen: Putting it all together

We've covered a lot of ground and many different types of security controls. Putting all of this together creates the *Maximum Security Architecture* – a blueprint for securing data and minimizing risk. This chapter discusses the maximum security architecture and possible strategies for prioritizing which security projects you tackle first. We also offer a few options for how you might roll new controls out in your organization.

Table of contents

Protecting Data **20**

Data is the universal currency 21

Security is job #1 21

Threat actors and the “Dirty Dozen” 22

Addressing the dirty dozen through data security controls 24

Technical or process control? 26

Coming up 27

Why You Should Read This book **28**

Why should you read this book? 29

The need to protect data has never been greater 29

What do all these regulations require? 30

Encrypt your data 30

Control access to your data 30

Audit access to your data 30

It’s not just regulations 31

It isn’t all bad news 31

Coming up 32

Assessing Database Security **33**

Why is assessment important? 34

Why should you focus on assessment? 34

Evaluate and assess database configuration 34

Assessing configuration security of the Oracle Database with DBSAT 35

DBSAT security assessment reports 36

Assessing your database fleet using Oracle Data Safe 38

Using Data Safe to assess database security 39

Understanding user risk with Data Safe 42

Assessment with Oracle Audit Vault and Database Firewall 44

Configuration Assessment with Enterprise Manager 47

Choosing the right database security assessment tool 50

Reducing the blast radius with Privilege Analysis 51

Security patch management 52

Proactive Maintenance with RUs and MRPs 52

Patching tools 53

Oracle LiveLabs 54

Summary 54

Discovering Sensitive Data **55**

Why is sensitive data important? 56

Why should you focus on discovering sensitive data? 56
What is sensitive data? 56
Discovering sensitive data using DBSAT 57
Discovering sensitive data using Data Safe 59
Discovering sensitive data using Audit Vault and Database Firewall 61
Discovering sensitive data using Enterprise Manager 62
Discovering sensitive columns and referential relationships 63
Oracle LiveLabs 65
Summary 66
Authenticating Database Users **67**
Why is authentication important? 68
Why should you focus on authentication? 68
Database authentication methods 69
Making users resistant to attacks 70
Don’t store plain-text passwords 70
Don’t share accounts and passwords 71
Increase the strength of passwords 71
Simplify account administration 73
Protecting your users from getting hacked 73
Protecting business users 73
Protecting DBA accounts 74
Protecting application accounts 74
Protecting security administrator accounts 75
Oracle LiveLabs 75
Summary 75
Controlling Database Access **76**
Why is access control important? 77
Why should you focus on access control? 77
Database Privileges 77
Database Roles 79
Types of database users 79
Who can do what in your database? 81
Managing users and their privileges centrally 83
Protecting database accounts 84
Oracle LiveLabs 86
Summary 86
Enforcing Separation of Duties **87**
Why is separation of duties important? 88
Why should you focus on separation of duties? 88

Multiple people? Or multiple accounts? 88

Controlling powerful administrators 88

Enforce separation of administrative privileges 88

Control the database owner account 89

Implement a trusted path 89

Enforce least privilege 89

Do not share accounts 89

Safeguard the audit 89

Enhancing separation of duties with Oracle Database Vault 90

Block administrator access to data with Oracle Database Vault 90

Enforce a trusted path to data 91

Enforce separation of duties within Database Vault 93

Enforce separation of duties with containerized databases 94

Control SQL database commands with Oracle Database Vault 95

Break-Glass scenarios 96

Operationalizing Oracle Database Vault 97

Database Vault and packaged applications 98

Oracle LiveLabs 98

Summary 98

Minimizing Risk from SQL Injection **99**

Why is SQL injection important? 100

Why should you focus on SQL injection? 100

What is SQL injection? 100

Approaches to addressing SQL injection attacks 101

Network-based Database Firewall 102

Reporting on Database Firewall Violations 103

Managing Database Firewall 104

Database-resident SQL Firewall 104

Reporting on SQL Firewall violations 105

Administering SQL Firewall 105

Deciding which to use: Database Firewall or SQL Firewall 105

Oracle LiveLabs for hands-on experience 106

Summary 106

Data-Driven Authorization **107**

Why is data-driven authorization important? 108

Why should you focus on data-driven authorization? 108

What is data-driven authorization? 108

Data-Driven, fine-grained authorization 108

Challenges with implementing data authorization in applications 109

Controlling data access using Virtual Private Database 110

Controlling data access using data labels 110

Controlling data access using Real Application Security 113

Which data-driven access control should I use? 115

Oracle LiveLabs 116

Summary 116

Masking Sensitive Data 117

Why is masking sensitive data important? 118

Why should you focus on masking sensitive data? 118

What is data masking? 118

Use cases for static data masking 119

Use cases for data redaction 119

Static data masking 119

Data redaction 120

Data subsetting 121

Masking formats 121

Masking techniques 122

Masking sensitive data using Oracle Data Safe 124

Masking reports 124

Masking sensitive data using Oracle Data Masking and Subsetting 124

Data masking 124

Differences between Data Masking and Subsetting & Data Safe 126

Masking data dynamically 126

Use cases for redaction 126

Data Redaction 127

Data Subsetting 131

Oracle LiveLabs 133

Summary 133

Data Encryption and Key Management 134

Why is encryption important? 135

Why should you focus on encryption? 135

Encrypting data in motion – database connections 135

Encrypting data at rest – database files and backups 136

Transparent Data Encryption 138

TDE and the Database Configuration Assistant (DBCA) 139

Migrating clear-text data to encrypted tablespaces 140

TDE two-tier key architecture 140

Key Management 142

Local key management with Oracle Wallets 142

Centralized key management with Oracle Key Vault 142

Key Vault with DBMS_CRYPTO 144

Key Vault beyond encryption 144

Key Vault APIs for scripting and automation 145

Oracle LiveLabs 145

Summary 145

Database Activity Monitoring 147

Why is database activity monitoring important? 148

Why should you focus on activity monitoring? 148

What is database activity monitoring? 148

Oracle Database with unified auditing 149

What should you audit? 149

Security-relevant events auditing 150

Privileged user activity auditing 150

Sensitive data access auditing 151

Effective audit policies 152

Audit policy provisioning from Data Safe or AVDF 153

Transitioning from legacy to unified auditing 154

Oracle LiveLabs 154

Summary 154

Using and Managing Audit Data 155

Why is managing and using audit data important? 156

Why should you focus on managing and using audit data? 156

What is audit data management? 156

How do you effectively use audit data? 157

Managing and using audit data with Oracle Data Safe 157

Audit insights 159

Activity auditing dashboard 159

Audit reports 160

Audit retention periods 161

Alerts 162

Auditing with Audit Vault and Database Firewall (AVDF) 163

External monitoring 163

Activity data collection 164

Audit reports 164

Before/after data value changes 166

AVDF special support for Oracle Database 166

Sensitive data discovery 167

Oracle LiveLabs 167

Summary 167

Database Security in a Multicloud Environment 168

Why is multicloud important? 169

Why should you focus on multicloud? 169

Identity cloud service 169

Managing security across clouds 170

Oracle Database is available in most clouds 171

Multicloud service access 171

Multicloud identity management for Oracle Database 171

Oracle Cloud Infrastructure Identity and Access Management (IAM) 171

Microsoft Entra ID Integration 172

Oracle LiveLabs for multicloud integration 173

Simplify User Management by integrating Oracle Autonomous Database and Oracle Cloud Identity 173

Summary 173

Ransomware and Zero Trust 174

Why is ransomware important? 175

Why should you focus on Zero Trust? 175

What is ransomware? 175

Protecting databases from ransomware 175

Mitigating the risk of data destruction 176

Mitigating the risk of data theft 177

Plan for success 177

What is Zero Trust 177

Typical Zero Trust projects 178

Applying Zero Trust to the database 179

Assess your configuration and monitor for configuration drift 179

Minimize the attack surface 180

Encrypt data at rest and in motion 180

Strengthen authentication 181

Control access to data 182

Monitor database activity 182

Is that all there is to zero trust? 183

Use a two-pronged approach to address ransomware risk 183

Oracle LiveLabs 183

Securing the Autonomous Database 184

Why is Autonomous Database security important? 185

Why should you focus on Autonomous Database security? 185

Why Autonomous Database? 185

The security benefits of Autonomous Databases 185

The security capabilities of Autonomous Databases 186

Shared responsibility 187

Oracle LiveLabs 188

Summary 188

Putting it all together 189

Index 193

Appendix: Tools – Features, Options, Products, and Packs 198

Database Security Assessment Tool (DBSAT) 198

Oracle Data Safe 198

Enterprise Manager Database Lifecycle Management 198

Privilege Analysis 198

Native Network Encryption 198

Transport Layer Security 199

Centrally Managed Users 199

Enterprise User Security 199

Traditional Auditing 199

Fine-Grained Auditing 199

Unified Auditing 199

Enterprise Manager Data Discovery 199

Oracle Advanced Security 200

Oracle Key Vault 200

SQL Firewall 200

Oracle Database Vault 200

Oracle Audit Vault and Database Firewall 200

Oracle Data Masking and Subsetting 200

About the authors 201

Acknowledgments 203

List of figures

Figure 1-1: Types of personal data 21

Figure 1-2: Threat actors 22

Figure 1-3: How hackers attack the database 24

Figure 1-4: A goal-based approach to combatting the Dirty Dozen 26

Figure 3-1: Database Security Assessment Tool (DBSAT) 36

Figure 3-2: DBSAT report summary 37

Figure 3-3: DBSAT Sample finding - Users with the DBA role 38

Figure 3-4: Oracle Data Safe provides essential security for Oracle Databases, both in the cloud and on-premises 39

Figure 3-5: Data Safe assessment summary 39

Figure 3-6: Data Safe security assessment findings 40

Figure 3-7: This finding needs to be evaluated to assess its risk 41

Figure 3-8: Data Safe: Configuring acceptable risks 41

Figure 3-9: Finding modified risks is easy 42

Figure 3-10: Security assessment risk modification report 42

Figure 3-11: Data Safe: User risk assessment 43

Figure 3-12: Data Safe: User profile insight 44

Figure 3-13: Audit Vault and Database Firewall security posture management 45

Figure 3-14: AVDF: detailed assessment report 46

Figure 3-15: AVDF: Detailed information about an individual finding 46

Figure 3-16: AVDF: Configuring stored procedure auditing 47

Figure 3-17: Oracle Enterprise Manager Compliance Dashboard 48

Figure 3-18: Enterprise Manager general security reports 48

Figure 3-19: Enterprise Manager compliance results for the STIG standard 49

Figure 3-20: Configuration Drift and Consistency Management Dashboard 50

Figure 3-21: Report on DBA_UNUSED_PRIVS 51

Figure 3-22: Report on DBA_USED_PRIVS showing object access 52

Figure 4-1: Examples of sensitive data categories and types 57

Figure 4-2: Recommendations for protecting data 59

Figure 4-3: Data discovery report in Oracle Data Safe 60

Figure 4-4: Data discovery sensitive data models 61

Figure 4-5: Sensitive data report 62

Figure 4-6: Sensitive data discovery in Enterprise Manager 63

Figure 4-7: Schedule a sensitive column discovery job 64

Figure 4-8: Sensitive columns discovered using Enterprise Manager ADM 64

Figure 4-9: Referential relationships 65

Figure 4-10: Results of data discovery are useful in configuring other security controls 65
Figure 5-1: Granting proxy privileges 71
Figure 5-2: Connecting using proxy privileges 71
Figure 5-3: Sample password profile 72
Figure 6-1: Scope of privileges 77
Figure 6-2: Granting an object privilege 78
Figure 6-3: Granting a schema privilege 78
Figure 6-4: Granting a system privilege 78
Figure 6-5: Granting an administrative privilege 78
Figure 6-6: Granting an administrative privilege 79
Figure 6-7: Sample role and privilege hierarchy 79
Figure 6-8: Analyzing a user's privileges with Data Safe 82
Figure 6-9: Analyzing a user's privileges with AVDF 83
Figure 6-10: Centrally Managed Users (CMU) 84
Figure 7-1: Database Vault Realms block administrator access to data 91
Figure 7-2: Database Vault uses rules to make access control decisions 92
Figure 7-3: Oracle Database can have multiple realms in the same database 92
Figure 7-4: Operations Control protects pluggable databases 94
Figure 7-5: Database Vault Command Rule 96
Figure 8-1: SQL injection attack pattern 100
Figure 8-2: Approaches to combating SQL injection 101
Figure 8-3: Database Firewall blocks SQL injection attempts 102
Figure 8-4: Oracle SQL Firewall is built into the Oracle Database kernel 104
Figure 9-1: Fine-grained access control applied to a table 109
Figure 9-2: VPD fine-grained access control policy automatically blocks access to salary data 110
Figure 9-3: Label Security components: levels, compartments, groups 111
Figure 9-4: Label component organization 111
Figure 9-5: Using the group component in Label Security 112
Figure 9-6: Using the compartment and group components in Label Security 112
Figure 9-7: Using the level, compartment, and group components in Label Security 113
Figure 9-8: Controlling data access with Real Application Security 114
Figure 9-9: RAS access control list (ACL) 115
Figure 10-1: Static data masking 120
Figure 10-2: Data redaction 120
Figure 10-3: Predefined masking formats in Oracle Data Safe 122

Figure 10-4: Masking formats and techniques in action 123
Figure 10-5: Data Safe masking reports summarize key information from masking jobs 124
Figure 10-6: Overview of Data Masking in Enterprise Manager 125
Figure 10-7: Masking modes for Data Masking and Subsetting 125
Figure 10-8: Data Redaction policy creation example for partial redaction 128
Figure 10-9: Data Redaction formats 129
Figure 10-10: Unredacted application data 129
Figure 10-11: Redacted application data 130
Figure 10-12: Data subsetting strategies 132
Figure 11-1: Encryption options 137
Figure 11-2: Users and applications work against decrypted data in database memory 139
Figure 11-3: Creating an encrypted tablespace 139
Figure 11-4: Online tablespace encryption 140
Figure 11-5: Offline tablespace encryption 140
Figure 11-6: Two-tier key architecture 141
Figure 11-7: TDE encryption keys 142
Figure 11-8: Oracle Key Vault securely delivers MEKs to Oracle Databases on demand. 143
Figure 11-9: Oracle Key Vault manages Oracle TDE keys as well as other keys and security objects 145
Figure 12-1: Enabling an audit policy 150
Figure 12-2: Auditing local access to the database 151
Figure 12-3: Auditing all activity by a user 151
Figure 12-4: Auditing activity on sensitive data 152
Figure 12-5: Auditing activity from outside of an application 152
Figure 12-6: Audit policy provisioning from the AVDF Console 153
Figure 13-1: Data Safe audit data global retention settings 158
Figure 13-2: Data Safe individual database retention settings 158
Figure 13-3: Data Safe audit insights 159
Figure 13-4: Data Safe example of a custom alert policy 162
Figure 13-5: Oracle Audit Vault and Database Firewall 163
Figure 13-6: AVDF Audit Insights 166
Figure 14-1: Multicloud environment 169
Figure 14-2: OCI database integration with IAM 172
Figure 14-3: Oracle Database integration with Azure AD 173
Figure 15-1: CISA Zero Trust Maturity Model Pillars 178
Figure 15-2: Typical zero trust projects for Oracle Database 179
Figure 17-1: Maximum Security Architecture 190

Figure 17-2: Example of implementing security controls based on system sensitivity 192

List of tables

Table 1-1: The Oracle Database Security Dirty Dozen 23

Table 1-2: Examples of security goals and tasks 25

Table 2-1: Regulatory-influenced controls 31

Table 3-1: Difference between RUs and MRPs 53

Table 4-1: Sensitive Data Report - Category Summary 58

Table 5-1: Database user authentication and authorization methods 70

Table 6-1: Data dictionary views for database roles and privileges 81

Table 8-1: Deployment Modes of Database Firewall 103

Table 10-1: Simple masking formats 121

Table 10-2: Masking techniques 123

Table 10-3: Predefined redaction formats 131

Table 10-4: Subsetting conditions 132

Table 13-1: Data Safe audit reports 161

Table 13-2: Alert policies in Oracle Data Safe 162

Table 13-3: Built-in reports in AVDF 165

Chapter One

Protecting Data

Data is the universal currency

Cybersecurity incidents impact organizations at an astonishing rate. Every day, there are new stories about a service provider mishandling subscribers' personal information, an employer losing employee HR records, or a government contractor exposing sensitive intellectual property. Data is the universal currency, and bad actors often leverage stolen data for financial or political advantage for years after a breach. Where do organizations keep this valuable data? At the end of the day, data is primarily stored and managed in databases. This is why it's so important to protect those databases!

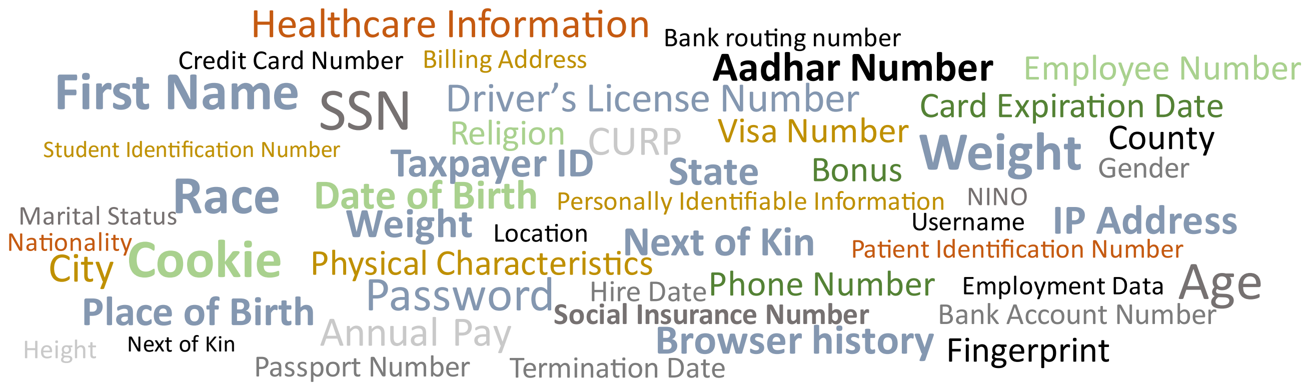


Figure 1-1: Types of personal data

Protecting data is what this book is all about. Your Oracle Databases hold a significant amount of data, much of it sensitive – intellectual property (e.g., source code, patents), regulated data (e.g., financial transactions, healthcare records), and business-critical information (e.g., customer insights, operational data). While some data may seem insignificant, remember that one person's trash can be another person's treasure. In other words, data that is unimportant to one organization can be very attractive to cybercriminals. Protecting that data may be your direct responsibility as the data owner, security administrator, or database administrator - or you may simply want to understand how data SHOULD be protected. This book takes you through the various aspects of Oracle's defense-in-depth security for databases and provides a high-level overview of how different controls work and the types of protection they provide.

Security is job #1

At a recent Oracle CloudWorld, I spoke to a group of DBAs from a very large financial institution. They shared that more than half of their time is now spent on security tasks—a stark contrast to a decade ago, when database administration tasks like performance tuning and storage management dominated their workload. This shift has been driven by regulatory pressures (GDPR, CCPA), rising threats like ransomware, and high-profile breaches.

Back then, security-related tasks received little (if any) time because other tasks were given a higher priority. The priorities have shifted, and security is now frequently a higher priority task than performance, and, in some organizations, even takes precedence over availability!

What are those security-focused tasks? If security is job one, what SHOULD we be working on to better protect our data? If there were a perfect formula that would result in a completely secure system, then our job would be easy – we'd all follow the formula, and there would never be another data theft. No such formula exists! Instead, we have guidelines, benchmarks, and best practices that we can apply to reduce risk, keeping in mind that the bad guys are always out there, working tirelessly to accomplish THEIR goals of breaking into our systems, disrupting our business, and stealing our data. Our security-related tasks should be focused on blocking the attackers – narrowing or closing gaps or weaknesses that can be used against us. To understand those gaps and weaknesses, you need to understand your own system's vulnerable points and how attackers leverage them.

Tools and methods used to break into a system vary widely depending upon the attackers. They may rely on readily available toolkits to exploit unpatched systems, or they may use advanced methods where hackers penetrate a network, search for vulnerabilities, and then covertly exfiltrate data from servers. There are even documented cases of physical penetration – criminal or nation-state actors placing agents into an organization as contractors or employees in order to steal data. These attacks can go undetected for weeks, months, or even years.

Threat actors and the "Dirty Dozen"

Perimeter security solutions like network firewalls were once considered sufficient for protecting internal systems and repositories such as databases from data theft. However, the threat environment has grown to the point where few organizations depend solely on firewalls to keep the bad guys out of their data.

The most effective way to protect data is to enable security controls at multiple levels of the application stack. If an attacker circumvents one security control, additional controls can address the threat. We describe this approach as defense-in-depth. To understand why a defense-in-depth approach to database security is essential, we should examine the threat actors who want your data and how they try to get it.

Threat actors can be broadly divided into "outsiders" and "insiders." Outsiders vary widely in their level of skill and resources. They include everyone from lone "hacktivists" and cyber criminals seeking business disruption or financial gain to criminal groups and nation-state-sponsored organizations seeking to perpetrate fraud and create disruption at a national scale. Insiders include current or former employees, curiosity seekers, and customers or partners who take advantage of their position of trust to steal data. Both groups' targets may include trade secrets as well as personal, financial, and regulated data.

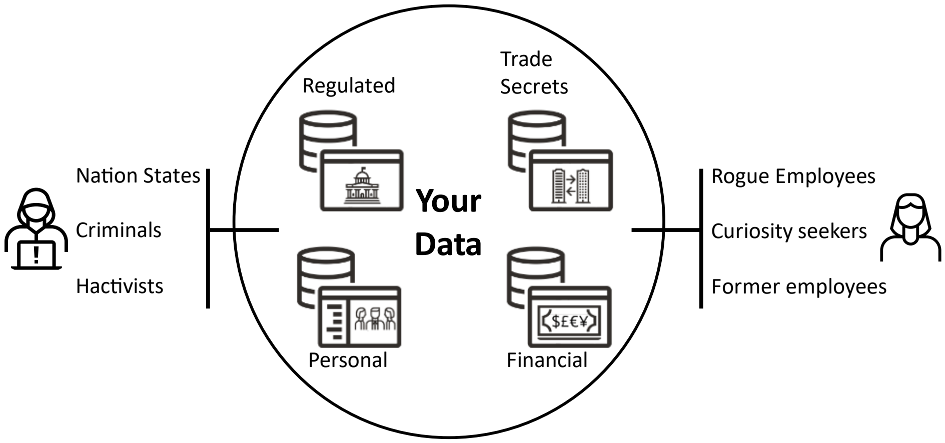


Figure 1-2: Threat actors

What techniques do these threat actors use to compromise data? Many information security professionals are familiar with the OWASP Top Ten. OWASP stands for the Open Worldwide Application Security Project — an online community founded in 2001 focused on improving application security. The OWASP Top Ten aims to raise awareness about application security by listing the most critical security risks to web applications according to broad consensus. This list is updated regularly by OWASP as the threat environment evolves. Part of the value of the OWASP Top Ten is that it guides web administrators and developers in where to spend their effort and resources to deploy more secure applications. In this way, it is an essential step towards a more secure web infrastructure.

Similarly, we have a list of the twelve most common database security risks that we call the “Dirty Dozen.” The items on this list are a hacker’s “tool chest” of tactics, techniques, and methods they might use to compromise the data stored in databases. These tactics include:

- Exploiting unpatched systems or misconfigured databases to bypass access controls.
- Escalating run-time privileges by exploiting vulnerable applications.
- Searching for sensitive data in unprotected databases, applications, and systems.
- Stealing the credentials of a privileged administrator or application user.
- Accessing accounts through password guessing or exploiting careless credential management.
- Exploiting application weaknesses with techniques like SQL injection.
- Exploiting unprotected systems as a bridge to launch attacks against more sensitive systems.
- Creating rogue user accounts on systems as a base for reconnaissance and possible escalation of privilege.
- Targeting copies of live production data used in development and test systems where the data is typically not as well protected as in production systems.
- Accessing unencrypted database system files on the disk or in backup files.
- Encrypting data or stealing the encryption keys from already-encrypted data, rendering it inaccessible to users, and demanding a ransom.

A complete list of the Dirty Dozen appears in Table 1-1 below.

A peek inside the Hacker’s tool chest:

THE DIRTY DOZEN

The most common database security risks

1. Insecure configuration and configuration drift
2. Unpatched and out-of-date systems
3. Lack of a consistently enforced security policy
4. Lack of visibility into sensitive data placement and quantity
5. Overprivileged database users and administrators
6. Weak authentication and shared accounts
7. SQL Injection vulnerabilities and insecure application design
8. Trusting vulnerable networks
9. Insufficient or inefficient monitoring and auditing
10. Sensitive data proliferation to non-production databases
11. Unprotected servers and database backups
12. Insecure encryption keys and secrets

Table 1-1: The Oracle Database Security Dirty Dozen

Attackers attempt to exploit those risks at any point of contact they can make with the database, as illustrated below in Figure 1-3. The most common attacks are against accounts with access to the database (users, application service accounts, and administrators), frequently using stolen login credentials. SQL injection attacks against applications or directly against database packages are also common because attackers generally find it easier to reach an application than to attempt a direct connection to the database or database server. Similarly, if an attacker has access to network segments where database traffic is flowing, they may simply sniff unencrypted network communications to steal data.

Once an attacker can create a session to the database or can log directly onto the database's server, they may begin to exploit unpatched database vulnerabilities or directly attack the database through the server. A now-common form of direct attack is data scraping, where the attacker extracts and exfiltrates the underlying database files without attempting to connect to the database through a session. This is how most ransomware attacks will steal data from the database.

The same attack points are available in database copies, like those made for test and development purposes. In targeted attacks where the attacker is aware of the database and actively working to break in (as opposed to opportunistic attacks like ransomware, where the database is normally stolen during a broader attack against IT infrastructure), attackers tend to prefer working against data copies because they are typically less well-defended (and monitored) than production systems.

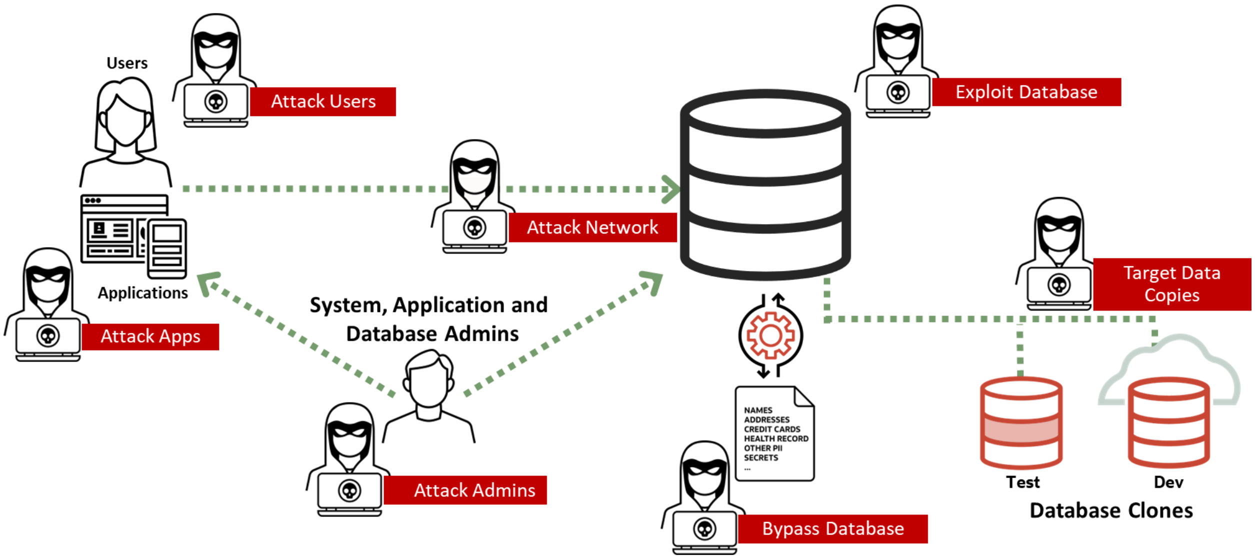


Figure 1-3: How hackers attack the database

Addressing the dirty dozen through data security controls

A well-structured data security posture helps mitigate the dirty dozen risks. Successful approaches incorporate multiple layers of security controls to provide defense-in-depth protection from threats. How should you get started? I recommend concentrating first on your security goals and then identifying tasks that support those goals. Database security goals can usually be divided into these four broad categories:

1. Assess security posture
2. Control access to data
3. Monitor user activity
4. Protect data against theft

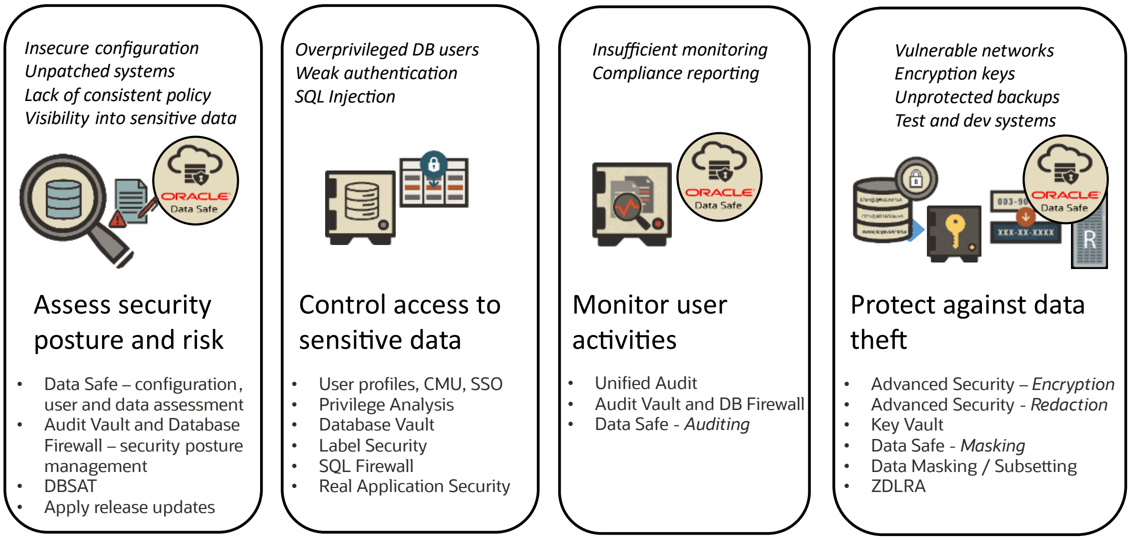
To mitigate database security risks, organizations must establish clear security goals and align them with actionable tasks. A goal defines the desired security outcome (e.g., ensuring secure user authentication), while a task is a specific action taken to achieve that goal (e.g., implementing multifactor authentication). Table 1-2 highlights examples of high-impact security tasks, prioritized by their effectiveness in reducing risk.

Category	Goal	Task	Threat addressed
Assess security posture	Ensure systems do not deviate from their approved security baseline	Monitor databases for configuration changes that introduce risk	Insecure configuration
Monitor user activities	Detect inappropriate activity by privileged users	Audit all DBA activity	Insufficient or inefficient monitoring and auditing
Protect data against theft	Remove risk from test and development environments	Scramble sensitive data in test and development environments	Sensitive data proliferation to non-production databases
Control access to data	Strengthen user authentication to prevent compromise of accounts	Implement multifactor authentication for the database	Weak authentication and shared accounts

Table 1-2: Examples of security goals and tasks

If you find yourself spending time on tasks that are not relevant to a specific goal, then it's likely you are in one of these situations:

1. You are satisfying an externally mandated regulatory requirement—it doesn't matter that it may not make sense in the context of your goals; you are required to do this task because the law says to. Accomplish the requirement as efficiently as possible and return to value-added tasks as quickly as you can.
2. You are satisfying an internally mandated regulatory requirement. First, check yourself—is it not relevant to your goals because you don't understand the goals and requirement? Or is it truly irrelevant? If the latter, is your organization open to change, possibly freeing up time for value-added tasks?
3. Your goals no longer reflect the current reality. This is a very common occurrence in our line of work, where technology constantly matures, tactics evolve, and new threats occur near-daily. Reevaluate your goals and reprioritize your efforts accordingly.
4. You've been diverted into a task that has little or no benefit to your organization. Depending on how you got here and why, you might want to continue working on the task (perhaps the task isn't relevant to your team's goals but is important to another team?), or you might want to pause work on this task and switch to a value-add task.



Insecure configuration
Unpatched systems
Lack of consistent policy
Visibility into sensitive data

Assess security posture and risk

- Data Safe – configuration, user and data assessment
- Audit Vault and Database Firewall – security posture management
- DBSAT
- Apply release updates

Overprivileged DB users
Weak authentication
SQL Injection

Control access to sensitive data

- User profiles, CMU, SSO
- Privilege Analysis
- Database Vault
- Label Security
- SQL Firewall
- Real Application Security

Insufficient monitoring
Compliance reporting

Monitor user activities

- Unified Audit
- Audit Vault and DB Firewall
- Data Safe - *Auditing*

Vulnerable networks
Encryption keys
Unprotected backups
Test and dev systems

Protect against data theft

- Advanced Security – *Encryption*
- Advanced Security - *Redaction*
- Key Vault
- Data Safe - *Masking*
- Data Masking / Subsetting
- ZDLRA

Figure 1-4: A goal-based approach to combatting the Dirty Dozen

Once you've determined your goals and the tasks necessary to support those goals, what remains is a matter of selecting the appropriate control (technology or process-driven) and implementing the control.

Technical or process control?

When trying to perform a task related to the Oracle Database, there are almost always multiple ways to accomplish the task. For example, suppose you wanted to monitor databases for configuration changes that introduce risk (this was the first example in our table relating tasks to goals and the dirty dozen). You might:

- Publish a policy that says your databases will be configured a certain way and that no changes will be made to the database configuration. Require each DBA to acknowledge their receipt of the policy and agreement to comply with it. This is an example of a *process control*. You are expecting your database administrators and users to follow the process you set in place. As long as they follow the process, your control will function as designed.
- Write a script that checks database parameters that you've identified as important from a security context. Then run that script against each of your databases and verify that all of the parameters are set appropriately. If you're a clever scriptwriter, you could even build your script in such a way that the script output gave you a pass/fail for each parameter. This can be better than just publishing a policy because you are checking the actual results instead of relying on people to remember the policy. This is an example of a *technical control* (your script) that augments a *process control* (setting a policy that the script be run periodically with the results of the script reviewed and assessed for compliance).
- Download a tool to scan your database for security issues. You might use the Oracle Database Security Assessment Tool (DBSAT - you'll learn more about this later in the book), or you might download a third-party tool like the Center for Internet Security (CIS) benchmark for the Oracle Database. Then, run that tool against your databases and verify that all the parameters are set appropriately. This is another example of a *technical control* (DBSAT) which augments a *process control* (setting a policy that DBSAT be run periodically with the DBSAT reports reviewed and assessed for compliance). Using a vendor-supplied tool can be better than writing your own script because someone else has already done the work of identifying which parameters and settings need to be checked.

- Purchase a tool that scans your databases for security issues automatically. Configure it to check the databases periodically (daily, weekly, or monthly are common) and notify you via email if there's any drift away from your preferred security posture. You might use something like Oracle Data Safe, Oracle Audit Vault and Database Firewall, or a third-party tool like IBM Guardium. This is an example of a *technical control* that operates independently of process. This approach is usually better than manually running a downloaded tool (or executing your own scripts) because the tool does the work of performing the scan and looking for changes.

In these examples:

Publish a policy	Process control
Write a script	Process control supplemented with technology (your script)
Download and manually run a tool	Process control supplemented with technology (vendor's tool)
Purchase a tool that automates the process	Technical control

Note that the middle two examples blend technology and process. You have a technical component (script or tool) that is being run manually, and the results from the tool are interpreted manually.

Purely process-driven controls are almost always ineffective. As an auditor, I evaluate process-based security controls by sampling a statistically relevant subset of databases. For smaller environments, such as those with 100 databases, I might verify each one. In larger deployments, like those with 10,000 databases, I typically select a representative random sample, often around one or two percent (100-200 databases in this case), to assess compliance.

If you use a process control supplemented by technology, I will validate the tool's effectiveness (is it something I'm already familiar with as an auditor, or is it new to the marketplace or developed internally?), review your process for running the tool, and then sample a statistically relevant portion of the available databases to ensure the tool was run and the results reviewed.

If you use a purely technical control, I will validate the tool's effectiveness and ensure that it is being used for all of your databases. This is usually much quicker than evaluating a purely process-driven control, and your organization's audit costs will decrease as a result.

If you deal with external auditors, you may already be intimately familiar with the difference in how an auditor views a process control compared to a technical control. The process control may be enough to get you through the audit, but the auditor is going to spend a lot more time with you to judge the control's effectiveness before signing off on it.

Note: I highly recommend the ISACA Certified Information Systems Auditor (CISA) course of study if you work with security assessments and audits. The CISA certification offers great insight into running an effective assessment program.

You might say that a technology solution is always better – but suppose you have only a single Oracle Database. Just one. In that case, you'd probably spend more time and effort setting up and maintaining a purchased solution than you would just downloading DBSAT and running it once a week. So, you'd probably be better off with a technology-supported process control than a pure technical control.

Coming up

Let's examine what drives most security projects. In the next chapter, we'll discuss the three most common topics at the heart of security initiatives.

Chapter Two

Why You Should Read This book

“If cybercrime was a state, it would be the third largest economy in the world after the US and China.”

— Edi Rama, Prime Minister of Albania, speaking at the 53rd annual meeting of the World Economic Forum, January 2023

Why should you read this book?

Since you’ve gotten this far, I’m guessing that you have some responsibility for securing data. It’s likely that some of that data is in a database, and probable that you are using Oracle Database. Chances are you're busy. You have a lot to do, and never enough time to do it all in. So why should you expend some of your valuable time reading this book? Because if I’ve done my job well, what I’ve written here will make your life easier in the long run, saving you time and effort. As a side benefit, you’ll make your organization more secure and reduce the chances of security or regulatory issues related to your Oracle Databases. If you are not using Oracle Database, but still need to secure OTHER databases, don’t despair. The fundamentals of security are the same for all databases, you’ll just want to skip the parts where I talk about specific Oracle Database-centric features and products.

The need to protect data has never been greater

In the first chapter, I discussed the constantly expanding threat environment. I assure you that (if anything), I understated the severity of the problem. In 2024, the World Economic Forum valued the global cybercrime economy at USD $9.5 trillion. If cybercrime were a nation-state, it would rank as the third-largest economy globally, following the United States and China.

When a problem is that big and shows no signs of abating naturally or being solved by private industry, governments step in to try and correct the problem. Even if it weren’t for the damage to their economies, in representative forms of government there is also the concern of citizens demanding action and punishing non-responsive officials at the ballot box. For these reasons, we operate in an increasingly stringent and fast-evolving regulatory landscape. The United States alone has more than 100 different privacy and data security laws that attempt to protect citizen’s data, and in the European Union, the venerable General Data Protection Regulation (GDPR) is being reinforced with a growing body of cybersecurity legislation, including the Digital Operational Resilience Act (DORA) and an updated Network Information Security Directive (NIS2). Since the European Union first harmonized the scattering of privacy regulations from its member nations into a single GDPR regulation, over 130 countries have enacted similar privacy legislation.

With most privacy regulations, data breaches can lead to fines. For example, under GDPR, a violation can incur penalties of up to four percent of a company’s global annual turnover or €20 million, whichever is greater. In the past five years, GDPR fines alone totaled more than €4.5 Billion (USD 4.8 Billion).

Most organizations, especially those conducting business internationally or serving an international clientele (e.g., tourists), are subject to multiple regulations simultaneously.

What do all these regulations require?

There are so many regulations relating to data privacy and security that it does not make sense for us to try and even list them all in this book, much less delve into their requirements. Instead, keeping in mind that this is not a legal recommendation (please consult your own organization's legal staff before adopting anyone's advice on dealing with regulatory requirements) what I'd like to suggest is to treat your regulatory environment as requiring a framework of controls. Since most of the data privacy and security regulations are attempting to solve the same problem, there is a great deal in common between the mandates they impose – at least when it comes to the technical controls necessary to enforce compliance. Reporting and assessment requirements tend to differ greatly between different regulations, but at the end of the day MOST (but not all) regulations want you to do these things:

- Encrypt your data
- Control access to your data
- Audit access to your data

Encrypt your data

Encryption secures data by applying cryptographic algorithms that transform information into an unreadable format, accessible only to those with the correct decryption key. It is an essential technical control, required by almost all regulations. Usually, unstated is an accompanying requirement to manage the encryption keys so they cannot be accessed by unauthorized persons. Some security experts classify encryption as another form of access control, but most treat encryption as a separate discipline with its own unique benefits and challenges.

Control access to your data

Access control is the process of controlling who and under what circumstances an account can access the database or objects within the database. It is the broadest area of technical control, encompassing everything from the initial authentication process to a system (controlling who can create a database session) through determining which database objects are visible to a user to deciding which rows or columns of data within a table a user is allowed to interact with.

Audit access to your data

Auditing is the process of recording what was done in a system. Audit records normally include:

- What was done (insert, update, delete, execute, etc.)
- When it was done (date and time)
- Who did it (which database account, which operating system user initiated the session)
- How they did it (which program was used to connect to the database)
- Where the action originated from (IP address and/or the hostname where the session originated)

Auditing can be used to reconstruct what was done after the fact. Unlike access control, auditing does not prevent unauthorized actions but provides a detailed record for forensic analysis and compliance reporting. Auditing is useful in breach investigations and troubleshooting problems.

Beyond these three core requirements, some regulations may require other types of controls, but most regulations will require these three basics. A few examples of regulations requiring these and other types of control are shown in Table 2-1.

Regulation	Country	Required control	Remarks	Applies to
Payment Card Industry Data Security Standards	Global	Redaction - A specialized form of access control.	Only part of a payment account number should be visible.	Anyone accepting or processing credit cards
Digital Operational Resilience Act (DORA)	European Union	Zero Data Loss Recovery	Implicit in the goals is the ability to recover from a ransomware attack	Financial institutions in the European Union
Ministry of Corporate Affairs Companies Act Rule 11	India	Before/After value tracking - A specialized form of auditing	Track changes to entries in books of account.	Every company using software to maintain its books of account
Sarbanes Oxley Act of 2002	United States	Separation of Duties - A specialized form of access control.	Ensure financial records are not altered or viewed by unauthorized personnel.	All publicly traded companies

Table 2-1: Regulatory-influenced controls

As you can see from the table, even for regulations with requirements that, at first glance, seem unusual, the requirements still generally fall into one of the three main control types.

It's not just regulations

In addition to the stringent regulatory environment and ever-evolving data privacy concerns, threats to data have grown to top-of-mind for most organizations. Ransomware drives much of this, accounting for about 17% of cyber-attacks (a slight decrease from our last edition when ransomware was estimated to be about 20% of cyber-attacks), at an annual cost of tens of billions of dollars. Don't be fooled by that decrease, ransomware is still a major concern for almost all organizations.

Growing political instability also concerns many organizations, with nation-state operators working to steal data to gain economic advantage and corrupt or destroy data to weaken opponents.

There are no signs that these trends are slowing – the regulatory environment continues to increase control, ransomware shows no signs of going away, and political instability is spreading. Protecting data and preventing theft, destruction, corruption, or misuse is more important than ever.

It isn't all bad news

While regulatory requirements and concerns about data theft/misuse drive most database security initiatives, there is a third topic that is growing in importance. Business (or organizational) agility. All of us have customers, even those of us not involved in a commercial enterprise. And it's very likely that *your* customers are increasingly paying attention to security when they do business with you. Being able to demonstrate a robust security program, especially a successful security program, makes it more likely that you will win new business. Think of it like this – who are you more likely to want to bank with? A bank that has a public track record of being hacked every few years, with major losses of data in each hack? Or a bank that is never in the news for security issues? I know which one has MY money!

And if you are part of an organization not involved in traditional business, it's likely you still need to seek funding for your organization. That could be from the government or the private sector. Again, who are you more likely to help fund? The organization that has a good security track record? Or the one that is frequently in the news due to data breaches?

Coming up

The rest of this book takes you through the various aspects of Oracle's defense-in-depth security for databases, focusing on technology controls. I'll provide a high-level overview of how the controls work, the goals they support, and the types of protection they provide. Let's begin.

Chapter Three

Assessing Database Security

Why is assessment important?

Databases are complex, with many configuration settings that can impact security. Recent data breaches have demonstrated all too well how critical properly configured and secure systems are to protecting data. Human errors can leave your database open to everyone, and attackers will maliciously exploit configuration mistakes to steal your valuable data.

Failing to implement basic security controls risks exposing sensitive data, including names, addresses, birth dates, account information, etc. This can have a devastating impact on both your organization's reputation and bottom line. Many regulations, such as EU GDPR, PCI DSS, Sarbanes-Oxley, and various breach notification laws, promote regular security assessments for systems that contain personal data. Therefore, you should regularly scan your databases, remediating deviations following security best practices and considering recommendations from different regulations, security frameworks, Oracle best practices, and your own organization's standards.

This chapter covers how Oracle Database security solutions can help you quickly evaluate your database security posture, categorize the findings, and recommend suitable actions to keep your databases secure.

Why should you focus on assessment?

As you'll discover when reading this chapter, there are many different tools you can use to assess your database security. Some are included with your database, others are additional cost products or services. For those additional cost items, most people reading this chapter already have access to one or more of them. Knowing the capabilities of each tool, and the pros/cons of your different options can help you choose the right tool for your unique situation.

Evaluate and assess database configuration

Attackers take their time to prepare for an attack and usually spend considerable time doing reconnaissance. They use tools that automate the discovery of databases, find open ports, identify unpatched vulnerabilities, automate application and SQL injection attacks, and execute brute-force password attacks. Once they finish probing, they assess the weakest links and determine the next steps. In essence, the attackers evaluate your security posture to find a way to get to your sensitive data.

Some common questions attackers try to answer while probing your databases:

- What version of the database is running? – Identifying outdated versions with known vulnerabilities.
- Are default or weak credentials in use? – Exploiting accounts with weak or unchanged passwords.
- Which privileged users exist? – Escalating access by compromising high-privilege accounts.
- Is auditing enabled? – Determining whether actions will be logged and monitored.
- Is the data encrypted? – Assessing if they can steal raw data from storage or backups.
- Is there a copy of this database that is less likely to be audited? – Finding the least risky approach to data.

All these questions are inside the hacker's mind, and the answers help them devise a plan to break into your database and steal data. As data custodians, you need to think like a hacker to harden your database's security posture. One of the more recent data privacy laws, India's Digital Personal Data Protection Act, tries to clarify your responsibility, defining a new phrase, *data fiduciary*, to refer to those who determine the purpose and means of processing personal data. In other words, the law expects organizations to act as if they have a legal responsibility to manage systems that contain personal data in such a way as to safeguard the interests of the data subjects.

Because Oracle Database is highly customizable, assessing security requires understanding the impact of configuration choices on the overall security posture of the database. Hardening and securing a database can be a challenging task. Success requires understanding the database users and their roles, the data and its sensitivity, the security configuration parameters, the enabled features, knowledge about the database attack vectors, and the available security controls to protect the database.

Here are some key considerations for protecting your databases:

- Almost all databases hold sensitive data, but the level of importance of individual data attributes may differ. For example, a customer's date of birth may be more sensitive than their email address. It is essential to find out which databases contain what type of sensitive data so that controls can be implemented accordingly.
- Common points of attack against the database include unpatched systems, poor application design, weak user credentials, excessive privilege grants, lack of a trusted path to data, separation of duties, encryption, and inadequate auditing.
- Security configuration parameters are tightly related to how the database behaves and require understanding the parameters, what they do, the impact of changing them, and their dependencies.
- Not all database users are equal. Apart from the DBAs, several other actors/processes interact with your data through database user accounts—the application, application administrators, security administrators, and others, including service accounts, batch programs, etc. Identifying the different types of database users and the activities they need to execute on the database helps you properly manage privileges and roles and implement the principle of least privilege.
- Not all databases are created equal. Some databases may be more business-critical or contain more sensitive or highly regulated data. Your investment in security controls (which could be in tools, time, or operational resource commitment) is usually commensurate with the criticality or sensitivity of the database.

Several software tools and services can help you assess the security of your databases. We will discuss four tools:

1. Database Security Assessment Tool (DBSAT) – a command-line utility available at no cost to all Oracle Database customers with an active support agreement
2. Oracle Data Safe – an Oracle Cloud service included with all Oracle Cloud database services and available for use with on-premises Oracle Database and Oracle Database running in third-party clouds
3. Audit Vault and Database Firewall (AVDF) – a database security posture management (DSPM) product available for installation on-premises or in Oracle Cloud
4. Oracle Database Lifecycle Management (DBLM) – a management pack for Oracle Enterprise Manager

If you are an Oracle Database user, you probably have one or more of these tools available.

Assessing configuration security of the Oracle Database with DBSAT

DBSAT identifies areas where your database's configuration, operation, or implementation introduces risk. DBSAT collects and analyzes configuration data and parameters from the database. DBSAT then recommends changes and controls to mitigate those risks.

Apart from database and listener configuration, DBSAT collects information on user accounts, privileges and roles, authorization control, separation of duties, fine-grained access control, data encryption and key management, auditing policies, and operating system file permissions. DBSAT applies rules that assess a database's current security status quickly and recommends changes based on best practices. Updated best-practice evaluation rules are delivered periodically with new versions of the tool.

DBSAT scans the database for weaknesses and vulnerabilities and categorizes findings by risk level to help you prioritize work on the most critical weaknesses. DBSAT also provides high-level summaries and specific recommendations for each issue, making it simpler and quicker for you to act.

DBSAT is a free command-line tool available to all Oracle customers with a current support agreement. Oracle consultants, partners, and auditors worldwide use DBSAT to perform database security assessments.

DBSAT security assessment reports

DBSAT has three components: *collector*, *reporter*, and *discoverer*. The *collector* and *reporter* generate database security risk assessments, and the *discoverer* scans the database, producing a report on the different types of sensitive data found within the database.

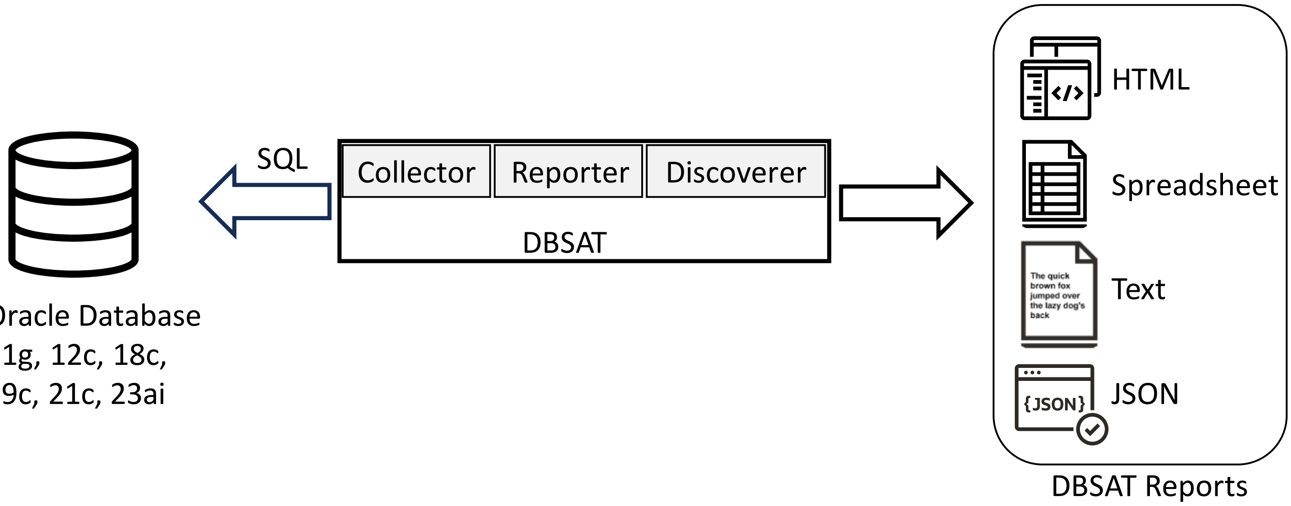
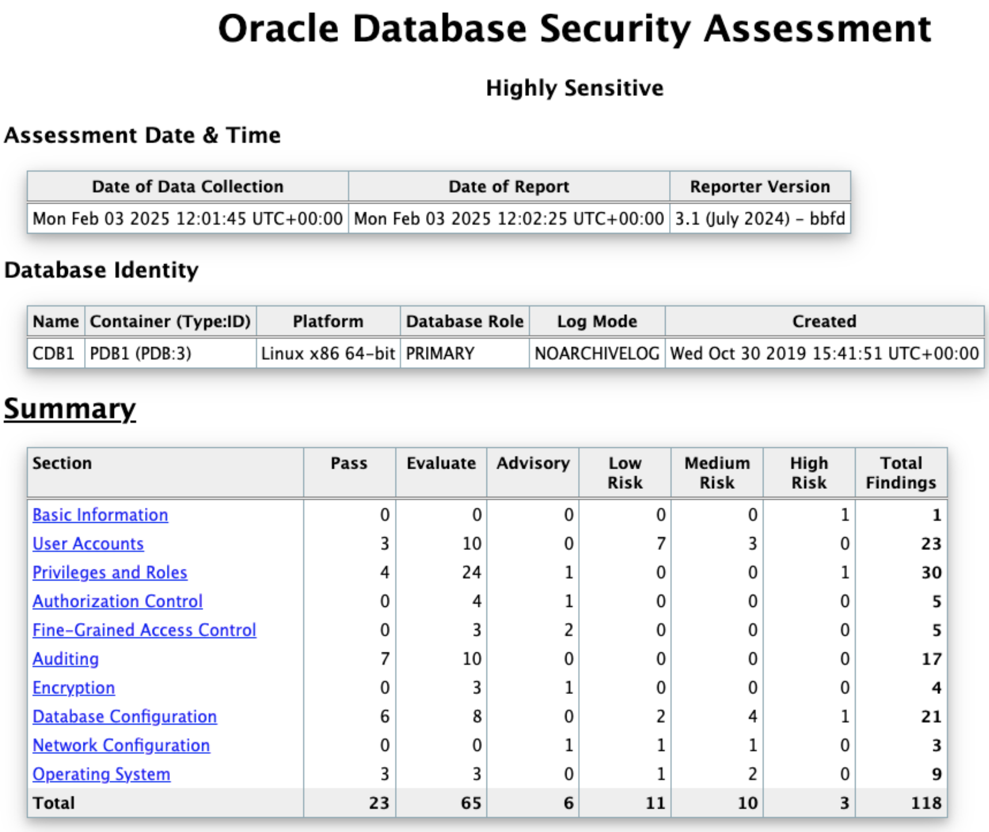


Figure 3-1: Database Security Assessment Tool (DBSAT)

The *collector* gathers security configuration information from the database and underlying operating system. DBSAT's *reporter* then analyzes the collected data and generates a report containing detailed findings and recommendations. The reports are delivered in HTML, spreadsheet, text, and JSON formats. DBSAT's *discoverer* (described later in Chapter Four) helps identify sensitive data and classifies and summarizes its findings in HTML and spreadsheet reports.

The HTML reports provide detailed assessment results in a format that is easy to navigate with summarized data as you can see in Figure 3-2. The spreadsheet format provides a high-level summary of each finding so that you can add columns for your tracking and prioritization purposes. A report in text format makes it convenient to copy portions of the output for other usages. The JSON output is suitable for data aggregation and integration purposes.



Oracle Database Security Assessment

Highly Sensitive

Assessment Date & Time

Date of Data Collection	Date of Report	Reporter Version
Mon Feb 03 2025 12:01:45 UTC+00:00	Mon Feb 03 2025 12:02:25 UTC+00:00	3.1 (July 2024) - bbfd

Database Identity

Name	Container (Type:ID)	Platform	Database Role	Log Mode	Created
CDB1	PDB1 (PDB:3)	Linux x86 64-bit	PRIMARY	NOARCHIVELOG	Wed Oct 30 2019 15:41:51 UTC+00:00

Summary

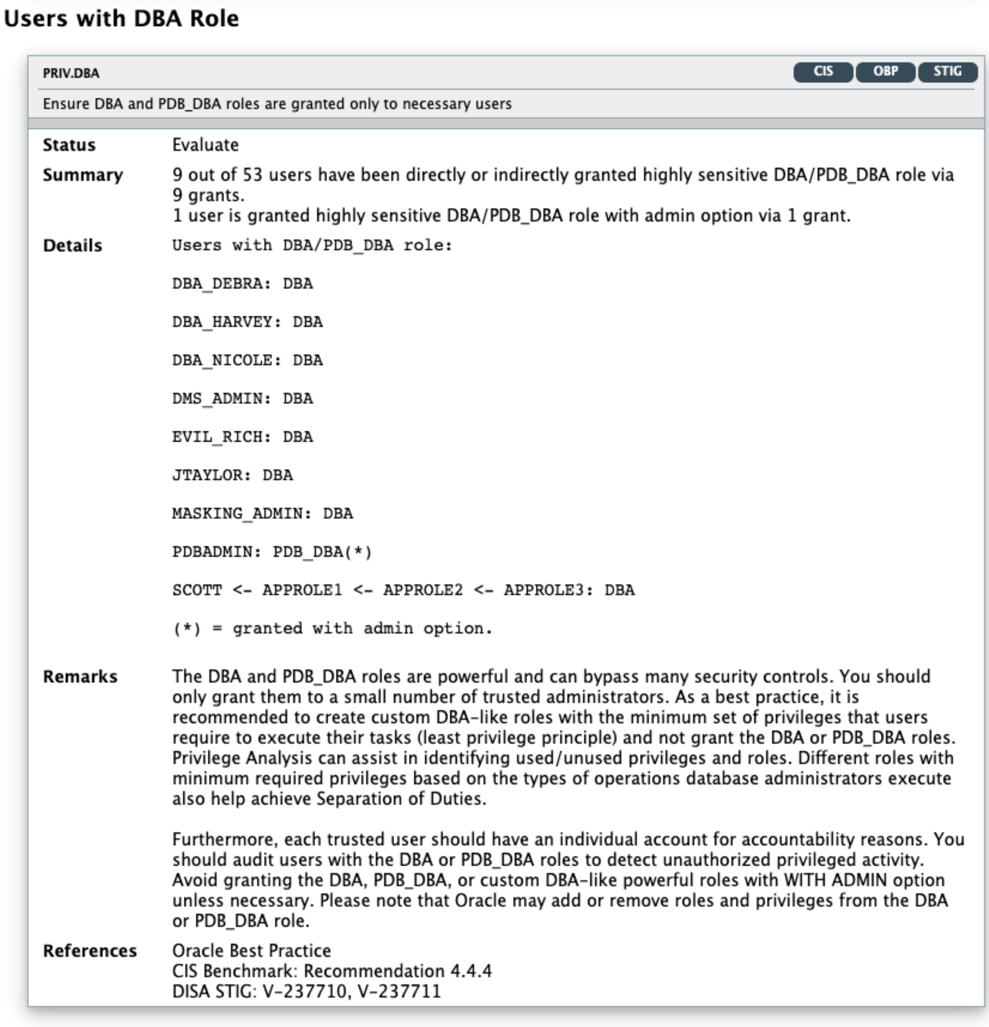
Section	Pass	Evaluate	Advisory	Low Risk	Medium Risk	High Risk	Total Findings
Basic Information	0	0	0	0	0	1	**1**
User Accounts	3	10	0	7	3	0	**23**
Privileges and Roles	4	24	1	0	0	1	**30**
Authorization Control	0	4	1	0	0	0	**5**
Fine-Grained Access Control	0	3	2	0	0	0	**5**
Auditing	7	10	0	0	0	0	**17**
Encryption	0	3	1	0	0	0	**4**
Database Configuration	6	8	0	2	4	1	**21**
Network Configuration	0	0	1	1	1	0	**3**
Operating System	3	3	0	1	2	0	**9**
Total	**23**	**65**	**6**	**11**	**10**	**3**	**118**

Figure 3-2: DBSAT report summary

The most useful report is usually the HTML format. In addition to summary information, the report contains detailed information for each finding. A sample finding is shown below in Figure 3-3. For each finding, you'll find:

- Rule ID: contains a prefix that identifies the report section for the finding, followed by a period and a name to uniquely identify the finding.
- One-liner: A single sentence that describes the purpose of the finding.
- Status: This indicates the level of risk associated with the finding (Pass, Low Risk, Medium Risk, High Risk) or indicates that the finding is an Advisory for improvement, such as information about an optional security feature currently not in use. In cases where further analysis is needed, the status is shown as "Evaluate."
- Summary: Presents an overview of the finding.
- Details: Presents the details of the results from DBSAT's investigation, followed by recommendations.
- Remarks: Explain the reason for the rule and recommended actions for remediation.
- References: When applicable, lists the corresponding CIS Oracle Database benchmark recommendation, Oracle Database STIG Rule, and the EU GDPR article/recital.

The finding shown in Figure 3-3, DBSAT identifies eight users with the DBA role and one with the PDB_DBA role. It lets you know that one of those users, SCOTT, has the DBA role through a series of nested grants of other roles (this is a common tactic when trying to hide grants of the DBA role from view). The finding informs the reader that further analysis is needed (Status = Evaluate) in order to validate that grants of the role are to appropriate users. The remarks section of the finding provides more information on why it is important to limit the usage of this role to a small number of trusted administrators. References flag this finding as originating with CIS Oracle Database Benchmark recommendation 4.4.4, DISA STIG rules V-237710 and V-237711, and an Oracle-recommended best practice.



Users with DBA Role

Ensure DBA and PDB_DBA roles are granted only to necessary users

Status Evaluate

Summary 9 out of 53 users have been directly or indirectly granted highly sensitive DBA/PDB_DBA role via 9 grants.
1 user is granted highly sensitive DBA/PDB_DBA role with admin option via 1 grant.

Details

```
Users with DBA/PDB_DBA role:

DBA_DEBRA: DBA

DBA_HARVEY: DBA

DBA_NICOLE: DBA

DMS_ADMIN: DBA

EVIL_RICH: DBA

JTAYLOR: DBA

MASKING_ADMIN: DBA

PDBADMIN: PDB_DBA(*)

SCOTT <- APPROLE1 <- APPROLE2 <- APPROLE3: DBA

(*) = granted with admin option.
```

Remarks The DBA and PDB_DBA roles are powerful and can bypass many security controls. You should only grant them to a small number of trusted administrators. As a best practice, it is recommended to create custom DBA-like roles with the minimum set of privileges that users require to execute their tasks (least privilege principle) and not grant the DBA or PDB_DBA roles. Privilege Analysis can assist in identifying used/unused privileges and roles. Different roles with minimum required privileges based on the types of operations database administrators execute also help achieve Separation of Duties.

Furthermore, each trusted user should have an individual account for accountability reasons. You should audit users with the DBA or PDB_DBA roles to detect unauthorized privileged activity. Avoid granting the DBA, PDB_DBA, or custom DBA-like powerful roles with WITH ADMIN option unless necessary. Please note that Oracle may add or remove roles and privileges from the DBA or PDB_DBA role.

References Oracle Best Practice
CIS Benchmark: Recommendation 4.4.4
DISA STIG: V-237710, V-237711

Figure 3-3: DBSAT Sample finding - Users with the DBA role

Assessing your database fleet using Oracle Data Safe

DBSAT is a great tool for quickly assessing a few databases, but what if you have dozens, hundreds, or thousands of databases? Running DBSAT manually and analyzing each report individually is time-intensive, and manual analysis of the reports might miss important issues. Enter Oracle Data Safe, an Oracle Cloud Infrastructure (OCI) service focused on Oracle Database security.

Data Safe shines a light on your database fleet's security posture, helping you uncover areas of risk and prioritize mitigation efforts. Data Safe works with Oracle Databases anywhere, whether running on-premises, in the Oracle Cloud, or 3rd party clouds. Data Safe provides a comprehensive suite of security capabilities such as security and user assessment, activity auditing, SQL firewall management, data discovery, and data masking. Tightly integrated assessment capabilities provide the ability to simultaneously run assessments on multiple databases, schedule assessments, establish a security baseline, and get a comparison report highlighting the drift between that baseline and the current database security assessment.

Data Safe is easy to use, even for those with no specialized security expertise. Data Safe saves time with an intuitive interface that minimizes error and shortens learning curves. It mitigates security risks by making various aspects of configuration, data, and user security risks immediately visible.

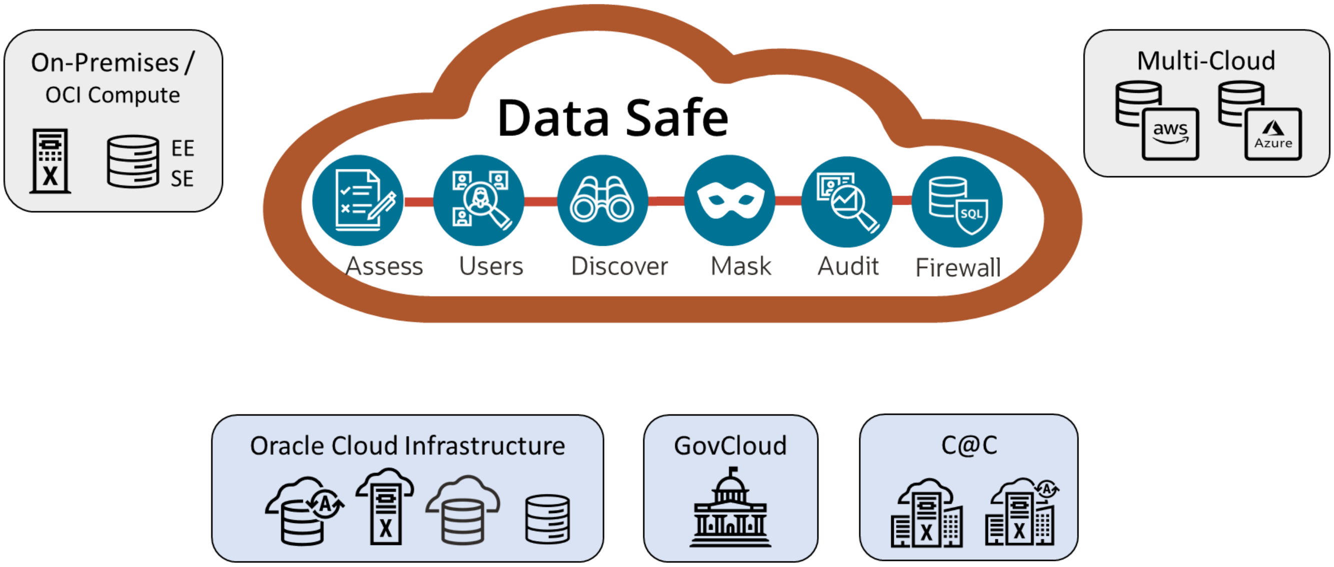
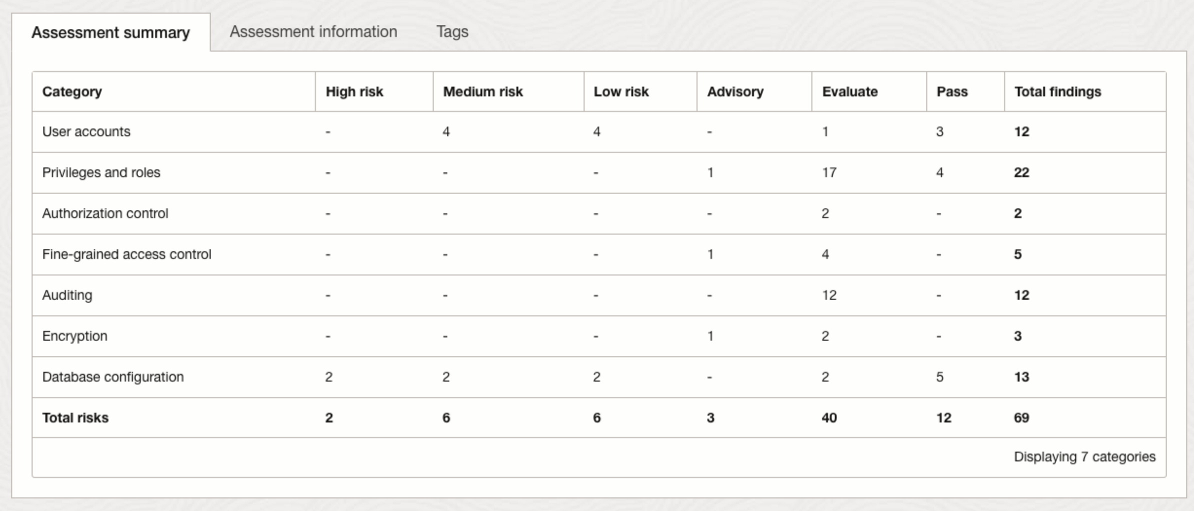


Figure 3-4: Oracle Data Safe provides essential security for Oracle Databases, both in the cloud and on-premises

Using Data Safe to assess database security

Data Safe's security assessment performs a comprehensive check of your database configuration and identifies problems that could introduce a vulnerability. It examines user accounts, privilege and role grants, authorization controls, fine-grained controls, auditing, encryption, and configuration parameters. Data Safe identifies gaps in your database's current state compared to organizational best practices. It delivers actionable reports with prioritized recommendations and mappings to common compliance frameworks like EU GDPR, DISA STIG, and CIS benchmarks. Like DBSAT, Data Safe's security and user assessments start with a summary of the issues, as seen in Figure 3-5.

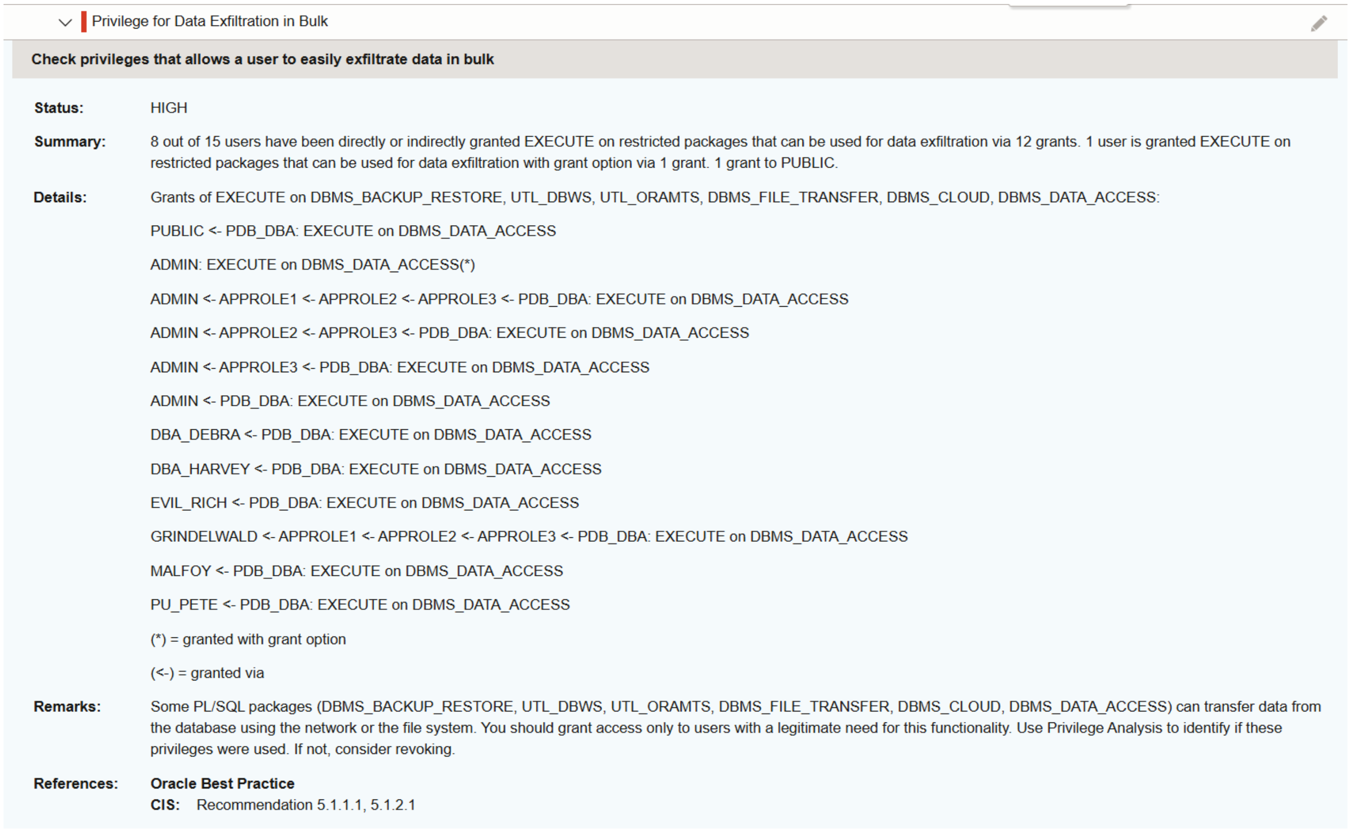


Category	High risk	Medium risk	Low risk	Advisory	Evaluate	Pass	Total findings
User accounts	-	4	4	-	1	3	**12**
Privileges and roles	-	-	-	1	17	4	**22**
Authorization control	-	-	-	-	2	-	**2**
Fine-grained access control	-	-	-	1	4	-	**5**
Auditing	-	-	-	-	12	-	**12**
Encryption	-	-	-	1	2	-	**3**
Database configuration	2	2	2	-	2	5	**13**
Total risks	**2**	**6**	**6**	**3**	**40**	**12**	**69**

Displaying 7 categories

Figure 3-5: Data Safe assessment summary

Figure 3-6 shows security assessment details, including a high-risk finding. The findings include whatever the issue is, its severity, and recommendations on how to remediate it. Data Safe leverages DBSAT and adds enterprise-grade features such as periodic assessment, fleet view (aggregations), filtering, baselining, drift reports, history of assessments, events, and notifications.



Privilege for Data Exfiltration in Bulk

Check privileges that allows a user to easily exfiltrate data in bulk

Status: HIGH

Summary: 8 out of 15 users have been directly or indirectly granted EXECUTE on restricted packages that can be used for data exfiltration via 12 grants. 1 user is granted EXECUTE on restricted packages that can be used for data exfiltration with grant option via 1 grant. 1 grant to PUBLIC.

Details: Grants of EXECUTE on DBMS_BACKUP_RESTORE, UTL_DBWS, UTL_ORAMTS, DBMS_FILE_TRANSFER, DBMS_CLOUD, DBMS_DATA_ACCESS:

PUBLIC <- PDB_DBA: EXECUTE on DBMS_DATA_ACCESS

ADMIN: EXECUTE on DBMS_DATA_ACCESS(*)

ADMIN <- APPROLE1 <- APPROLE2 <- APPROLE3 <- PDB_DBA: EXECUTE on DBMS_DATA_ACCESS

ADMIN <- APPROLE2 <- APPROLE3 <- PDB_DBA: EXECUTE on DBMS_DATA_ACCESS

ADMIN <- APPROLE3 <- PDB_DBA: EXECUTE on DBMS_DATA_ACCESS

ADMIN <- PDB_DBA: EXECUTE on DBMS_DATA_ACCESS

DBA_DEBRA <- PDB_DBA: EXECUTE on DBMS_DATA_ACCESS

DBA_HARVEY <- PDB_DBA: EXECUTE on DBMS_DATA_ACCESS

EVIL_RICH <- PDB_DBA: EXECUTE on DBMS_DATA_ACCESS

GRINDELWALD <- APPROLE1 <- APPROLE2 <- APPROLE3 <- PDB_DBA: EXECUTE on DBMS_DATA_ACCESS

MALFOY <- PDB_DBA: EXECUTE on DBMS_DATA_ACCESS

PU_PETE <- PDB_DBA: EXECUTE on DBMS_DATA_ACCESS

(*) = granted with grant option

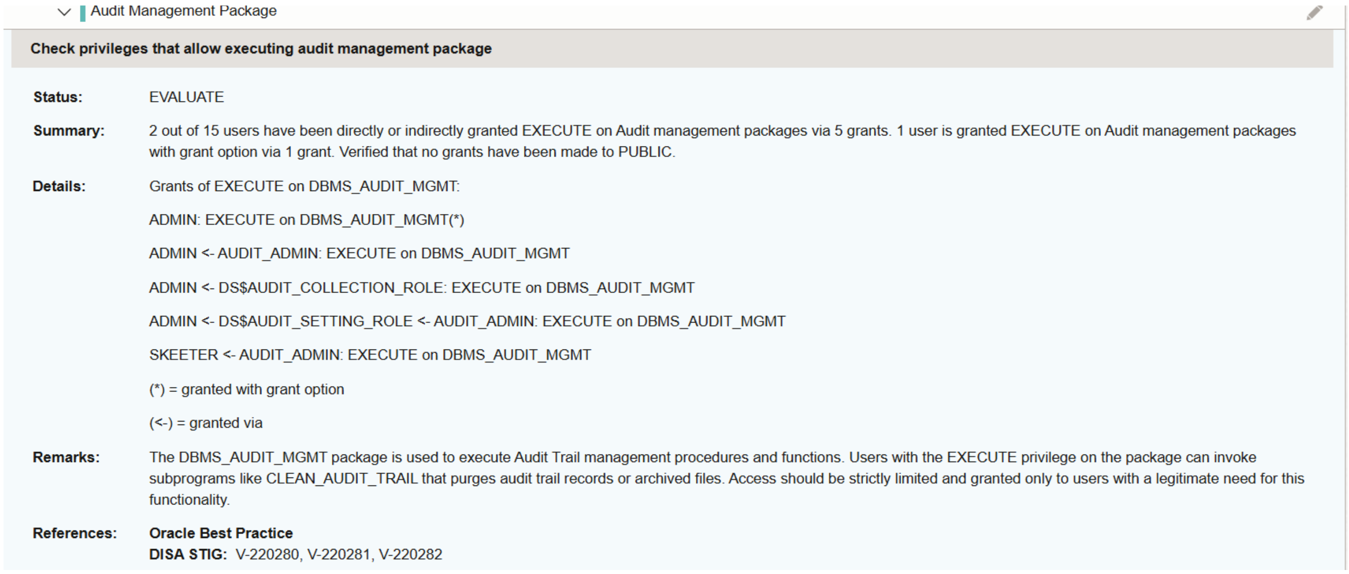
(<-) = granted via

Remarks: Some PL/SQL packages (DBMS_BACKUP_RESTORE, UTL_DBWS, UTL_ORAMTS, DBMS_FILE_TRANSFER, DBMS_CLOUD, DBMS_DATA_ACCESS) can transfer data from the database using the network or the file system. You should grant access only to users with a legitimate need for this functionality. Use Privilege Analysis to identify if these privileges were used. If not, consider revoking.

References: **Oracle Best Practice**
CIS: Recommendation 5.1.1.1, 5.1.2.1

Figure 3-6: Data Safe security assessment findings

Data Safe also allows you to configure acceptable risks. For example, if the assessment lists a finding that needs to be evaluated (Status EVALUATE) as shown in Figure 3-7, then the Data Safe user needs to review the finding to see if it's a problem. In this example, Data Safe can tell that two users (ADMIN and SKEETER) have the privilege to manage audit settings, but it's left to the reviewer to decide if SKEETER really should have the privilege.



Audit Management Package

Check privileges that allow executing audit management package

Status: EVALUATE

Summary: 2 out of 15 users have been directly or indirectly granted EXECUTE on Audit management packages via 5 grants. 1 user is granted EXECUTE on Audit management packages with grant option via 1 grant. Verified that no grants have been made to PUBLIC.

Details: Grants of EXECUTE on DBMS_AUDIT_MGMT:

ADMIN: EXECUTE on DBMS_AUDIT_MGMT(*)

ADMIN <- AUDIT_ADMIN: EXECUTE on DBMS_AUDIT_MGMT

ADMIN <- DS$AUDIT_COLLECTION_ROLE: EXECUTE on DBMS_AUDIT_MGMT

ADMIN <- DS$AUDIT_SETTING_ROLE <- AUDIT_ADMIN: EXECUTE on DBMS_AUDIT_MGMT

SKEETER <- AUDIT_ADMIN: EXECUTE on DBMS_AUDIT_MGMT

(*) = granted with grant option

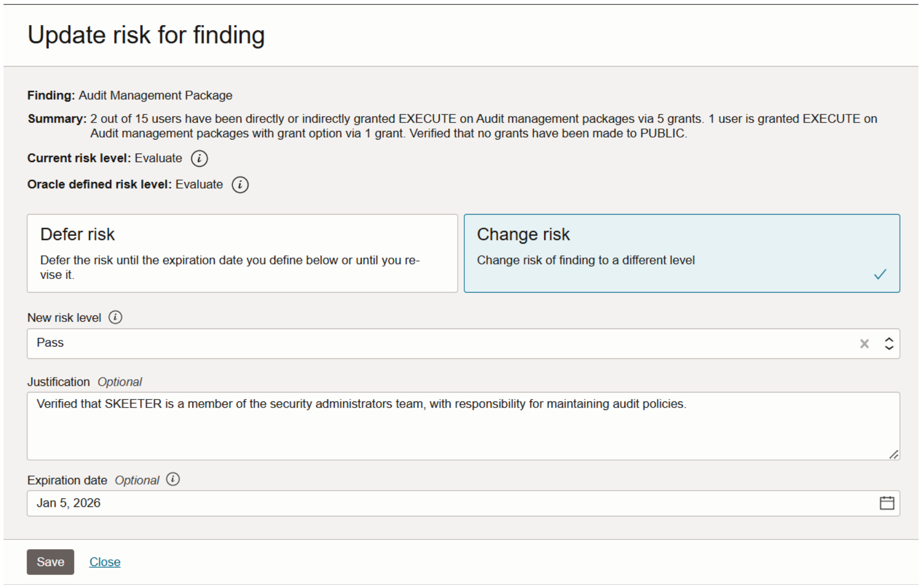
(<-) = granted via

Remarks: The DBMS_AUDIT_MGMT package is used to execute Audit Trail management procedures and functions. Users with the EXECUTE privilege on the package can invoke subprograms like CLEAN_AUDIT_TRAIL that purges audit trail records or archived files. Access should be strictly limited and granted only to users with a legitimate need for this functionality.

References: **Oracle Best Practice**
DISA STIG: V-220280, V-220281, V-220282

Figure 3-7: This finding needs to be evaluated to assess its risk

If a review determines that this finding presents no risk (SKEETER really *should* have the privilege) you can mark the finding as “Pass.” You can also lower a finding’s risk if you have other compensating controls that reduce the risk. You can even choose to accept the risk if you do not plan to address it. In this case, we have validated that SKEETER should have the privilege and configured Data Safe to remind us to review the finding again at a later time.



Update risk for finding

Finding: Audit Management Package

Summary: 2 out of 15 users have been directly or indirectly granted EXECUTE on Audit management packages via 5 grants. 1 user is granted EXECUTE on Audit management packages with grant option via 1 grant. Verified that no grants have been made to PUBLIC.

Current risk level: Evaluate

Oracle defined risk level: Evaluate

○ Defer risk
Defer the risk until the expiration date you define below or until you revise it.

● Change risk
Change risk of finding to a different level ✓

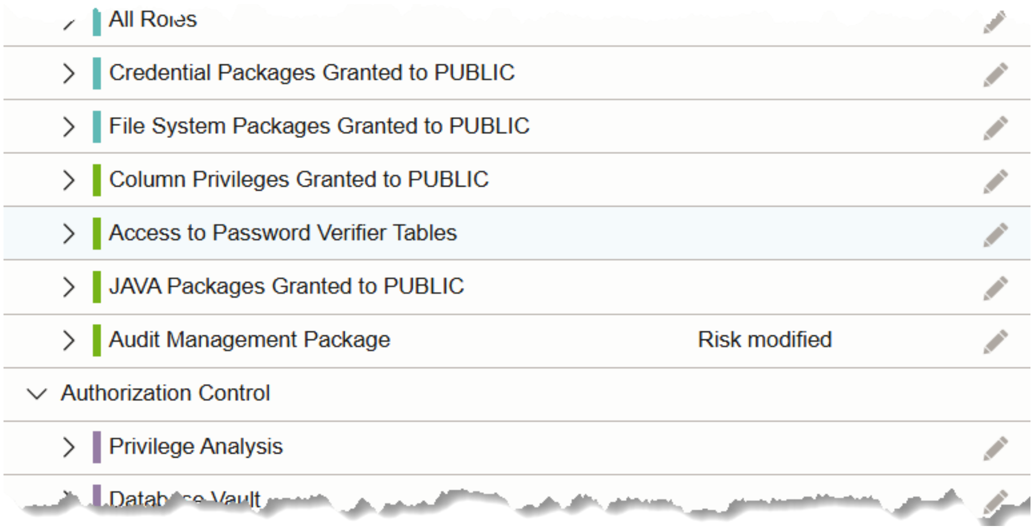
New risk level: Pass

Justification *Optional*: Verified that SKEETER is a member of the security administrators team, with responsibility for maintaining audit policies.

Expiration date *Optional*: Jan 5, 2026

Figure 3-8: Data Safe: Configuring acceptable risks

You can easily see which risks have been updated from their default values as shown in Figure 3-9.



All Roles

Credential Packages Granted to PUBLIC

File System Packages Granted to PUBLIC

Column Privileges Granted to PUBLIC

Access to Password Verifier Tables

JAVA Packages Granted to PUBLIC

Audit Management Package — Risk modified

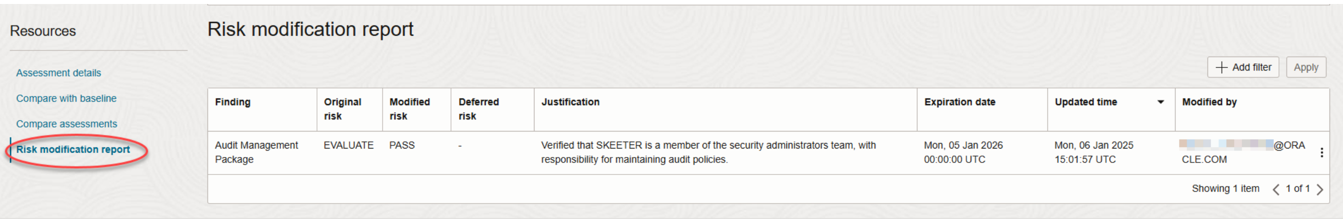
Authorization Control

Privilege Analysis

Database Vault

Figure 3-9: Finding modified risks is easy

And security assessment's *risk modification report* lets you quickly drill into details on the modification.



Risk modification report

Finding	Original risk	Modified risk	Deferred risk	Justification	Expiration date	Updated time	Modified by
Audit Management Package	EVALUATE	PASS	-	Verified that SKEETER is a member of the security administrators team, with responsibility for maintaining audit policies.	Mon, 05 Jan 2026 00:00:00 UTC	Mon, 06 Jan 2025 15:01:57 UTC	[redacted]@ORACLE.COM

Showing 1 item 1 of 1

Figure 3-10: Security assessment risk modification report

Understanding user risk with Data Safe

Stealing privileged user credentials is the most common method attackers use to access sensitive data. Data Safe's *user assessment* module helps you make your applications more secure by providing the unique ability to assess and visualize user risk.

Note: For the purposes of this discussion, the terms user and account are used interchangeably. A database user might actually be an application service account or batch process service account, not a human actor.

User assessment evaluates database accounts, looking at static and dynamic profile characteristics to identify the highest-risk users. User risk is presented alongside other account details, allowing you to quickly determine which users may be over-privileged or require compensating controls such as auditing. It helps you understand, for example, how many users have not logged in the last three (3) months or longer or have not changed their passwords recently. Suppose there is some suspicion about a user's activity. In that case, Data Safe helps you understand all details about the user, including when they were created, their roles and privileges, and even takes you to a report of related audit records showing the user's activity.

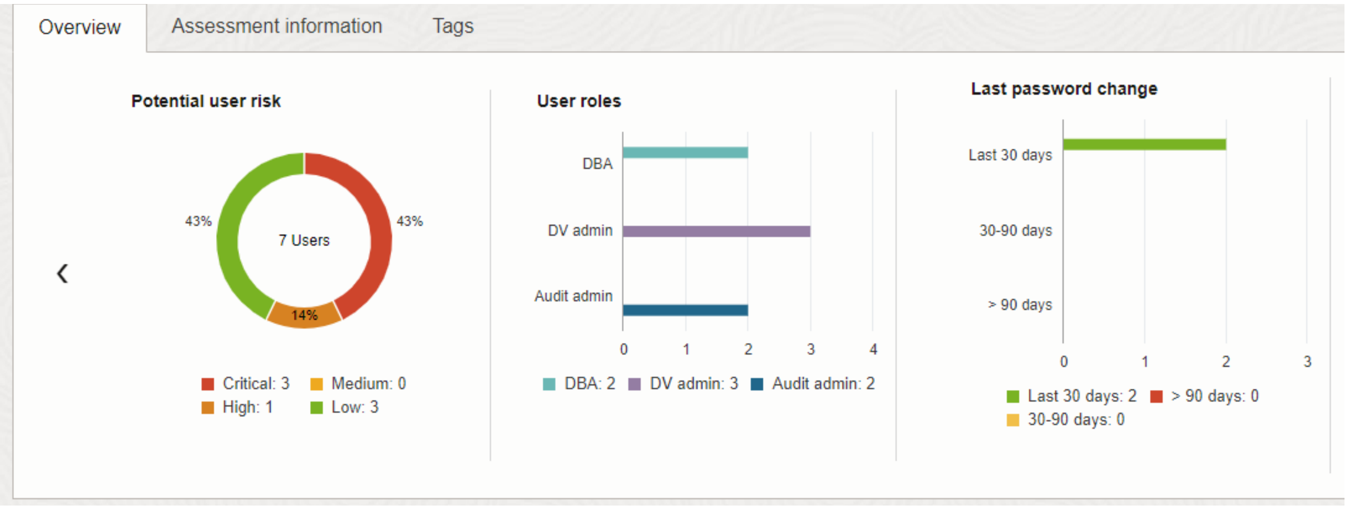
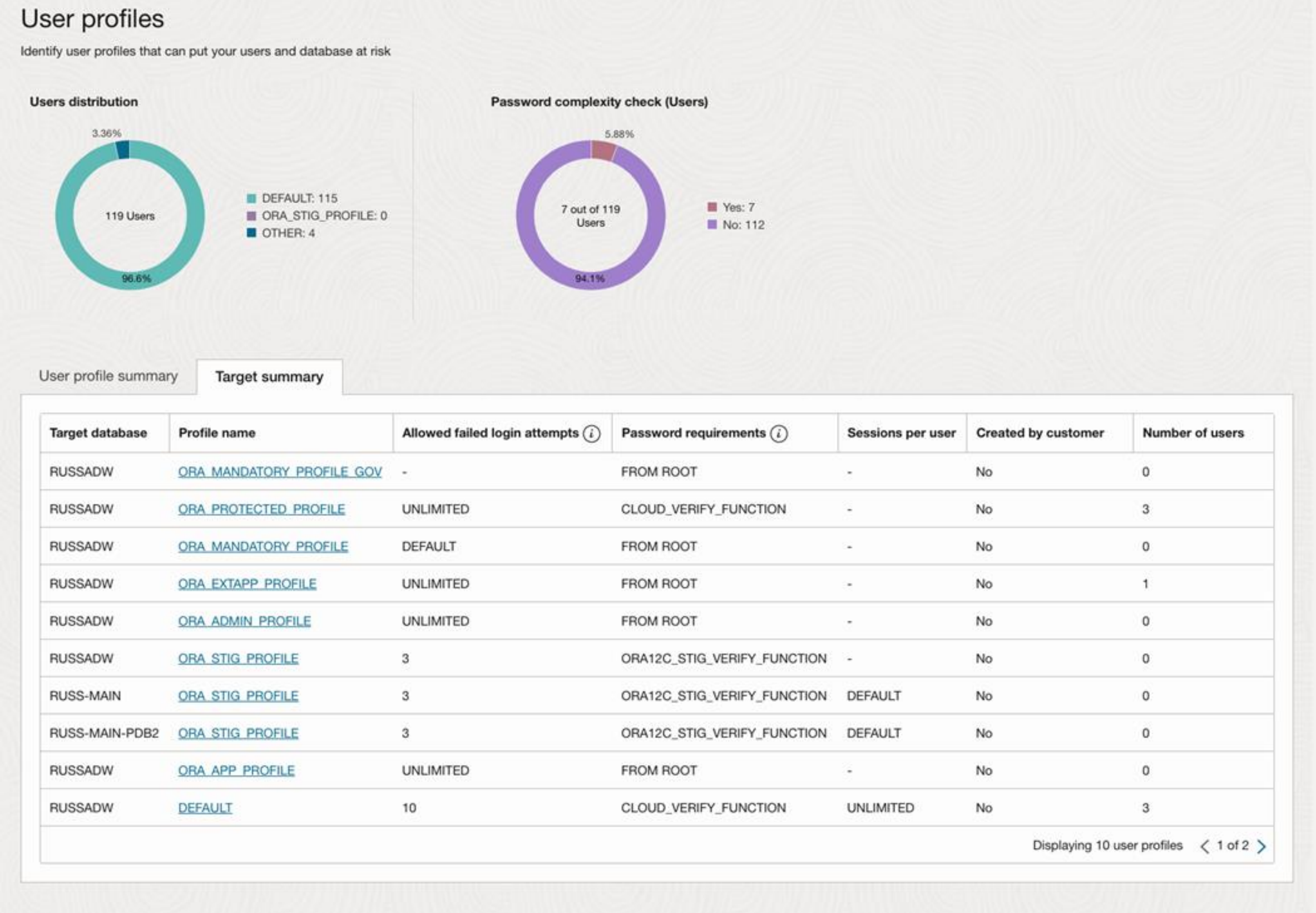


Figure 3-11: Data Safe: User risk assessment

Data Safe's user assessment also provides you with visibility into database user profiles. User profiles include password-related settings that are essential for strengthening passwords. User profiles allow you to limit some actions that users can perform on the database. For example, you can use user profiles to limit the number of failed login attempts before a user is locked out for a configurable time period, set password governance settings, including complexity requirements, and define password expiration policies. We will talk more about user profiles in Chapter Five.

Data Safe's User Profile Insight helps unlock the power of profiles across your fleet of databases, giving you the capability to:

- Review existing user profiles and their parameters, including the password verification function being used to enforce password complexity
- Contrast user profiles with similar names across multiple databases to spot differences or gaps
- Identify which profiles are assigned to which users
- Quickly identify users and profiles with inadequate password governance policies



User profiles

Identify user profiles that can put your users and database at risk

Target database	Profile name	Allowed failed login attempts ⓘ	Password requirements ⓘ	Sessions per user	Created by customer	Number of users
RUSSADW	ORA_MANDATORY_PROFILE_GOV	-	FROM ROOT	-	No	0
RUSSADW	ORA_PROTECTED_PROFILE	UNLIMITED	CLOUD_VERIFY_FUNCTION	-	No	3
RUSSADW	ORA_MANDATORY_PROFILE	DEFAULT	FROM ROOT	-	No	0
RUSSADW	ORA_EXTAPP_PROFILE	UNLIMITED	FROM ROOT	-	No	1
RUSSADW	ORA_ADMIN_PROFILE	UNLIMITED	FROM ROOT	-	No	0
RUSSADW	ORA_STIG_PROFILE	3	ORA12C_STIG_VERIFY_FUNCTION	-	No	0
RUSS-MAIN	ORA_STIG_PROFILE	3	ORA12C_STIG_VERIFY_FUNCTION	DEFAULT	No	0
RUSS-MAIN-PDB2	ORA_STIG_PROFILE	3	ORA12C_STIG_VERIFY_FUNCTION	DEFAULT	No	0
RUSSADW	ORA_APP_PROFILE	UNLIMITED	FROM ROOT	-	No	0
RUSSADW	DEFAULT	10	CLOUD_VERIFY_FUNCTION	UNLIMITED	No	3

Displaying 10 user profiles < 1 of 2 >

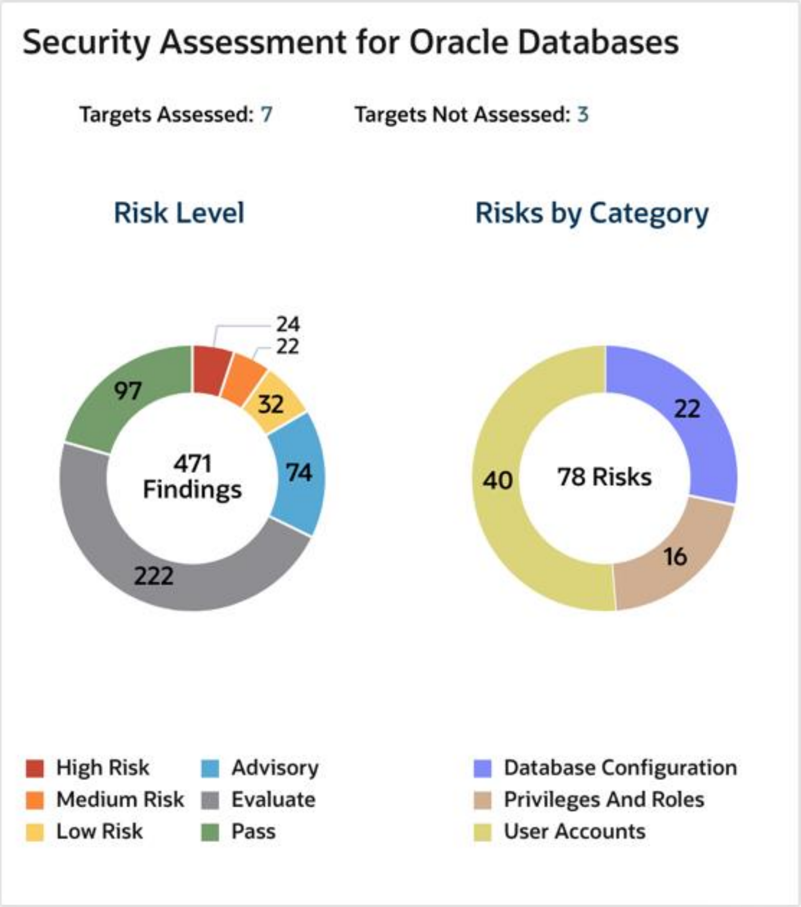
Figure 3-12: Data Safe: User profile insight

Data Safe includes a broad range of capabilities beyond assessment. See how else Data Safe can help:

- To discover sensitive data, please see Chapter Four.
- To detect or block SQL injection attacks, please see Chapter Eight.
- To anonymize sensitive data in non-production environments, please see Chapter Nine.
- To collect audit records and centralize reporting and alerting to meet regulatory requirements, please see Chapter Thirteen.

Assessment with Oracle Audit Vault and Database Firewall

Audit Vault and Database Firewall (AVDF) gives you a fleet-wide simplified and centralized view of security configuration assessments for all your Oracle Databases that is similar in functionality to Data Safe's security assessment module. The full-featured assessment with compliance mappings and recommendations helps you understand the security posture for all your Oracle Databases in one central place. You can also define an assessment baseline and determine deviation from that baseline by viewing security assessment drift reports. Insights from the drift reports help you focus only on the changes since the last assessment.



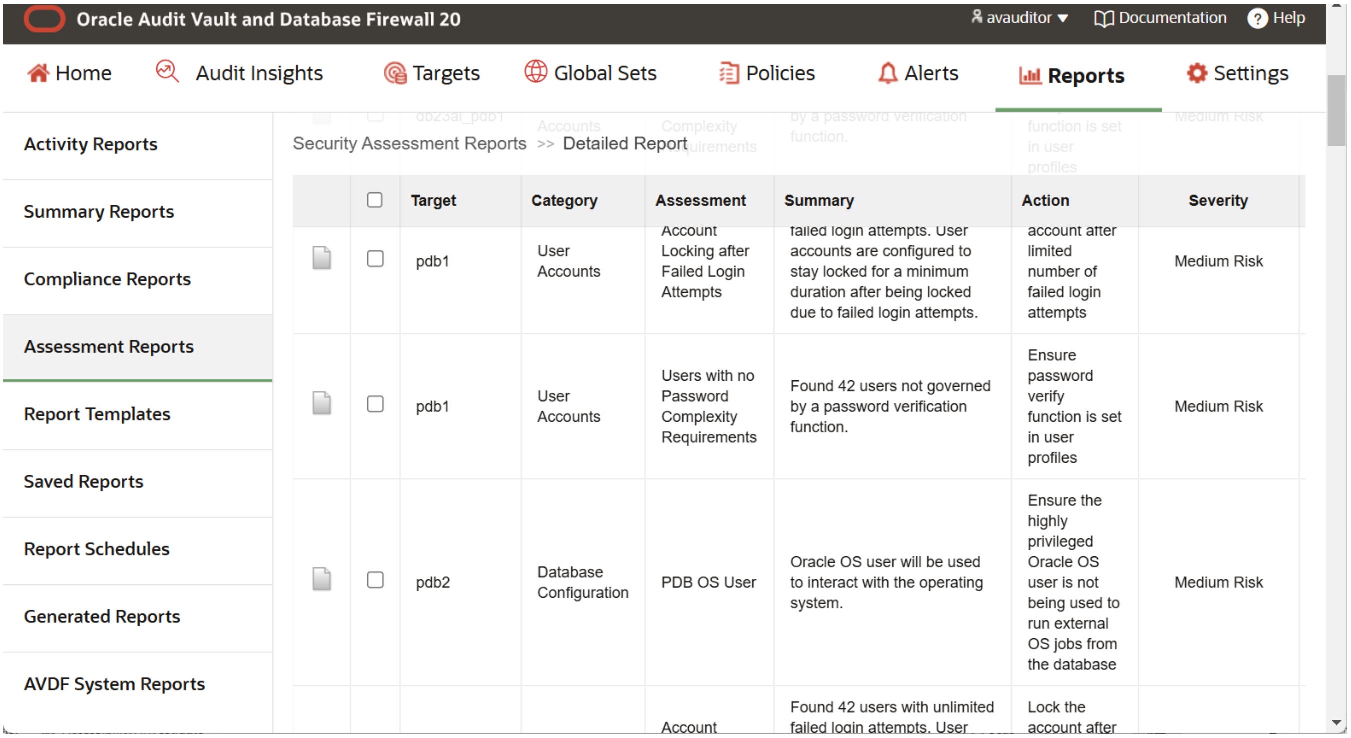
Security Assessment for Oracle Databases

Targets Assessed: 7

Targets Not Assessed: 3

Figure 3-13: Audit Vault and Database Firewall security posture management

AVDF’s reporting engine is based on Oracle APEX, and it’s easy to drill down to exactly the information you want to see. In Figure 3-14 you see the detailed assessment report, giving you an overview of all findings.

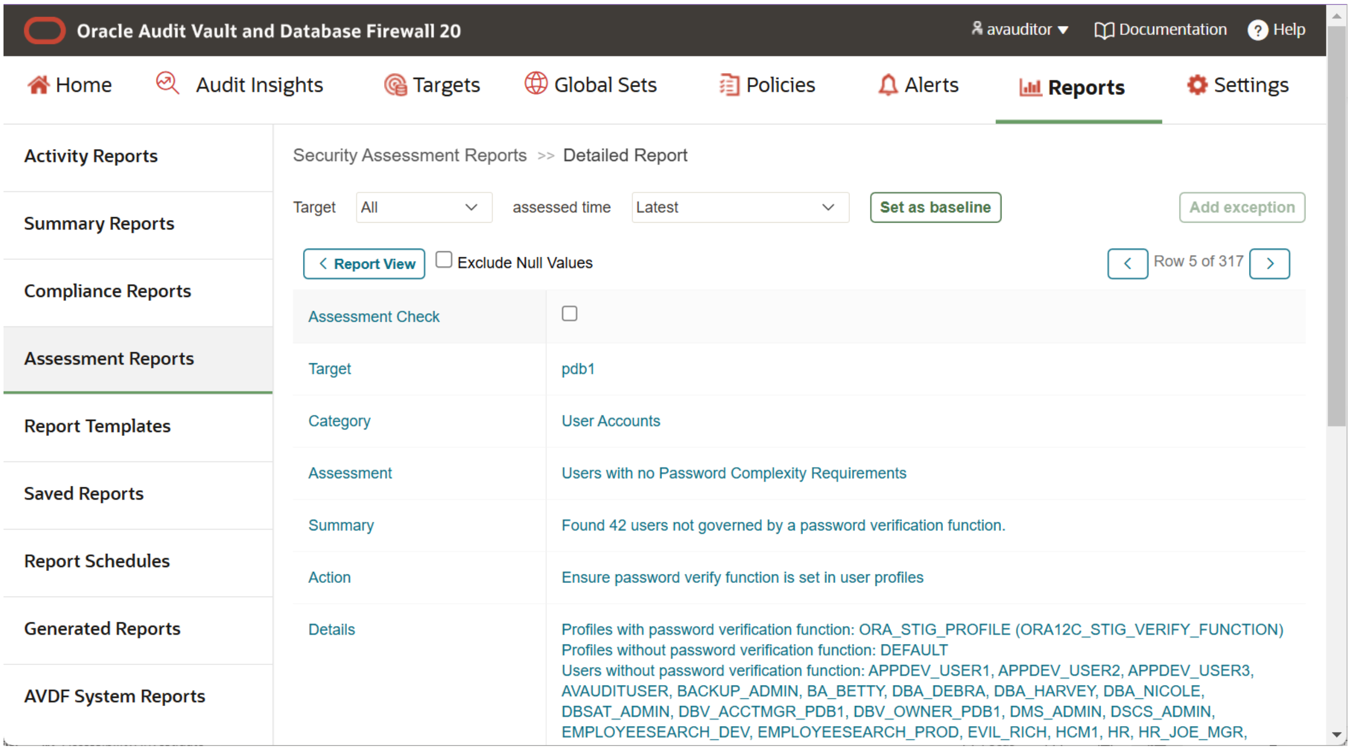


Security Assessment Reports >> Detailed Report

	☐	Target	Category	Assessment	Summary	Action	Severity
	☐	pdb1	User Accounts	Account Locking after Failed Login Attempts	failed login attempts. User accounts are configured to stay locked for a minimum duration after being locked due to failed login attempts.	account after limited number of failed login attempts	Medium Risk
	☐	pdb1	User Accounts	Users with no Password Complexity Requirements	Found 42 users not governed by a password verification function.	Ensure password verify function is set in user profiles	Medium Risk
	☐	pdb2	Database Configuration	PDB OS User	Oracle OS user will be used to interact with the operating system.	Ensure the highly privileged Oracle OS user is not being used to run external OS jobs from the database	Medium Risk
				Account	Found 42 users with unlimited failed login attempts. User	Lock the account after	

Figure 3-14: AVDF: detailed assessment report

You can drill-down even deeper to get detailed information about a finding, with all of the information you'd see in the Database Security Assessment Tool's HTML report at your fingertips. Figure 3-15 shows details of a finding on users with no password complexity requirements set.



Security Assessment Reports >> Detailed Report

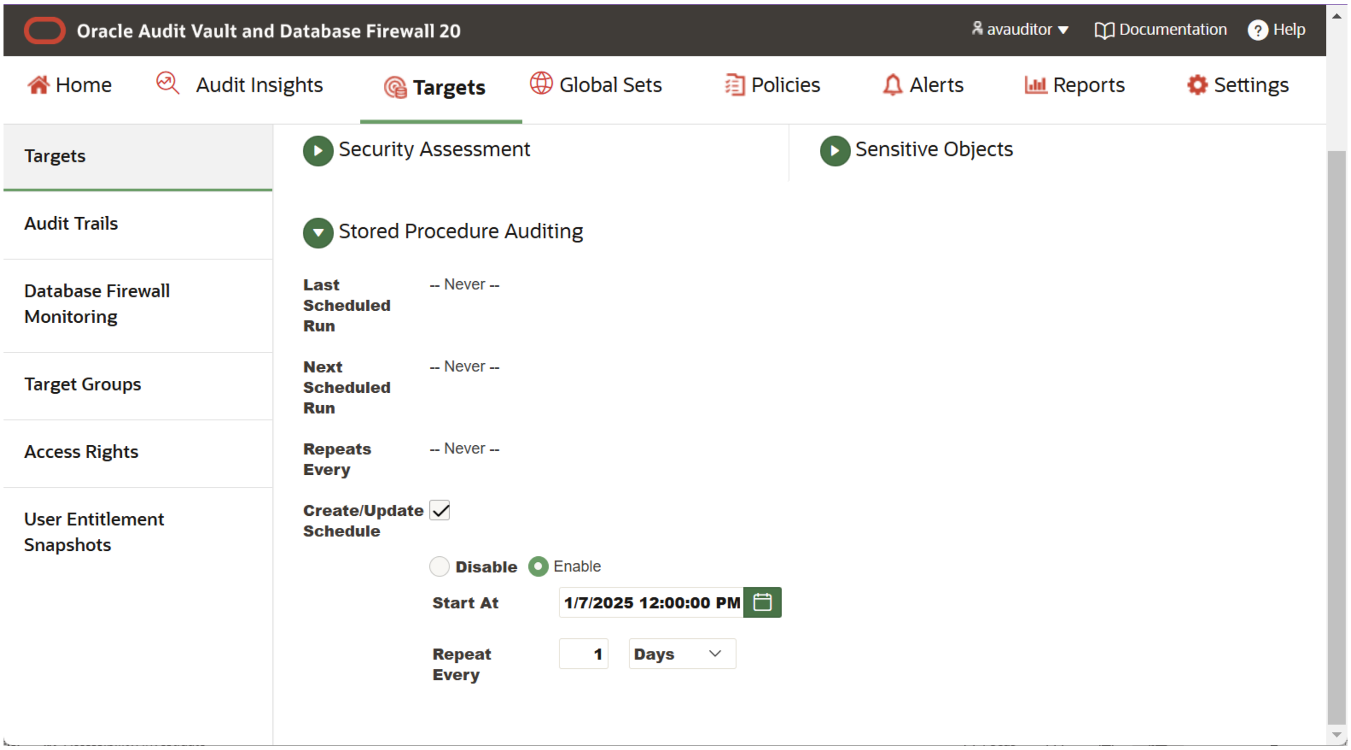
Target: All   assessed time: Latest

☐ Exclude Null Values

Assessment Check	☐
Target	pdb1
Category	User Accounts
Assessment	Users with no Password Complexity Requirements
Summary	Found 42 users not governed by a password verification function.
Action	Ensure password verify function is set in user profiles
Details	Profiles with password verification function: ORA_STIG_PROFILE (ORA12C_STIG_VERIFY_FUNCTION) Profiles without password verification function: DEFAULT Users without password verification function: APPDEV_USER1, APPDEV_USER2, APPDEV_USER3, AVAUDITUSER, BACKUP_ADMIN, BA_BETTY, DBA_DEBRA, DBA_HARVEY, DBA_NICOLE, DBSAT_ADMIN, DBV_ACCTMGR_PDB1, DBV_OWNER_PDB1, DMS_ADMIN, DSCS_ADMIN, EMPLOYEESEARCH_DEV, EMPLOYEESEARCH_PROD, EVIL_RICH, HCM1, HR, HR_JOE_MGR,

Figure 3-15: AVDF: Detailed information about an individual finding

AVDF adds a few capabilities that Data Safe has not yet incorporated as of the time I'm writing this. One of those is the extremely important ability to track stored procedures and detect changes to them. Assessing changes and detecting drift in stored procedures can be an important factor in identifying malicious activity. Once an attacker gains access to the database, they are likely to update a seemingly benign stored procedure, inserting a backdoor that will automatically restore their access. Identifying changes to stored procedures, particularly those that occur outside of a normal patching operation, can give you early visibility into this risk.



Last Scheduled Run: -- Never --

Next Scheduled Run: -- Never --

Repeats Every: -- Never --

Create/Update Schedule ☑

○ Disable ● Enable

Start At: 1/7/2025 12:00:00 PM

Repeat Every: 1 Days

Figure 3-16: AVDF: Configuring stored procedure auditing

AVDF also helps you analyze user entitlements (role and privilege grants). AVDF lets you take a snapshot of user privileges at a specific time and label it to compare it with other snapshots to see how entitlements have changed over time.

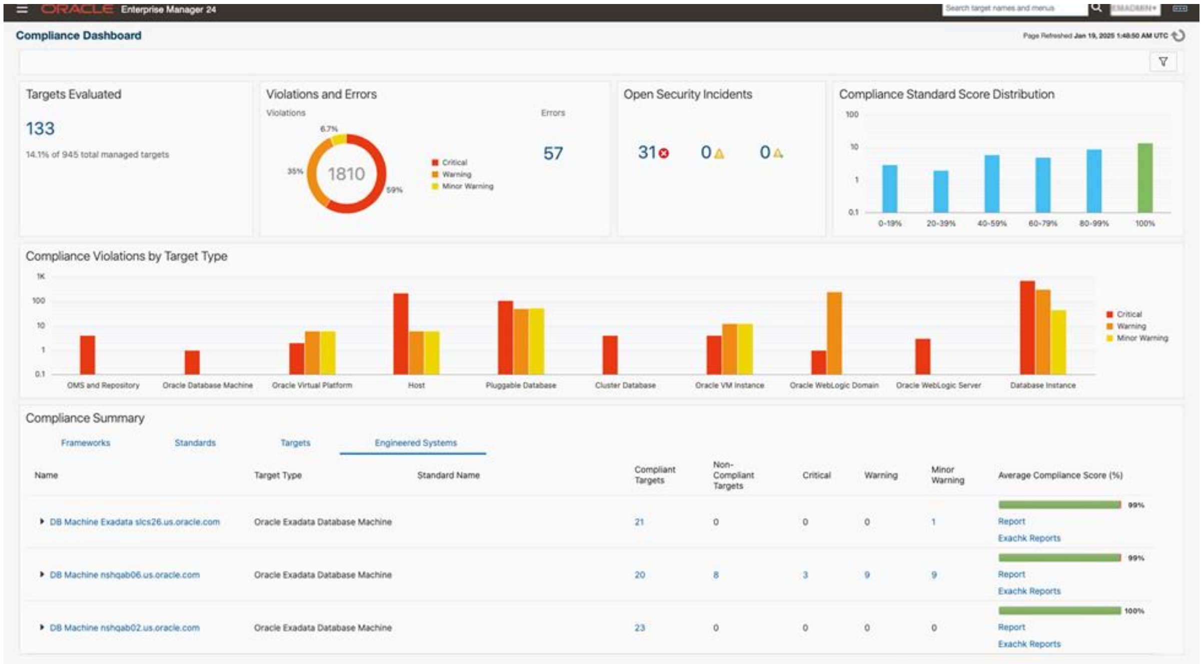
To know more about how Oracle Audit Vault and Database Firewall can help:

- To discover sensitive data, please see Chapter Four.
- To detect or block SQL injection attacks, please see Chapter Eight.
- To collect audit records and centralize reporting and alerting to meet regulatory requirements, please see Chapter Thirteen.

Configuration Assessment with Enterprise Manager

Oracle also offers the Enterprise Manager Database Lifecycle Management (DBLM) pack to address your enterprise needs for assessing security configuration. DBLM streamlines compliance management by automating the monitoring, assessment, and enforcement of industry standards and regulatory requirements. It promotes secure, consistent configurations across IT environments, offering detailed reports, customizable frameworks, and predefined compliance standards to enhance governance, security, and operational efficiency.

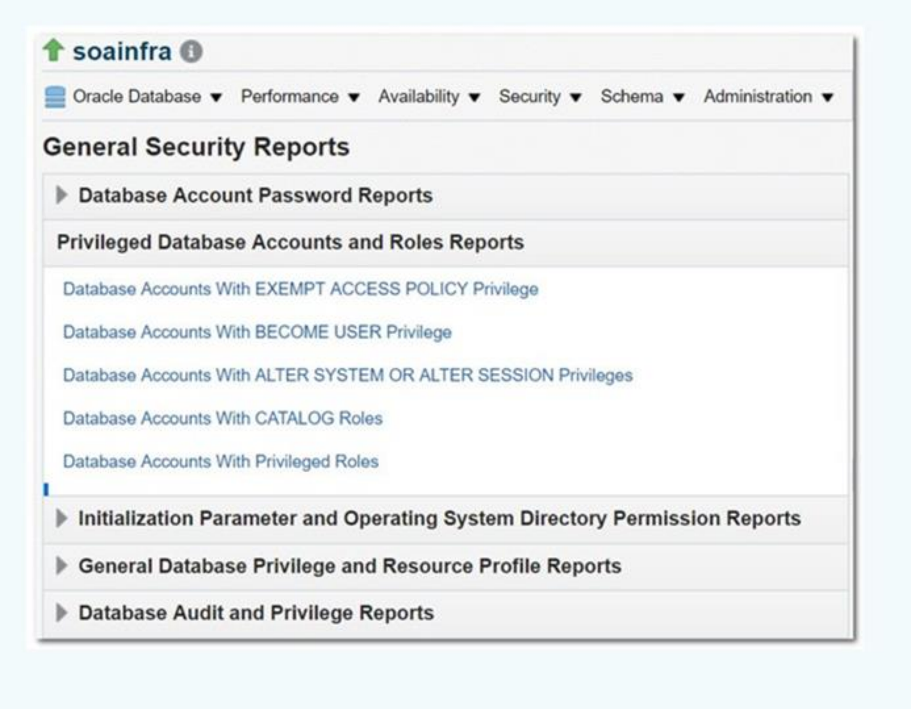
The compliance dashboard provides a centralized view of compliance status across IT environments. The dashboard displays scores, highlights non-compliant targets, and enables drill-down for detailed reports.



Name	Target Type	Standard Name	Compliant Targets	Non-Compliant Targets	Critical	Warning	Minor Warning	Average Compliance Score (%)
▸ DB Machine Exadata slcs26.us.oracle.com	Oracle Exadata Database Machine		21	0	0	0	1	99% Report Exachk Reports
▸ DB Machine nshqab06.us.oracle.com	Oracle Exadata Database Machine		20	8	3	9	9	99% Report Exachk Reports
▸ DB Machine nshqab02.us.oracle.com	Oracle Exadata Database Machine		23	0	0	0	0	100% Report Exachk Reports

Figure 3-17: Oracle Enterprise Manager Compliance Dashboard

DBLM's compliance frameworks include detailed reports on initialization parameters, OS directory permissions, user account profiles, user access and authorization restrictions, parameter settings and sensitive objects, offering a comprehensive compliance framework for effective database estate management with greater reliability and security.



General Security Reports

▸ Database Account Password Reports

Privileged Database Accounts and Roles Reports

- Database Accounts With EXEMPT ACCESS POLICY Privilege
- Database Accounts With BECOME USER Privilege
- Database Accounts With ALTER SYSTEM OR ALTER SESSION Privileges
- Database Accounts With CATALOG Roles
- Database Accounts With Privileged Roles

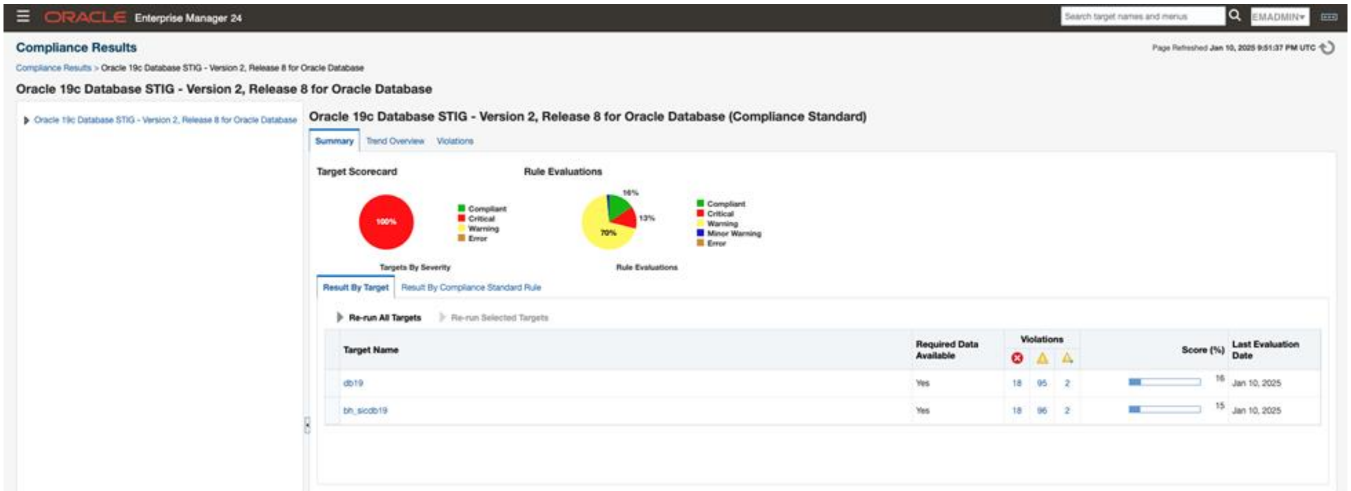
▸ Initialization Parameter and Operating System Directory Permission Reports

▸ General Database Privilege and Resource Profile Reports

▸ Database Audit and Privilege Reports

Figure 3-18: Enterprise Manager general security reports

DBLM tracks trends, helping with regulatory adherence and proactive compliance management. It includes support for CIS Benchmarks and the DISA Security Technical Implementation Guide (STIG), with predefined rules to validate compliance and promote Oracle best practices.



Compliance Results

Compliance Results > Oracle 19c Database STIG - Version 2, Release 8 for Oracle Database

Oracle 19c Database STIG - Version 2, Release 8 for Oracle Database

Oracle 19c Database STIG - Version 2, Release 8 for Oracle Database

Oracle 19c Database STIG - Version 2, Release 8 for Oracle Database (Compliance Standard)

Target Name	Required Data Available	Violations			Score (%)	Last Evaluation Date
		⊗	⚠	⚠		
db19	Yes	18	95	2	16	Jan 10, 2025
bh_sicdb19	Yes	18	96	2	15	Jan 10, 2025

Figure 3-19: Enterprise Manager compliance results for the STIG standard

DBLM supports dozens of out-of-the-box compliance standards, covering basic security configurations for Oracle Database, RAC nodes, and Oracle Listener.

Oracle Database Security Assessment Tool (DBSAT) integrates seamlessly with Oracle Enterprise Manager to enhance database security through automated assessments. This integration allows administrators to run DBSAT directly within Enterprise Manager, enabling centralized management and streamlined operations. It empowers organizations to continuously identify, assess, and mitigate risks, ensuring robust security across their database estate.

The Oracle Autonomous Health Framework (AHF) integrates with compliance frameworks to enhance Exadata security through automated configuration security and health checks. This integration, managed via Enterprise Manager, enables centralized oversight, continuous monitoring, and detailed reporting, streamlining remediation. It promotes proactive risk mitigation, regulatory adherence, and operational efficiency for secure, compliant Exadata environments.

Configuration Drift and Consistency Management with Enterprise Manager

DBLM configuration management provides robust capabilities, including baseline creation, continuous drift detection, compliance enforcement, automated remediation, and detailed reporting. It ensures consistent configurations across environments, integrates seamlessly with patching workflows, and delivers real-time alerts for proactive issue resolution. By automating monitoring and remediation, DBLM minimizes manual effort, enhances system reliability, ensures compliance with regulations, and bolsters security. Organizations gain improved operational stability, streamlined audits, and reduced risks, all from a centralized, user-friendly platform.

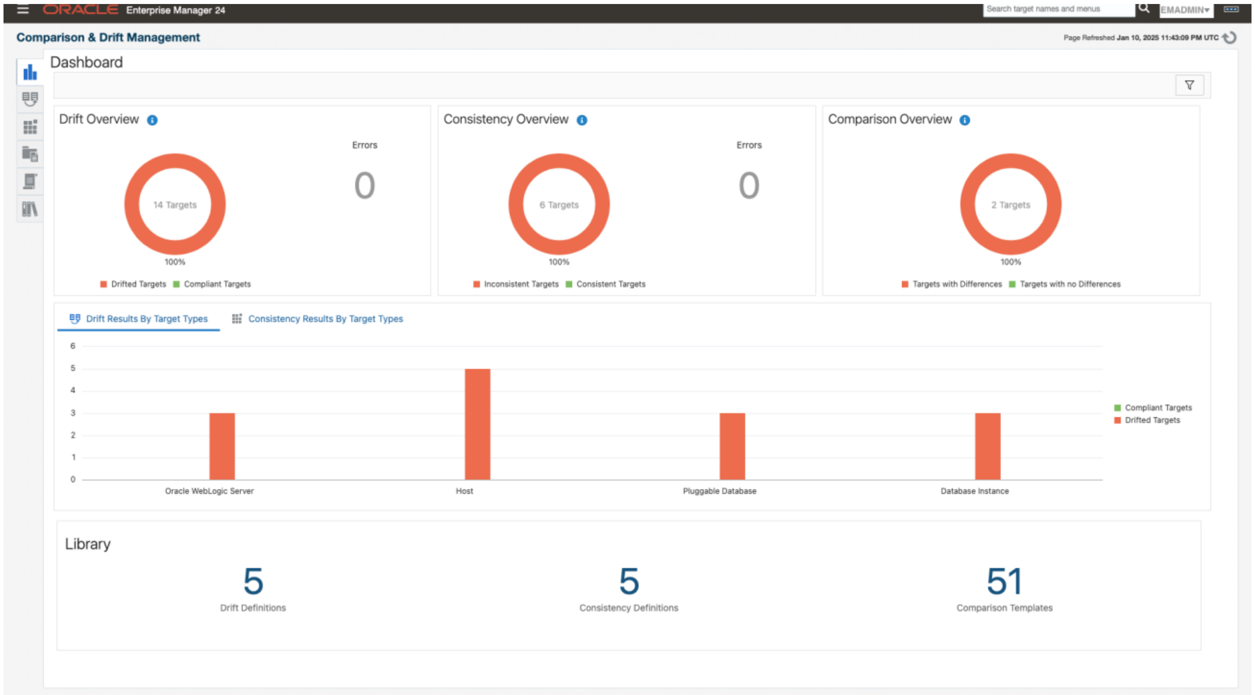

ORACLE Enterprise Manager 24
Search target names and menus
EMADMIN▾
Comparison & Drift Management
Page Refreshed Jan 10, 2025 11:43:09 PM UTC
Dashboard
Drift Overview
Errors
0
14 Targets
100%
🟥 Drifted Targets 🟩 Compliant Targets
Consistency Overview
Errors
0
6 Targets
100%
🟥 Inconsistent Targets 🟩 Consistent Targets
Comparison Overview
2 Targets
100%
🟥 Targets with Differences 🟩 Targets with no Differences
Drift Results By Target Types
Consistency Results By Target Types
6
5
4
3
2
1
0
Oracle WebLogic Server
Host
Pluggable Database
Database Instance
🟩 Compliant Targets
🟥 Drifted Targets
Library
5
Drift Definitions
5
Consistency Definitions
51
Comparison Templates

Figure 3-20: Configuration Drift and Consistency Management Dashboard

Asset discovery and grouping

DBLM eliminates the need to track IT assets, including databases, manually. It provides centralized inventory and usage tracking for IT assets, offering real-time visibility into hardware, databases, middleware, and applications. It automates discovery, maintains up-to-date inventories, monitors resource usage, and ensures compliance with licensing agreements. By tracking essential asset details such as versions, configurations, and interdependencies, DBLM supports capacity planning, cost optimization, and performance tuning. Integrated with lifecycle management tools, Enterprise Manager streamlines patching, provisioning, and configuration processes, enabling organizations to enhance efficiency, reduce costs, maintain compliance, and make informed decisions for reliable and optimized IT operations.

Choosing the right database security assessment tool

Oracle offers a range of assessment tools to help evaluate and enhance your security posture. Here is a rough guideline to assist you in selecting the right tool.

- The Database Security Assessment Tool is a simple standalone tool to assess the security configuration of a single Oracle database. But if you want to assess your fleet or track deviations, consider using Data Safe.
- Suppose you want to run security assessments at scale across many databases, track drift, and benefit from a unified console with other security services. In that case, Data Safe cloud service is your answer. Data Safe is also the tool of choice if your Oracle Databases run in a multicloud or hybrid deployment model and you are looking for one tool across all those databases.
- Another option for running security assessments at scale is Audit Vault and Database Firewall (AVDF). If you prefer not to use a cloud service, then AVDF is your best choice. Like Data Safe, AVDF offers much more than security assessment – we'll be returning to AVDF several times throughout this book.
- Oracle Enterprise Manager's Database Lifecycle Management (DBLM) pack is another option for assessing Oracle Database security. If you already use Enterprise Manager extensively and there is no requirement to

separate database management and security management duties within your organization, then DBLM could be a good option for you.

Reducing the blast radius with Privilege Analysis

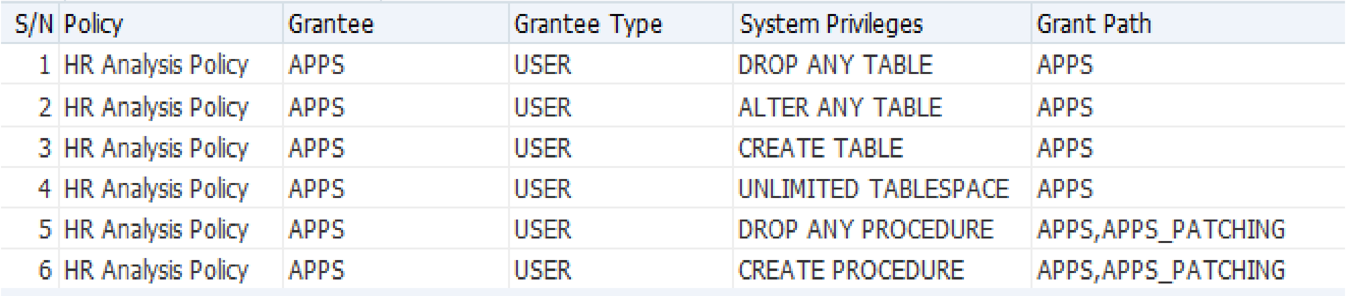
Assessing a database's security posture deals with more than security configuration settings. The end goal of the assessment process is to reduce security risk. A large part of the risk is tied up in database user accounts. As mentioned earlier, most database breaches involve the use of stolen credentials – the bad guys just log into the database and steal your data. Removing unnecessary privileges from user accounts reduces the damage those accounts could cause if they were compromised (and thus reduce the "blast radius" of the event).

The problem is that it is tough to determine whether a user has more privileges than they need. One way to identify overprivileged users is through entitlement reviews, where a knowledgeable person reviews the privileges and roles granted to a user and identifies privileges that are not needed. Tools like Data Safe's user assessment or AVDF's entitlement reviews facilitate that approach. However, managers may be reluctant to remove privileges for fear it will disrupt operations, and DBAs may hesitate to revoke roles and privileges from user accounts as the potential impact is unknown.

A better way to reduce unnecessary privileges is to track which privileges accounts are actually using and remove those not being used. Oracle Database's privilege analysis (PA) feature analyzes privilege and role usage for database users and application service accounts. PA reports on unused privileges and roles based on the actual user or application usage of the roles and privileges over time. Removing privileges you know are not being used is likely to be more productive than just asking a user's manager if they still need the privileges they already have.

PA reports reflect the actual privileges and roles used/unused by users and applications. The `DBA_USED_PRIVS` and `DBA_UNUSED_PRIVS` views show which privileges and roles have been used or not used, respectively.

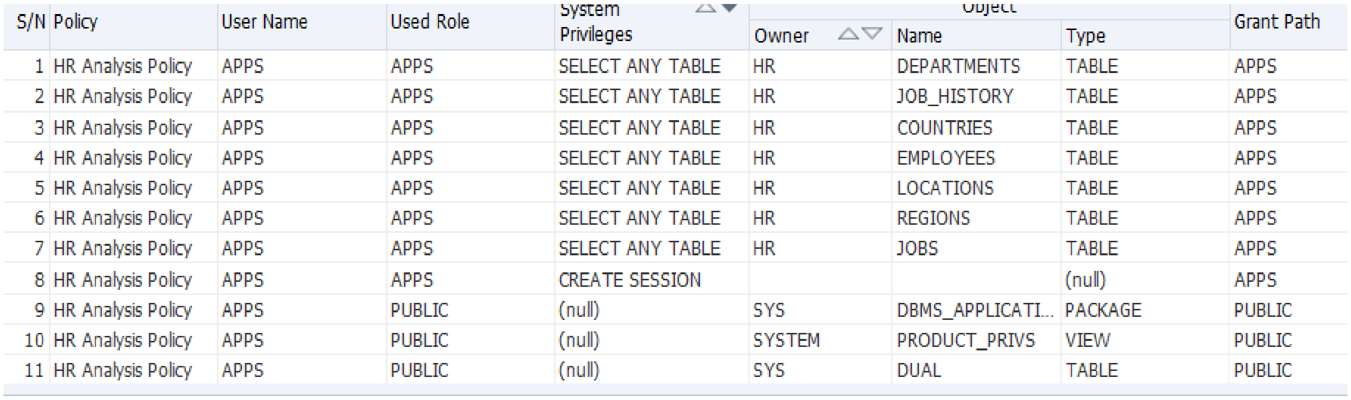
Figure 3-21 shows that the `APPS` user has privileges that are not being used: `DROP ANY TABLE`, `ALTER ANY TABLE`, `CREATE TABLE, and UNLIMITED TABLESPACE`. Also, `DROP ANY PROCEDURE and CREATE PROCEDURE` granted through both the APPS and `APPS_PATCHING` roles were not in use. Could that mean that the `APPS` user was granted a role once to apply a patch and that the role never got revoked?



S/N	Policy	Grantee	Grantee Type	System Privileges	Grant Path
1	HR Analysis Policy	APPS	USER	DROP ANY TABLE	APPS
2	HR Analysis Policy	APPS	USER	ALTER ANY TABLE	APPS
3	HR Analysis Policy	APPS	USER	CREATE TABLE	APPS
4	HR Analysis Policy	APPS	USER	UNLIMITED TABLESPACE	APPS
5	HR Analysis Policy	APPS	USER	DROP ANY PROCEDURE	APPS,APPS_PATCHING
6	HR Analysis Policy	APPS	USER	CREATE PROCEDURE	APPS,APPS_PATCHING

Figure 3-21: Report on DBA_UNUSED_PRIVS

Privilege analysis lets you go even deeper. Figure 3-22 shows interesting details of used roles and privileges as they relate to specific database objects. Here, we can see that the `APPS` user was granted `SELECT ANY TABLE` when, in fact, the user is selecting from specific tables on the HR schema (`DEPARTMENTS`, `JOB_HISTORY`, `COUNTRIES`, `EMPLOYEES`, `LOCATIONS`, `REGIONS`, and `JOBS`). In this case, the risk inherent in the APPS account might be reduced by revoking the `SELECT ANY TABLE` system privilege that allows the `APPS` user to select from any table in the database and replacing it with grants of object privileges on the required tables (e.g., `GRANT SELECT on HR.DEPARTMENTS to APPS`) instead. If APPS requires access to ALL tables in the HR schema, you may `GRANT SELECT ANY TABLE on HR to APPS` – this is an example of granting schema-level privileges, a new feature in Oracle Database 23ai.



S/N	Policy	User Name	Used Role	System Privileges	Object			Grant Path
					Owner	Name	Type	
1	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	DEPARTMENTS	TABLE	APPS
2	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	JOB_HISTORY	TABLE	APPS
3	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	COUNTRIES	TABLE	APPS
4	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	EMPLOYEES	TABLE	APPS
5	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	LOCATIONS	TABLE	APPS
6	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	REGIONS	TABLE	APPS
7	HR Analysis Policy	APPS	APPS	SELECT ANY TABLE	HR	JOBS	TABLE	APPS
8	HR Analysis Policy	APPS	APPS	CREATE SESSION			(null)	APPS
9	HR Analysis Policy	APPS	PUBLIC	(null)	SYS	DBMS_APPLICATI...	PACKAGE	PUBLIC
10	HR Analysis Policy	APPS	PUBLIC	(null)	SYSTEM	PRODUCT_PRIVS	VIEW	PUBLIC
11	HR Analysis Policy	APPS	PUBLIC	(null)	SYS	DUAL	TABLE	PUBLIC

Figure 3-22: Report on DBA_USED_PRIVS showing object access

Static-based role/privilege analysis tools (e.g., DBSAT or Data Safe) provide a good starting point but can only show which roles and privileges are granted to users. With a deeper understanding of the privileges required for an application to run with the least privileges needed to function, database administrators can confidently refine roles and privileges granted to limit unnecessary grants. Reducing unnecessary privileges reduces the attack surface and the potential impact of a stolen database user account credentials or account misuse.

Privilege Analysis lets you:

- Report on actual privileges and roles used in the database.
- Identify unused privileges and roles by users and applications.
- Reduce risk by helping enforce least privilege for users and applications.

Security patch management

Patching can be considered part of the assessment since that's where you're most likely to learn you are missing security updates. Every quarter, Oracle routinely provides database fixes for functional, performance, or security issues discovered by internal testing or reported by customers and external researchers. The security fixes can cover a wide range of topics, including:

- Vulnerable SQL statements, buffer overflows, SQL injections, etc.
- Vulnerable database clients, JDBC drivers, third-party code, etc.
- Weaknesses in cryptography, networking, remote code execution, etc.

The timely application of patches is necessary for organizations to maintain a proper security posture. If you are not applying patches, then you are accepting the risk of known vulnerabilities!

Proactive Maintenance with RUs and MRPs

For proactive maintenance, apply the quarterly patch bundle (Release Update) available from the My Oracle Support (MOS) Customer Portal for each Oracle Database software release. The RUs are released quarterly on the third Tuesday of January, April, July, and October. Each RU gets a maximum of six Monthly Recommended Patches (MRPs).

Release Updates (RUs)

RUs are highly tested bundles of critical fixes that enable you to avoid known issues. They usually contain the following types of fixes: security, regression (bug), optimizer, and functional (which may include feature extensions).

Oracle recommends that you stay current by using RUs. This minimizes the chance of encountering known bugs and security vulnerabilities. If you run your databases on Linux x86-64 platforms, have tested an application against an RU, and want to go live, you should check for the latest MRP and apply it.

Quarterly release updates are announced on the Critical Patch Updates, Security Alerts, and Bulletins page each January, April, July, and October.

Monthly Recommended Patches (MRP)

With update 19.17, Oracle began releasing MRPs for Oracle Database installed on Linux x86-64 to provide proactive patching between Release Updates. MRP bundles a collection of recommended one-off fixes, including security fixes, delivered monthly as a merge patch via a single downloadable patch for a given RU. **Unlike an RU, an MRP does not affect the release number**. This distinction may be important if you have to run your application quality assurance/testing cycles with a specific major version and do not want to do it again.

MRPs are delivered for each RU in the six months following each RU's release. MRPs include the fixes documented in "Oracle Database Important Recommended Patches" (MOS note 555.1), plus the prior MRPs for the RU.

Each MRP includes the latest critical and regression fixes and critical content released up to six months prior. By waiting to apply release updates for up to six months, you can take a more conservative approach to Oracle Database software maintenance while still minimizing your security risk. Table 3-1 contrasts release updates with MRPs.

Criteria	Release Update (RU)	MRPs
Cadence	Quarterly	Monthly for long-term releases on Linux x86-64
Zero downtime	RAC Rolling	RAC Rolling
Security fixes	Included	May include CPU Alerts and fixes for vulnerabilities with high CVSS scores
Regression fixes	Included	Included
Proactive functional fixes	Included	Not included
Optimizer plan changes (off by default)	Included	Not included
Minor functional enhancements	Included	Not included
Emergency one-offs	Included	Included
Supported operating systems	All supported platforms	Linux x86-64

Table 3-1: Difference between RUs and MRPs

We strongly recommend keeping your database and Oracle Grid Infrastructure up to date by applying RUs to fix known security vulnerabilities and minimize the risk of a successful attack. RUs include the most recent security, regression, and critical fixes. Staying current with RUs reduces the likelihood of requiring separate interim one-off patches, which lead to unique software baselines and the potential for ongoing costly patch maintenance.

Patching tools

Patching an Oracle Database requires planning as the process is complex and can lead to downtime if not properly managed. We recommend out-of-place patching for rolling patching, granular patching, and easier rollback.

Oracle Fleet Patching and Provisioning (FPP) helps control your database fleet lifecycle using automation, standardization, and out-of-place patching. The FPP drift detection capabilities can verify the environment's compliance. Routine operations like provisioning new clusters and databases, installing patched Oracle binaries, patching clusters and databases, or upgrading them are completely automated with FPP. These blueprints guarantee that all planned maintenance operations are executed in the correct order with the least impact on the business.

Larger organizations implementing FPP can patch hundreds or thousands of databases per maintenance window with the minimum human interaction, enabling consistent time and money savings. Small organizations with limited resources can also benefit from it.

You can use FPP if your targets have Oracle RAC or Oracle RAC One licenses or have licensed the Enterprise Manager Database Lifecycle Management (DBLM) Pack for single-instance databases.

Oracle LiveLabs

Oracle LiveLabs gives you access to Oracle's tools and technologies to run a wide variety of labs and workshops at your own pace. If you want try the assessment technologies discussed in this chapter, please go to:

- Database Security Assessment Tool
- Get Started with Oracle Data Safe Fundamentals
- Audit Vault and Database Firewall workshop
- Privilege Analysis workshop
- Oracle Fleet Patching and Provisioning workshop

Summary

Knowing how securely your databases are configured is the foundation for a defense-in-depth strategy. Configuration drift needs to be monitored, the database must be patched, and appropriate controls need to be implemented. No system is 100% secure, but overlooking basic security controls only makes life easier for attackers.

Oracle Database Security Assessment Tool (DBSAT), Oracle Data Safe, Oracle Audit Vault and Database Firewall (AVDF), and Oracle Database Lifecycle Management Pack (DBLM) help you identify the gaps in your Oracle database security configuration and the recommendations to overcome these gaps.

Chapter Four

Discovering Sensitive Data

Why is sensitive data important?

In today's data-driven world, organizations that effectively collect, analyze, and utilize data are more likely to succeed and thrive. The amount of data you collect and manage grows every day. For most organizations, that data is vital to the organization's central function. It might be your customer list, with the information needed to effectively support and market to your customers. It might be the data your organization needs to make data-driven decisions that let you reduce operating costs, enter new markets, bill for services rendered, or track clinical trials of the next great wonder drug. It doesn't matter what your business is, chances are good that the data you collect and store in your databases is important to your organization's success.

Much of the collected data is sensitive or personal. The more sensitive the data is, the more likely that someone else will find it valuable and attempt to steal it, and the greater the level of protection you'll want to apply to it to prevent theft or misuse.

Why should you focus on discovering sensitive data?

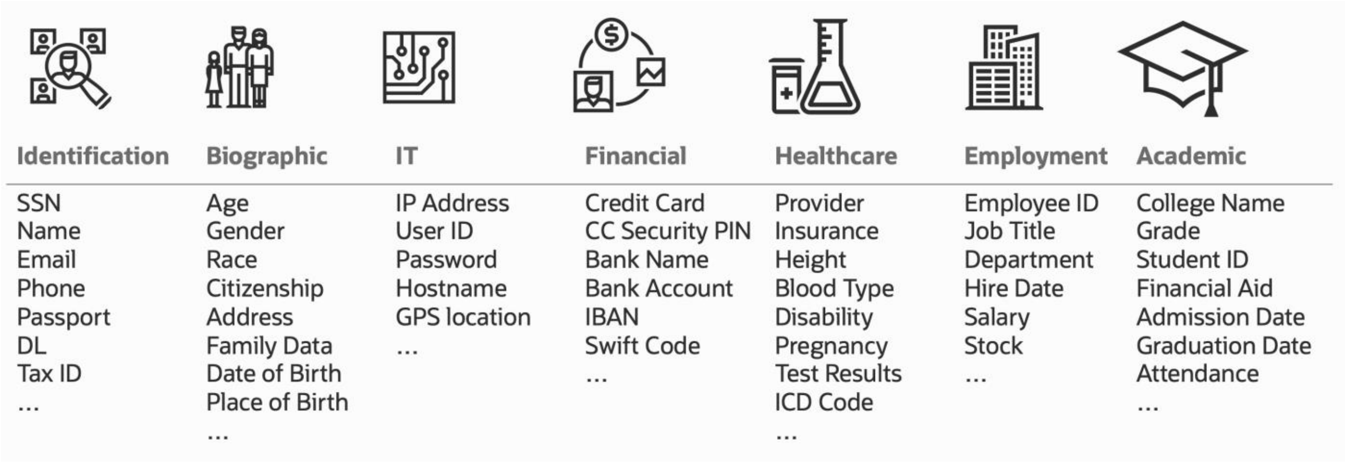
How much time and effort should you expend on securing your databases? As we discussed in Chapter 2, the answer to that question MIGHT be driven by regulations, but even if your database is governed by regulatory requirements, you should also consider how much business risk a compromise of the database would produce. The higher the level of risk, the greater the level of control you'll implement to reduce that risk to an acceptable level. Before you can protect sensitive data, you need to know what kind and how much sensitive data a database contains and where in the database that data is located. This knowledge helps you implement appropriate security controls to protect the data.

This chapter introduces the basic elements of sensitive data discovery and gives you an overview of the Oracle technologies that can be used to discover sensitive data, including:

- Database Security Assessment Tool (DBSAT)
- Oracle Data Safe
- Oracle Audit Vault and Database Firewall
- Enterprise Manager Application Data Modeling

What is sensitive data?

What is considered sensitive data? The answer to that depends on your organization, but a good rule of thumb is that if you don't want to see your data published openly on the internet, the data is sensitive, and should be protected from access by unauthorized people, whether inside or outside your organization. Sensitive data can be classified into categories such as identification, biographic, healthcare, financial, employment, and academic data. The following figure shows a few examples of categories and types of sensitive data.



Identification	Biographic	IT	Financial	Healthcare	Employment	Academic
SSN	Age	IP Address	Credit Card	Provider	Employee ID	College Name
Name	Gender	User ID	CC Security PIN	Insurance	Job Title	Grade
Email	Race	Password	Bank Name	Height	Department	Student ID
Phone	Citizenship	Hostname	Bank Account	Blood Type	Hire Date	Financial Aid
Passport	Address	GPS location	IBAN	Disability	Salary	Admission Date
DL	Family Data	…	Swift Code	Pregnancy	Stock	Graduation Date
Tax ID	Date of Birth		…	Test Results	…	Attendance
…	Place of Birth			ICD Code		…
	…			…		

Figure 4-1: Examples of sensitive data categories and types

It's unlikely that you'll only be concerned with one of these categories. Different regulations are likely to focus on different types of data. For example, PCI-DSS is highly focused on payment account numbers. Sarbanes-Oxley focuses on financial data. GDPR on identification and biographic data.

Oracle offers four different sensitive data discovery tools:

- Database Security Assessment Tool (DBSAT)
- Oracle Data Safe
- Oracle Audit Vault and Database Firewall (AVDF)
- Enterprise Manager (EM) Application Data Modeling (ADM)

At least one of them will be available for you to use for your Oracle Database, and in most cases, you will be able to choose between several of them. Which tool you use depends on your goal and (in most cases) which is easiest for you to use.

One way to discover sensitive data is to search for column names and comments using keywords or search patterns (regular expressions). Another is to use data patterns to check column values. When combined with a column name or comment match, a data pattern helps raise confidence that a certain column has sensitive data. All four of Oracle's sensitive data discovery tools use one or both methods to locate sensitive data.

Discovering sensitive data using DBSAT

Chapter Three introduced you to the Oracle Database Security Assessment Tool (DBSAT), which you can use to assess database security posture and identify areas where you can reduce risk. DBSAT can also scan your database to identify tables that contain sensitive data by inspecting column names and comments to determine if they hold sensitive data. DBSAT checks table statistics in the data dictionary to determine the quantity of sensitive data (number of rows) in the table. DBSAT generates detailed data assessment reports in HTML and spreadsheet formats, with sensitive columns classified into categories for easier management.

DBSAT helps discover sensitive columns in English and provides sample files for seven major European languages: Dutch, French, German, Greek, Italian, Portuguese, and Spanish.

DBSAT's is flexible, with a library of sensitive data definitions that is extensible to include your unique types of sensitive data. Assigned categories can be easily modified to suit your needs.

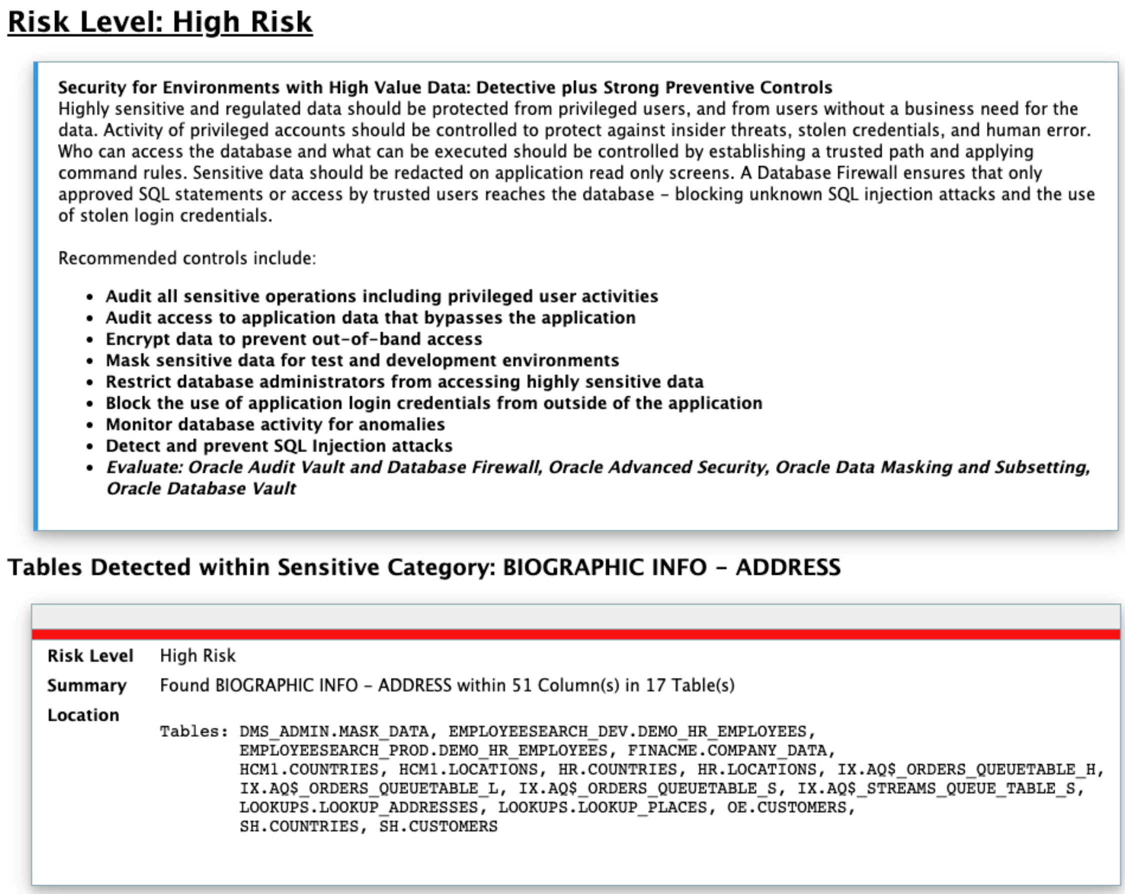
Sensitive data assessment report using DBSAT

DBSAT's sensitive data assessment report helps you understand what kind and how much sensitive data a database has, and where it is located. Table 4-1 shows the summary section that provides information about the number of tables, columns, and rows identified as sensitive, grouped by sensitive category and subcategory.

Sensitive Category	# Sensitive tables	# Sensitive columns	# Sensitive rows
BIOGRAPHIC INFO - ADDRESS	9	36	6307209
BIOGRAPHIC INFO – EXTENDED PII	2	2	2000
FINANCIAL INFO – BANK DATA	2	2	830
FINANCIAL INFO – CARD DATA	7	7	3235
HEALTH INFO – PROVIDER DATA	1	1	149
IDENTIFICATION INFO – NATIONAL IDS	2	6	2000
IDENTIFICATION INFO – PERSONAL IDS	3	3	405
IT INFO – USER DATA	13	15	13228
JOB INFO – COMPENSATION DATA	10	12	3380
JOB INFO – EMPLOYEE DATA	8	16	406
JOB INFO – ORG DATA	5	6	278
Total	**29**	**132**	**8617644**

Table 4-1: Sensitive Data Report - Category Summary

Each sensitive category has a preconfigured risk level (high, medium, or low) which you can customize if desired. The report recommends protecting sensitive data based on the associated risk level. Figure 4-2 shows some recommendations for protecting high-risk data, such as personal health information.



Risk Level: High Risk

Security for Environments with High Value Data: Detective plus Strong Preventive Controls
Highly sensitive and regulated data should be protected from privileged users, and from users without a business need for the data. Activity of privileged accounts should be controlled to protect against insider threats, stolen credentials, and human error. Who can access the database and what can be executed should be controlled by establishing a trusted path and applying command rules. Sensitive data should be redacted on application read only screens. A Database Firewall ensures that only approved SQL statements or access by trusted users reaches the database – blocking unknown SQL injection attacks and the use of stolen login credentials.

Recommended controls include:

- **Audit all sensitive operations including privileged user activities**
- **Audit access to application data that bypasses the application**
- **Encrypt data to prevent out-of-band access**
- **Mask sensitive data for test and development environments**
- **Restrict database administrators from accessing highly sensitive data**
- **Block the use of application login credentials from outside of the application**
- **Monitor database activity for anomalies**
- **Detect and prevent SQL Injection attacks**
- ***Evaluate: Oracle Audit Vault and Database Firewall, Oracle Advanced Security, Oracle Data Masking and Subsetting, Oracle Database Vault***

Tables Detected within Sensitive Category: BIOGRAPHIC INFO – ADDRESS

Risk Level	High Risk
Summary	Found BIOGRAPHIC INFO – ADDRESS within 51 Column(s) in 17 Table(s)
Location	

```
Tables: DMS_ADMIN.MASK_DATA, EMPLOYEESEARCH_DEV.DEMO_HR_EMPLOYEES,
        EMPLOYEESEARCH_PROD.DEMO_HR_EMPLOYEES, FINACME.COMPANY_DATA,
        HCM1.COUNTRIES, HCM1.LOCATIONS, HR.COUNTRIES, HR.LOCATIONS, IX.AQ$_ORDERS_QUEUETABLE_H,
        IX.AQ$_ORDERS_QUEUETABLE_L, IX.AQ$_ORDERS_QUEUETABLE_S, IX.AQ$_STREAMS_QUEUE_TABLE_S,
        LOOKUPS.LOOKUP_ADDRESSES, LOOKUPS.LOOKUP_PLACES, OE.CUSTOMERS,
        SH.COUNTRIES, SH.CUSTOMERS
```

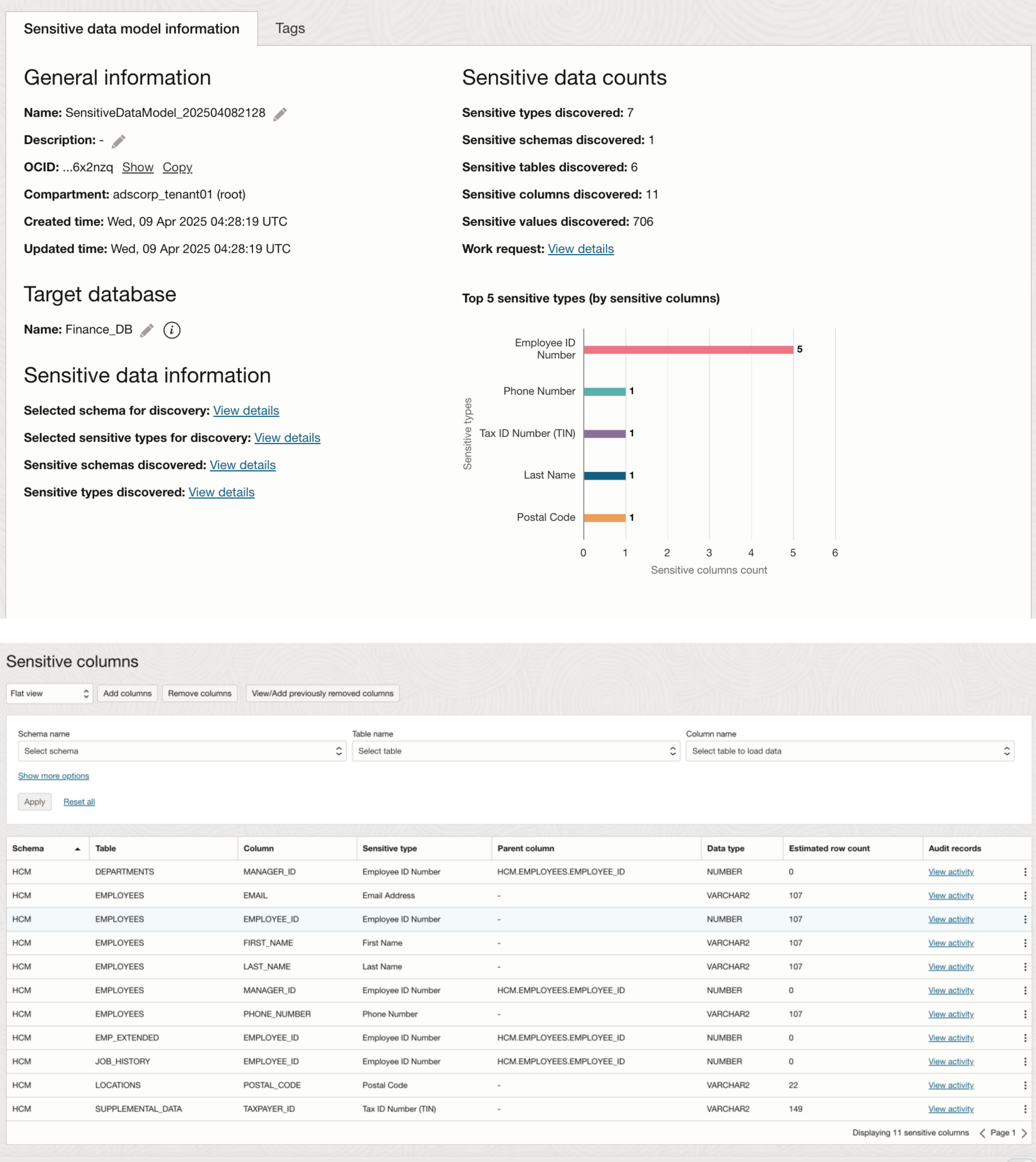
Figure 4-2: Recommendations for protecting data

The Sensitive Data Assessment report also provides schema, table, column level details, and statistics to help understand where the sensitive data is and how much you have in the database. These results can be used to implement appropriate security controls to protect sensitive data.

Discovering sensitive data using Data Safe

Oracle Data Safe, a database security cloud service introduced in Chapter 3, includes discovery capabilities for several country-specific identifiers such as Brazil, Canada, France, Germany, India, Italy, México, Netherlands, Portugal, Spain, the UK, and the US. You can also define your own custom types and categories of sensitive data.

Figure 4-3 shows a sample data discovery report generated by Data Safe.



Sensitive data model information | Tags

General information

Name: SensitiveDataModel_202504082128

Description: -

OCID: ...6x2nzq Show Copy

Compartment: adscorp_tenant01 (root)

Created time: Wed, 09 Apr 2025 04:28:19 UTC

Updated time: Wed, 09 Apr 2025 04:28:19 UTC

Target database

Name: Finance_DB

Sensitive data information

Selected schema for discovery: View details

Selected sensitive types for discovery: View details

Sensitive schemas discovered: View details

Sensitive types discovered: View details

Sensitive data counts

Sensitive types discovered: 7

Sensitive schemas discovered: 1

Sensitive tables discovered: 6

Sensitive columns discovered: 11

Sensitive values discovered: 706

Work request: View details

Top 5 sensitive types (by sensitive columns)

Sensitive columns

Schema name: Select schema

Table name: Select table

Column name: Select table to load data

Schema	Table	Column	Sensitive type	Parent column	Data type	Estimated row count	Audit records
HCM	DEPARTMENTS	MANAGER_ID	Employee ID Number	HCM.EMPLOYEES.EMPLOYEE_ID	NUMBER	0	View activity
HCM	EMPLOYEES	EMAIL	Email Address	-	VARCHAR2	107	View activity
HCM	EMPLOYEES	EMPLOYEE_ID	Employee ID Number	-	NUMBER	107	View activity
HCM	EMPLOYEES	FIRST_NAME	First Name	-	VARCHAR2	107	View activity
HCM	EMPLOYEES	LAST_NAME	Last Name	-	VARCHAR2	107	View activity
HCM	EMPLOYEES	MANAGER_ID	Employee ID Number	HCM.EMPLOYEES.EMPLOYEE_ID	NUMBER	0	View activity
HCM	EMPLOYEES	PHONE_NUMBER	Phone Number	-	VARCHAR2	107	View activity
HCM	EMP_EXTENDED	EMPLOYEE_ID	Employee ID Number	HCM.EMPLOYEES.EMPLOYEE_ID	NUMBER	0	View activity
HCM	JOB_HISTORY	EMPLOYEE_ID	Employee ID Number	HCM.EMPLOYEES.EMPLOYEE_ID	NUMBER	0	View activity
HCM	LOCATIONS	POSTAL_CODE	Postal Code	-	VARCHAR2	22	View activity
HCM	SUPPLEMENTAL_DATA	TAXPAYER_ID	Tax ID Number (TIN)	-	VARCHAR2	149	View activity

Displaying 11 sensitive columns Page 1

Figure 4-3: Data discovery report in Oracle Data Safe

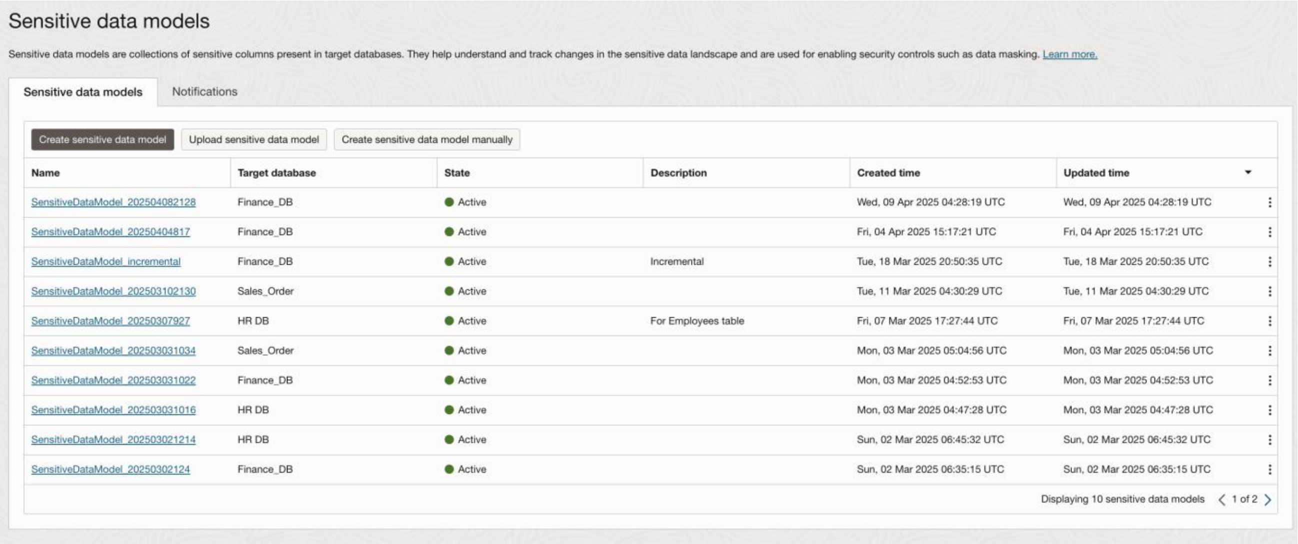
Data Safe scans column names, comments, and data values. Data Safe identifies relationships between primary and foreign key columns, which will be important in Chapter Ten when we discuss masking sensitive data. Data Safe also helps you identify non-dictionary referential relationships defined in the application. As you can see in the right-most column of Figure 4-3, the Data Safe console also makes it easy to view activity and drill down into audit records related to the data. We will discuss Data Safe's capabilities for audit data collection in Chapter Thirteen.

Sensitive Data Models in Data Safe

Data Discovery reports provide summary totals of sensitive tables, columns, and values and details about the sensitive columns. The sensitive columns are categorized based on their sensitive types. These results, along with the primary/foreign key information, are stored as a sensitive data model. We'll return to the sensitive data model in Chapter Ten when we discuss data masking. You can optionally store metadata in a sensitive data model, including sample data and estimated row counts as you can see above in Figure 4-3. Collecting sample data and row counts gives you a perspective on the quantity of the different types of sensitive data in your target databases. Because many organizations prefer not to copy actual data values outside of the database containing them, Data Safe defaults to not collecting sample data. If you'd like to know more about what information Data Safe collects from your databases, refer to the reference section of the Data Safe Administrator's guide.

When changes occur in a target database, you can perform incremental updates to its sensitive data model, add and remove sensitive columns from the sensitive data model, and manage the referential relationships between the sensitive columns. Data Safe also allows you to start discovery with a smaller scope (e.g., just looking for sensitive types of just one sensitive category or only scanning select database schemas) and then using incremental discovery to broaden the scope of the discovery and the sensitive data model.

You can download a sensitive data model, modify it offline, and then upload it into the same or other Oracle Data Safe regions. A sensitive data model is associated with one target database at a time, although you can change that target database as needed.



Sensitive data models

Sensitive data models are collections of sensitive columns present in target databases. They help understand and track changes in the sensitive data landscape and are used for enabling security controls such as data masking. Learn more.

Sensitive data models
Notifications
Create sensitive data model
Upload sensitive data model
Create sensitive data model manually

Name	Target database	State	Description	Created time	Updated time
SensitiveDataModel_202504082128	Finance_DB	● Active		Wed, 09 Apr 2025 04:28:19 UTC	Wed, 09 Apr 2025 04:28:19 UTC
SensitiveDataModel_20250404817	Finance_DB	● Active		Fri, 04 Apr 2025 15:17:21 UTC	Fri, 04 Apr 2025 15:17:21 UTC
SensitiveDataModel_incremental	Finance_DB	● Active	Incremental	Tue, 18 Mar 2025 20:50:35 UTC	Tue, 18 Mar 2025 20:50:35 UTC
SensitiveDataModel_202503102130	Sales_Order	● Active		Tue, 11 Mar 2025 04:30:29 UTC	Tue, 11 Mar 2025 04:30:29 UTC
SensitiveDataModel_20250307927	HR DB	● Active	For Employees table	Fri, 07 Mar 2025 17:27:44 UTC	Fri, 07 Mar 2025 17:27:44 UTC
SensitiveDataModel_202503031034	Sales_Order	● Active		Mon, 03 Mar 2025 05:04:56 UTC	Mon, 03 Mar 2025 05:04:56 UTC
SensitiveDataModel_202503031022	Finance_DB	● Active		Mon, 03 Mar 2025 04:52:53 UTC	Mon, 03 Mar 2025 04:52:53 UTC
SensitiveDataModel_202503031016	HR DB	● Active		Mon, 03 Mar 2025 04:47:28 UTC	Mon, 03 Mar 2025 04:47:28 UTC
SensitiveDataModel_202503021214	HR DB	● Active		Sun, 02 Mar 2025 06:45:32 UTC	Sun, 02 Mar 2025 06:45:32 UTC
SensitiveDataModel_20250302124	Finance_DB	● Active		Sun, 02 Mar 2025 06:35:15 UTC	Sun, 02 Mar 2025 06:35:15 UTC

Displaying 10 sensitive data models 1 of 2

Figure 4-4: Data discovery sensitive data models

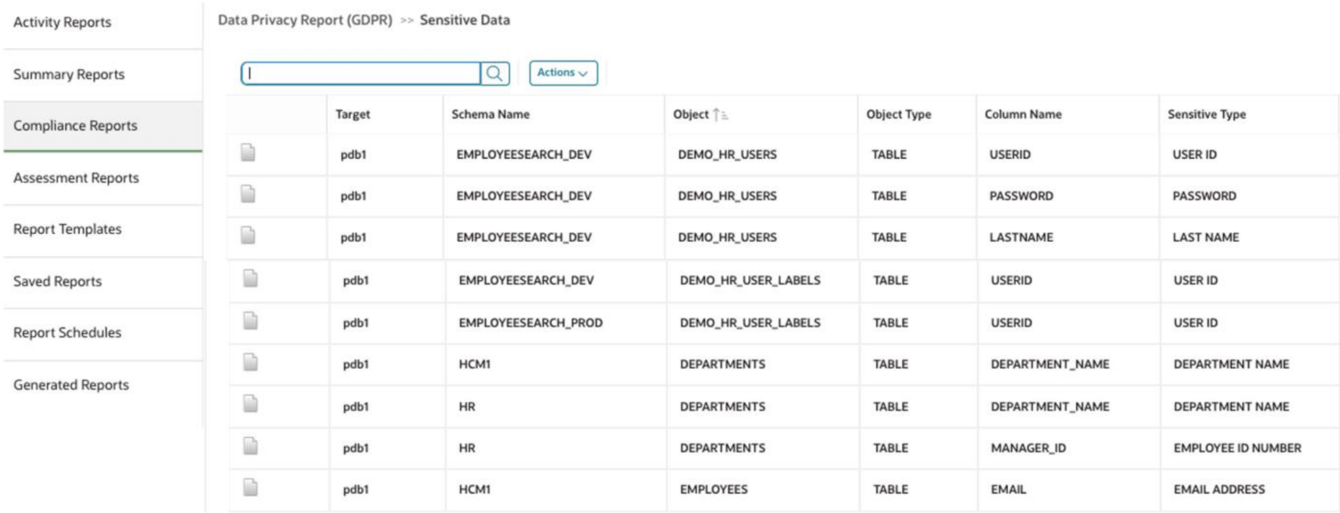
Sensitive data models help you design other security controls, such as data masking, auditing, Database Vault policies, and fine-grained access controls. For example, you can define a masking policy using a sensitive data model to mask the sensitive data on non-production database clones (like test and development instances).

Discovering sensitive data using Audit Vault and Database Firewall

Oracle Audit Vault and Database Firewall (AVDF) offers another option for discovering sensitive data in Oracle Databases. Unlike Data Safe, AVDF does not scan column data – like DBSAT, it produces results based on column names and comments. Sensitive data scans in AVDF are initiated by scheduling a Sensitive Object discovery job.

Data Privacy reports leverage the discovered sensitive data and audit policies that capture actions on the identified sensitive objects. With the Data Privacy reports, you can view the following:

- Sensitive Data: Target name, schema name, column name, and sensitive type
- Activity on Sensitive Data: Displays details about activity on sensitive data by all users
- Activity on Sensitive Data by Privileged Users: Displays activity on sensitive data by privileged users



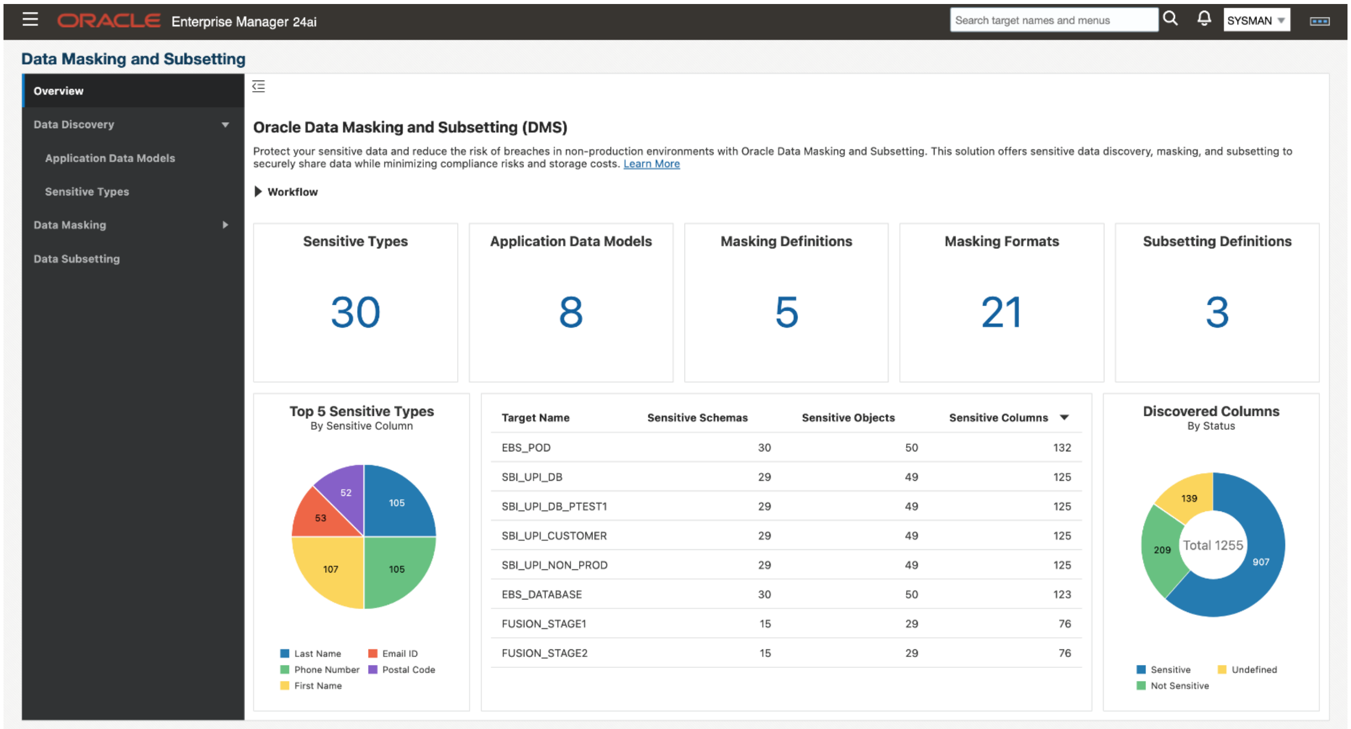
	Target	Schema Name	Object	Object Type	Column Name	Sensitive Type
	pdb1	EMPLOYEESEARCH_DEV	DEMO_HR_USERS	TABLE	USERID	USER ID
	pdb1	EMPLOYEESEARCH_DEV	DEMO_HR_USERS	TABLE	PASSWORD	PASSWORD
	pdb1	EMPLOYEESEARCH_DEV	DEMO_HR_USERS	TABLE	LASTNAME	LAST NAME
	pdb1	EMPLOYEESEARCH_DEV	DEMO_HR_USER_LABELS	TABLE	USERID	USER ID
	pdb1	EMPLOYEESEARCH_PROD	DEMO_HR_USER_LABELS	TABLE	USERID	USER ID
	pdb1	HCM1	DEPARTMENTS	TABLE	DEPARTMENT_NAME	DEPARTMENT NAME
	pdb1	HR	DEPARTMENTS	TABLE	DEPARTMENT_NAME	DEPARTMENT NAME
	pdb1	HR	DEPARTMENTS	TABLE	MANAGER_ID	EMPLOYEE ID NUMBER
	pdb1	HCM1	EMPLOYEES	TABLE	EMAIL	EMAIL ADDRESS

Figure 4-5: Sensitive data report

In addition to creating reports of sensitive data within the database, AVDF can use the collected information in Database Firewall policies for specifying alert conditions, compliance, and activity reporting and as part of the criteria for generating alerts.

Discovering sensitive data using Enterprise Manager

Our final option for discovering sensitive data leverages the data discovery module in Oracle Enterprise Manager (EM). Data discovery, and especially the sensitive types feature of data discovery, is part of Oracle Data Masking and Subsetting, an Enterprise Manager pack. Licensing for data discovery is also included with database security options like Advanced Security, Database Vault, and Label Security (we'll be discussing these options later in the book). Like Oracle Data Safe, Enterprise Manager's data discovery examines column names, comments, and data to discover sensitive columns.



Oracle Data Masking and Subsetting (DMS)

Protect your sensitive data and reduce the risk of breaches in non-production environments with Oracle Data Masking and Subsetting. This solution offers sensitive data discovery, masking, and subsetting to securely share data while minimizing compliance risks and storage costs. Learn More

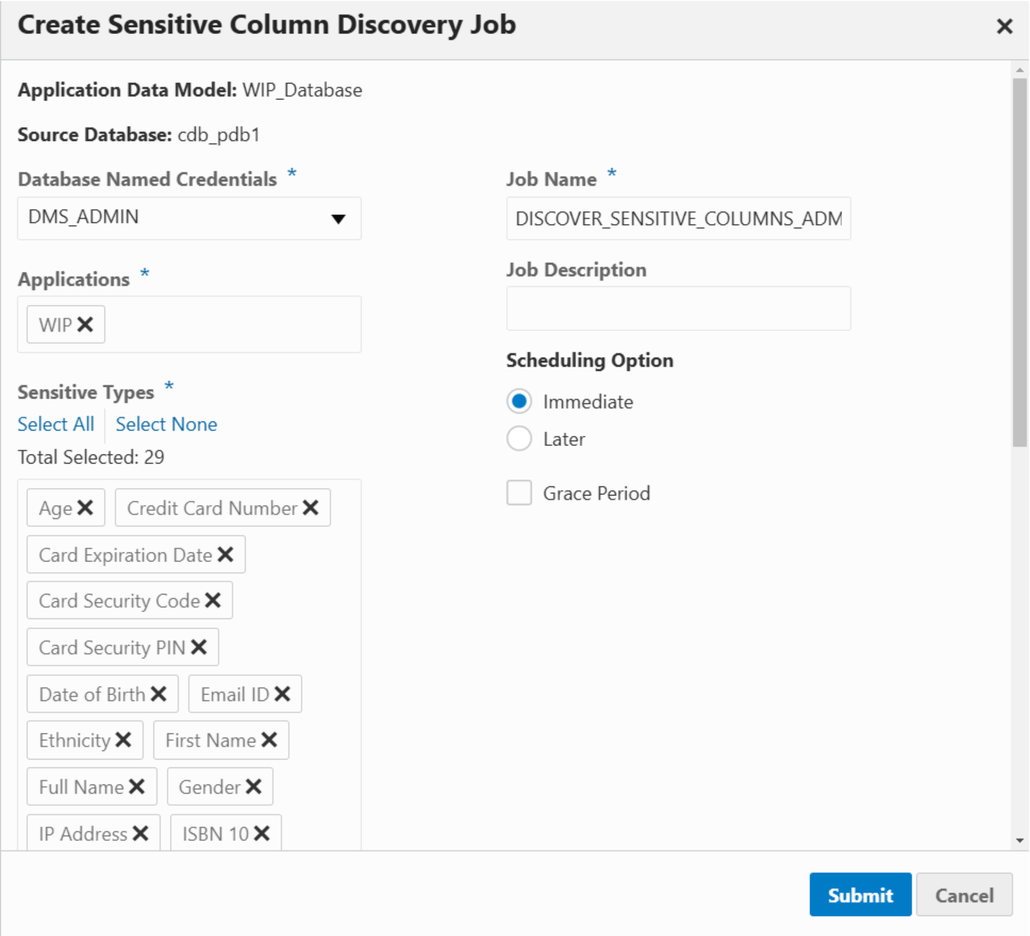
Sensitive Types	Application Data Models	Masking Definitions	Masking Formats	Subsetting Definitions
30	8	5	21	3

Target Name	Sensitive Schemas	Sensitive Objects	Sensitive Columns
EBS_POD	30	50	132
SBI_UPI_DB	29	49	125
SBI_UPI_DB_PTEST1	29	49	125
SBI_UPI_CUSTOMER	29	49	125
SBI_UPI_NON_PROD	29	49	125
EBS_DATABASE	30	50	123
FUSION_STAGE1	15	29	76
FUSION_STAGE2	15	29	76

Figure 4-6: Sensitive data discovery in Enterprise Manager

Discovering sensitive columns and referential relationships

Data discovery uses its sensitive types feature to perform pattern matching and identify sensitive columns. Schedule a sensitive data discovery job to scan the database for data that might need additional protection.



Create Sensitive Column Discovery Job

Application Data Model: WIP_Database

Source Database: cdb_pdb1

Database Named Credentials * DMS_ADMIN

Job Name * DISCOVER_SENSITIVE_COLUMNS_ADM

Applications * WIP

Job Description

Sensitive Types *

Total Selected: 29

Age, Credit Card Number, Card Expiration Date, Card Security Code, Card Security PIN, Date of Birth, Email ID, Ethnicity, First Name, Full Name, Gender, IP Address, ISBN 10

Scheduling Option

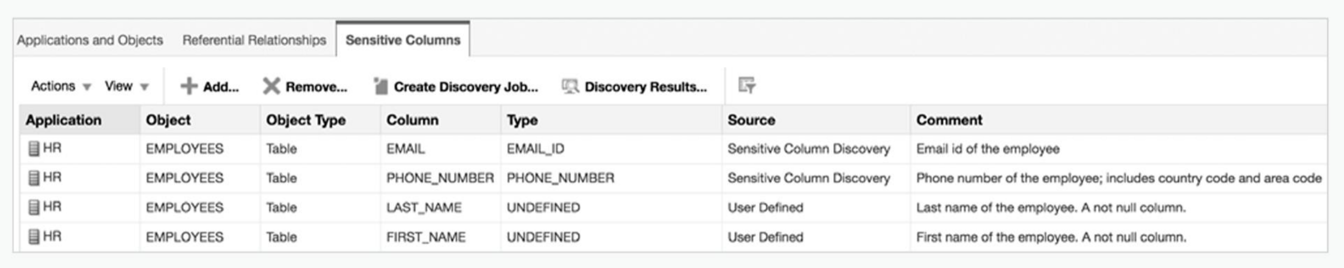
● Immediate

○ Later

□ Grace Period

Figure 4-7: Schedule a sensitive column discovery job

You can review the sensitive columns discovered and, if necessary, manually add additional columns to the list. Figure 4-8 shows a list of discovered as well as user-defined sensitive columns.

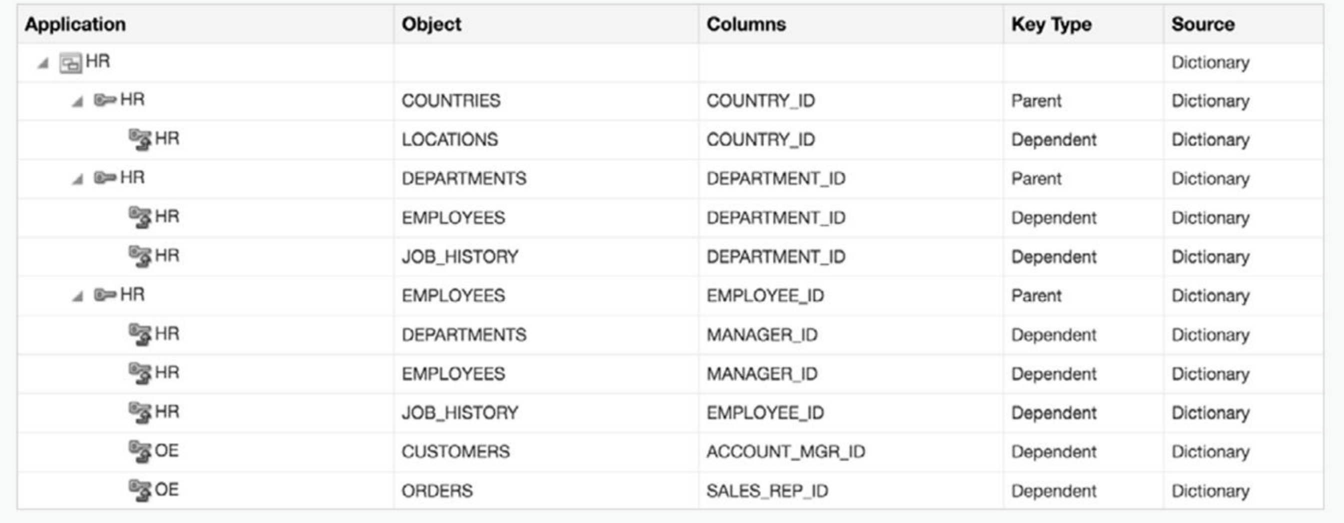


Application	Object	Object Type	Column	Type	Source	Comment
HR	EMPLOYEES	Table	EMAIL	EMAIL_ID	Sensitive Column Discovery	Email id of the employee
HR	EMPLOYEES	Table	PHONE_NUMBER	PHONE_NUMBER	Sensitive Column Discovery	Phone number of the employee; includes country code and area code
HR	EMPLOYEES	Table	LAST_NAME	UNDEFINED	User Defined	Last name of the employee. A not null column.
HR	EMPLOYEES	Table	FIRST_NAME	UNDEFINED	User Defined	First name of the employee. A not null column.

Figure 4-8: Sensitive columns discovered using Enterprise Manager ADM

Data discovery analyzes the referential relationships between application objects using foreign key constraints defined inside the database. It also allows automatically discovering application-level referential relationships not defined in the database. Users can review the discovered referential relationships and add additional relationships manually.

Understanding such dependencies helps preserve application integrity during data masking by ensuring that the data in the related columns is masked consistently. Figure 4-9 shows a list of parent-child relationships found inside the data dictionary.



Application	Object	Columns	Key Type	Source
HR				Dictionary
HR	COUNTRIES	COUNTRY_ID	Parent	Dictionary
HR	LOCATIONS	COUNTRY_ID	Dependent	Dictionary
HR	DEPARTMENTS	DEPARTMENT_ID	Parent	Dictionary
HR	EMPLOYEES	DEPARTMENT_ID	Dependent	Dictionary
HR	JOB_HISTORY	DEPARTMENT_ID	Dependent	Dictionary
HR	EMPLOYEES	EMPLOYEE_ID	Parent	Dictionary
HR	DEPARTMENTS	MANAGER_ID	Dependent	Dictionary
HR	EMPLOYEES	MANAGER_ID	Dependent	Dictionary
HR	JOB_HISTORY	EMPLOYEE_ID	Dependent	Dictionary
OE	CUSTOMERS	ACCOUNT_MGR_ID	Dependent	Dictionary
OE	ORDERS	SALES_REP_ID	Dependent	Dictionary

Figure 4-9: Referential relationships

Figure 4-10 puts everything together. Data discovery automatically discovers sensitive columns, database-defined referential relationships, and application-level referential relationships and creates an application data model, which it then stores in the Enterprise Manager repository. This application data model can be used to help implement security controls such as Oracle Data Masking and Subsetting or Oracle Database Vault.

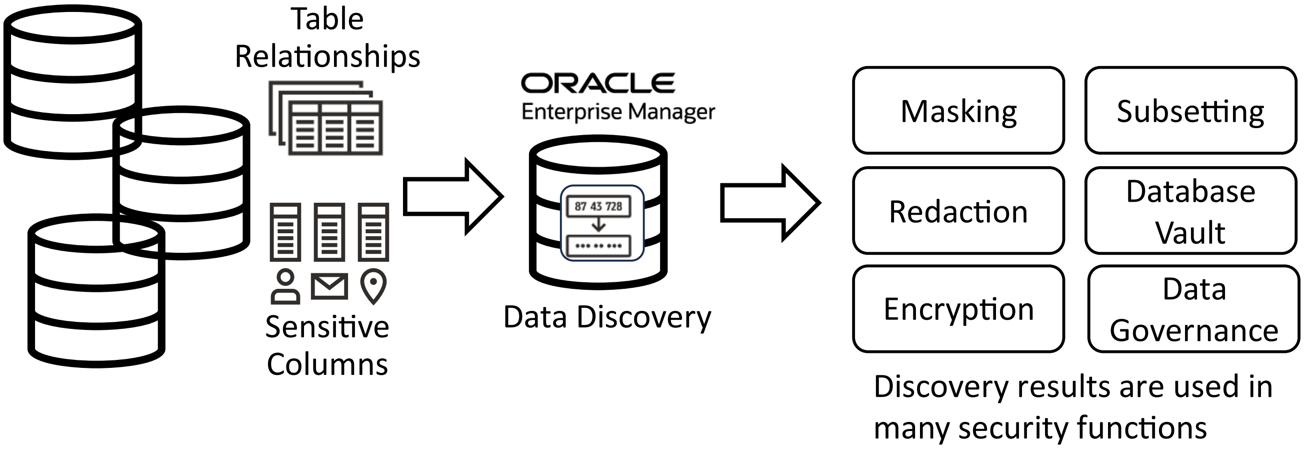


Figure 4-10: Results of data discovery are useful in configuring other security controls

Oracle LiveLabs

Oracle LiveLabs gives you access to Oracle's tools and technologies to run a wide variety of labs and workshops at your own pace. If you want to try out the technologies discussed in this chapter, please go to:

- Database Security Assessment Tool
- Get Started with Oracle Data Safe Fundamentals
- Audit Vault and Database Firewall
- Data Masking and Subsetting (including sensitive column discovery)

Summary

Understanding sensitive data is an important step in implementing appropriate security controls to protect data. Oracle provides multiple sensitive data discovery tools: Database Security Assessment Tool (DBSAT), Oracle Data Safe, Oracle Audit Vault and Database Firewall (AVDF), and Oracle Enterprise Manager data discovery.

- DBSAT is a lightweight, easy-to-use tool that helps you quickly analyze the sensitive data in a database. It automatically identifies sensitive columns, classifies them into risk categories, and provides detailed reports.
- Data Safe offers a comprehensive data discovery feature and is recommended for discovering sensitive data for Oracle Databases in the Oracle Cloud, in third-party clouds, and on-premises. The discovered sensitive objects can be leveraged to review access to sensitive objects in the activity auditing reports and to mask sensitive information in your non-production databases with Data Safe. Because Data Safe is a cloud service, you do not need to manage infrastructure and can use Data Safe APIs to automate complex use cases.
- AVDF integrates sensitive data discovery into activity and compliance reporting, alert conditions, and Database Firewall policies.
- Enterprise Manager's data discovery scans are similar to Oracle Data Safe and suitable for customers required to keep all security data locally.

Overall, these tools help discover and classify sensitive data, enabling you to implement security controls effectively, minimize your security risk, and address requirements associated with regulations like India's DPDPA, the European Union's GDPR, the Payment Card Industry Council's PCI-DSS, and the United States HIPAA.

Chapter Five

Authenticating Database Users

Why is authentication important?

The easiest (and unfortunately most common) way to hack into the database is to impersonate an authorized user of that database. The stronger your authentication is, the less likely this type of attack will succeed. Some of the common techniques attackers use to find credentials to connect to your database include:

- Apply social engineering to capture account credentials: With spear phishing attacks, hackers can target end users or database administrators in an organization (who are easy to find via social media channels such as LinkedIn) and steal their credentials. Generative AI (GenAI) makes finding those targeted users and creating targeted phishing attacks even easier!

  Find hardcoded database connection information: Applications frequently connect to a database using database usernames and passwords that are embedded in the program code or stored in a clear text configuration file. Because application servers tend to be closer to the network edge than database servers, they are frequently easier to hack into. Compromising application service accounts allows hackers to exfiltrate, modify, or delete any data that the account can access.
- Use default or published passwords: Hackers can try common default passwords to connect as users and use their privileges to access sensitive data.
- Run brute force password attacks: By cycling through combinations of characters and trying all possible variants, hackers can break into database accounts with weak passwords when there are no limits on password retries. Oracle Database makes this type of attack difficult by implementing exponential backoff on incorrect passwords. Exponential backoff means that there is an increasing wait time between failed login attempts, so trying millions of random character combinations is unlikely to succeed as a hacking strategy, but the possibility for an eventually successful attack still exists. Brute force attacks usually front load their attack by starting with commonly used passwords like "password," "oracle," or "qwerty123" to see if they will work before shifting to a true brute force approach. Without enforcement of complex passwords, the attackers may get lucky and find a way into your database despite the exponential backoff.
- Try passwords harvested from compromised sites: This is a common variation on a brute force attack often referred to as credential stuffing. People often reuse the same password across multiple applications or websites, and if one of those sites is compromised, attackers can collect username/password information from the site and then try the collected passwords to attack your database. Enormous databases of compromised username/password pairs are readily available for sale on the dark web, as are automated tools that will try tens of millions of credentials in an attempt to break into a system. If the attackers have done their homework and know the login name of the user they are trying to compromise, then a search of the stolen password database may reveal some likely candidates for them to try against your systems.

These attacks are not especially sophisticated and automated tools are available to simplify attacks using this vector. If successful, accessing a database via a compromised account gives the attacker as much access as the compromised user possesses.

This chapter discusses how your authentication strategy helps protect users of databases and the data within from attackers. It also explains how to manage user accounts, whether locally within the database or with centralized external services, such as a directory service or a cloud identity provider.

The next chapter will focus on user and application authorization to the database, controlling what the user or application can do within the database.

Why should you focus on authentication?

Controlling access to data is one of our fundamental security goals, both to satisfy regulatory requirements and to prevent theft or misuse of data. A core tenet of security is that users should only have access to the data and functionality they require to perform their job function. But, before you can begin to determine what data a user should be able to access, you need to know who that user is. This is where authentication comes into the picture. Authentication provides you assurance of the identity of who (or what) is connecting to your database. Identity assurance is the bedrock of access control – most access control relies on a user's identity.

Database authentication methods

Notice that the attacks listed above are all against password-based authentication. Passwords are by far the most commonly used authentication mechanism, but the frequency of successful attacks using compromised passwords is fueling a growing movement to shift away from passwords to stronger methods of authentication. Many organizations, including the United States federal government and the Oracle Corporation, are taking steps to remove passwords as an authentication method in favor of tokens, pass keys, and biometrics, preferably using multiple factors so that compromise of one factor does not result in a compromised account. By requiring two or more independent verification methods, multi-factor authentication (MFA) creates a more robust defense against cyber threats.

In place of passwords checked against the database's internal hashed credential store or through a centralized directory or identity services, Oracle Database users can be authenticated by external authentication services, including the OCI IAM and Microsoft Entra ID cloud identity providers, Kerberos, public key certificates, and RADIUS.

Once the user is authenticated, the user is mapped to a database schema consisting of tables, views, indexes, and procedures and then granted appropriate authorization through roles and privileges. In the early years of the Oracle Database, the concept of schema and user account were merged. Each user had their own schema. Even today, the command syntax to create a new schema in the database is `CREATE USER`. There is no `CREATE SCHEMA` command. Instead, a specialized form of the `CREATE USER` command creates a "schema-only" account – a user with no authentication credentials assigned. Schema and the user are still the same entity when using local database password authentication. When authenticating users with a directory or identity service, users are mapped to a schema. They may get their own private database schema (dedicated schema mapping) or be mapped to a schema that is shared by multiple users (shared schema mapping).

Although we won't talk about authorization until Chapter Six, this is a good time to mention that database roles (collections of privileges) can also be managed in the database or, in some cases, managed externally in a directory service, through OS groups, or within a RADIUS server. We'll talk more about roles later.

The following table lists the database authentication methods and associated mappings to schemas and roles.

Authentication Method	Schema Mapping	Role Mapping
Database Password	Schema is the same as user	Managed in the database
Operating System	Database maps the user to a schema	Managed in database or through membership in OS groups
Kerberos	Database maps the user to a schema	Managed in database
Public Key (PKI)	Database maps the user to a schema	Managed in database
RADIUS	Database maps the user to a schema	Managed in database or through RADIUS server
Oracle Directory Services	Managed in database and directory service	Managed in database or through membership in directory service groups

Microsoft Active Directory	Managed in database and directory service	Managed in database or through membership in Active Directory groups
Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM)	Managed in database and identity service	Managed in database or through membership in OCI IAM groups
Microsoft Entra ID	Managed in database and identity service	Managed in database or through membership in Entra ID groups

Table 5-1: Database user authentication and authorization methods

Making users resistant to attacks

As mentioned earlier, despite the inherent issues with passwords, they are still the most commonly used authentication token. If your database users are authenticating via password, it's important to ensure those passwords are well-managed and as secure as you can make them. Below are a few things to look out for if you are using passwords to authenticate.

Don't store plain-text passwords

With password-based authentication, users provide a password when they connect to the database, but applications, middle-tier systems, and batch jobs cannot depend on a human to type in the password. In the past, a common but insecure way to provide passwords was to embed usernames and passwords in code or scripts. However, this increased the attack surface of the database, and people had to ensure the scripts were not exposed. Also, changes to the scripts were required if passwords were ever changed.

Secure External Password Store (SEPS) attempted to solve this problem by storing various authentication credentials, including passwords, private keys, and certificates in an encrypted file known as an Oracle Wallet. With SEPS, the users need to remember only the wallet password, which then unlocks all remaining user credentials - potentially for multiple databases. Applications and database servers could also use the auto-login form of the wallet for 24/7 access to credentials. With wallets, the database passwords were no longer exposed on command-line history or in clear text configuration files. Someone with access to the operating system could not easily discover them. Unfortunately, the SEPS wallet still resided on the application server or client workstation and was only protected by the strength of the wallet passphrase. An attacker who stole the wallet files might be able to use brute force tactics to eventually open the wallet and read the credentials, but they would be more likely to simply monitor activity on the compromised server and capture the wallet password when it is used.

Over the last few years, secrets management – a more modern and secure way to handle password storage and distribution – emerged as a more secure way to manage the process of securely storing and distributing passwords to applications. Using a secrets manager like Oracle Key Vault, users and applications may retrieve credentials from the Key Vault using an API call. This way, the password is centrally managed and never stored on the application server or user workstation.

Secret managers like Key Vault eliminate the overhead of managing, updating, and protecting Secure External Password Stores (SEPS). Key Vault allows you to centralize your credentials instead of having local SEPS wallets spread across all your application servers. Key Vault, acting as a secrets manager, also simplifies password rotation and doesn't require a permanent password footprint on every application server. For those applications that still use password-based authentication, using a secrets manager is the recommended way to manage the storage and distribution of credentials. We'll talk more about Key Vault when we discuss encryption and key management in Chapter Eleven.

Don’t share accounts and passwords

Having multiple people share the same account is a security nightmare. If something goes wrong – mistakes, sabotage, data theft – how can you tell which of the many people who are using the shared account are responsible? Most organizations have policies forbidding the use of shared accounts, but we still find these policies are commonly violated, often by some of the highest-privileged database users - developers and database administrators!

Database administrators frequently connect to the database using a shared account, most often the database super-user (SYS) or the default SYSTEM database administrator account. These are shared accounts that should not be used for everyday DBA activities! Instead, DBAs should each have their own personal account with the roles or privileges necessary for them to do their job function – that could be the out-of-box DBA role, or it could be a role more narrowly targeted at the functions the DBA does.

Application administrators and developers also should not use shared accounts. They often need to connect to an application schema for maintenance, but connecting to the application schema does not mean they need to log in using the application service account’s credentials. A bit later in this book we’ll talk about ways to block the misuse of service accounts using Database Vault, but for now, let’s focus on how we can enable the administrators and developers to do their job using their own private accounts without imposing too large of a burden on them.

Proxy authentication solves the problem of allowing an administrator to act as the application account without the administrator knowing the application account’s password. With proxy authentication, a database user can connect through their own account, but act as if they were a different user. Audit records still reveal the actual end user, and access control policies can consider both the true end user and the proxied account. Proxy authentication helps hold individual administrators and developers accountable. When authorized for proxy authentication to the application account, administrators first authenticate to the database with their credentials and then proxy to the application schema without needing to know the password for the application schema account. For example, in Figure 5-1 we grant `alice_appdba` `CONNECT THROUGH` privilege to connect as `hrapp.`

```
SQL> ALTER USER alice_appdba
GRANT CONNECT THROUGH hrapp;
```

Figure 5-1: Granting proxy privileges

`alice_appdba` then connects using their password and assumes the identity and privileges of the `hrapp` schema by proxy as shown in Figure 5-2:

```
SQL> CONNECT alice_appdba[hrapp]
Enter password: <alice_appdba_password>
```

Figure 5-2: Connecting using proxy privileges

The audit records now show `hrapp` as the `DBUSERNAME` while the `alice_appdba` user is recorded as the `DBPROXY_USERNAME`. The proxy username is also available for policy-driven access control mechanisms like Database Vault, Label Security, Data Redaction, and Real Application Security (discussed later in this book).

Increase the strength of passwords

There are many occasions when you have no choice but to use passwords for authentication. Older applications and clients frequently don’t support any other type of authentication. In that case, you want to make those passwords as strong as your users and applications can support, and exercise good password discipline (things like not reusing passwords, not sharing accounts, and controlling the password lifecycle). Like most systems that allow password authentication, Oracle Database lets you control the minimum length, complexity, and lifecycle of passwords. This control is exercised through the use of user profiles. Every database account is assigned to a user profile. Theoretically, each user in the database could have their own unique profile, but that would be very unusual. Usually, there are just a few profiles, with users grouped under different profiles according to their needs. User profiles control password and resource authorization parameters for all accounts assigned to them. The default profile is automatically assigned if no profile is specified when a user is created. You can create custom profiles as needed.

Note: Oracle Database 21c and earlier support passwords up to 30 bytes in length. Beginning with Oracle Database 23ai, the maximum length of a password is increased to 1024 bytes.

In most cases, you will modify the default profile to meet your organization's requirements and assign accounts to other profiles when an exception to your organizational standard is required. Profiles can consist of up to 18 different parameters. Nine of these are related to resource limits, and the other nine are related to authentication. If you create a profile without specifying some of the 18 parameters, the new profile will use the value in the default profile for the undeclared parameters.

An example of a custom profile is shown in Figure 5-3 below, where `app_profile` incorporates both authentication and resource allocation parameters. This custom profile only specifies six of the possible 18 parameters – the other 12 parameters will use the value specified in the default profile.

```
SQL> CREATE PROFILE app_profile LIMIT
PASSWORD_VERIFY_FUNCTION ora12c_stig_verify_function   -- STIG password complexity rules
PASSWORD_ROLLOVER_TIME 30                 -- Allows both old and new passwords for 30 days
FAILED_LOGIN_ATTEMPTS 6                   -- Lock the account after 6 attempts
SESSIONS_PER_USER 2                       -- Allow two sessions for each user
IDLE_TIME 30                              -- Automatic logout after 30 min idle
PASSWORD_LIFE_TIME 180;                   -- Force password change after 180 days
```

Figure 5-3: Sample password profile

While each of the 18 profile parameters can be important in different situations, there are three profile parameters I'd like to highlight:

`PASSWORD_VERIFY_FUNCTION` - Profiles enforce password complexity by a password verification function like `ora12c_stig_verify_function` shown in Figure 5-3. `ora12c_stig_verify_function` is one of several out-of-box password complexity functions included with the database. This function imposes password requirements that correspond with the United States Department of Defense Security Technical Implementation Guide (STIG) recommendations, including a minimum length, minimum number of alphanumeric characters, and at least one special character. If none of the pre-created password verification functions meet your needs, you can create your own password verification functions to enforce your organization's unique requirements. The source code for the Oracle-supplied password verification functions is available in `$ORACLE_HOME/rdbms/admin/catpvf.sql`, and using one of the functions in that script as a starting point it is relatively easy to create a custom function.

`PASSWORD_ROLLOVER_TIME` - One common issue with the database accounts used by applications is the need to update their passwords periodically to comply with corporate policies or follow security best practices guidelines. Changing an application account password requires the password to be changed in the database and with the application (hopefully stored in a client wallet or OKV). Organizations used to have to schedule downtime for their applications to eliminate the chance that the application might fail when it tried to log in during password rotation. Some organizations would even forego password updates to minimize application downtime.

Starting with Oracle Database 19c, you can assign a new password to an application while allowing application clients to use the old password for a limited period defined by PASSWORD_ROLLOVER_TIME. During that period, both the old and new passwords will work. With gradual password rollover, the new password can be updated with all application clients or mid-tiers without having to coordinate downtime or risk an adverse application event. This parameter should be set in profiles used by application service accounts, but should not be set in profiles used by human actors. A value of zero disables the password rollover feature.

`FAILED_LOGIN_ATTEMPTS` - the number of consecutive failed attempts to log in to the user account before the account is locked. This parameter is important to block brute-force password-cracking attacks on accounts.

Simplify account administration

Most databases have few human-actor accounts – typically just the database administrators, perhaps a developer or two, and power users that connect for reporting purposes. But there are many outliers that don't follow that convention – we've seen databases with tens of thousands of valid human-actor accounts. Even if there are only a few user accounts in the database, it isn't uncommon for there to be MANY databases with those accounts. Customers with thousands of Oracle Databases are not unusual.

In both cases, administering user identities within the database can be challenging. People come and go from organizations. People change roles, in some cases moving away from roles that require a database account or to roles that require different or fewer privileges. Managing that natural user lifecycle is challenging when you have a lot of users, or when a user belongs to many different databases. What we see in practice is that accounts that are no longer relevant often live on for a very long time after their legitimate owner has left the organization or changed roles, making the accounts an attractive target for hackers to exploit and gain unauthorized access to data.

This is why Oracle Database supports centralized user management, with users and their credentials managed outside of the database in some other system – most frequently a directory such as Microsoft Active Directory, or a cloud identity service like Oracle Cloud Infrastructure's Identity and Access Management (IAM), Okta, or Microsoft Entra ID.

A central user management service enforces consistent policies across different users and databases and makes it easy to handle people joining, leaving, or changing roles within an organization. Many centralized services, especially cloud identity providers, can also enforce multi-factor authentication. And don't despair – even if you've moved beyond password authentication you can still take advantage of centralized user management. For example, when integrating the database with Microsoft Active Directory, it's more common to use Kerberos as the authentication mechanism. PKI certificates (often via the use of smart cards, like the US Department of Defense Common Access Card (CAC)) are also popular.

We'll talk more about centralized user management in Chapter Six.

As more applications, tools, and databases are moved to the cloud, multicloud identity integration becomes critical. Users can be managed in a single identity service even though their applications, databases, and tools are in different clouds or on-premises. We'll talk more about multicloud in Chapter Fourteen.

Protecting your users from getting hacked

Protecting business users

You should not expect business users to remember and voluntarily follow all the security guidelines regarding password strength, password management hygiene, and the roles/privileges granted to them. These security guidelines need to be enforced by the database to make sure users follow them. Policies such as the following help protect your users from being hacked:

- Avoid password authentication. Implement strong authentication mechanisms with cloud identity service-provided tokens, Kerberos, RADIUS, or PKI certificates.

- If you must use passwords, impose a strong password profile that controls password strength, the inactivity period, and the number of password retries (failed logins).
- Manage users centrally with an identity service like Active Directory to reduce the probability of orphan accounts when someone leaves the organization or changes their role.

Protecting DBA accounts

Because database administrator accounts typically have broad access within the database they are very attractive targets for hackers trying to compromise an account. DBA authentication should be controlled using the following best practices:

- Database administrators should use named accounts with strong authentication to provide accountability. Use of shared accounts or default accounts like `SYSTEM` should be prohibited. Where practical, use strong authentication mechanisms such as cloud identity service tokens, PKI certificates, or Kerberos.
- Use a Privileged Account Management (PAM) system when operating system accounts for root and database owner are needed for `SYSDBA` access (typically only during upgrade, patching, and database restarts).
- Use sudo if a DBA needs to access the operating system account for the database so there is a record of the user (in other words, the operating system account is also specific to the user).
- If DBA are accessing the database by logging into the database server's operating system and then connecting to the database:
  - Ensure they connect to the server using a named account that is private to them, not using the account that owns the database binaries.
  - Install some other client, like the Oracle Instant Client or SQLcl, for them to use instead of allowing them access to the database owner account or binaries.
- If your organization divides the DBA function into specialized teams like backup and recovery DBAs, performance tuning DBAs, and security managers, then don't grant the default DBA role, but instead create specialized administrator roles for each function with only the privileges required for each specialized team (more about this in Chapter Six and Chapter Seven).

Protecting application accounts

Application service accounts in the database not only have all the roles and privileges required to run the application on behalf of every application user, but for the sake of convenience, they may also have roles and privileges to perform application installation, upgrades, patching, and other maintenance activities. Such accounts need to be protected using the following best practices:

- Use strong authentication mechanisms (cloud identity service tokens or PKI certificates) to authenticate to the database. Use controls like SQL Firewall, Database Vault connect rules, or logon triggers to restrict where connections by these service accounts can originate from. We will talk more about SQL Firewall and Database Vault in Chapter Seven (for Database Vault) and Chapter Eight (for SQL Firewall).
- Use a secrets manager like Oracle Key Vault to store credential information and have the application retrieve the credentials via API calls. Do not hard-code credentials into the application code or store them in clear-text properties files on the application server. Application servers are more likely to be hacked than database servers because they are usually placed within network zones with an increased threat surface.
- Use gradual password rollover to reduce the risk of application outages when rotating the application's database password. This feature allows you to have both old and new passwords until the application account password has been changed in all the application servers.

- Have application administrators use proxy authentication instead of connecting using the application service account.
- Create a schema-only account for the application objects and procedures to prohibit direct login access for the application schema. Then, use a separate run-time application service account to access the application objects through controlled procedures and views. The schema-only account feature was introduced in Oracle Database 18c, where accounts have a named schema but no password or other ability to log in. Starting with Oracle Database 19c, almost all Oracle accounts installed with the database are schema-only accounts to prevent login to these privileged accounts. In earlier versions, these accounts would have passwords that would need to be periodically rotated. Other users can still be granted read/write access to these schemas.

Protecting security administrator accounts

Security administrators manage security controls, encryption keys, database users, and audit records. They protect the security of the database along with ensuring that there are user constraints on tablespaces, idle time, and number of concurrent sessions to prevent impact on other users. Such accounts need to follow these best practices:

- Use strong authentication mechanisms (PKI, Kerberos, tokens) to authenticate to the database.
- Enforce proper separation of duty between database administrators/users and security administrators to prevent DBA/user accounts from being able to alter security records, create fake users, and change security controls.
- If you must use passwords for authentication, assign the security users to profiles that ensure they use strong passwords and automatically lock if too many failed login attempts are made.

Oracle LiveLabs

Oracle LiveLabs gives you access to Oracle's tools and technologies to run a wide variety of labs and workshops at your own pace. If you want to try out the technologies discussed in this chapter, please go to:

- DB Security – Privilege Analysis
- Proxy authentication
- Centralize database user management with OCI IAM5

Summary

Strong authentication and effective privilege management sets the foundation for a secure database.

Because most attacks target authorized database users, properly managing those users is the first step in protecting the database. Oracle Database provides many options for user authentication, including external or local management of users.

Oracle Data Safe includes a User Assessment feature to analyze cloud and on-premises database user risk. User Assessment identifies users with DBA privileges in each database to quickly identify user accounts that can pose a risk if account credentials are compromised. Oracle Audit Vault and Database Firewall also helps with tracking user risk, allowing you to report on changes to user entitlements. See Chapter Three for more details on Data Safe and Audit Vault and Database Firewall.

Chapter Three also discussed using the Database Security Assessment Tool (DBSAT) to scan databases and gather information about user account configuration and privilege grants. It also covered how to easily find the right set of privileges and roles using Privilege Analysis.

Chapter six

Controlling Database Access

Why is access control important?

At the risk of sounding like a broken record, and as already stated in Chapter Five, the easiest way to hack a database is to impersonate an authorized user of that database, log on, and steal, alter, or destroy whatever data you are interested in. That is why controlling what users can do is so vital to safeguarding your data. Every unnecessary privilege available to a user increases the risk to your data and database if that account is compromised or if the user behind that account acts in ways that violate organizational policy.

Why should you focus on access control?

Access controls regulate who can act upon which data, and what types of actions they can perform within the database. You use access controls to limit accounts to only the capabilities needed for their intended job or function because overprivileged users are an unacceptable risk to your data and your database. Access control helps you enforce regulatory requirements and prevent the theft, destruction, or misuse of data.

Because access control is so important, Oracle Database gives you many ways to control access. In some cases, the differences between the access control features will be obvious. In others, the differences are subtle and you'll want to refer to this and the next few chapters to identify the right feature or tool to help you accomplish your objective.

We'll start with the basics, privileges and roles, and move into more advanced topics in later chapters.

This chapter explains how a robust authorization strategy protects the database from mistakes, misuse, and attackers. It also covers managing user account authorizations locally and through directory or identity services.

The following four chapters will explore more access control features and explain how they work together to minimize an account's ability to operate beyond the constraints required for its job function.

Database Privileges

A database privilege grants the ability to perform specific operations on data objects or execute certain statements. Oracle Database has four types of privileges: object, schema, system, and administrative. The first three types provide a gradually increasing scope for the privilege.

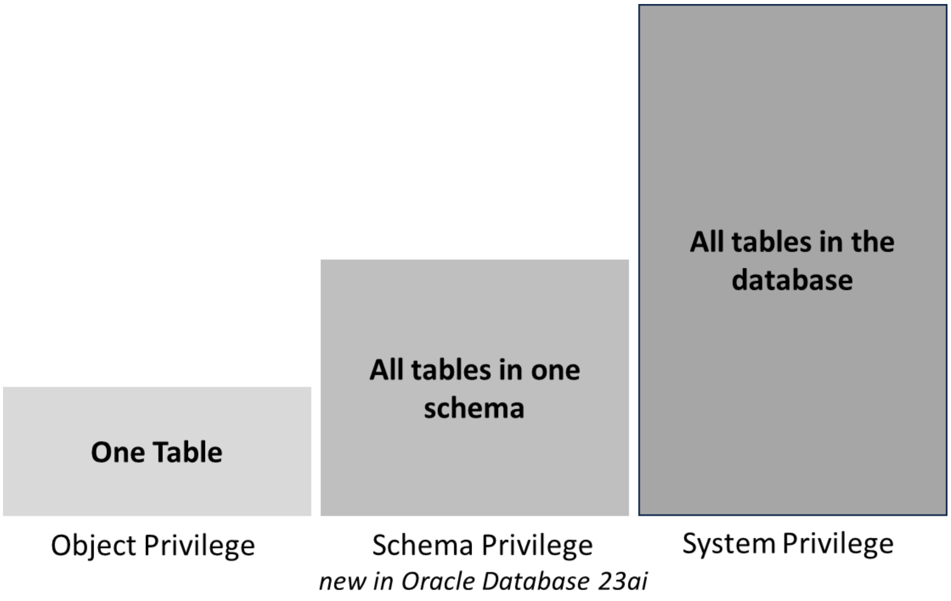


Figure 6-1: Scope of privileges

Object privileges are fine-grained privileges used to perform actions on database objects or execute stored procedures. Examples include privileges to view data from a table (`SELECT` or `READ`), execute a PL/SQL statement

(`EXECUTE`), and alter table structure (`ALTER`). The command to grant an object privilege giving the user Scott a privilege to select data in the `customers` table, which is part of the app schema, would look like this:

```
GRANT SELECT ON app.customers TO scott;
```

Figure 6-2: Granting an object privilege

Schema privileges (new in Oracle Database 23ai) are used by applications and users that need to manage objects in another schema. An application run-time or application administrator may be granted schema privileges to another schema where the data is stored. Granting schema privileges is preferred to granting system privileges because the privileges are constrained to a schema. Schema privileges are more secure than system privileges because their scope is significantly less. Schema privileges are easier to maintain than individual object privileges because you don't need to revisit privilege grants each time new objects are added to a schema. This next database command gives the user Scott the ability to select data from all tables or views in the schema owned by app.

```
GRANT SELECT ANY TABLE ON SCHEMA app TO scott;
```

Figure 6-3: Granting a schema privilege

System privileges allow broader access to database objects, usually to ALL objects relevant to the privilege. System privileges like `SELECT ANY TABLE` allow the user to read data from almost any table in the database, including sensitive data stored in application schemas. `CREATE USER` is another example of a system privilege allowing new users to be created in the database. In most cases, non-administrators should not be granted system privileges. The command to grant the powerful `SELECT ANY TABLE` privilege to the user Scott would look like this:

```
GRANT SELECT ANY TABLE TO scott;
```

Figure 6-4: Granting a system privilege

Administrative privileges are different than object/schema/system privileges. Administrative privileges are focused on database administration tasks, not database objects. You'll find administrative privileges used for tasks such as database backup, encryption key management, and database operations (e.g., startup, shutdown). Examples of administrative privileges include SYSDBA, SYSOPER, SYSKM, and SYSDG. The command to grant an administrative privilege looks very similar to granting an object privilege, as shown in this next example that gives Scott the `SYSKM` privilege – giving them the ability to manage encryption keys for the entire database:

```
GRANT SYSKM TO scott;
```

Figure 6-5: Granting an administrative privilege

Certain maintenance activities like database upgrades and patching can only be done by the `SYS` account (Database owner). The administrative privilege `SYSDBA` allows a user to become `SYS` for these tasks. The `SYS` account and the related `SYSDBA` administrative privilege should only be used when necessary and be safeguarded. Use of administrative privileges should be fully audited.

Administrative privileges are each associated with an operating system group (e.g., OSBACKUP or OSKM). Depending on your organizational structure, you may assign all these administrative privileges to the same group (e.g., DBA) or assign each privilege to its own unique OS group.

Administrative privileges are extremely powerful. Even when they are granted to a user they are not active within a session by default. The user granted those privileges must explicitly activate them when starting their session with the database.

```
CONNECT scott@pdb1
-- Scott’s SYSKM privileges are not activated and cannot be used during this session
CONNECT scott@pdb1 AS SYSKM
-- Scott’s SYSKM privilege is active in this session and Scott can perform encryption key
management operations
```

Figure 6-6: Granting an administrative privilege

Database Roles

Roles are named groups of privileges. We use roles to simplify privilege management, making it easier to grant or revoke identical privileges to multiple users. Object, schema, and system privileges can be grouped into task-based roles. Roles can also be hierarchically grouped under other roles. Granting roles to other roles allows privileges to be grouped into a task role and multiple task roles to be grouped for an organizational role.

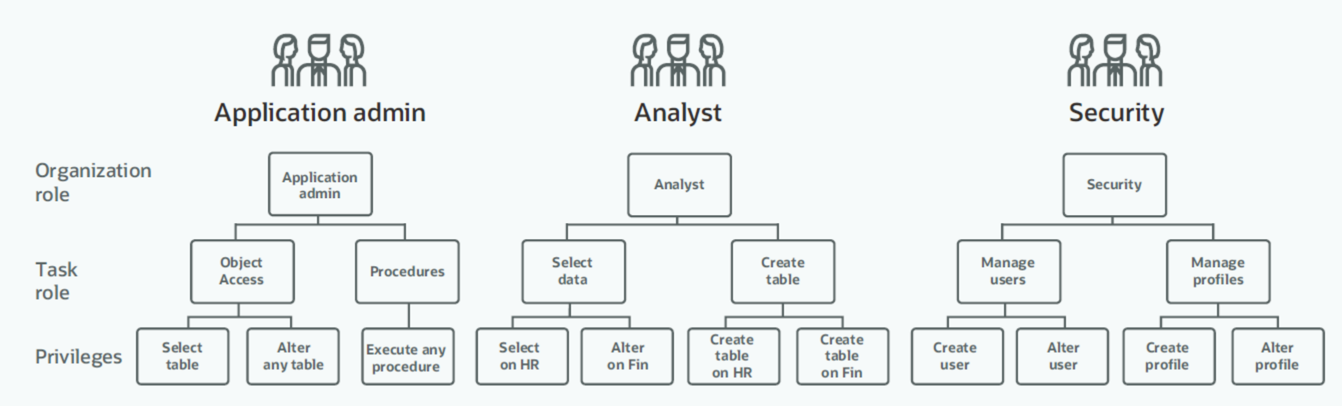


Figure 6-7: Sample role and privilege hierarchy

A single user can be granted multiple database roles. Using privileges and roles in this fashion makes adding a new person to an organization easy. They would be granted the role for the position instead of having to discover all the needed privileges and grant them one by one. As organizations change and tasks move from one group to another, the task roles can be moved instead of managing individual privileges or redefining all the organizational-level roles. Roles simplify the management of privileges when there is an organizational change.

Types of database users

Users are granted privileges directly or through a role. In general, a user can connect to their schema and access, modify, or delete all objects in their schema. Through privileges and roles, users can get permission to perform a specific operation, such as updating an object in some other schema, or certain database-specific rights, such as the ability to export or import data. And, as mentioned in Chapter Five, these users may be human actors (individual users), applications, tools, automated processes, or other programs acting on behalf of users.

A fundamental security precept is that users should only have the privileges necessary for their jobs. What privileges are necessary?

That depends on the account’s function. When discussing access controls, it’s helpful to divide accounts into different categories based on how they are used and what they are going to do within the database. These categories include:

- Database administrators (DBAs): Every database has one or more administrators responsible for a wide variety of tasks within the database, including performance management, diagnostics and tuning, upgrade and patching, database startup and shutdown, and database backup. Unless otherwise constrained, their highly privileged access lets them see (and modify) all data within the database, including sensitive data like personal, health, corporate finance records, etc., even though that access is not required to perform DBA tasks. Because of their privileged access, DBAs are a hackers' prime target. In most cases, DBAs will have the VERY highly privileged DBA role.

- Security administrators: Many organizations have specialized DBAs who focus on security-related tasks, such as user account management, access control policy development and implementation, encryption key management, and audit management. These accounts generally need a lower level of privileges within the database than a general-purpose DBA account because security administrators are performing a narrower range of functions within the database. For example, a security administrator has no need to create schema objects or access application data. The use of specialized DBA accounts is common, especially in large organizations with lots of people and lots of databases. There might be a group of DBAs responsible for backup and recovery, another for performance tuning, and yet another for patching and upgrades. Each of these specialized administrators does a subset of the overall DBA function and only needs a subset of the DBA privileges, not the full DBA role.
- Application service accounts: We refer to them as "Service accounts" (some organizations use the term "robotic" or "machine" accounts) because they connect to the database without a human directly initiating the connection. Most databases have applications, business intelligence tools, or reporting systems connected via one or more of these service accounts. A compromise of these database accounts can lead to loss of data for the entire application. Because applications are usually configured for high availability, their passwords are often stored on multiple middle-tier servers, which can increase the risk of the account being compromised (see the discussion of secrets management in Chapter Five for a better way to manage service account credentials). Since you are not counting on a human to remember and type in the password, service account passwords should be as long and complex as the database can support.
- Application administrators: These accounts manage, patch, and upgrade your application and usually have full access to all the data and the stored procedures used for the application. Application administrators have privileges similar to DBAs when it comes to their application's data but don't need the extensive set of database management privileges used by a DBA.
- Developers and testers: These accounts are used to create and modify the database objects required for an application. Traditionally, developers did not have routine access to production databases. They made changes in lower-level environments and promoted those changes through well-defined change management processes. In DevOps (or SecDevOps) environments, those traditional lines are often blurred, and it is now common to find developers with production database access. Even if the developers do not have access to production, the systems they work on are frequently cloned from production and contain the same data (Chapter Ten will discuss how to remove risk from test and development copies of a database). Developer accounts normally have privileges equivalent to the application service account and may utilize proxy authentication (discussed in Chapter Five) to access the application schema. Oracle Database 23ai introduces a new `DB_DEVELOPER_ROLE` that contains the privileges a developer will need to do their job function. Like DBA accounts, developers are usually easy to identify from their social media presence and are favored targets for hackers attempting to break into a system.
- Data analysts or business intelligence users: Most databases allow access by one or more data analysts, either directly or through a business intelligence tool. These accounts typically need read-only access to the application schema. They pose a slightly higher risk than users connecting through the application because they do not go through the application-level access controls.
- Other database users: This is everyone else with a database account. The users who do not fall into the categories listed above. They may be accounts used by batch processes, integration services like Oracle Golden Gate, or tools like Data Safe, Enterprise Manager, or Audit Vault and Database Firewall. Their access needs can vary widely, and many tools or programs will include a script that grants appropriate access to the accounts. Keep in mind that "appropriate" is a loaded term – your vendor may think it's appropriate for their

tool to have the DBA role, but chances are that you'll find that is not really the case when you use privilege analysis to determine what privileges are actually used. These accounts may be restricted to their own schema or may need broader access to multiple schemas.

Who can do what in your database?

One of the most critical components of an Oracle Database is its data dictionary, a set of tables and views that provide information about the database, including definitions (metadata) about all objects and users. The dictionary views listed below in Table 6-1 allow you to investigate the roles and privileges granted to users or roles.

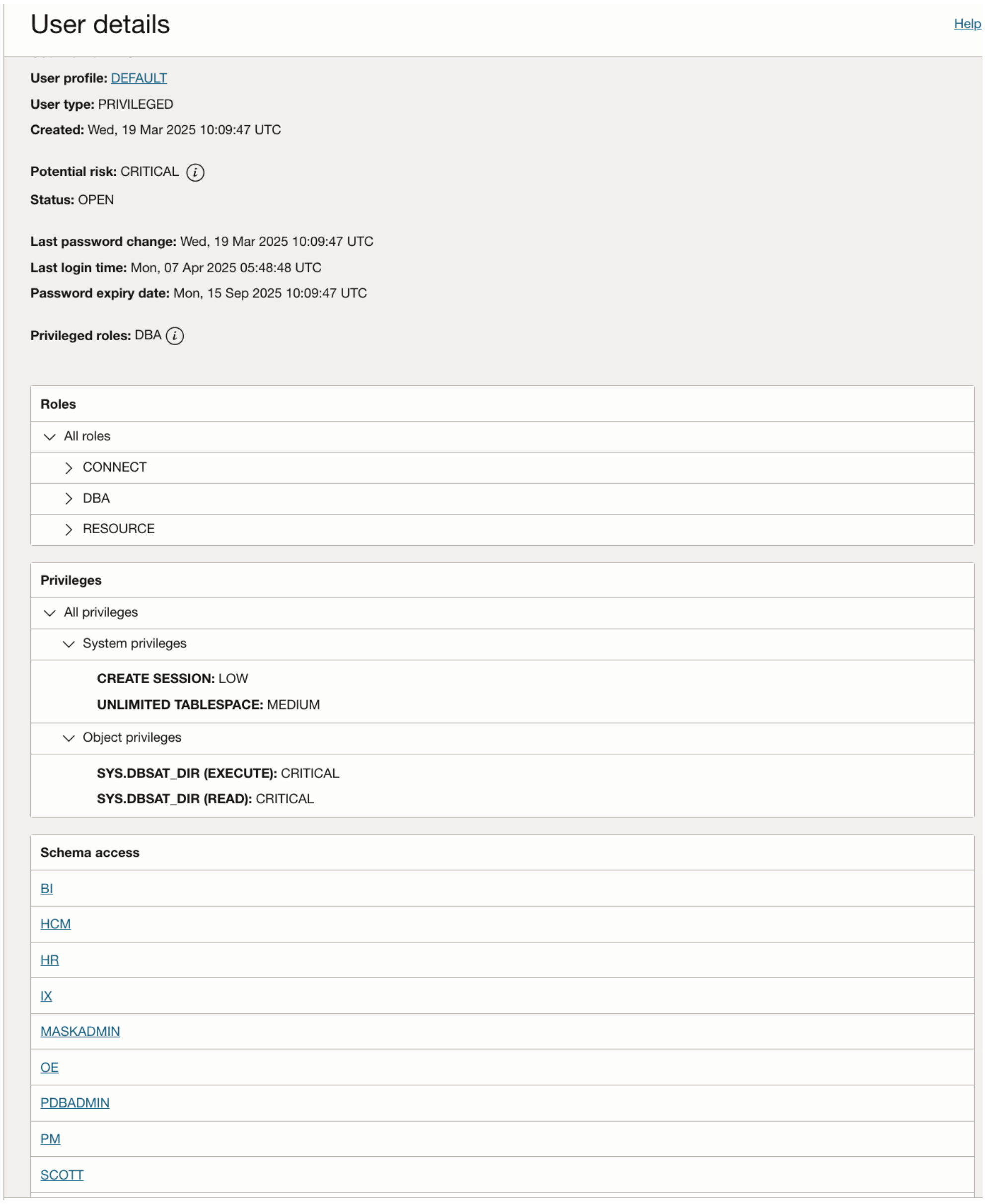
Dictionary views	Contents
`DBA_TAB_PRIVS`	Object privilege grants to roles or users
`DBA_SCHEMA_PRIVS`	Schema privilege grants to roles or users
`DBA_SYS_PRIVS`	System privilege grants to roles or users
`DBA_ROLE_PRIVS`	Roles granted to all users and roles
`DBA_ROLES`	All defined roles
`ROLE_ROLE_PRIVS`	Roles granted to other roles
`ROLE_SYS_PRIVS`	System privileges granted to roles
`ROLE_TAB_PRIVS`	Object privileges granted to roles
`V$PWFILE_USERS`	Administrative privileges granted to users

Table 6-1: Data dictionary views for database roles and privileges

If you were to analyze the information in these dictionary views, you would see that the out-of-the-box `DBA` role is extremely powerful, with more than two hundred system privileges, including `ALTER SESSION`, `CREATE, ALTER`, and `DROP USER`, `CREATE` and `ALTER ANY TABLE`, `SELECT`, `INSERT`, `UPDATE`, and `DELETE ANY TABLE`, the `EXPORT` and `IMPORT FULL DATABASE` roles, and over a dozen additional roles, each with its own set of privileges.

If your organization uses specialized DBAs, where the job function requires only a subset of the full DBA role, you should create one or more custom DBA roles (i.e., `ops_dba`, `backup_dba`) with the roles and privileges required for that job function. Privilege Analysis (discussed in Chapter Three) can help you develop those custom DBA roles.

When you examine a database user to identify their complete set of privileges, you must review grants of roles, system privileges, and object privileges to get the complete picture. Below is an example taken from Oracle Data Safe's user assessment module.



User details

User profile: DEFAULT

User type: PRIVILEGED

Created: Wed, 19 Mar 2025 10:09:47 UTC

Potential risk: CRITICAL

Status: OPEN

Last password change: Wed, 19 Mar 2025 10:09:47 UTC

Last login time: Mon, 07 Apr 2025 05:48:48 UTC

Password expiry date: Mon, 15 Sep 2025 10:09:47 UTC

Privileged roles: DBA

Roles

- All roles
  - CONNECT
  - DBA
  - RESOURCE

Privileges

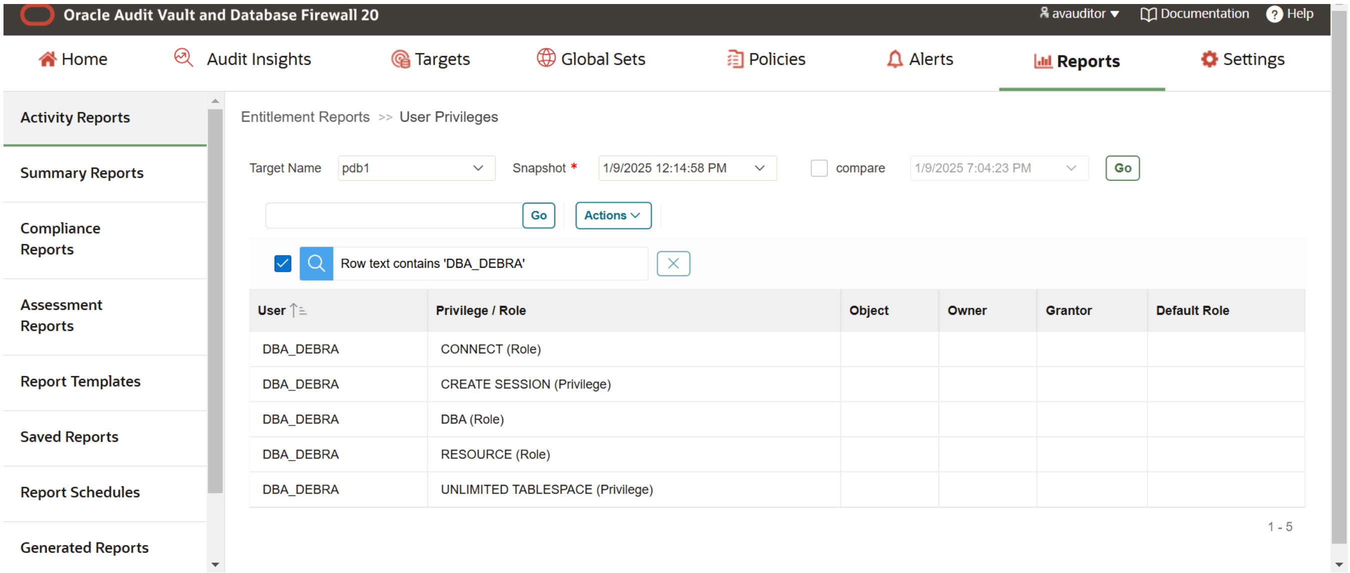
- All privileges
  - System privileges
    - **CREATE SESSION:** LOW
    - **UNLIMITED TABLESPACE:** MEDIUM
  - Object privileges
    - **SYS.DBSAT_DIR (EXECUTE):** CRITICAL
    - **SYS.DBSAT_DIR (READ):** CRITICAL

Schema access

- BI
- HCM
- HR
- IX
- MASKADMIN
- OE
- PDBADMIN
- PM
- SCOTT

Figure 6-8: Analyzing a user's privileges with Data Safe

Viewing privileges through Audit Vault and Database Firewall gives similar information:



Entitlement Reports >> User Privileges

Target Name: pdb1 Snapshot *: 1/9/2025 12:14:58 PM □ compare 1/9/2025 7:04:23 PM

☑ Row text contains 'DBA_DEBRA'

User	Privilege / Role	Object	Owner	Grantor	Default Role
DBA_DEBRA	CONNECT (Role)				
DBA_DEBRA	CREATE SESSION (Privilege)				
DBA_DEBRA	DBA (Role)				
DBA_DEBRA	RESOURCE (Role)				
DBA_DEBRA	UNLIMITED TABLESPACE (Privilege)				

1 - 5

Figure 6-9: Analyzing a user's privileges with AVDF

Managing users and their privileges centrally

Chapter Five discussed the benefits of managing database users in a central directory or identity service. Database authorizations can also be controlled centrally, increasing security while reducing the DBA workload.

In an enterprise with many users accessing several databases, users have difficulty keeping passwords compliant with each database's policy and remembering different passwords for their various accounts. Further, it is difficult for administrators to manage accounts for each user in every database as each user needs to be provisioned separately along with their password. More importantly, each account must be deprovisioned when the user changes roles or leaves the organization. Accounts that are not de-provisioned after a user no longer needs it are known as orphaned accounts. Hackers often exploit orphaned accounts to gain unauthorized access to data.

Enterprise User Security (EUS) centrally manages users and roles across multiple databases in one of the Oracle directory services like Oracle Internet Directory or Oracle Unified Directory. The Oracle directory can be a stand-alone LDAP server or can integrate with other corporate directories. After successfully authenticating the user, the database refers to the directory for authorization (roles) information. Database users managed through the directory service are called enterprise users, as they span multiple databases across the enterprise. Enterprise users can be assigned enterprise roles or groups that determine access privileges across multiple databases. Enterprise roles in the directory map to one or more roles in your databases. Enterprise User Security is deprecated in Oracle Database 23ai.

Centrally Managed Users (CMU) directly integrates the database with Microsoft Active Directory. Active Directory users can have their own schema or share a schema through membership in Active Directory groups. Active Directory groups can also map directly to database roles. A big difference between EUS and CMU is that with EUS, schema and role mappings can be in the database or the directory. With CMU, schema and role mappings are always in the database.

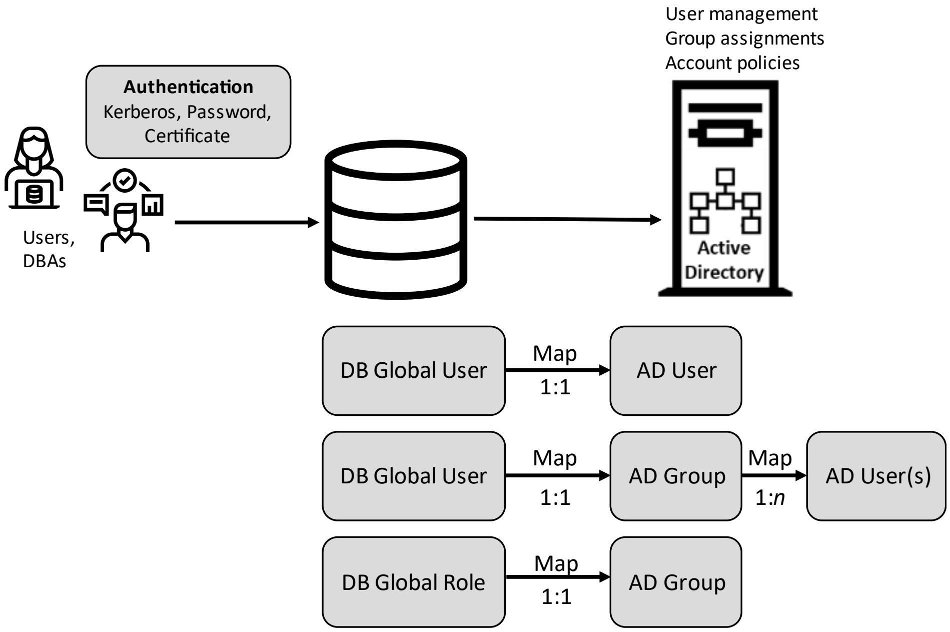


Figure 6-10: Centrally Managed Users (CMU)

EUS and CMU allow users to authenticate using passwords, Kerberos, and Public Key Infrastructure (PKI) certificates. Human users commonly use Kerberos, while passwords and certificates are typically used for application accounts.

Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users can access the database using IAM tokens. Similar to CMU, authorization is based on mappings to database schemas and roles. IAM users can map exclusively (1:1) to database schemas, or IAM users in an IAM group can be mapped to a shared database schema (many:1). With a shared schema, IAM security administrators can add an IAM user to a shared database schema by adding the IAM user to the appropriate IAM group. Shared schemas remove the burden of creating/managing/dropping database schemas for every IAM user. IAM groups can also be mapped to database roles so IAM users can be granted privileges and roles through membership in different IAM groups. IAM authentication and authorization is available for all OCI databases, including Autonomous Database, Oracle Base Database Service, and others.

Microsoft Entra ID users can access the database using Entra ID tokens and leverage Entra ID app roles for database authorization. App roles are included with the Entra ID token and are used by the database to map Entra ID users to a database schema and, optionally, database roles. Entra ID authentication and authorization is available for Oracle Database 19c and higher—these can be databases installed in OCI, on third-party clouds, or on-premises.

Protecting database accounts

Protecting DBA accounts

Database Administrator (DBA) accounts typically have wide access within the database, usually much more than is needed to administer the database. This access should be controlled using the following best practices:

- Tailor privileges for individual tasks and responsibilities to reduce the attack surface associated with these accounts instead of granting the DBA role. If extra privileges and roles are needed for troubleshooting, revoke them after the task is done.
- Use task-specific administrative privileges (SYSKM, SYSBACKUP…) instead of the SYSDBA privilege.
- Block access to user schemas or SQL commands using Database Vault—more on this in Chapter Seven.
- Fully audit all DBA activities for accountability and tracking.

Protecting security administrator accounts

Security administrators manage security controls, encryption keys, database users, and audit records. They manage the database's security posture and ensure there are appropriate constraints on authentication, failed logins, idle time, and the number of concurrent sessions. Security administrator accounts need to follow these best practices:

- Limit the privileges and roles granted to security administrators so that they don't have wide access to the sensitive data in the database. It would be unusual to grant a security administrator the DBA role.
- All their activities should be fully audited for accountability and tracking.

Protecting application service accounts

Application service accounts in the database not only have all the roles and privileges required to run the application on behalf of every application user, but for convenience, they may also have roles and privileges to perform application installation, upgrades, patching, and other maintenance activities. Such accounts need to be well protected using the following best practices:

- Revoke the privileges for application upgrade, patching, and other maintenance activities from the application runtime account and instead create a separate administrator user who is separately audited and managed. Without this separation, a hacker who has compromised an application user account can use SQL injection attacks to take over the application, change stored procedures, steal or destroy data, and delete tables.
- Grant application DBAs access to only the application schema objects and not database-wide privileges, even though this may be less convenient. Earlier in this chapter we discussed a new feature (schema privileges) in Oracle Database 23ai that makes this easier.
- Fully audit all their activities for accountability and tracking.

Protecting Data analysts or business intelligence users

These accounts allow access to data but, in most cases, should not allow data to be altered, deleted, or inserted.

- Avoid grants of system privileges like SELECT ANY TABLE. Instead, use direct object grants (either to the user or to a role) or schema-level grants.
- Audit these users for misuse – are they logging in from unusual IP addresses? Running different programs than they normally do? These could be indicators of account compromise. If the users have static IP addresses, consider using SQL Firewall or Database Vault connect rules to lock the accounts to specific workstations. We'll talk more about SQL Firewall and Database Vault later in the book.
- Audit these users for insert/update/delete activity.
- Consider making these accounts read-only (read-only accounts are a new feature of Oracle Database 23ai).

Protecting other database users

Because these accounts are outliers, where policies for the more common types of database accounts may not be appropriate, it's important to pay closer attention to them.

- Regularly validate the requirement for the account, especially if it is assigned to a human actor.
- Track grants of system privileges like `SELECT ANY TABLE`. Where possible, use object or schema privileges to reduce the user's access scope.
- Audit these accounts for misuse. Are they logging in from unusual IP addresses? Are there indications the credentials are being shared between multiple people? Are they running different programs than they normally do?

- For accounts tied to batch processing where the same SQL statements are consistently issued, consider protecting them with SQL Firewall (discussed in Chapter Eight)
- Use privilege analysis to identify unnecessary privilege grants. We discussed privilege analysis in Chapter Three.

Oracle LiveLabs

Oracle LiveLabs gives you access to Oracle's tools and technologies to run a wide variety of labs and workshops at your own pace. If you want to try out the technologies discussed in this chapter, please go to:

- Privilege Analysis
- Data Safe Fundamentals
- Audit Vault and Database Firewall

Summary

Since most attacks target authorized users of the database, properly managing accounts and their authorization is an important step in protecting the database. Oracle provides options to manage user authorizations to meet your requirements with central or local management of users. Well-scoped access controls using privileges and task-oriented roles give a great level of control that is easy to manage.

In Chapter Three we showed how tools can help identify areas of concern more quickly. For example, the Database Security Assessment Tool (DBSAT) scans databases and provides information about user accounts and privilege grants. Using privilege analysis, the right set of privileges and roles can be found easily.

Oracle Data Safe includes a User Assessment feature to analyze cloud and on-premises database user risk, letting you quickly identify user accounts that can pose a risk if account credentials are compromised. Oracle Audit Vault and Database Firewall tracks user entitlements and can help you quickly identify when user privileges change.

Proper use of privileges and roles to limit user access, combined with strong authentication, sets the foundation for a secure database.

Chapter Seven

Enforcing Separation of Duties

Why is separation of duties important?

Stealing sensitive data using compromised privileged user accounts is the most common attack vector for database breaches. While database administrators may want to use a single account to perform multiple functions for convenience, dividing duties among multiple accounts and understanding exactly which privileges are in use can dramatically improve security. For example, the person who creates a new database account and assigns that account its initial credentials might not be the person who can grant that account access to data. By separating the duties of account creation and assigning data access, you block a single person from creating a new pathway for data access. It would require two people working together to introduce that new pathway. Another example might be encryption key management – the person who controls the ability to create and rotate encryption keys should not normally be the same person who manages the rest of the database. This separation of duties reduces the chance of a compromise of both the database (or database backup) and the encryption keys at the same time.

Separation of duties is also useful in preventing human error. It can increase the chances a mistake will be caught and fixed before it causes harm.

Why should you focus on separation of duties?

Separation of duties divides tasks among multiple users to reduce the risk of fraud, errors, or malicious activities. Separating duties across multiple users helps minimize the potential damage from compromised accounts. Cybersecurity and regulatory concerns drive the use of strong security controls for accounts used by insiders and privileged administrative users.

This chapter discusses how Oracle Database's capabilities let you implement separation of duties and enforce least privilege to minimize losses from compromised accounts, reduce the impact of malicious administrators, and mitigate the risk of human error.

Multiple people? Or multiple accounts?

Your ability to separate duties across different people will be constrained by the size of your organization. If you have only one or two DBAs for the entire organization, then the degree of separation you can apply will be limited – you probably won't have a separate backup and recovery DBA, performance tuning DBA, etc. In cases like this, we've seen some organizations follow a multiple-account strategy, with the same person performing most functions using one account and certain critical functions (security administration as an example) using another account with different privileges. The same person has credentials for both accounts but must log in using the account dedicated to the function they intend to perform. In this way, you reduce the risk of human error and cut down on the blast radius of a compromised account, but forgo risk mitigation against a malicious administrator.

Controlling powerful administrators

We can strike a balance between the need for privileged users to do their jobs and the need to protect sensitive data by doing the following:

Enforce separation of administrative privileges: Most DBA operations can be completed with narrow and very specific privileges. With these specific privileges, multiple administrators can perform all the normal operations to manage a database without the risk inherent in using the all-powerful `SYSDBA` administrative privilege. For example, the `SYSOPER` administrative privilege allows an administrator to perform limited tasks, such as starting and stopping the database, without having the full range of powers of the `SYSDBA` privilege. Oracle Database provides additional administrative privileges, including `SYSBACKUP, SYSDG, SYSKM,` and `SYSRAC,` to enable database backups, Oracle Data Guard administration, key management, and Oracle RAC management, respectively.

Dividing database administration duties into distinct administrative, operational, and security responsibilities helps prevent privilege abuse by ensuring that no single administrator has unrestricted control over the database. As discussed in Chapter Six, Oracle Database allows all administrative privileges to be assigned to a single operating system (OS) group, but this is not recommended because that would not enforce separation of duties. Instead, the best practice is to create separate OS groups for each administrative privilege and associate them with the corresponding privilege at the database binary level during installation.

Additionally, administrators who connect to the database locally (by first accessing the database host server and then logging into the database) should avoid using the database owner's OS account (typically `'ORACLE'`). Instead, they should use individual OS accounts with membership in the appropriate OS group(s) to ensure accountability and minimize security risks. Another best practice is not to have these administrators connect to the database using the client that is part of the database binary installation but instead install a separate client for their use.

Control the database owner account: The `SYSDBA` administrative privilege grants full access to the `SYS` (database owner) account and should be restricted to use during database upgrades and patching. Change management processes and privileged account management (PAM) systems should be used to control remote access to accounts with this privilege, including the default SYS account.

Implement a trusted path to data: Even if administrator credentials are compromised, access to the database should only be allowed if it follows a trusted path. The trusted path might include IP address, program used, time of day, authentication type, etc. Adding additional authorization checks reduces the probability that an attack using compromised credentials succeeds. Trusted path enforcement is also an excellent practice for application service accounts, as they are another prime target for attackers.

Enforce least privilege: Grant database users the minimum set of privileges needed to accomplish their intended tasks or functions and no more. Use privilege analysis (discussed earlier in Chapter Six) to identify unnecessary privilege and role grants. When granting privileges to a user or role, grant specific object or schema privileges rather than broad system privileges that allow access to all objects in the database. Similarly, you should create database roles with the minimum privileges necessary for a particular function instead of using powerful roles like the built-in `DBA` role. Granting one or more of these task-specific roles to a user allows for a closer match to the tasks the user needs to perform without granting unnecessary privileges. The least privilege model helps reduce the blast radius of a compromised account by limiting what the attacker can do within the database.

Do not share accounts: It is not uncommon to see administrators share database accounts for convenience. However, this removes accountability and increases risk as many people have access to the same account. Each DBA should have an individual, named account with appropriately tailored privileges and roles. If you have a lot of DBAs or a large number of databases, you could simplify management by centralizing database authentication and authorization in an external identity service like Active Directory. See Chapter Five for more information.

Safeguard the audit trail: Audit records provide a record of actions on a database, directory, or operating system. Activity by privileged administrators should be audited. Information such as privileged user actions that were taken (`CREATE USER, CREATE ANY TABLE, ALTER SYSTEM, ALTER SESSION`) coupled with the context of the event, such as the initiating IP address, event time, and actual SQL statement, are just a few examples of audit information needed in compliance and forensic reports. The `AUDIT_ADMIN` role grants privileges to change audit policies and purge audit records. This role should not normally be granted to anyone other than the security DBA and any tools or services that are used to collect and aggregate audit data. Further, if audit records can be created when the administrative users access application accounts, alerts can be raised for further action. More information about audit can be found in Chapters Twelve and Thirteen.

Enhancing separation of duties with Oracle Database Vault

Database Vault is one of the most powerful security options for Oracle Database. Database Vault is a multi-purpose access control mechanism that addresses many different use cases. Three primary use cases for Database Vault are:

- Blocking privileged user access to data: Database Vault prevents privileged users, such as DBAs, from accessing sensitive application data. This is crucial for meeting regulatory requirements and reducing risk. Even if DBA credentials are compromised, sensitive data remains protected, minimizing the potential for data breaches.
- Enforcing a Trusted Path: Database Vault ensures that application service accounts, often targeted by attackers, can only connect to the database under strict conditions. Rules can restrict these accounts to specific IP addresses, programs, or operating system users.
- Controlling SQL command execution: Database Vault allows precise control over database commands, enabling conditions like requiring multiple administrators for critical actions (e.g., no table can be dropped unless two DBAs are logged in) or restricting user actions to specific times or circumstances.

Block administrator access to data with Oracle Database Vault

In most cases, there is no reason for database administrators to have access to the application schemas and data. Using the different administrative privileges does not prevent the administrators from accessing user-specific data. Read on to see how to enforce separation of duties further and break the link between database administration and data access.

The out-of-box DBA role is granted most system privileges, including the powerful `SELECT ANY TABLE` privilege. With this privilege, a DBA can view any data in the database, including salary, taxpayer IDs, phone numbers, corporate financial forecasts, intellectual property, and other sensitive data. If a cyber-criminal successfully compromises the credentials of a DBA, they can now easily access and exfiltrate your valuable data.

Other privileged accounts may also be granted `SELECT ANY TABLE`, frequently by administrators who are trying to quickly solve problems related to their access. Complex applications with multiple schemas frequently use system privileges instead of object privileges, making it possible for exploits like SQL injection attacks to access data throughout the database (we'll talk more about ways to deal with SQL injection in Chapter Eight). Like DBAs, these accounts create an elevated risk to data and are prime targets for attackers.

Oracle Database Vault can restrict privileged users, including database administrators, from accessing sensitive data with an access control mechanism called a *security realm*. Security realms are collections of schemas or specific sensitive objects where access is controlled by Database Vault. Database Vault policies override system privileges like `SELECT ANY TABLE` for objects and schemas protected by a security realm, With Database Vault, database administrators can manage the database but cannot access sensitive schemas/objects protected by realms.

Figure 7-1 shows an example of a realm protecting sensitive data within the human resources (HR) schema. The Database Vault realm prevents users with the powerful DBA role from accessing data inside the HR realm yet still allows them to do their database administration tasks like creating indexes. Business users with a valid requirement to view data are still able to access data because they have permissions within the realm.

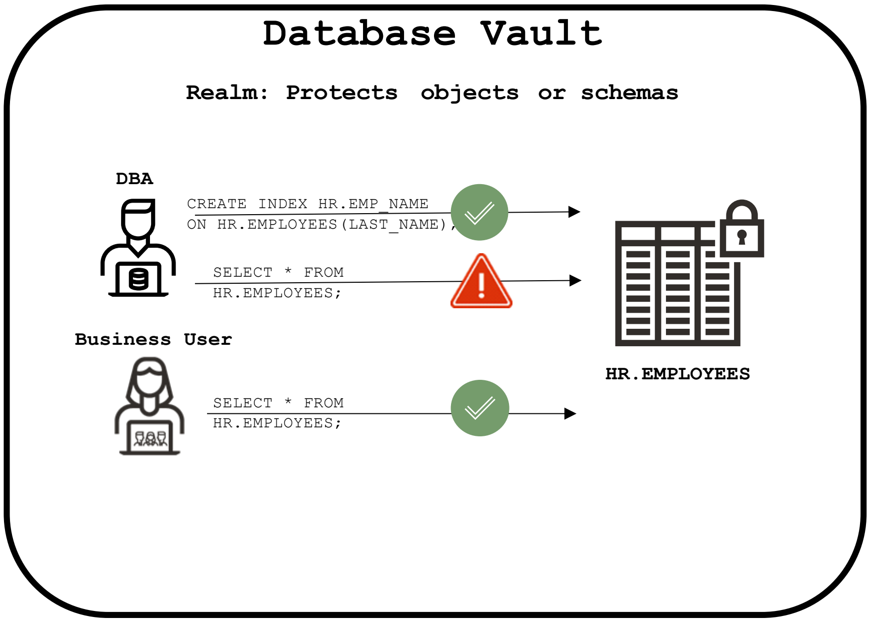


Figure 7-1: Database Vault Realms block administrator access to data

Enforce a trusted path to data

Access within a realm is constrained by Database Vault rule sets. A rule set is a collection of one or more rules that determine the conditions for access. Anything that can be checked programmatically can be used in a rule. In the example shown in Figure 7-2, the rule set has two rules. One rule checks the IP address that the user's connection is made from, the other checks that the time of day is within allowed business hours. The business user must satisfy BOTH rules in order to access data within the realm using the rule set. Using rules, Database Vault shrinks the attack surface, enforcing a trusted path to data by applying multi-factor authorization policies to reduce the risk of a compromised account even further.

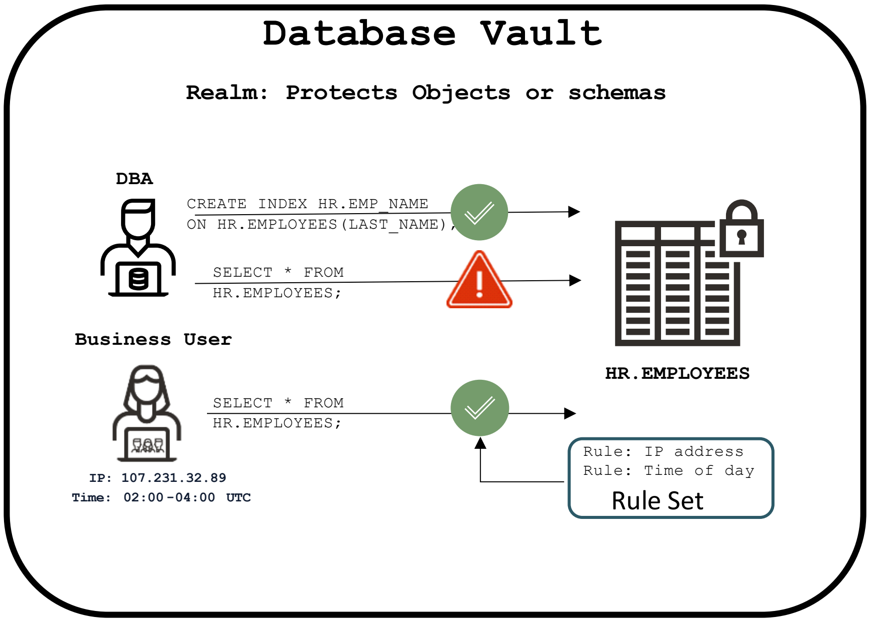


Figure 7-2: Database Vault uses rules to make access control decisions

A database can contain many different Database Vault realms, each with its own set of authorized users controlled by rule sets appropriate to those users and the realm. In Figure 7-3, we can see that the human resources application administrator can manage the HR data but cannot access objects or data within the `FIN` realm, even though the `HR` application administrator has broad system privileges. Similarly, the highly privileged finance application administrator can view data and objects within the `FIN` realm but not within the `HR` realm. Thus, Database Vault enforces separation of duties at the database object and schema level, even overriding system privileges, and isolates the DBA from application data.

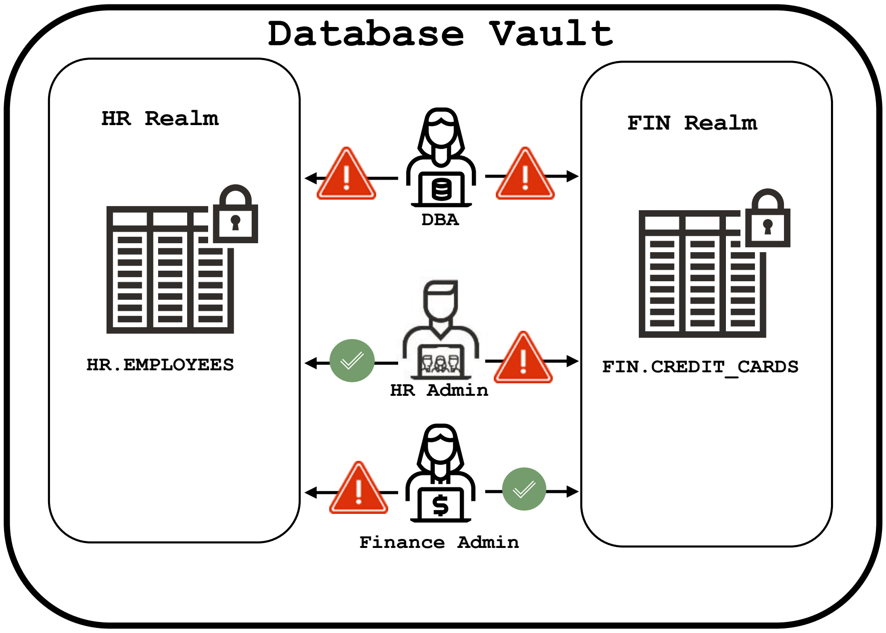


Figure 7-3: Oracle Database can have multiple realms in the same database

By default, realms are overridden by schema ownership and direct object grants. In Figure 7-3 above, that would mean that the HR account (the schema owner of the EMPLOYEES table) would be able to access data in the EMPLOYEES table and the FIN account (owner of the CREDIT_CARDS table) would be able to see data in the CREDIT_CARDS table even though they were not authorized within the realms. Sometimes this is the behavior you want, but there are situations where you want the realm to exercise control over ALL users, regardless of ownership or direct object privileges. Having the realm override object privileges is useful when an administrator needs to make changes during an application upgrade or patching. During some application upgrades, only the application procedures and functions need to be modified—not the tables and views that hold the sensitive data. In this case, stronger protection can be put around sensitive tables and views while allowing access to update the procedures.

For this use case, Database Vault realms can be made "mandatory," where not even the schema owner can access objects protected by the realm unless they are explicitly granted access through Database Vault. Mandatory realms override both schema ownership and direct object grants. With a mandatory realm, no one is allowed access – even to objects they own – unless they are specifically granted membership to the realm via Database Vault.

With Database Vault, the only user that can grant access to protected data is the security administrator and not the data owner. Database Vault simplifies identifying which users have access because auditors don't need to traverse a potentially complex chain of direct object grants to different users and roles. Instead, permissions to access or manipulate objects or schemas are maintained in easy-to-understand security realms.

Enforce separation of duties within Database Vault

As you might expect, separation of duties is baked into Database Vault. Enabling Database Vault creates additional roles intended to enforce the separation of duties. These roles include:

- **Database Vault account manager** provides out-of-the-box enforced separation of duties that controls the creation and management of users in the database along with their passwords. By default, when Database Vault is enabled, the DBA role loses the ability to create users or change their passwords.
- **Database Vault administrator and owner** roles allow a security administrator to create, modify, and manage Database Vault security controls, including the ability to manage realms and add realm participants. The Database Vault administrator role controls the various PL/SQL packages associated with Database Vault. The Database Vault owner role includes the Database Vault Administrator role, but also allows a user with the role to manage and monitor the Database Vault configuration.

This separation of database and user administration prevents a common attack pattern where a malicious actor steals the credentials of a DBA and then uses this legitimate account to create a rogue user account and grant powerful privileges to this account, enabling the user to steal sensitive data. This rogue account gives the attacker a way back into the database if the DBA credentials are later changed.

For obvious reasons, membership in the Database Vault roles needs to be carefully controlled. In cases where separation of duties needs to be relaxed (perhaps your organization does not have enough DBAs to support a separate security administrator), you can simply grant the Database Vault roles to the DBA role, allowing the DBAs to manage users and Database Vault policies. Keep in mind that relaxing the separation of duties by granting the Database Vault roles to the DBA role also re-introduces the risk that a compromise of a database administrator's account could lead to a compromise of your data.

The database security assessment tools discussed in Chapter Three will all report on which users have the Database Vault roles. Both Data Safe and Audit Vault and Database Firewall will detect drift in Database Vault role assignment, making it easy to spot when conditions change.

Oracle Database Vault also allows you to separate the viewing and management of audit records from privileged users, such as database administrators. This separation adds a layer of security, as purging audit records is a common technique to cover malicious or inappropriate activity. Oracle Database 23ai adds Database Vault control over the `AUDIT_ADMIN` and `AUDIT_VIEWER` roles. We'll talk more about those roles when we discuss auditing in Chapter Twelve.

The database administrator team cannot access application data unless they are explicitly granted permission within the realm that protects the data, but they can still perform common DBA tasks like performance tuning, memory management, backup, and others. Certain DBA tasks like exporting data or scheduling automated jobs may expose sensitive data to the DBA and will require additional authorization by the Database Vault security administrator.

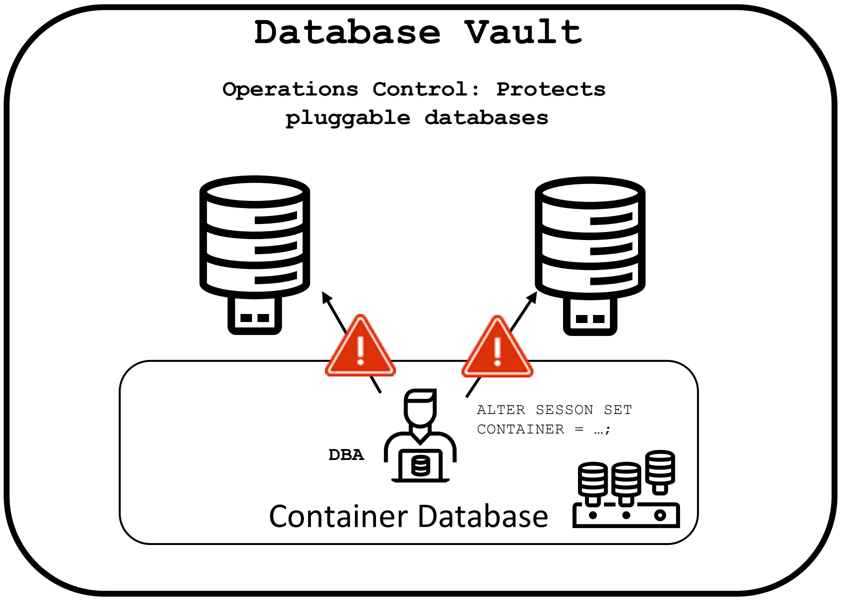
Enforce separation of duties with containerized databases

Oracle introduced the concept of containerized databases in Oracle Database 12c. In a containerized environment, a container root database (CDB) holds common metadata and settings that are applicable throughout the containerized databases. A pluggable database (PDB) is a collection of schemas, schema objects, and non-schema objects that appear to an application or user as a separate database. A single CDB can host many PDBs. There are numerous advantages to the containerized architecture – simpler maintenance, portability across servers, increased resource utilization with lower overhead, and more. From a security standpoint, the container architecture has advantages because it provides another way to separate data and users into different pluggable databases, where each pluggable database can have its own policies, administrators, and configuration.

Starting with Oracle Database 19c, Database Vault's *operations control* feature can block common users (infrastructure DBAs, for example) from accessing application data in pluggable databases. There is no need to create separate security realms to use Database Vault Operations Control to separate common users from local data in PDBs. As you adopt Database 23ai, which always uses container databases, operations control becomes more important than ever.

With operations control, the Database Vault administrator can:

- Perform all administrative activities on the container database.
- Perform all administrative activities on the pluggable database, except accessing or managing local user objects or local user data.
- Grant a common administrator an exception to access PDB local data or exempt a PDB from operations control enforcement.



```
ALTER SESSON SET
CONTAINER = …;
```

Figure 7-4: Operations Control protects pluggable databases

Control SQL database commands with Oracle Database Vault

This isn’t really part of separation of duties, but since we’re discussing Database Vault, this seems like a good place to mention it. Most organizations have change management policies that forbid dropping tables, modifying stored procedures or other database objects, and changing system parameters outside of defined maintenance windows. Database Vault can enforce these policies, preventing accidental or malicious violations. The chance of accidents is very real – while compromised accounts are the most common contributor to a database breach, human error is the most frequent cause of production outages. DBAs frequently have terminal windows open in various dev, test, and production systems, and it’s easy for an administrator to execute a command in what they think is a test system’s terminal session but is actually running against a production environment.

Database Vault provides fine-grained SQL command rules to prevent this type of accidental (or malicious) activity. These command rules can prohibit certain commands from running or use rule sets to establish conditions under which commands are allowed to run. As with realms, the rule sets attached to a command rule can be based on anything that can be checked programmatically. With command rules, changes to production schemas, like dropping a table, might be forbidden unless two application administrators are logged in at the same time, with the person executing the change connecting from an approved jump server operating outside of normal business hours and having a change number set for their session.

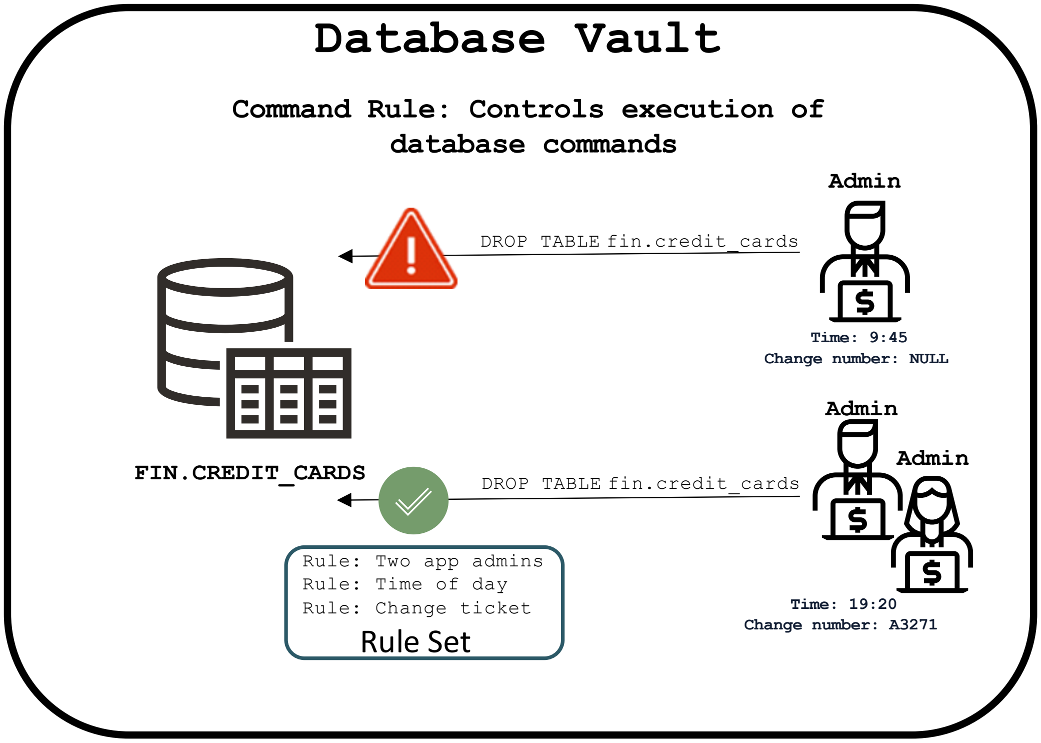


Figure 7-5: Database Vault Command Rule

For example, you can limit DBAs to running a `DROP TABLE` command only from their desktops at work and only during working hours to prevent unauthorized remote access off-hours. Other examples are to limit SQL commands like `CREATE`, `TRUNCATE`, or `ALTER TABLE` to maintenance windows. Database Vault can also enforce conditions like requiring that changes have to be associated with a trouble ticket number or ensuring that two DBAs have logged in at the same time before a particularly risky activity (like rekeying the database) can succeed.

Break-Glass scenarios

As a general principle, operating system administrators, database administrators, and developers should have limited or no access to production data. However, there are occasions when access is required, such as to perform application upgrades or troubleshoot application schema or application data issues.

When these scenarios occur, the administrative and development staff should have a mechanism to gain access to the production schema and production data. This is called a break-glass scenario. When an administrator breaks the glass, they are temporarily acquiring elevated privileges to perform a specific task. They might do this by checking out a privileged account and using it to conduct the necessary actions or by enabling an Oracle Database role to grant their day-to-day account a higher level of database privileges and production data access.

In the former, checking out an account, you could consider having a highly privileged user account stored in a Privileged Access Management (PAM) solution. The person needing access would check out this account, perform their task, and then check the account back into the PAM solution. In this scenario, you can use Database Vault to limit where and how the checked-out account can be used. For example, the checked-out account might only be allowed to connect from a specific subnet or from a specific trusted host. You might limit the checked-out account to perform `CREATE` or `ALTER` commands but not `DROP TABLE`, `TRUNCATE TABLE`, or `DELETE` commands on production schemas. This ensures the account can see the data but cannot perform destructive actions on the data. Once the emergency is resolved, the checked-out database account can be automatically locked by the PAM solution.

In the later scenario, the person needing access can enable a database role called a secure application role. Secure application roles are not granted to a user, they are enabled by running a PL/SQL procedure that must execute successfully in order to enable the role. You can learn more about secure application roles here. In this scenario, the role can only be enabled if the requesting database user meets the criteria you have documented in a Database Vault rule set. The rule set might only allow the role to be enabled if the user is from a set of trusted hosts, IP addresses, or subnets. The rule set might also only allow the role to be enabled during specific hours or on specific days. You could restrict access to a chosen list of database users, perhaps only senior database administrators. When the emergency is resolved, the database user logs off and the role is automatically revoked until it is needed again. It is very similar to using sudo on Linux systems or "open as Administrator" on Windows systems. The privileges are available to the user, but they are not always enabled.

In day-to-day operations, each administrator should use a least-privilege, named account with tightened domain-specific roles. Break-glass privileges are specifically for unplanned crises and controlling them behind robust administrative policies helps maintain a secure database environment. This way, if an attacker compromises ordinary credentials, they still face the barriers of multiple approval levels, rotating credentials, and the blend of preventative and detective controls that enable a defense-in-depth approach.

Operationalizing Oracle Database Vault

There are three key elements to consider when operationalizing Oracle Database Vault:

- What needs to be protected, and under what conditions?
- Which SQL commands should be allowed in your database environment?
- What process and organizational changes are needed because of the newly enforced separation of duties?

Before creating Oracle Database Vault realms, you need to know which data is sensitive. Some organizations operate under the assumption that all application data is sensitive and that a realm should be created on every application schema. Others prefer a finer-grained approach, using sensitive data discovery to identify which data needs to be protected. Both approaches are valid, and Database Vault can support either of them.

To determine which SQL commands to allow, you can examine audit records to discover what critical SQL commands (i.e., `ALTER SYSTEM`, `ALTER DIRECTORY`, …) were run and the context they were run under (IP address, program name, database, or operating system username, etc.). Oracle Database Vault realms or SQL command rules can constrain authorized commands to the time of day, incoming IP address, and other rules to ensure changes are done in accordance with policy.

Most consideration, however, should be given to the separation of duties within the Database Vault since Database Vault enforces the separation of duties outside for the rest of the database. Some organizations take multiple steps before they implement full separation of duties. For example, you may choose to initially assign the new Database Vault roles to the DBA role, and then later take steps towards separating duties. This allows some controls in Oracle Database Vault to be implemented quickly and for SoD changes to be spread over a longer time frame.

Database administrative staff may hesitate to turn on preventive controls like Database Vault because they are worried about causing application outages or breaking administrative scripts. To minimize this impact, Database Vault provides a **simulation mode** that allows you to verify that realms and SQL command rules do not impact application run-time and operations. Instead of blocking, simulation mode logs Database Vault policy violations, allowing a full end-to-end regression test without interfering with existing application operations. Once testing is complete, and you've validated that you would not have blocked legitimate activity, Database Vault policies can be updated from simulation to blocking mode.

Finally, audit records need to be captured and alerted when a violation occurs. These audit records are of high value since Database Vault alerts either indicate an initial probe by a malicious user or the need for additional training for existing users on accessing sensitive data. Both Data Safe and AVDF can be used to collect and report on Database Vault violations.

Database Vault and packaged applications

Several Oracle and third-party applications are certified to work with Oracle Database Vault, including Oracle Fusion Human Capital Management, Oracle Enterprise Resource Planning Cloud Services, Oracle E-Business Suite, Oracle PeopleSoft, and SAP. Please review the certification matrix on Oracle Support for more information about application certifications and the Oracle Cloud website for cloud support.

Oracle LiveLabs

Oracle LiveLabs gives you access to Oracle's tools and technologies to run a wide variety of labs and workshops at your own pace. If you want to try out the technologies discussed in this chapter, please go to:

- DB Security- Database Vault

Summary

One of the most common attack vectors is stealing sensitive data using compromised privileged user accounts. Oracle Database Vault security controls, such as realms and command rules, protect sensitive data and database operations from privileged or compromised insiders.

As a standard security practice, database administrators should not have access to sensitive data. Separating duties and least privilege can minimize the losses from compromised accounts

Chapter Eight

Minimizing Risk from SQL Injection

Why is SQL injection important?

The Open Worldwide Application Security Project (OWASP) lists SQL injection as one of the top ten web application vulnerabilities, and it's been on that list (usually as the #1 vulnerability) since the list was first published in 2004! SQL injection is one of the oldest and most frequently encountered database attack methods, primarily because it operates through applications, which tend to be exposed to a broader range of potential attackers, often to the entire internet. Theoretically, this vulnerability shouldn't exist – developers have been trained for many years about the dangers of SQL injection, and there are even automated code-scanning tools that look for SQL injection vulnerabilities. However, despite years of education and training to solve the problem, SQL injection continues to plague data-driven web applications.

Why should you focus on SQL injection?

SQL injection is often the entry point for a more extensive compromise. In addition to corrupting, destroying, or exfiltrating data from the compromised database, attackers may use SQL injection as a path to move laterally to other hosts inside your network.

This chapter discusses two approaches to mitigate the risk of SQL injection at or near the database layer: network-based Database Firewall or built-in SQL Firewall. We will review the differences between the two approaches and suggest strategies for selecting the approach that best fits your needs.

What is SQL injection?

Strictly speaking, SQL injection is not a database vulnerability, it's an application vulnerability that allows an attacker to work through the application to steal or corrupt data, insert false or misleading information into the database, or even destroy the database. Most multi-tier applications connect to the database using a powerful application account with access to the entire application schema and procedures.

There are many SQL injection vulnerabilities, attacks, and techniques. But all of them follow a similar attack pattern as shown in Figure 8-1:

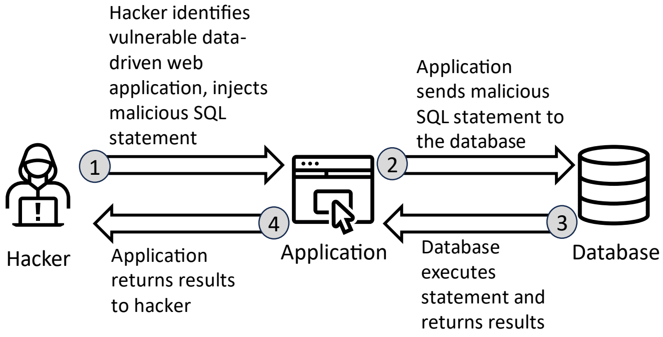

Hacker identifies vulnerable data-driven web application, injects malicious SQL statement
1
Application sends malicious SQL statement to the database
2
Hacker
Application
Database
4
3
Application returns results to hacker
Database executes statement and returns results

Figure 8-1: SQL injection attack pattern

The application tier enforces access control in the middle tier based on the end user's identity and authorization. SQL injection attacks target applications and the underlying database by injecting malicious SQL into input fields or parameters of data-driven applications. The attack forces the application to respond to the attacker-injected SQL. Injected commands can do anything that the application could have done. Thus, the attacker circumvents the authentication and authorization of a web application and retrieves the contents of the database – giving them access to sensitive data: customer information, personal data, trade secrets, intellectual property, and more. Because attackers may try to use SQL injection to add, modify, and delete records in the database, it's a good practice to guard against SQL injection on BOTH sides of the application, just in case there is an unknown vulnerability at the application layer. Before the application (closest to the application user), we employ web application firewalls (WAF) that look for SQL injection signatures in incoming HTTP traffic. The WAF will stop some of the most common attacks, but because WAFs are signature-based, they will not catch everything, particularly zero-day attacks that are not considered in their signatures. This is why it's so important to also combat SQL injection after the application – inspecting the actual SQL commands issued by the application.

Approaches to addressing SQL injection attacks

Preventing or mitigating SQL injection attacks is not easy. The need for input validation is well known and has been for many years – but developers make mistakes, which code reviews and automated code scans sometimes miss, creating new vulnerabilities. For existing and legacy applications, you probably do not have access to the source code required to modify vulnerable applications.

Web Application Firewalls (WAF) are one way to block SQL injection attempts by filtering out suspicious HTTP traffic before it reaches the application. Most WAFs depend on signature pattern matching - they may be able to detect and block well-known SQL injection payloads but are helpless in the face of zero-day exploits or complex SQL injection attacks. A WAF cannot evaluate the actual content of the injection payload and cannot use the full SQL context when making decisions.

Another approach to blocking SQL injection attacks is to filter database traffic before the database processes it. This is where a database firewall comes into the picture. As you can see in Figure 8-2, a prudent security architect will plan to use both approaches, attempting to stop SQL injection before it reaches the application, but also protecting the database.

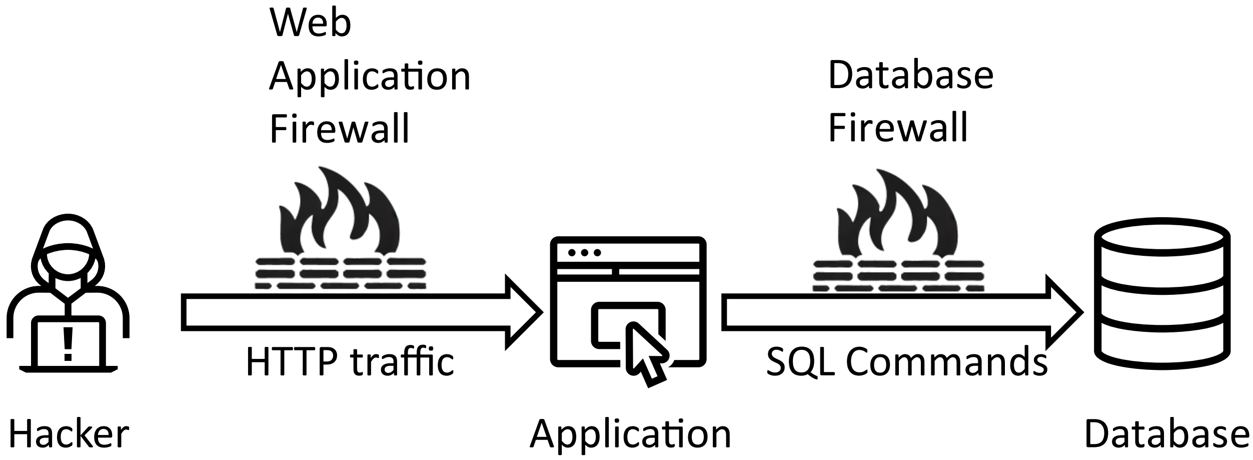


Figure 8-2: Approaches to combating SQL injection

Database firewalls offer a simple but effective way to analyze incoming SQL to detect injection attacks, raise alerts when required, and block injection attacks from reaching the database. Oracle Database security provides two approaches to mitigate the risk of SQL injection against data-driven web applications.

1. Network-based Database Firewall in Oracle Audit Vault and Database Firewall (AVDF)
2. Database-resident SQL Firewall operating inside the database kernel (a new feature in Oracle Database 23ai)

Unlike web application firewalls, neither of the above approaches relies on regular expressions or looks for signature-based matches. Instead, the database firewall treats the challenge of blocking SQL injection as an access control problem. It learns typical application SQL traffic and creates an allow-list of permissible statements. The database firewall can reject or alert if the SQL statement does not match its list of permissible statements.

AVDF applies that protection at the network layer, working between the database and the application. SQL Firewall moves that protection from the network into the database kernel, avoiding the complexity of routing database traffic through a separate Database Firewall. Both approaches have advantages and disadvantages. Let's look at each of these solutions to understand how they mitigate the risk of SQL injection.

Both network and database-resident firewalls are functionally similar and can address these three use cases:

- Provide real-time protection against attacks by restricting database access to only authorized SQL statements/connections
- Mitigate risks from SQL injection attacks, anomalous access, and credential theft/abuse
- Enforce trusted database connection paths

Network-based Database Firewall

Database Firewall, a component of Audit Vault and Database Firewall, acts as the database's first line of defense. Database Firewall monitors incoming SQL traffic and enforces expected database behavior while helping prevent SQL injection, application bypass, and other malicious activities from reaching the database. A single Database Firewall can protect multiple databases of different types from a central location. It monitors enterprise databases, including Oracle Database, MySQL, Microsoft SQL Server, SAP Sybase, and IBM Db2.

As shown in Figure 8-3, when an application user attempts to insert malicious input into an application field, while the application may not prevent it, Database Firewall does block it because the SQL submitted by the application does not match Database Firewall's allow-list of permitted SQL. Information about the attempted violation is sent to the Audit Vault server and can generate an alert. The Audit Vault server also provides a platform for analysis and reporting of the malicious activity. We will talk more about the Audit Vault server in Chapter Thirteen when we discuss auditing reports and alerts.

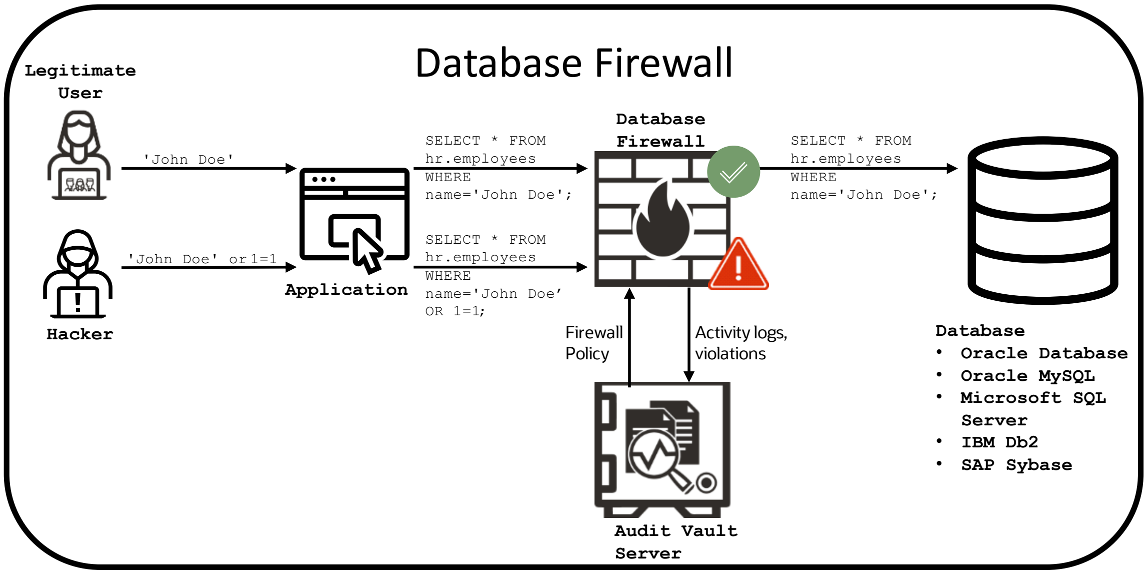


Figure 8-3: Database Firewall blocks SQL injection attempts

Database Firewall protects databases using one of three different modes, as shown below in Table 8-1: Deployment Modes of Database Firewall.

MODE	Details	Monitoring or blocking
Proxy	All traffic to the database server routes through the Database Firewall, including return traffic.	Monitoring Blocking
Host Monitor	Host Monitor is deployed on the same machine as the database. It captures SQL traffic going to the database and then securely forwards it to the Database Firewall.	Monitoring
Out-of-band	Database Firewall listens to the network traffic sent to the database. Several technologies, such as span ports, port replicators, etc., can send a copy of the database traffic to the Database Firewall.	Monitoring

Table 8-1: Deployment Modes of Database Firewall

The choice of deployment mode for mitigating risks with SQL injection depends on whether you want to detect/monitor potential SQL injection attacks or if you also wish to block unauthorized SQL.

If you only want to monitor for attacks (without blocking), then any of the modes are acceptable. All modes work by "sniffing" network traffic and looking for SQL statements that violate the Database Firewall's policy.

- Proxy mode gathers information directly at the Database Firewall, with database traffic flowing through the Database Firewall, and the firewall sniffing SQL traffic as it passes through.
- Host monitor sniffs network activity from the database server, using the host monitor agent to capture incoming SQL traffic and relay it to the Database Firewall for analysis.
- Out-of-band mode has the Database Firewall sniffing network traffic from the network – either because the firewall is on the same network segment as the database or, more commonly because the network switch is sending a copy of the database traffic to the firewall. Depending on the switch or router this replication may be referred to as a span port, mirror port, port replica, or some similar term.

If you intend to block unauthorized SQL traffic, the firewall must be in line with the incoming SQL traffic using proxy mode. Proxy enforcement can initially be set to monitor traffic (without blocking) and just alert on unauthorized activity. Once you are confident in the SQL allow-list, you can switch the firewall to blocking mode, enforcing the firewall policy and preventing SQL injection and other unauthorized SQL from reaching the database.

Database Firewall inspects SQL traffic coming into the database and determines whether to allow, log, alert, substitute, or block the SQL. The firewall evaluates SQL traffic through multiple stages, including checks for the IP address, database user, OS user, program name, SQL statement category, data definition language (DDL, data manipulation language (DML), and database tables accessed. This information determines whether the SQL statement should be logged, generate an alert, allowed to pass through to the database, or blocked.

Database Firewall supports both allow-list and deny-list policies, but in most cases, we recommend using allow-lists. After all, the universe of "bad" SQL statements is nearly infinite. In contrast, the universe of statements an application should be making is a small subset of the total number of possible variations. Put another way, it's much easier to describe the things the firewall SHOULD let through than it is to list all the things the firewall SHOULD NOT allow to pass!

Reporting on Database Firewall Violations

Database Firewall collects violations and activity logs and transfers them to the Audit Vault server for analysis, alerting, and reporting. We will talk more about the Audit Vault server in Chapter Thirteen when we discuss auditing reports and alerts.

Managing Database Firewall

The database firewall policies are managed from the Audit Vault server's console or via APIs handled by the Audit Vault server. There is no separate console for the Database Firewall. For more information on Database Firewall policies, refer to the Audit Vault and Database Firewall Auditor's Guide. You may also find this technical paper useful.

Database-resident SQL Firewall

SQL Firewall is a new capability in Oracle Database 23ai that is licensed as part of Oracle Database Vault or with Oracle Audit Vault and Database Firewall (there is no need to license both, SQL Firewall is included with either of them). Like the Database Firewall, SQL Firewall helps mitigate the risks of web application attacks such as SQL injection on data-driven web applications. Unlike the Database Firewall, SQL Firewall is internal to the database and does not require additional product installation or network configuration. Because SQL Firewall is embedded within the Oracle Database kernel, operating closer to where data resides, it cannot be bypassed. SQL Firewall inspects all incoming SQL statements, whether local or over the network, encrypted or clear text, and ensures the database executes only explicitly authorized SQL arriving via a trusted path.

Like the Database Firewall, SQL Firewall uses an allow-list approach. Unlike the Database Firewall, SQL Firewall has access to the user session's context and the database's internal metadata. The combination of the allow-list approach and SQL Firewall's access to metadata and session context means hackers cannot fool the firewall by encoding their SQL statements, referencing synonyms, or using dynamically generated object names.

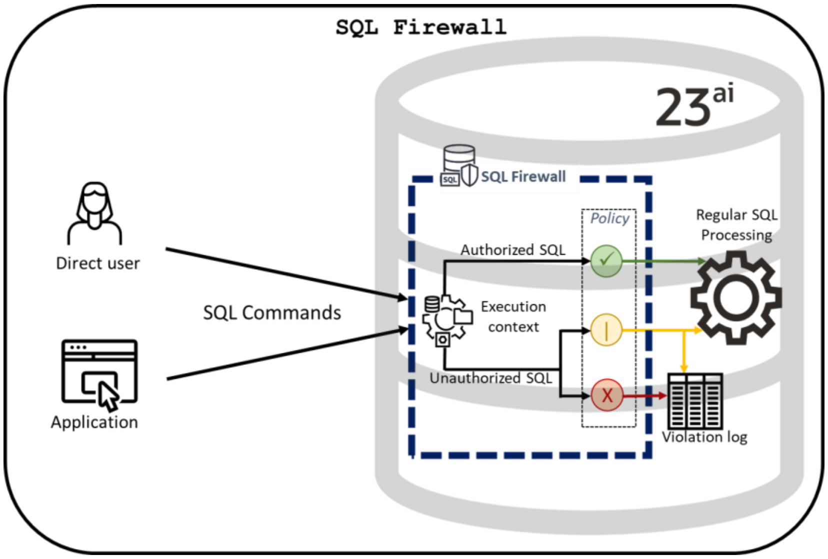


Figure 8-4: Oracle SQL Firewall is built into the Oracle Database kernel

SQL Firewall can also monitor (or block) unusual access patterns. SQL Firewall checks the source IP address, operating system username, and program used by incoming connections. If a database account is used from a previously unknown address or program, the SQL Firewall can treat that as a violation. SQL command rules require extensive training of the firewall to ensure that all possible SQL statements have been included in the allow-list. Session context rules are easy to capture, requiring minimal training of the SQL Firewall. They can even be entered manually if desired.

Unlike Database Firewall, which is heterogeneous and works on many different types and versions of databases, SQL Firewall is part of Oracle Database and only works for Oracle Database 23ai and higher.

Reporting on SQL Firewall violations

SQL Firewall violations are written to `DBA_SQL_FIREWALL_VIOLATIONS` using Oracle Database's fast ingest capabilities. If desired, SQL Firewall violations can be audited so that records of violations are protected by the database audit trail.

If you are working with many databases, you should have the violation logs collected for consolidated analysis, alerting, and reporting by Data Safe or Audit Vault and Database Firewall.

Administering SQL Firewall

There are two options for administering SQL Firewall – command line (using PL/SQL procedures) or graphically (using Oracle Data Safe). Both allow you to enable SQL Firewall, create and manage the SQL Firewall allow lists, and manage the violation logs.

Deciding which to use: Database Firewall or SQL Firewall

Database Firewall is a network-based SQL Firewall that can monitor SQL traffic of Oracle and non-Oracle Databases. For non-Oracle Databases or Oracle Databases older than 23ai, AVDF is the only choice. For Oracle Database 23ai and above, there are multiple factors that you might want to consider when deciding between the two options:

- **Performance Impact:** The database server has no additional CPU overhead with a network-based Database Firewall. All work is done off-server on the Database Firewall machine. For most deployment modes, Database Firewall introduces zero performance overhead. In proxy mode, the Database Firewall introduces minor additional network latency, with the degree of latency primarily controlled by the position of the Database Firewall on the network. SQL Firewall introduces no observable latency but does create minimal CPU overhead, not exceeding 1% to 1.5%.
- **Deployment Options:** Database Firewall is deployed on a dedicated virtual machine or hardware (separate from the database server). Database Firewall has multiple deployment options: Proxy, Host Monitor, and Out-of-Band. Proxy is the only in-line configuration that supports blocking use cases, but using the proxy requires client-side connection changes to route traffic through the proxy. SQL Firewall is part of the database, requiring no client-side changes. SQL Firewall is the best solution for deployments where blocking is the predominant use case or if it is impractical or undesirable to change client-side configurations.
- **Scope of inspection:** Database Firewall can only monitor SQL traffic coming over the network. It is unaware of database synonyms, internal jobs/ stored procedure executions. For local connections, it is only able to monitor the traffic using the host monitor mode, and is not able to any block SQL from local connections. On the other hand, the SQL Firewall runs inside the database and has access to the full session context and database metadata. It can monitor and block SQLs internally. If your application server and database server are on the same host, you should use SQL Firewall.
- **Processing of encrypted traffic:** Handling encrypted SQL traffic requires additional processing in the Database Firewall but is seamless and straightforward in the SQL Firewall because the traffic has already been decrypted by the time it reaches the SQL Firewall.
- **Disruption in blocking:** Blocking SQL on the network (with Database Firewall) may disrupt transactions because each operation is handled individually as a stand-alone statement. In contrast, SQL Firewall understands database transactions and honors atomicity, consistency, isolation, and durability.
- **High Availability:** High availability for the Database Firewall is provided by redundant firewalls, with clients distributing connects/failing over based on client connect parameters or using a third-party load balancer. SQL Firewall is part of Oracle Database and takes advantage of high availability architectures like Real Application Clusters and Data Guard.

- **Separation of Duties:** Database Firewall can monitor database traffic without any input or control by the database administrators. SQL Firewall is part of Oracle Database and requires cooperation with the database administrators to deploy.
- **License cost**: If your organization already licenses AVDF, there is no license consideration in the decision since SQL Firewall is included with AVDF. If your organization already licenses Database Vault (discussed in Chapter 7), then you would probably want to use SQL Firewall instead of Database Firewall since SQL Firewall is included with Database Vault (note: most Oracle Database cloud services include Database Vault). If your organization does not license either AVDF or Database Vault, and you are not running an Oracle Database cloud service, then you will need to either procure AVDF or Database Vault.

Oracle LiveLabs for hands-on experience

Oracle LiveLabs gives you access to Oracle's tools and technologies to run a wide variety of labs and workshops at your own pace. If you want to try out the technologies discussed in this chapter, please go to:

- Oracle Audit Vault and Database Firewall
- Oracle SQL Firewall
- Managing SQL Firewall with Oracle Data Safe is covered in lab 7 of Get Started with Oracle Data Safe Fundamentals.

Summary

SQL injection is one of the most common and dangerous database attack patterns for data-driven web applications. To mitigate the risk for Oracle Database, leverage Database Firewall in Oracle Audit Vault and Database Firewall, or SQL Firewall in Oracle Database 23ai. Neither relies on heuristic pattern matches typically used by most network firewall solutions. Instead, they rely on an allow-list of SQL statements, making them effective against SQL-based zero-day vulnerabilities. They combine allowed SQL with session context to learn the normal application SQL traffic and detect and alert/block violations in real-time.

Chapter Nine

Data-Driven Authorization

Why is data-driven authorization important?

There are times when your application needs to control access to individual rows or columns of data within a table. The access control mechanisms we've discussed so far can control access at the data object level but cannot dig down into the contents of a table or view to isolate specific data rows and columns.

Fortunately, your Oracle Database offers several ways to handle this type of fine-grained authorization. This chapter discusses several methods you can use to implement fine-grained access controls.

Why should you focus on data-driven authorization?

Building data-level access controls into the application can be costly because changes to the security model almost always require changes to the application code. This may be merely difficult, or (if it is a packaged application purchased from a third party) may not even be possible. Enforcing access control only at the application layer has another downside – it practically guarantees inconsistent enforcement of policies because different applications are almost certain to have different security models. And of course, if someone connects to the database outside of the applications (say with a database client or business intelligence tool) then there is no enforcement at all!

Enforcing fine-grained data access within the database saves you time and effort in the long run and can dramatically increase security. With your access control policies in one place (the database), you have one place to manage policy changes for ALL applications that use the data. Since the database enforces data access policies centrally, those policies are applied consistently to all tools and applications.

There are several ways to exercise fine-grained control of data access. Knowing the features and options available to you helps you choose the best method to meet your control objectives.

What is data-driven authorization?

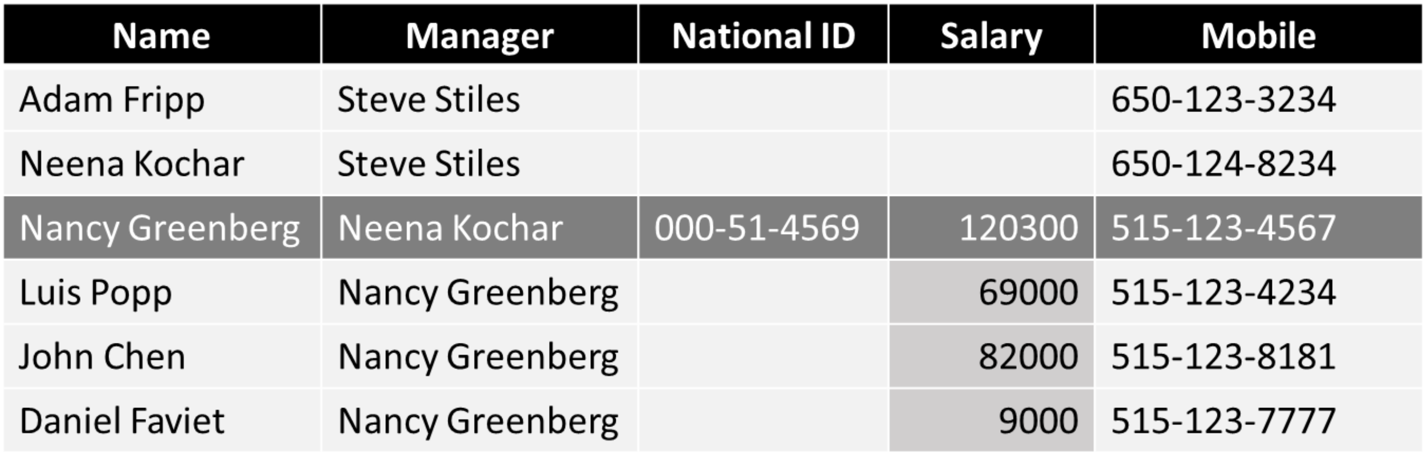
Data-driven authorization controls access at the data row and/or column level. In addition to leveraging privilege grants and session context, data-driven controls can consider the data values in making access control decisions. A very simple example might be a table of employee information, with a column that indicates who the employee's manager is. A data-driven policy might restrict access to only records for a manager's direct reports, with no ability to see information about other employees. The policy is data-driven because it relies on the value in that manager column.

Data-Driven, fine-grained authorization

Today's applications are highly complex, with different authorization policies based on the user's role, user attributes, organization attributes, session attributes, and other factors. Implementing appropriate access control checks at all the right places is cumbersome and difficult to maintain over the lifecycle of an application.

Application users should have access to only the rows and columns containing data they need to perform their tasks. However, when an administrator grants an object privilege such as `SELECT` or `INSERT` to a database user for a specific table, the privilege provides access to everything within that table. Database tables for most applications contain much more data than any single application user should be able to access. For example, a customer should be able to review their records in a support application, but they shouldn't be able to see another customer's records. A help desk technician needs to see tickets assigned to them but should not be able to view other open tickets. Sales managers need to see sales opportunities for their direct reports but not for other sales staff. These are all examples of fine-grained authorizations where the application user needs to be restricted to only the relevant rows and columns.

Figure 9-1 shows a table with both sensitive and non-sensitive HR data. Nancy Greenberg is an employee, and should be able to see phone numbers, names, and manager names for everyone, but all other sensitive data should be hidden. Nancy should be able to see her own national ID number. Because she is a manager, she should be able to select her direct report's salaries.



Name	Manager	National ID	Salary	Mobile
Adam Fripp	Steve Stiles			650-123-3234
Neena Kochar	Steve Stiles			650-124-8234
Nancy Greenberg	Neena Kochar	000-51-4569	120300	515-123-4567
Luis Popp	Nancy Greenberg		69000	515-123-4234
John Chen	Nancy Greenberg		82000	515-123-8181
Daniel Faviet	Nancy Greenberg		9000	515-123-7777

Figure 9-1: Fine-grained access control applied to a table

Challenges with implementing data authorization in applications

Consider how developers would implement access control policies on the `EMPLOYEE` table in the `HR` sample schema in Figure 9-1. They may write complex application authorization code to determine which rows and columns each user can access and under what conditions. An extra database schema or set of tables would typically be dedicated to the user and role authorization information, which the application uses to build the SQL for a particular application user request. Some common problems with this approach:

- An application developer must write this complex code and apply the application logic to all scenarios where the application user needs to access a given table.
- Every application accessing the same data must re-implement its version of the authorization policy, as the database itself knows nothing about this security policy.
- Tools that directly access the database (analytical tools, SQL*Plus, ...) have full, unfettered access to the data.

Oracle Database addresses this challenge with several technologies known collectively as data-driven authorization. Centralizing application authorization controls in the database simplifies and accelerates application development and provides a consistent set of policies to define and maintain.

Oracle introduced the well-known feature Virtual Private Database (VPD), an automated predicate-based row-filtering security technology, twenty-five years ago in Oracle Database 8*i*—at that time, it was the only database with this innovative capability. The next version of the database, Oracle Database 9*i*, introduced Oracle Label Security (OLS) to automatically filter rows based on arbitrary data and user labels. The most recent technology in this line-up is Real Application Security (RAS), which was introduced in Oracle Database 12c. Like Label Security, RAS is declarative - making it much easier to develop secure applications. We will now look at each of these technologies in more detail.

Of the three controls we are considering for data-driven access control (VPD, Label Security, and RAS), VPD and RAS are both features of Oracle Database Enterprise Edition. Label Security is an extra-cost option for on-premises databases but is included with all but the lowest levels of database cloud services. Refer to the Oracle Database Features and Licensing application if you have questions about what is included with your database.

Controlling data access using Virtual Private Database

Virtual Private Database (VPD) enforces row and column-level security policies based on a custom PL/SQL function that you develop. A VPD policy associates the function with a database table, view, or synonym. The function automatically executes for SQL statements that impact the database object the VPD policy is attached to. The function appends a WHERE clause to the user-submitted SQL statement so that the statement only returns the authorized rows and columns. For example, if the submitted statement were:

```
SELECT * FROM hr.employees;
```

The VPD function might cause the database to rewrite the SQL statement with a where clause so that something like this is what the database executes:

```
SELECT * from hr.employees WHERE department <> 'SUPER SECRET DEPARTMENT';
```

With VPD, it is possible to provide different types of access to specific rows depending on which operation the end user performs. VPD can restrict users to view information about all employees but only update their own rows. A VPD PL/SQL policy can include sensitive columns so that only authorized end users can access the sensitive column's values under controlled conditions while others get null values.

With VPD, no matter how the application accesses the table, this fine-grained authorization policy is always executed. In Figure 9-2, the user is allowed to access `HR.EMPLOYEES` table data only if the record is part of `DEPARTMENT_ID = '80'`. Additionally, VPD hides the `SALARY` data, replacing the actual value with a NULL. If a user executes a generic query without specifying any condition or `WHERE` clause, they still only get the rows and columns they are authorized to see.

```
SQL> SELECT last_name, email, department_id, salary FROM hr.employees;

LAST_NAME EMAIL                 DEPARTMENT_ID     SALARY
--------  --------------------  ----------------- ------
Hunold    AHUNOLD@EXAMPLE.COM   80
Ernst     ERNST@EXAMPLE.COM     80
Austin    DAUSTIN@EXAMPLE.COM   80
Pataballa VPATABAL@EXAMPLE.COM  80
Lorentz   DLORENTZ@EXAMPLE.COM  80
```

Figure 9-2: VPD fine-grained access control policy automatically blocks access to salary data

Controlling data access using data labels

Another method of fine-grained access control involves attaching a label to each row that describes its sensitivity or importance. In many government and corporate environments, a document might be labeled as TOP SECRET or INTERNAL USE ONLY. Then, only people who have a sufficiently high "clearance" level are allowed access to those documents. Typically, the labels reflect an ordered set of levels, and each user is assigned a maximum level they can access. When labels are attached to the rows in a table, this capability allows the database to inherently know which data is sensitive and restrict access to it accordingly.

Oracle Label Security (OLS) simplifies assigning labels to data and users and enforces access control based on those labels. A label that indicates the sensitivity of the data is associated with every row in a table protected by OLS. Each row's label can be set explicitly based on business logic. If desired, the system can set a default label automatically based on the application's label or user session that inserted the row. The label of the user session, in turn, is calculated from various factors, including the label assigned to the user, the session, the type of connection to the database, and so on. A label can be considered an extension to standard database privileges and roles. A label can be associated with a database user, and starting with Oracle Database 12c Release 2, a label can also be associated with application users supported by Real Application Security (discussed later in this chapter).

The format of the OLS label is flexible enough to accommodate virtually any data classification scheme. There are three components to a label:

- Level (hierarchical – always one and only one level within a label)
- Compartment (optional - zero or more compartments)
- Group (optional - zero or more groups)

Every label includes a level, ordered from lowest to highest, to indicate the overall sensitivity of the data. Optional components called compartments and groups may segregate information based on attributes such as projects or departments. Compartments may segregate and compartmentalize data such as special project information. Groups provide a convenient way to represent multi-dimensional hierarchical authorization based on factors like geography or organizational structure.

Figure 9-3 shows an example of a global retail store classification scheme using all three label components. Three levels are defined: HIGHLY SENSITIVE, SENSITIVE, and PUBLIC. Only users with the HIGHLY SENSITIVE label can access unreleased quarterly financial data and reports. A FINANCE compartment grants access to those needing to see financial data. Upcoming pricing changes are also sensitive, so a PRICING compartment protects pricing data. Groups represent the retail stores' geographic hierarchical structure. Take a minute to understand the group hierarchy shown in the figure – we'll be using this hierarchy throughout the next several examples.

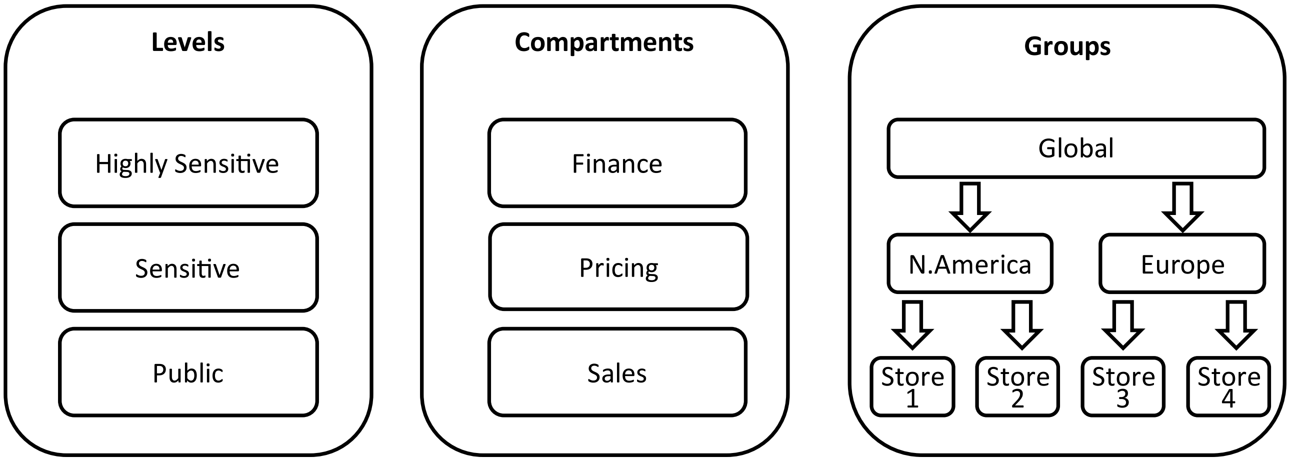


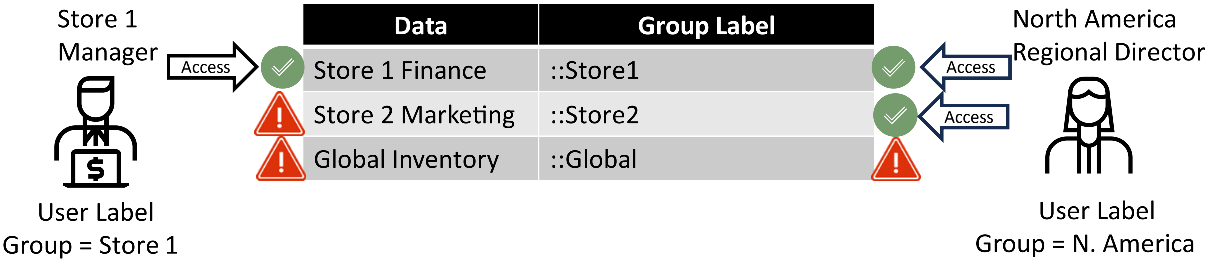
Figure 9-3: Label Security components: levels, compartments, groups

Labels are represented by the three components separated by a colon (:) as shown in Figure 9-4.

```
LEVEL:COMPARTMENTS:GROUPS
```

Figure 9-4: Label component organization

A data label considered sensitive, with finance data for store one would be represented as `Sensitive:Finance:Store1`. Pricing data for store one would be labeled `Sensitive:Pricing:Store1`. For the store one marketing manager who needs pricing information to create weekly sales newspaper inserts and emails, they will have the pricing compartment as part of their user label. We will review how labels work by starting the example with the group component and then adding in compartment and level.

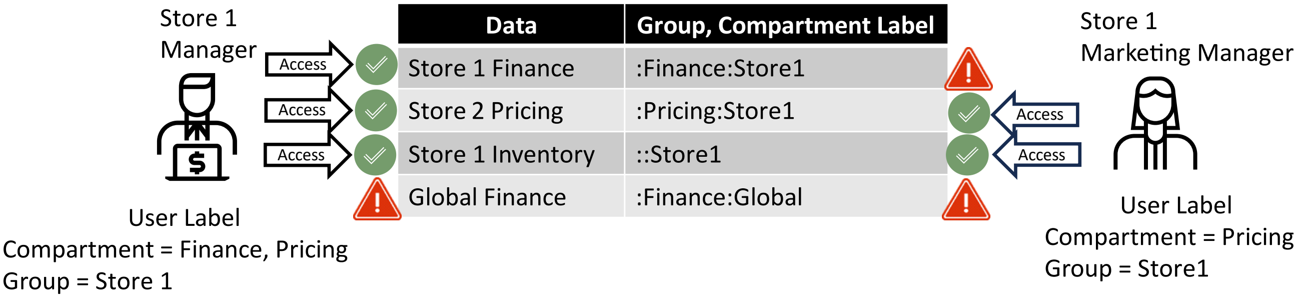


Data	Group Label
Store 1 Finance	::Store1
Store 2 Marketing	::Store2
Global Inventory	::Global

Figure 9-5: Using the group component in Label Security

In the label design, users in each store can only see data regarding their store due to the restrictions imposed by the group label component. Users with Store1 group label component can only see data that has the Store1 group data label. A North America regional Director would have N. America as their group label, allowing them to see all the elements that belong to North America (Store1, Store2) since the N. America label is defined hierarchically over Store1 and Store2 in Label Security.

The retail company required that financial and pricing data should only be seen by staff that need to see the respective data. Finance and pricing compartments were created, and financial and pricing data records were updated with the appropriate compartment label (example below). Users were reviewed to see who should have access to the compartmented data. The store one marketing manager needs to see pricing data to build sales material for the weekly store sales events, and the store one manager needs to see both pricing and financial data for store one. But neither manager can see finance data for the N.America group.



Data	Group, Compartment Label
Store 1 Finance	:Finance:Store1
Store 2 Pricing	:Pricing:Store1
Store 1 Inventory	::Store1
Global Finance	:Finance:Global

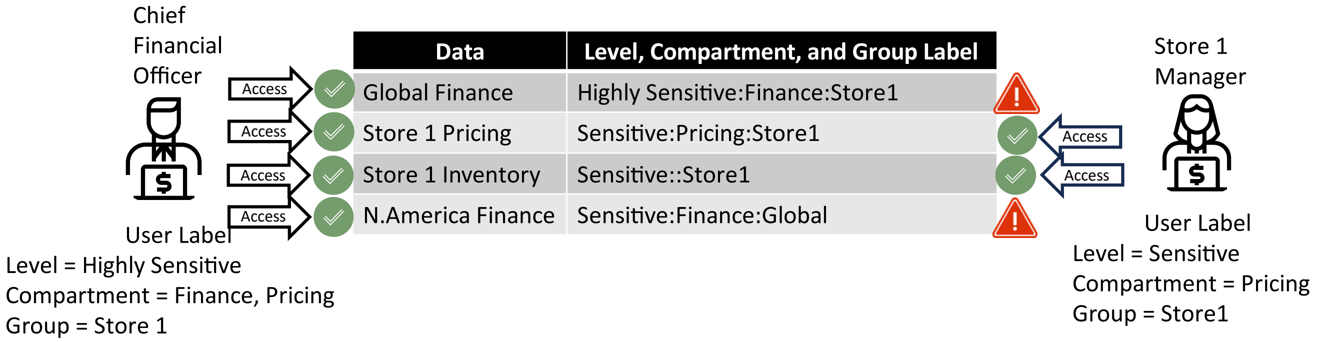
Figure 9-6: Using the compartment and group components in Label Security

Since this is a public company, global financial data and reports are highly sensitive. All data is considered sensitive except for the global financial data, which is highly sensitive with the finance compartment and global group. Only users with the level highly sensitive with the finance compartment and group global could see the data.

For the user to access data, their user label must meet the requirements for all three components (level, compartment, and group):

- Be at the same or higher level than the level for the data. For example, a user with a sensitive level label can access data labeled sensitive or public, but not highly sensitive.
- Include all the compartments in the data label. A data label with the finance compartment will require a user with the finance compartment. If a data label had TWO compartments- Finance and Pricing – then the user would need BOTH compartments in their user label to view data with that label.

- Include at least one group listed on the data label or have a group label that hierarchically contains one of the data label groups. If the data label includes Store1, a user with any of the Store1, N.America, or Global group labels could view it.



Data	Level, Compartment, and Group Label
Global Finance	Highly Sensitive:Finance:Store1
Store 1 Pricing	Sensitive:Pricing:Store1
Store 1 Inventory	Sensitive::Store1
N.America Finance	Sensitive:Finance:Global

User Label
Level = Highly Sensitive
Compartment = Finance, Pricing
Group = Store 1

User Label
Level = Sensitive
Compartment = Pricing
Group = Store1

Figure 9-7: Using the level, compartment, and group components in Label Security

Label Security was developed to meet government, military, and intelligence agency requirements for data labeling and mandatory access control. Commercial organizations also find label security simplifies their data access authorizations. In the commercial use case above, a retail store used to have individual applications and databases for each store. Consolidated reporting was difficult, and maintaining duplicate systems came with a high price tag. Label Security consolidates the data into a single database while maintaining strong controls on data access. Another common commercial use case supports data protection laws that require restricting processing of data based on user consent (the European Union's GDPR is an example of this type of law). Groups and compartments can be used to record which types of processing a data subject consents to (creating an appropriate data label). Applications set a user session label so that the application simply doesn't see records it is not supposed to process.

A significant advantage of Label Security over VPD or RAS is a complete authorization infrastructure with automatic mediation and simplified administration. Label Security leverages the concept and management of user and data levels, compartments, and groups, which map closely to many real-world use cases and do not require the administrator to create PL/SQL policy functions to enforce the access rules. Once an access control model is in place for data and user labels, access control is uniformly enforced everywhere without requiring explicit decisions about who can access which data in each table. Label Security is recommended when the data access logic can be expressed using the data and user labels.

Controlling data access using Real Application Security

We saw earlier that Virtual Private Database (VPD) controls data access at the row and column level. In most applications, end users do not interact directly with the database, and in most cases, there is no database account that belongs to the user. Instead, a single database account representing the entire application issues queries and updates on behalf of individual end users. End user accounts are maintained by the application.

Since the end user's identity is not known to the database, the application must enforce per-user access control policies. This approach requires additional software development and leads to inconsistent enforcement, especially when tools, clients, or other applications can bypass the application and connect directly to the database.

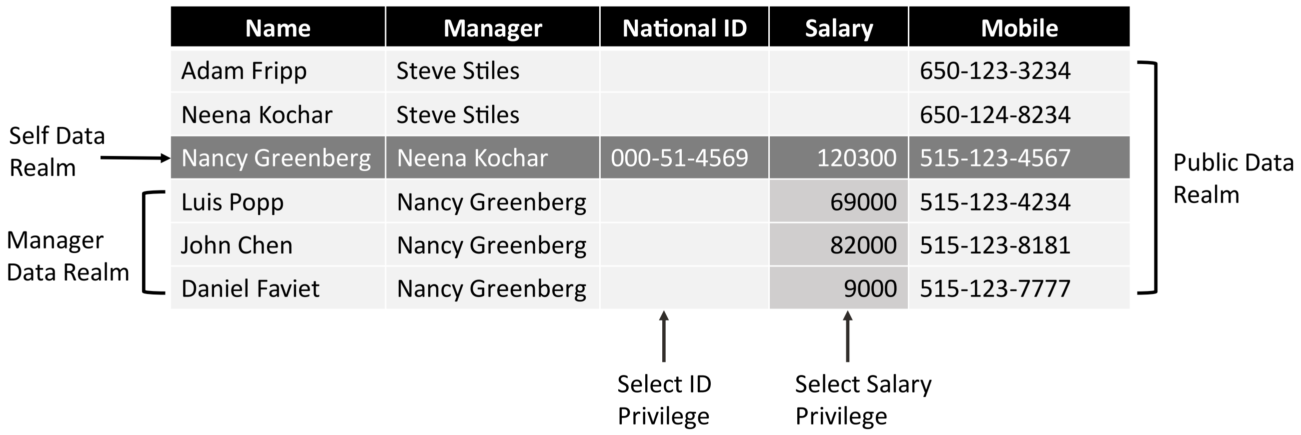
Real Application Security (RAS) was introduced in Oracle Database 12c. It provides the next generation of application access control frameworks within the database. RAS enables three-tier and two-tier applications to declaratively define, provision, and enforce access control requirements in the database layer. At the same time, RAS provides an easier-to-manage framework for data-driven access control that can replace VPD in almost all use cases.

RAS uses a policy-based authorization model that recognizes application-level users, privileges, and roles within the database and then controls access to static and dynamic collections of records representing business objects. RAS overcomes the maintainability, scalability, and security issues complex VPD policies face and supports common application patterns like master-detail tables and temporary assignments.

With RAS, the identity of the application end user is securely propagated to the database. Like VPD or OLS, RAS policies enforce access control policies within the database, regardless of which application accesses the data. The application can create any number of application user sessions and switch between them while using a single connection from a connection pool to the database. Note that the application user does not have its own schema to store data objects or a dedicated connection to the database as a regular database user. RAS can also enforce access control policies on database users.

RAS enforces fine-grained restrictions on access to both columns and rows within a database table, just like VPD. However, RAS specifies these restrictions using a more general, declarative syntax. First, the administrator creates one or more “data realms” to protect rows of data within a table. Each data realm identifies an applicable subset of the rows within the table using the same syntax as the `WHERE` clause in a SQL query. Then, an access control list (ACL) is attached to each data realm to identify which users or roles have permission to perform which operations on the data within that data realm.

The example in Figure 9-8 below shows three different data realms, all for rows within the same table. The public data realm includes all employee records. The self data realm is dynamic (it protects different rows for different users) and only consists of the user’s record. The manager data realm is also dynamic and includes all employees that report to the user.



Name	Manager	National ID	Salary	Mobile
Adam Fripp	Steve Stiles			650-123-3234
Neena Kochar	Steve Stiles			650-124-8234
Nancy Greenberg	Neena Kochar	000-51-4569	120300	515-123-4567
Luis Popp	Nancy Greenberg		69000	515-123-4234
John Chen	Nancy Greenberg		82000	515-123-8181
Daniel Faviet	Nancy Greenberg		9000	515-123-7777

Self Data Realm
Manager Data Realm
Public Data Realm
Select ID Privilege
Select Salary Privilege

Figure 9-8: Controlling data access with Real Application Security

For each data realm, an access control list (ACL) specifies who can access that data realm and under what conditions. For the ‘public’ data realm, all employees have only the ‘`SELECT`‘ privilege.

All non-protected data should be visible to any employee. Since `SSN` and `Salary` data are protected and not included in the ACL for the ‘public’ data realm, data from those columns does not appear when queried by someone with only public access. The ‘self’ data realm (see below) is associated with an ACL to view `SSN` (Authorize SSN) and `Salary` (Authorize Salary) data so an employee (role) can view their own sensitive data.

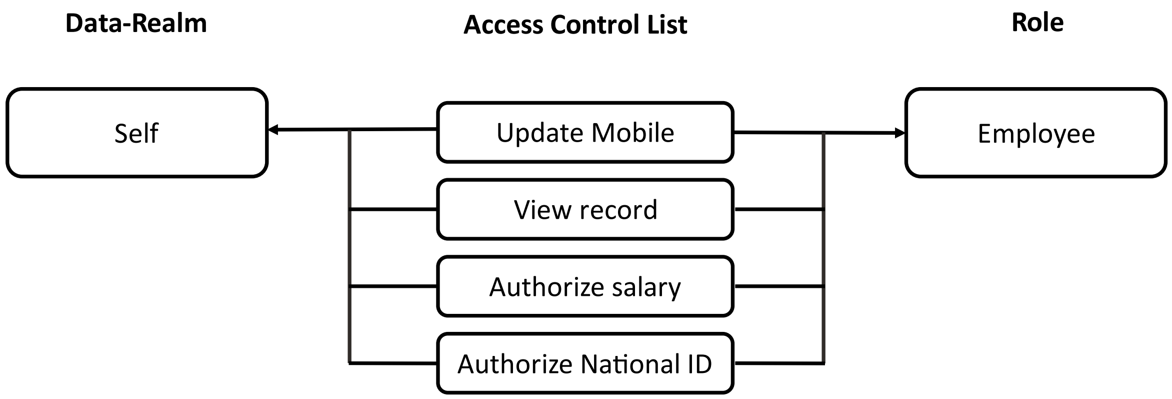


Figure 9-9: RAS access control list (ACL)

A manager can view the salary of their reports using the `manager` data realm, which has an ACL that allows the `salary` data to be shown. The `self` and `manager` data realms are dynamic—that is, RAS returns data from appropriate rows and columns based on the application user identity for that session. Furthermore, RAS allows the database to enforce additional security policies unique to each application. The application can define its privileges in addition to the usual `SELECT`, `INSERT`, `UPDATE`, and `DELETE` to represent operations that are specific to that application, such as vacation approval, check approval, and creating invoices. The administrator can specify that access to a column requires the user to have a particular application-defined privilege (i.e., `UPDATE_SALARY`, `VIEW_SSN`).

Connecting applications to the database through the RAS Java interface is more secure than traditional connections. A traditional application connection to the database needs to include every privilege and role that every application user needs to run the application, typically including those needed to install and maintain the application.

These highly privileged accounts are a top target for cybercriminals looking to break into a database. The RAS application connection to the database uses a minimally privileged account, so even if the account is compromised, attackers cannot use it to access sensitive data. RAS data realms, ACLs, and policy management can be entirely managed through the RAS management PL/SQL API.

With built-in support for propagating application user sessions to the database, RAS allows security policies on data to be expressed directly in terms of the application-defined users' roles and data operations. The RAS security model allows uniform specification and enforcement of access control policies on business objects irrespective of the access path. Using declarative access control policies on application data and operations, RAS enforces security close to the data and enables end-to-end security for both three-tier and two-tier applications. The declarative model for RAS is much simpler to maintain and extend than VPD.

Which data-driven access control should I use?

Virtual Private Database is the most straightforward access control feature to start using. However, when the VPD PL/SQL policies become complex, the VPD code to create the WHERE predicate clause becomes complex and, difficult to manage. If the access control policy is simple and will remain that way, VPD is a great choice.

Real Application Security is a more powerful data-drive access control. It is enterprise-ready with many enterprise patterns built in. It can serve as a direct replacement for VPD, working directly with database users, roles, and objects. If you are developing a new application, or have the ability to modify existing application code, RAS provides a robust data security framework that allows you to separate access control policies from the applications. All the application needs to do is invoke the RAS APIs to connect application users to the database session and set any needed context variables.

Label Security is an excellent choice because the code to adjudicate the label precedence is taken care of automatically. However, work is required to manage and maintain user and data labels. Both VPD and RAS provide row and column-level control, while Label Security only works at the row level.

Oracle LiveLabs

Oracle LiveLabs gives you access to Oracle's tools and technologies to run a wide variety of labs and workshops at your own pace. If you want to try out the technologies discussed in this chapter, please go to:

- Protecting row and column data with Virtual Private Database
- DB Security - Label Security

Summary

Hackers and malicious insiders can take advantage of weak application authorization policies if implemented inconsistently and improperly. Data-driven access controls like Label Security, Virtual Private Database, and Real Application Security let you centralize authorization policies within the database and ensure they are enforced consistently across different clients and applications. Centralized application authorization can also accelerate application development and reduce the complexity of maintaining and upgrading multiple authorization policies in different applications.

Chapter Ten

Masking Sensitive Data

Why is masking sensitive data important?

Applications manage large amounts of sensitive data, including PII, financial data, healthcare information, proprietary information, and more. Limiting the exposure of sensitive data to users who do not need to see it is a challenge many organizations face. The problem is that there are many cases where you want to leverage your application data outside of the application. For example, developing and testing new applications is much more efficient when the process can take advantage of some actual data. Likewise, marketing and customer service teams want to analyze application data to better understand customer needs and improve service delivery. Representatives in a call center can be much more effective in responding to customer inquiries if they can quickly access the customer's purchase history and previous customer interactions. In each of these cases, there are clear business benefits of using application data. The challenge is to enable these use cases without compromising sensitive information.

Data masking modifies sensitive data so that it is of no or little value to a data thief while remaining useful for your purposes. Masking data helps you reduce the risk of data breaches and protect the privacy of your customers and employees. Data masking helps you leverage the data you collect while protecting sensitive information such as credit card numbers, taxpayer identifiers, sales figures, and other personal or proprietary information. For example, testing a point-of-sale application may need real inventory records and store information, but not require the actual customer information (like names, loyalty numbers, and email addresses) contained in the application. Application testing can use fictitious but realistic customer information instead.

Why should you focus on masking sensitive data?

Most organizations create copies of their production databases for use in testing and development. These non-production copies may also be used to share data with other organizations (often for fraud analysis or other specialized analytics) or for training and user acceptance testing. Each new copy of a production database increases the risk that data within those databases will be compromised. Data Masking helps mitigate that risk.

This chapter discusses how static data masking can remove sensitive data from non-production environments. It further describes how data redaction prevents the proliferation of sensitive data to users in production environments. These tools are integral to a comprehensive data privacy strategy, helping you meet compliance requirements such as PCI-DSS and EU-GDPR.

What is data masking?

Data masking is a security control that hides sensitive data by replacing the original data with altered data. The altered data may be realistic looking, but it has zero or very little value to an attacker. Data masking may also be called redaction, scrambling, munging, or anonymization.

In most cases, the masked data must look real enough to remain and appear consistent across different tables, following referential integrity constraints. Masking differs from encryption because there is usually no intention or ability to unmask the data.

Data masking may be “static” – where the stored data is modified, changing it to remove security risk, or “dynamic” – where the stored data is not changed, but the presentation of the data to users is altered to hide some or all of the sensitive information.

- Static data masking is used in cases where sensitive data needs to be shared outside the organization (often for advanced analytics) or removed from non-production environments such as test, development, and training. With static data masking, all users who view the data see the masked data because unmasked data no longer exists.

- Dynamic data masking, also known as data redaction, prevents the proliferation of sensitive data to other systems by masking data in flight while keeping the original data intact. With data redaction, the result depends on the dynamic masking policy. Some users may see masked data, others may see unmasked data.

Oracle supports both static and dynamic masking:

- Static masking is done with Oracle Data Safe or Oracle Data Masking and Subsetting
- Dynamic masking is done with Data Redaction (part of Oracle Advanced Security), Virtual Private Database, or Real Application Security

These tools are integral to a comprehensive data privacy strategy. They reduce the risk of data theft or misuse and help you meet compliance requirements such as PCI-DSS and GDPR.

Use cases for static data masking

Common use cases include:

- Reduce risk and regulatory compliance scope by minimizing the amount of sensitive data you store: By masking sensitive personal data, you reduce the scope of a potential data breach (less data in the system means less data to steal). You may also be able to remove entire systems from the scope of privacy regulations such as GDPR and CCPA.
- Remove sensitive data from development and testing environments: In non-production environments, developers and testers must access realistic data to ensure that applications and systems work as intended. Data masking allows them to use realistic data without exposing sensitive information. Many regulations forbid the exposure of sensitive information in non-production environments.
- Save time and money: By cloning a production database and masking sensitive data, you can reduce the need to create and maintain test and development environments with purely artificial data sets.
- Data sharing and third-party analytics: When you outsource certain tasks to third-party vendors or partners, data masking safeguards the privacy and security of sensitive data. Reversible masking may provide a data set for analysis without exposing data subject information – this use case is prevalent in healthcare and financial service fraud analytics but exists in many other organizations.
- Training and education: Data masking in educational settings or training programs allows students or trainees to work with real-world volumes of data without exposing private information.

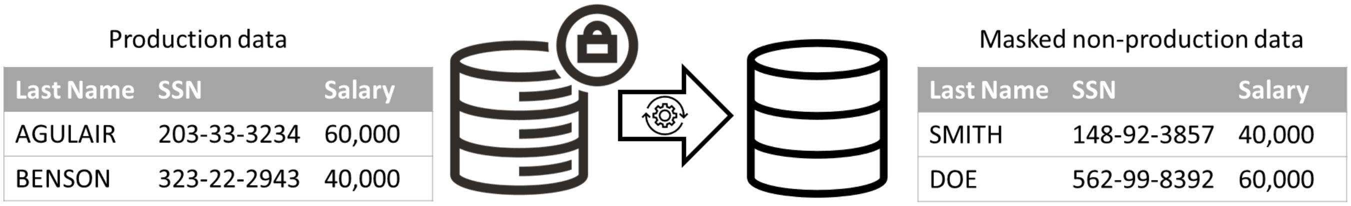
Use cases for data redaction

- Control access to specific data elements: Redact data as it is delivered to database clients so that users of those clients cannot view data they do not have a business need to see. Examples might include blocking access to sensitive columns for analytics tools or retrieval-augmented generative AI.
- Minimize access to certain data elements: Redact data as it is delivered to clients if conditions do not match organizational policy. For example, prevent viewing certain sensitive columns outside of approved network segments or outside normal working hours.
- Support training and education: Data redaction in educational settings or training programs allows students or trainees to work with real-world data without exposing private information.

Static data masking

Static data masking creates masked datasets where the original data no longer exists on that system. It doesn't matter who attempts to view the masked data; it is masked for all accounts. This type of masking is typically used for testing and development purposes. You can use static data masking to selectively replace sensitive data in the database while retaining application integrity.

Databases masked in this way can be shared with test, development, and business analytics teams while minimizing exposure of potentially sensitive data. Figure 10-1 shows an example of static data masking. In this case, masking replaces names from the production database with realistic but fake names and production social security numbers with random values formatted to look like social security numbers. Salary information is shuffled so that overall payroll remains constant, but actual pay for any given record is no longer real data. The production database contains sensitive information and needs to be secured appropriately. The non-production database has less risk inherent in its data and, therefore, can accept a lower level of security.



Production data

Last Name	SSN	Salary
AGULAIR	203-33-3234	60,000
BENSON	323-22-2943	40,000

Masked non-production data

Last Name	SSN	Salary
SMITH	148-92-3857	40,000
DOE	562-99-8392	60,000

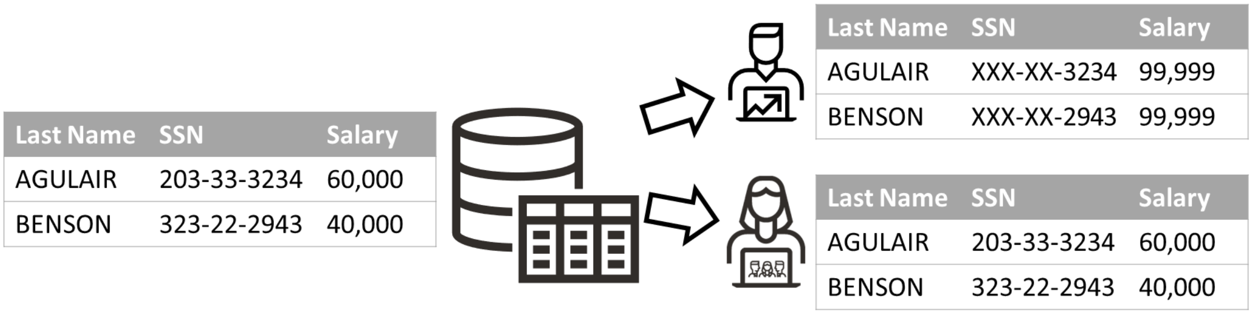
Figure 10-1: Static data masking

Static data masking is performed by a service external to the database, such as Oracle Data Safe or Oracle Data Masking and Subsetting.

Data redaction

Data redaction addresses use cases where we need to mask sensitive data for some users but display the original, unaltered data to others. Here, the data is masked on the fly when data is retrieved by a specific user but without changing the stored data. Data redaction limits the exposure of sensitive data within application user interfaces and helps reduce disclosure risks. This type of masking is typically used for production environments.

Take the case shown in Figure 10-2 where two different database users, one a data analyst and the other a human resources partner, are both accessing the same data from the same table. The data analyst needs to know the real last name of employees but does not need to know the full SSN or salary. Within the database, redaction policies hide the first portion of the SSN, and convert the salary to a fixed number. At the same time, the human resources partner, who is authorized by policy to view full data on an employee, can see the entire SSN and the actual salary. Unlike static masking, data within the table is not actually changed.



Last Name	SSN	Salary
AGULAIR	203-33-3234	60,000
BENSON	323-22-2943	40,000

Last Name	SSN	Salary
AGULAIR	XXX-XX-3234	99,999
BENSON	XXX-XX-2943	99,999

Last Name	SSN	Salary
AGULAIR	203-33-3234	60,000
BENSON	323-22-2943	40,000

Figure 10-2: Data redaction

The database performs data redaction at runtime (dynamically) using Data Redaction (part of Oracle Advanced Security), Virtual Private Database, or Real Application Security.

Data subsetting

Data subsetting helps with data minimization requirements and reduces risk in non-production database copies. If the entire data set is not required for test and development, removing unneeded data reduces the amount of data that could be stolen if the test or development environment were compromised, with the additional benefit of reducing the cost to store the database copy. Subsetting extracts a portion or subset of data instead of sharing the whole production dataset. For example, the analytics team might only need 20% of the production data, data collected over the last year, or data specific to a region. Data subsetting provides relevant data and reduces security risks.

Together, data masking, redaction, and subsetting limit hackers' avenues to steal sensitive data, reduce the impact of a database compromise, and help facilitate compliance with privacy regulations such as PCI-DSS and the EU GDPR.

Masking formats

Masking formats define how to transform the original data to create an anonymized and sanitized dataset during the masking process. Oracle provides a comprehensive set of both simple and predefined masking formats. There are simple masking formats, which are basic transformations applied to data, and predefined masking formats which are supplied with the masking tool to help you solve common use cases.

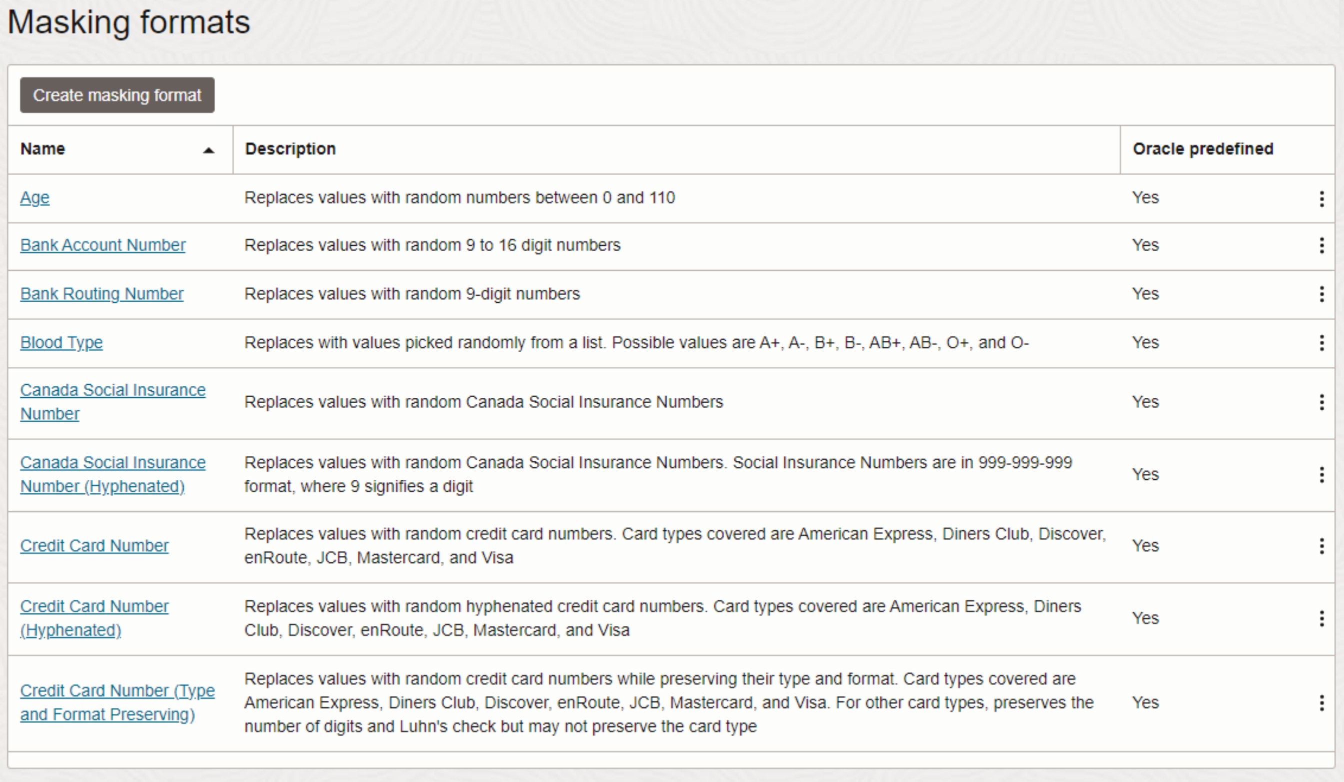
Simple formats are basic transformations that can be applied to achieve a desired output. Examples of simple formats include:

Masking format	Description
Fixed value	Replaces original values with a fixed value. e.g., real email address replaced with fake@no.co
Random values	Replaces original values with random values, usually within some specified range
Random list	Replaces values in a column using a list or array of values
Substitution	Replaces values in a column with values selected from another table
Encryption	Applies a cryptographic algorithm to mask the data. This type of masking is often used when there is a need to later reverse the masking (e.g., data shared with a third party for analysis)
Format preserving randomization	Randomizes data while preserving its format. Replaces letter with letter and digit with a digit but keeps the data length, special characters, and the case (upper or lower) of characters. This masking technique is often used for data elements like national identifiers, postal codes, and license plate numbers
Shuffle	Randomly reorders the values within a column so that statistical relevance is maintained. This masking technique might be applied to columns like age or gender
SQL expression	Uses a user-supplied SQL expression to generate masked values to replace the original values. For example, email addresses can be generated using values from columns containing first and last names. One benefit of SQL expression masking is that the masking format is applied AFTER other columns are masked. So, if you mask first and last name, you could generate an email address in the form first_name.last_name@company.com while maintaining consistency between the name and email columns. This masking technique may also be used to mask data contained in large objects (LOBs), including BLOBs, CLOBs, and NCLOBs

Table 10-1: Simple masking formats

It is common to apply multiple basic transformations to a single column, grouping them to create masked data that follows the formatting and syntax rules needed for your organization. For example, you might use a four-digit random number between 1 and 9999, concatenated with a random list of common road names like MAIN, ELM, BROADWAY… (you get the idea), concatenated in turn with a random array of street types like ST, RD, AVE, BLVD, CIR to construct fake street addresses that look realistic.

In addition to simple masking formats, there are predefined formats for common types of sensitive data, such as credit card numbers, telephone numbers, social security numbers, and other national identifiers. Predefined formats usually combine one or more simple formats, often with some sort of processing to format the data to look more realistic. Figure 10-3 shows a few of the more than 50 predefined formats available in Oracle Data Safe. Predefined formats simplify the design of your masking process.



Masking formats

Name	Description	Oracle predefined
Age	Replaces values with random numbers between 0 and 110	Yes
Bank Account Number	Replaces values with random 9 to 16 digit numbers	Yes
Bank Routing Number	Replaces values with random 9-digit numbers	Yes
Blood Type	Replaces with values picked randomly from a list. Possible values are A+, A-, B+, B-, AB+, AB-, O+, and O-	Yes
Canada Social Insurance Number	Replaces values with random Canada Social Insurance Numbers	Yes
Canada Social Insurance Number (Hyphenated)	Replaces values with random Canada Social Insurance Numbers. Social Insurance Numbers are in 999-999-999 format, where 9 signifies a digit	Yes
Credit Card Number	Replaces values with random credit card numbers. Card types covered are American Express, Diners Club, Discover, enRoute, JCB, Mastercard, and Visa	Yes
Credit Card Number (Hyphenated)	Replaces values with random hyphenated credit card numbers. Card types covered are American Express, Diners Club, Discover, enRoute, JCB, Mastercard, and Visa	Yes
Credit Card Number (Type and Format Preserving)	Replaces values with random credit card numbers while preserving their type and format. Card types covered are American Express, Diners Club, Discover, enRoute, JCB, Mastercard, and Visa. For other card types, preserves the number of digits and Luhn's check but may not preserve the card type	Yes

Figure 10-3: Predefined masking formats in Oracle Data Safe

Using the masking formats library, you can easily apply data masking rules to sensitive data in databases and provide usable data sets for testing, analysis, and development.

You are not limited to the predefined formats Oracle provides. You can create new masking formats based on combinations of existing formats. For example, you may have a data element called an account number. There is a reasonable chance that your account number format is unique to your organization – if so, the predefined formats won't give you test account numbers that look realistic to your users. So, you would create a custom format that generates account numbers based on your needs and then reuse that custom format as you mask data.

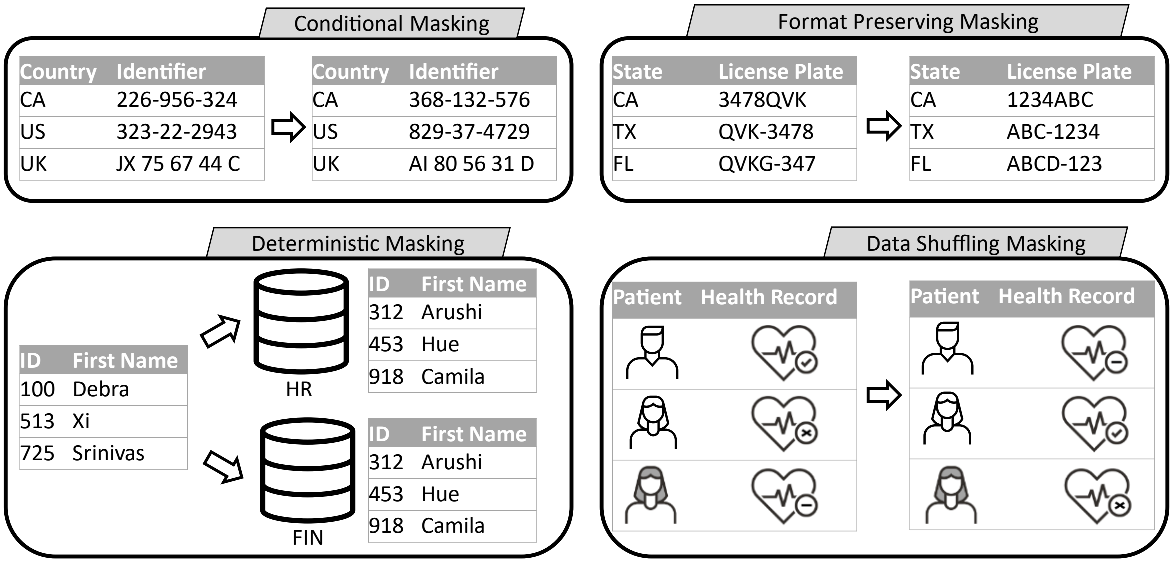
Masking techniques

The masking process can use different techniques for transforming data sets, enabling complex applications to operate with masked data in various non-production environments. Table 10-2 describes some masking techniques.

Masking technique	Description
Conditional masking	Applies different masking formats to different rows based on a specified condition. For example, data about citizens from multiple countries can have unique masked national identifiers based on their country. Similarly, credit card numbers can be masked while preserving their original type and format.
Compound masking	Masks related data contained in multiple columns as a group. For example, mask city, state, and zip code together to correlate the values so you don't place New York City in Texas.
Deterministic masking	Given the same input value, deterministic masking will produce the same output value. Deterministic masking masks data to the same consistent value across multiple databases, applications, or masking runs. It lets you maintain referential integrity in a multi-database test environment.
Reversible masking	Encrypts and decrypts sensitive data using a cryptographic key. This masking technique is helpful in scenarios where you want to retrieve the original data from the masked data. For example, you might have to share masked data with a third party for some business processing and want to recover the original data after receiving processed data.

Table 10-2: Masking techniques

Figure 10-4 shows some of the supported masking formats and techniques in action. For example, conditional masking produces different masked values for national identifiers using conditions based on country codes, random strings replace license plate numbers while preserving the original data format, deterministic masking consistently applies masked values for employee names across databases, and masking runs, and data shuffling disassociates health records from patient identities while still preserving complex data types like diagnostic images.



Country	Identifier
CA	226-956-324
US	323-22-2943
UK	JX 75 67 44 C

Country	Identifier
CA	368-132-576
US	829-37-4729
UK	AI 80 56 31 D

State	License Plate
CA	3478QVK
TX	QVK-3478
FL	QVKG-347

State	License Plate
CA	1234ABC
TX	ABC-1234
FL	ABCD-123

ID	First Name
100	Debra
513	Xi
725	Srinivas

ID	First Name
312	Arushi
453	Hue
918	Camila

ID	First Name
312	Arushi
453	Hue
918	Camila

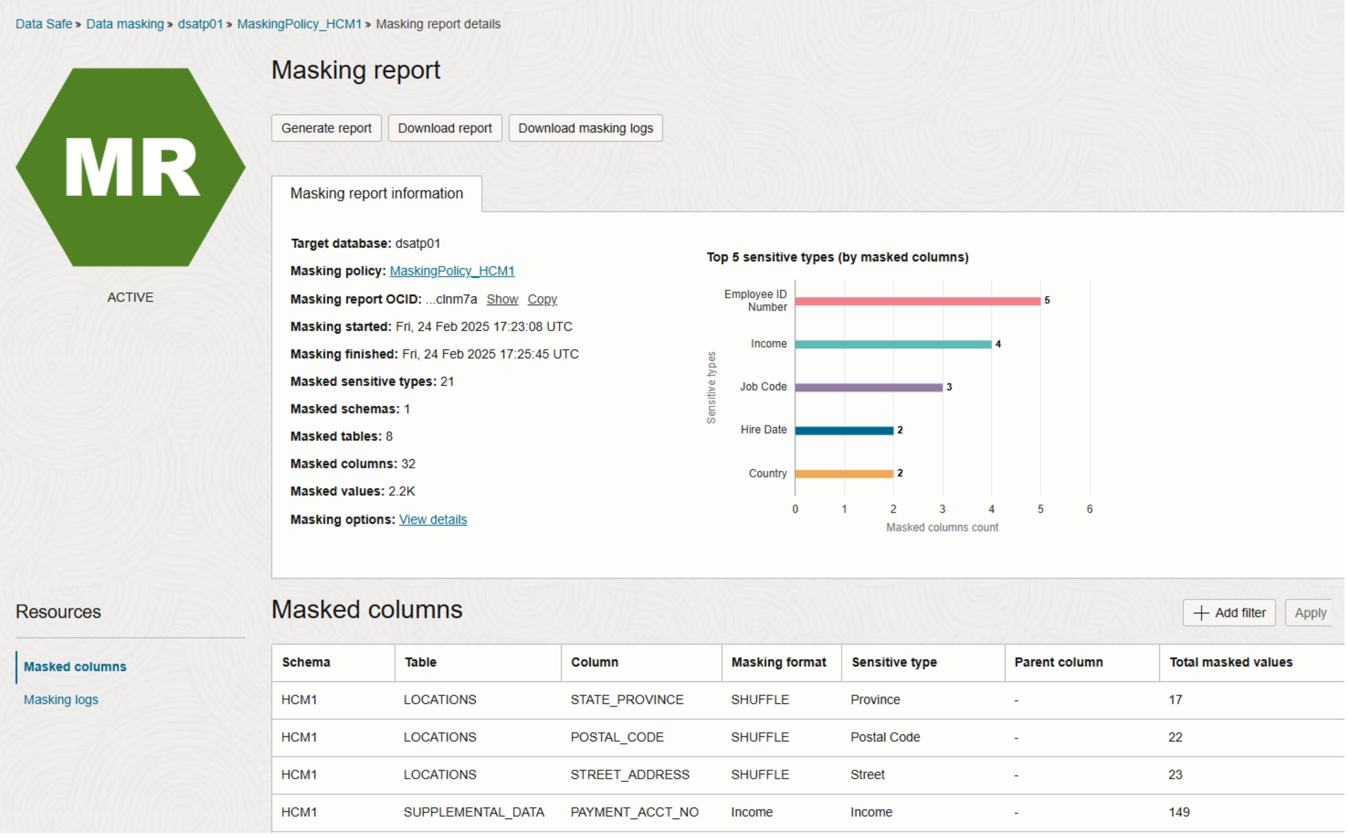
Figure 10-4: Masking formats and techniques in action

Masking sensitive data using Oracle Data Safe

Data Safe's static data masking feature allows you to quickly mask sensitive application data with a rich set of simple masking formats and a library of over 50 predefined formats. To mask a data column, select the desired masking format from a drop-down list. If you used the data discovery feature to locate sensitive data, Data Safe automatically suggests default masking formats based on the type of sensitive data discovered. Data Safe users can also add their own custom masking formats to the masking library.

Masking reports

The Data Safe console summarizes the results of masking runs in a report containing information about the database and masking policy applied, when the masking job was run, and statistics showing which columns were masked, how they were masked, and the volume of data that was masked. This information can be downloaded and saved as a PDF file to assist with reporting and compliance.



Masking report

Target database: dsatp01

Masking policy: MaskingPolicy_HCM1

Masking report OCID: ...clnm7a Show Copy

Masking started: Fri, 24 Feb 2025 17:23:08 UTC

Masking finished: Fri, 24 Feb 2025 17:25:45 UTC

Masked sensitive types: 21

Masked schemas: 1

Masked tables: 8

Masked columns: 32

Masked values: 2.2K

Masking options: View details

Masked columns

Schema	Table	Column	Masking format	Sensitive type	Parent column	Total masked values
HCM1	LOCATIONS	STATE_PROVINCE	SHUFFLE	Province	-	17
HCM1	LOCATIONS	POSTAL_CODE	SHUFFLE	Postal Code	-	22
HCM1	LOCATIONS	STREET_ADDRESS	SHUFFLE	Street	-	23
HCM1	SUPPLEMENTAL_DATA	PAYMENT_ACCT_NO	Income	Income	-	149

Figure 10-5: Data Safe masking reports summarize key information from masking jobs

Masking sensitive data using Oracle Data Masking and Subsetting

Oracle Data Masking and Subsetting is an Enterprise Manager management pack that statically masks data and can also create subsets of data.

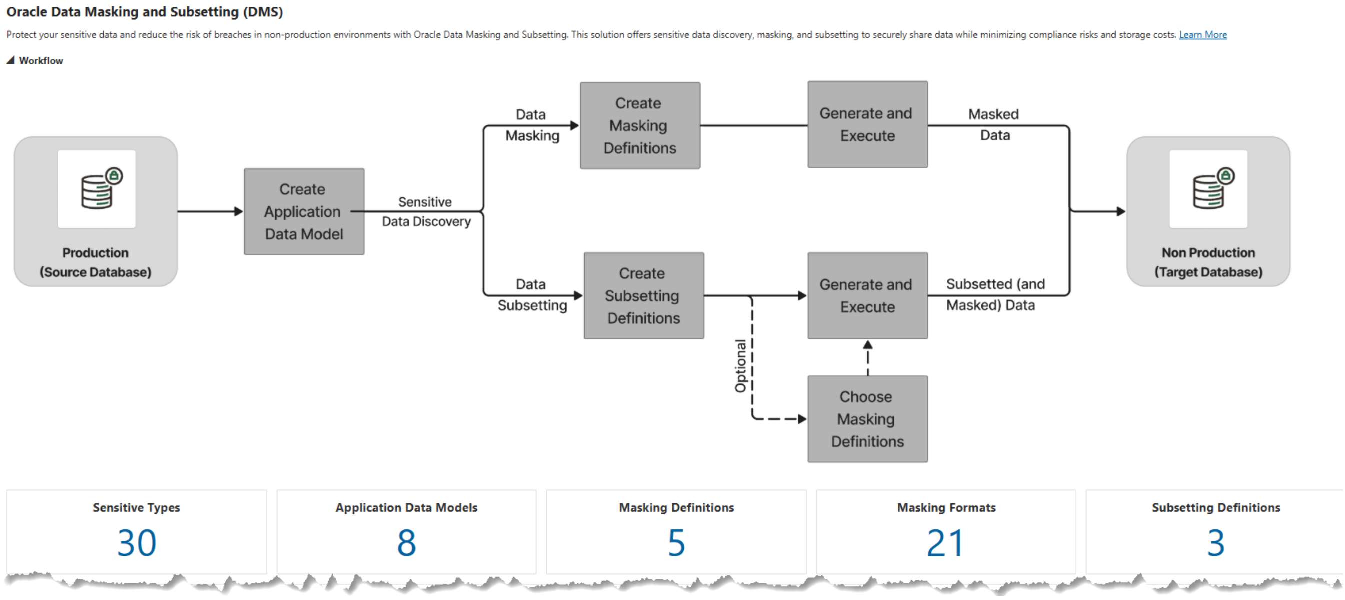
Data masking

Masking data with Data Masking and Subsetting follows a process similar to the one used in Data Safe. The first step uses data discovery to create an application data model that identifies sensitive columns and referential relationships.

Data discovery is a mandatory step with Data Masking and Subsetting. Only sensitive columns identified through data discovery are eligible for masking. We discussed data discovery earlier in Chapter Four.

Once your sensitive columns are discovered, you create a masking definition, assigning masking formats to the sensitive columns that should be masked. Like Data Safe, Data Masking and Subsetting includes both simple and predefined masking formats and supports creating custom masking formats.

To apply the masking definition, you generate a masking script and execute it within the Enterprise Manager framework or independently. Figure 10-6, taken from the Data Masking and Subsetting console, gives an overview of the process flow for both masking and subsetting.



Oracle Data Masking and Subsetting (DMS)

Protect your sensitive data and reduce the risk of breaches in non-production environments with Oracle Data Masking and Subsetting. This solution offers sensitive data discovery, masking, and subsetting to securely share data while minimizing compliance risks and storage costs. Learn More

◢ Workflow

Sensitive Types	Application Data Models	Masking Definitions	Masking Formats	Subsetting Definitions
30	8	5	21	3

Figure 10-6: Overview of Data Masking in Enterprise Manager

In-database and in-export data masking modes

Oracle Data Masking and Subsetting offers two modes for data masking: in-database and in-export. The different modes give you flexibility in executing masking jobs. Figure 10-7 shows the difference in the two modes.

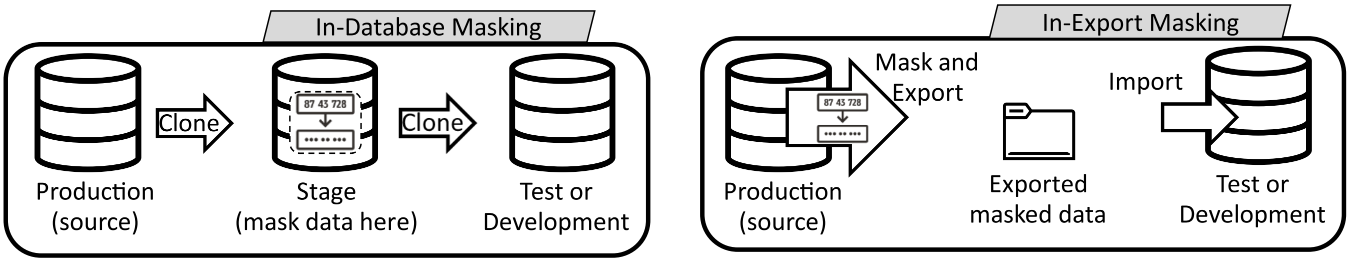


Figure 10-7: Masking modes for Data Masking and Subsetting

The in-database mode enables masking and subsetting data within a non-production database (usually a clone of production) with minimal or no impact on the production environment. This mode has you create an unmasked staging copy of the production database, mask and subset the data within the staging copy, and then create new database clones based on the staging copy as needed for test, development, or data sharing. Remember, data masing irreversibly modifies the data stored in the database and should not normally be performed on a production database.

Alternatively, Data Masking and Subsetting users can use the in-export mode to mask and subset data on the fly during a Data Pump export. In-export masks data directly from the production database while the data is being exported, leaving the original values untouched in the production database. The masked and exported data are in standard Data Pump files, which can then be imported into non-production databases or shared for analytics. This mode eliminates the need to create a production database clone on a staging server and ensures that sensitive data remains only within the production perimeter. With in-export masking, the masking formats applied should be reasonably simple, and the masking and export process should run outside of peak usage hours.

Differences between Data Masking and Subsetting & Data Safe

There are a few differences between masking with Data Safe and masking with Data Masking and Subsetting.

1. Data Masking and Subsetting only works with enterprise edition databases, while Data Safe can mask any Oracle Database, including standard edition.
2. Data Masking and Subsetting requires you to manually associate each column with the masking format you want to use, while Data Safe automatically associates a masking format with a sensitive data type, creating a masking policy in a few mouse clicks.
3. Data Masking and Subsetting can create a masked data pump export directly from production databases. Masking during export limits the complexity of the masking formats but is a great way to extract a pre-masked dataset to share with offsite developers or analysts.
4. Data Masking and Subsetting allows you to download the masking script and run it independently of the Enterprise Manager framework, simplifying the orchestration of test and development databases. Data Safe does not allow for the separation of the masking execution from the service.

Masking data dynamically

Oracle Database can redact (dynamically mask) data as results are delivered in response to a query. The decision to mask or not mask is based on factors such as the program used by a user, time of day, roles enabled within a session, or the IP address the session originates from. Unlike static data masking, redaction does not change the underlying data.

Use cases for redaction

There are a few common use cases for redaction:

- Preventing the proliferation of sensitive data: Business analytics tools or reports run through direct SQL client access seldom need the most sensitive data elements. For example, when reporting on an organization's customers, there is seldom a need to display credit card information or birth dates. Redaction can mask the most sensitive columns from the report so that the analysts and report readers don't get access to that sensitive data. Another example might be in artificial intelligence applications using retrieval augmentation of relational data.
- Controlling the display of sensitive information within applications: Legacy applications may have been created during a less stringent regulatory environment. A classic example is frequently found in call-center applications that display personal data that is now considered sensitive but wasn't when the application was developed. Those applications may still show sensitive data like email address, date of birth, or even taxpayer identifiers even though the users of the applications have no business need to see the data. It may be impossible or impractical to update the application screens, but data redaction can remove or transform sensitive data during display time to minimize the risk of misuse.

A commonly used technique for restricting sensitive data displayed in an application is to alter the application's access control logic to redact sensitive data. Redaction provides the option to obscure data differently by replacing it with random data, applying special characters to a portion of retrieved data (like the first twelve digits of a credit card number or the username portion of an email address), or hiding it entirely. Implementing redaction within application code can be hard to maintain, and developers must adhere to strict controls for any new application development to ensure that redaction is consistently applied. Redaction approaches implemented with custom application logic can result in inconsistent solutions across the enterprise, especially in consolidated environments where multiple applications access the same data. Moreover, you may not have access to the source code to implement redaction, especially in the case of packaged applications.

A better approach to controlling the exposure of sensitive data in applications is to apply redaction policies within the database before data is returned to the application.

Redaction in Oracle Database

There are several ways to mask sensitive columns dynamically within an Oracle Database:

- Database views: A database view can hide columns the user should not see by omitting them from the view or by using a function to transform a sensitive column into a less sensitive value.
- Virtual Private Database (VPD): While the most common use case for VPD is row-level security, VPD also can dynamically mask columns, replacing the original value with a NULL value. With VPD, the decision to mask or not mask can be based on session context and data values.
- Real Application Security (RAS): RAS is the next generation of VPD and allows for both row and column filtering. Like VPD, RAS is limited to replacing original values with a NULL. With RAS, the decision to mask or not mask can be based on session context and data values.
- Data Redaction (a feature of Oracle Advanced Security): This provides the greatest flexibility in masking formats. With data redaction, the decision to mask or not mask cannot consider the data values being returned.

Data Redaction

Data redaction is one of the most flexible ways to dynamically mask data. It does not alter data stored in the database but instead masks the information on the fly ("dynamically") before it's returned to the user following a query. Data redaction supports the most commonly used column data types and various database objects, including tables, views, and materialized views. The redacted values retain key characteristics of the original data, such as the data type and optional formatting characters.

The strength of data redaction lies in its efficient transformation and execution inside the database kernel, declarative policy conditions, and transparency. Unlike masking approaches that rely on application programming, data redaction policies are applied directly inside Oracle Database, ensuring consistent enforcement across application modules and delivering more robust security.

Oracle Database's data redaction is transparent to applications because it preserves the original data type and formatting when sending the transformed data to the application.

Data redaction is also transparent to the database since it doesn't alter data in the database buffers, caches, or storage. For database transparency, data redaction does not affect administrative tasks such as moving data using Oracle Data Pump or backup and restore of the database using Oracle Recovery Manager. It also does not interfere with advanced database configurations such as Oracle Real Application Clusters, Oracle Active Data Guard, and Oracle GoldenGate.

Data Redaction is part of Oracle Advanced Security, a database option. No installation is required to deploy data redaction; you merely need to configure redaction policies.

Data Redaction Policies

Data Redaction policies at the database table and column level are created with the `DBMS_REDACT` PL/SQL package. Redaction policies specify:

- Schema, object, and column to be redacted
- Format of redaction and any parameters associated with that format
- Enforcement expression

Figure 10-8 shows an example of creating a data redaction policy. In this example, the policy says that first five characters of the SSN column of the CUST_INFO table in the HR schema will be redacted whenever 1=1 (in other words, always).

```
DBMS_REDACT.ADD_POLICY(
   object_schema       => 'HR',
   object_name         => 'CUST_INFO',
   column_name         => 'SSN',
   policy_name         => 'REDACT_CUST_SSN',
   function_type       => DBMS_REDACT.PARTIAL,
   function_parameters => DBMS_REDACT.REDACT_US_SSN_F5,
   expression          => '1=1')
```

Figure 10-8: Data Redaction policy creation example for partial redaction

The enforcement expression allows you to specify a condition when you should be dynamically redacting the data. For example, suppose the call center application is submitting a query. In that case, the redaction policy can use that session context to decide whether to redact or send the original data. In the above example, the Social Security column is always read since the expression 1=1 always evaluates to TRUE.

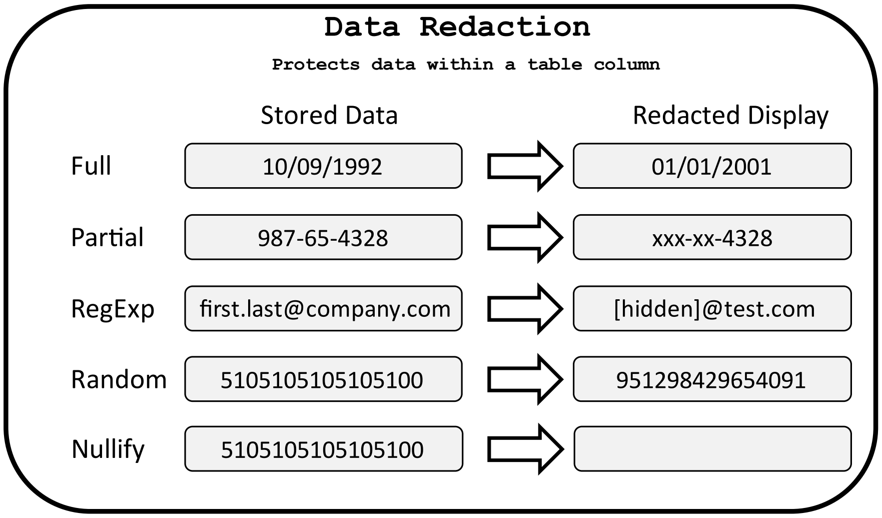
Data Redaction policies can consider factors available from the database and application in deciding whether data should be redacted. These factors include usernames, client identifiers, database roles, and session information, including client IP addresses and program modules. Policies may also use context information available from Oracle Application Express (APEX), Oracle Real Application Security (RAS), and Oracle Label Security (OLS). Policies may combine multiple execution conditions for precise control over when data is or is not redacted.

If you manage your databases with Oracle Enterprise Manager, you can use the built-in Redaction Policy Expression Builder. It guides you through creating policy conditions using one of the supported contexts. Oracle Enterprise Manager provides an easy-to-use interface for creating and applying redaction policies, allowing you to specify protected columns, transformation types, and conditions.

Data Redaction formats

Data Redaction policies allow different types of masking depending on the business needs:

- Full masking, where all sensitive data is masked.
- Partial masking, where only a portion of the data is masked.
- Rule-based masking, where data is masked based on rules defined by regular expressions.
- Random masking, where sensitive data is replaced with random data.
- Nullify, where the sensitive data is replaced with a NULL value



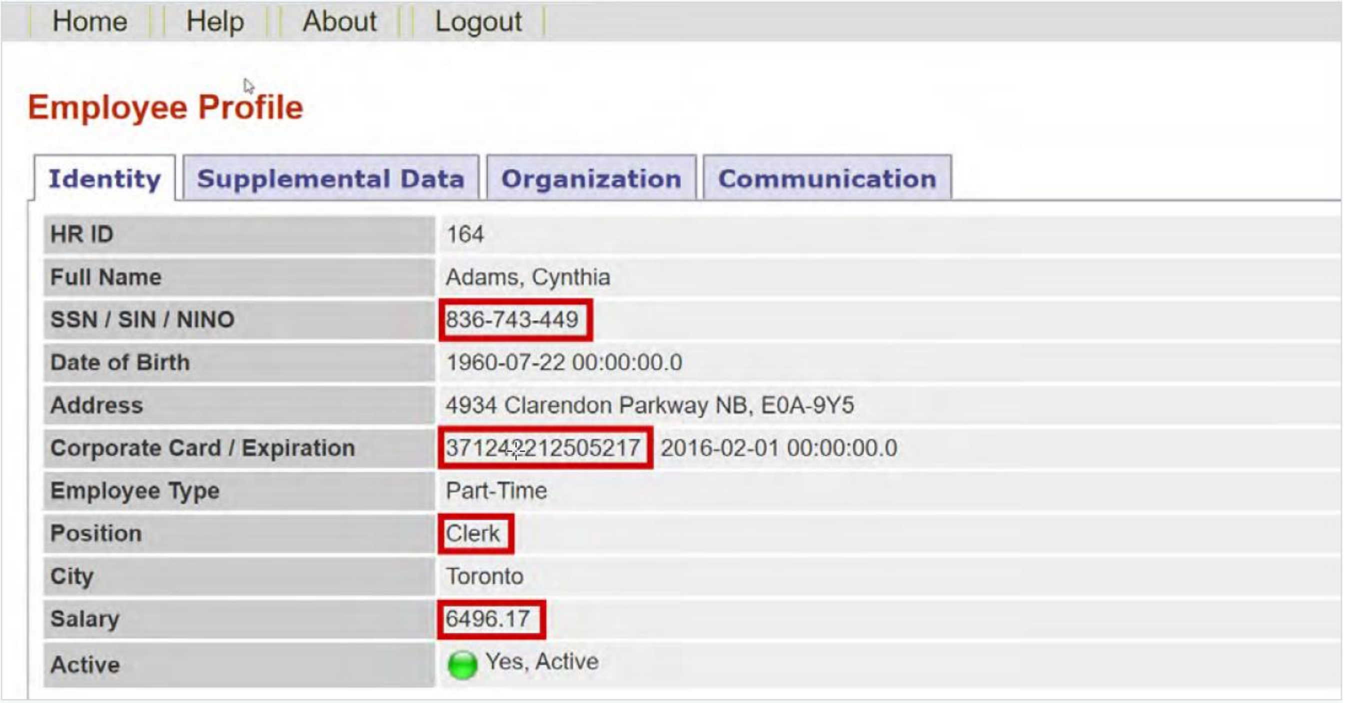
Data Redaction

Protects data within a table column

	Stored Data	Redacted Display
Full	10/09/1992	01/01/2001
Partial	987-65-4328	xxx-xx-4328
RegExp	first.last@company.com	[hidden]@test.com
Random	5105105105105100	951298429654091
Nullify	5105105105105100	

Figure 10-9: Data Redaction formats

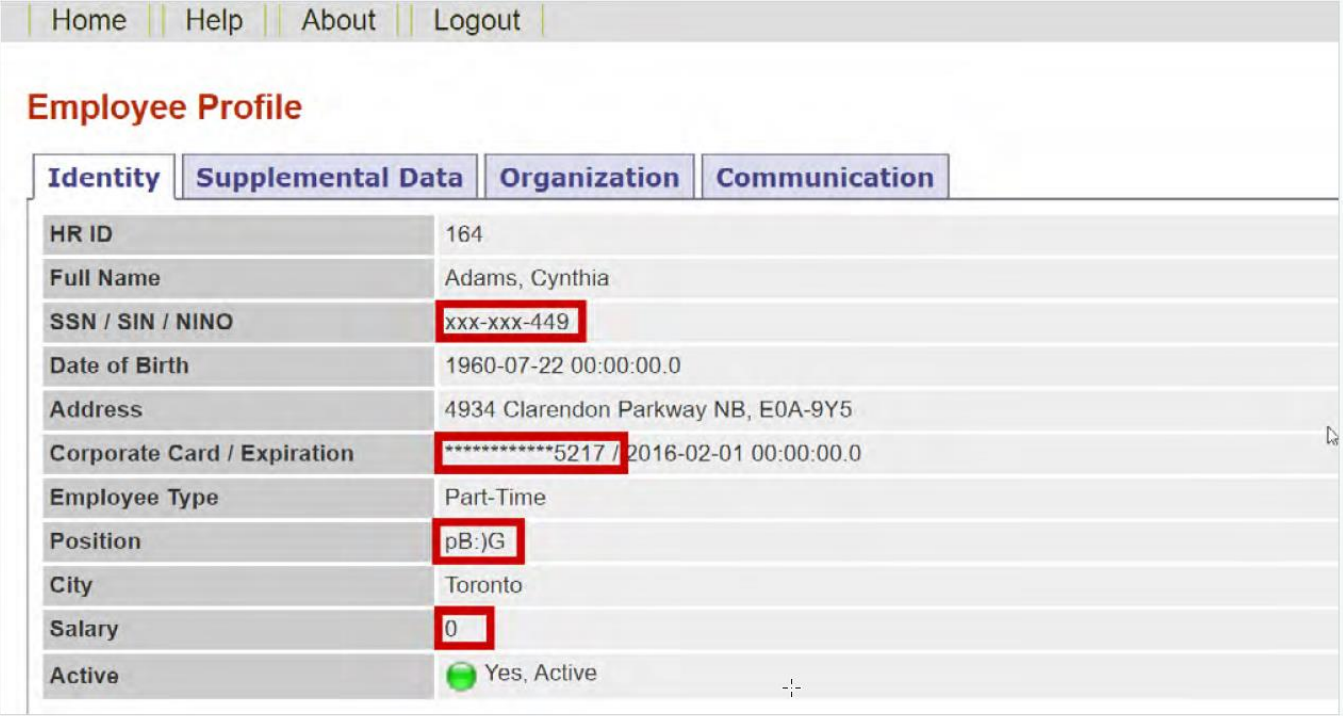
Figure 10-10 and Figure 10-11 help demonstrate how application users see the final data after data redaction. Figure 10-10 shows an HR application's original data without any data redaction. Figure 10-11 shows the redacted data in the SSN, Corporate Card, Position, and Salary fields using partial, regex, random, and full transformations.



Employee Profile

HR ID	164
Full Name	Adams, Cynthia
SSN / SIN / NINO	836-743-449
Date of Birth	1960-07-22 00:00:00.0
Address	4934 Clarendon Parkway NB, E0A-9Y5
Corporate Card / Expiration	371242212505217 2016-02-01 00:00:00.0
Employee Type	Part-Time
Position	Clerk
City	Toronto
Salary	6496.17
Active	● Yes, Active

Figure 10-10: Unredacted application data



HR ID	164
Full Name	Adams, Cynthia
SSN / SIN / NINO	xxx-xxx-449
Date of Birth	1960-07-22 00:00:00.0
Address	4934 Clarendon Parkway NB, E0A-9Y5
Corporate Card / Expiration	************5217 / 2016-02-01 00:00:00.0
Employee Type	Part-Time
Position	pB:)G
City	Toronto
Salary	0
Active	● Yes, Active

Figure 10-11: Redacted application data

In addition to simple formats like full redaction, nullify, partial redaction, and random values; Data Redaction supports predefined redaction formats available for many types of sensitive data, including credit card numbers, social security numbers, zip codes, email addresses, and phone numbers. Table 10-3 shows a few of the predefined redaction formats. Of course, you can also specify custom redaction formats to meet your specific requirements.

Format	Description
US SSN first five (formatted)	Redacts the first 5 numbers of Social Security numbers when the column is a VARCHAR2 data type. For example, the string 987-65-320 becomes XXX-XX-4320.
US SSN last four	Redacts the last 4 numbers of Social Security numbers when the column is a VARCHAR2 data type. For example, the string 987-65-4320 becomes 987-65-XXXX.
US SSN total (formatted)	Redacts the entire Social Security number when the column is a VARCHAR2 data type. For example, the number 987-65-4320 becomes XXX-XX-XXXX.
US SSN first five	Redacts the first 5 numbers of Social Security numbers when the column is a NUMBER data type. For example, the number 987654320 becomes XXXXX4320.
US SSN last four (formatted)	Redacts the last 4 numbers of Social Security numbers when the column is a NUMBER data type. For example, the number 987654320 becomes 98765XXXX.
US SSN total	Redacts the entire Social Security number when the column is a NUMBER data type. For example, the number 987654320 becomes XXXXXXXXX.
CA SIN six nines plus last three	Redacts the Canadian Social Insurance number by replacing the first 6 digits by 9 (number). For example, 123456789 is redacted to 999999789.
CA SIN last three	Redacts the Canadian Social Insurance number by replacing the first 6 digits by X (string). For example, 123456789 is redacted to XXXXXX789.

CA SIN last three (formatted)	Redacts the Canadian Social Insurance Number by replacing the first 6 digits by X (string). For example, 123-456-789 is redacted to XXX-XXX-789.
UK NIN alpha (formatted)	Redacts the UK National Insurance number by replacing the first 6 digits by X (string) but leaving the alphabetic characters as is. For example, ET 27 02 23 D is redacted to ET XX XX XX D.
UK NIN alpha	Redacts the UK National Insurance number by replacing the first 6 digits by X (string) and leaving the alphabetic characters as is. For example, ET270223D is redacted to ETXXXXXXD.
Credit Card last four (formatted)	Redacts the credit card number (other than American Express) by replacing everything but the last 4 digits by *. For example, the credit card number 5105–1051–0510–5100 is redacted to ****—****—****—5100.
Credit card last four	Redacts the credit card number (other than American Express) by replacing everything but the last 4 digits by 0. For example, the credit card number 5105105105105100 is redacted to ************5100.
AMEX last five (formatted)	Redacts the American Express credit card number by replacing the digits with * except the last 5 digits. For example, the credit card number 3782 822463 10005 is redacted to **** ****** 10005.
AMEX zeros plus last five	Redacts the American Express Credit Card Number by replacing the digits with 0 except the last 5 digits. For example, the credit card number 3782 822463 10005 is redacted to 0000 000000 10005.
US Zip code (varchar)	Redacts a 5-digit postal code when the column is a VARCHAR2 data type. For example, 95476 becomes XXXXX.
US Zip code (number)	Redacts a 5-digit postal code when the column is a NUMBER data type. For example, 95476 becomes XXXXX.
Date to 1970	Redacts dates to 01-JAN-1970.
Date to millennium	Redacts dates to 01-JAN-2000.
North America phone number (formatted)	Redacts the North American phone number by leaving the area code, but replacing everything else with X. For example, 650-555–0100 is redacted to 650-XXX-XXXX.
North America phone number (zeros)	Redacts the North American phone number by leaving the area code, but replacing everything else with 0. For example, 6505550100 gets redacted to 650000000.
North America phone number	Redacts the North American phone number by leaving the area code, but replacing everything else with X. For example, 6505550100 is redacted to 650XXXXXXX.

Table 10-3: Predefined redaction formats

Data Subsetting

The data subsetting component of Oracle Data Masking and Subsetting pack can create a reduced dataset derived from the original data while maintaining referential integrity. For instance, you could take a sizable dataset over 2TB and subset it down to a more manageable 100GB for development and analytics purposes. This smaller dataset enables more efficient operations while using fewer resources. As shown in Figure 10-12, data subsetting supports many different strategies for reducing the size of tables, such as extracting a fraction of a large table for application development, retrieving data from a specific region for analysis, or selecting rows from a particular partition or sub-partition of a table.

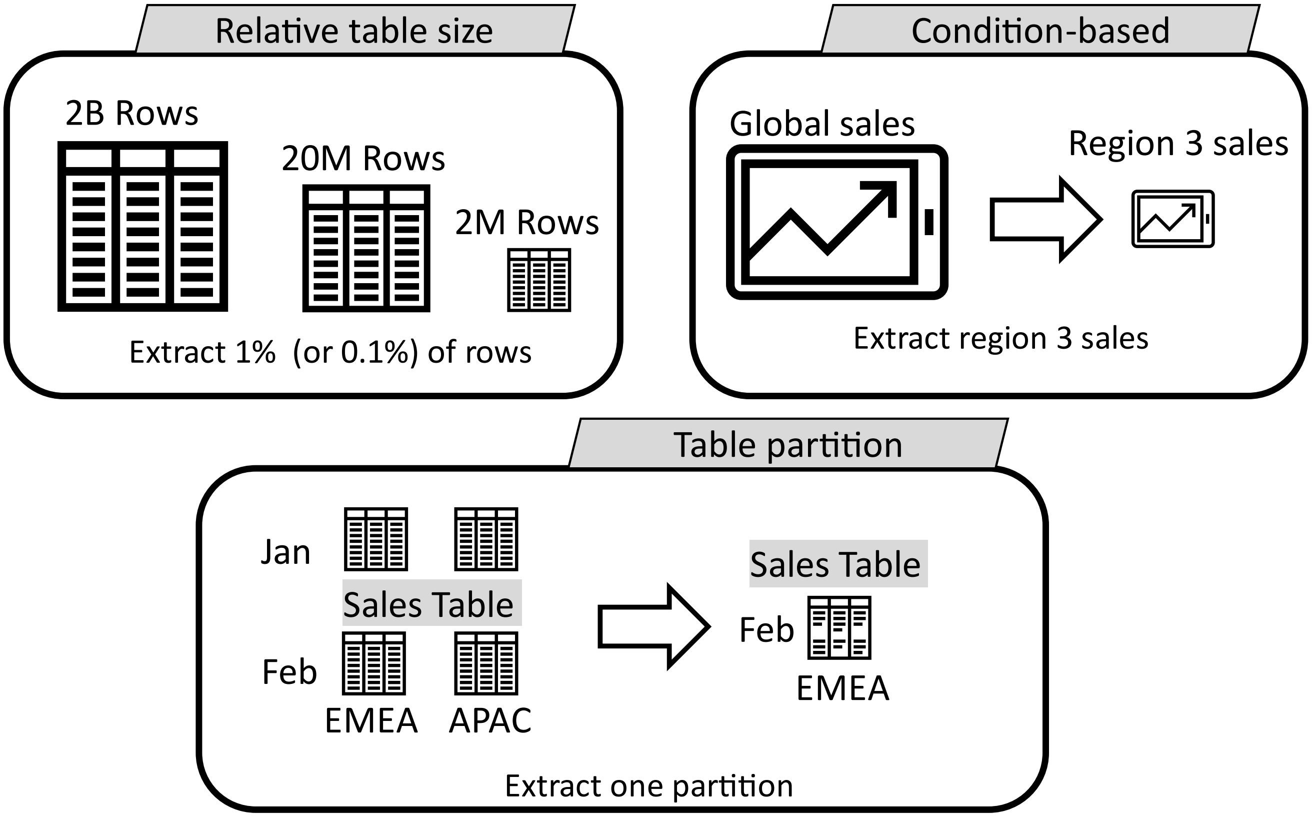


Figure 10-12: Data subsetting strategies

Table 10-4 lists different ways you can structure the conditions used to extract the data used to make up the subset.

Term	Description
Random Sampling	Randomly selecting a subset of data from a dataset. Randomness is crucial when you want to obtain a sample that is representative of the entire dataset, such as for development or testing purposes.
Stratified Sampling	When the dataset is diverse and contains different groups or 'strata,' stratified sampling ensures that each group is adequately represented. For stratified sampling, users can define rules for sampling different data groups.
Time-based Subsetting	You can subset data based on time criteria, such as selecting records from a particular date range. You may want the oldest or the latest data for analytics, for example.
Conditional Subsetting	Selecting data based on certain conditions or criteria. For example, you might extract records where a particular attribute meets a certain threshold.
Top-N Queries	Selecting a subset of data based on the top or bottom N records according to a particular attribute. For example, you might want to extract the top 1000 customers by sales volume.

Table 10-4: Subsetting conditions

Data Masking and Subsetting permits you to execute data subsetting immediately followed by masking. This combination first extracts some of the production data and subsequently masks the sensitive information within this subset for non-production applications. This approach maximizes the business value of production data without compromising sensitive information or wasting resources.

Oracle Data Masking and Subsetting provides comprehensive data subsetting capabilities, ensuring you can extract a representative sample of data while maintaining referential integrity and complying with security requirements. Whether you need to extract a fraction of a large table, focus on data belonging to a specific region, or work with partitioned data, Data Masking and Subsetting offers the flexibility needed to create smaller, more space-efficient databases.

Oracle LiveLabs

Oracle LiveLabs give you access to Oracle's solutions and technologies to run a wide variety of labs and workshops at your own pace. If you want to try the solutions discussed in this chapter, please go to:

- Get Started with Oracle Data Safe Fundamentals
- Data Redaction
- Redaction for Autonomous Databases
- Data Masking and Subsetting

Summary

Data masking, data subsetting, and data redaction are essential in protecting sensitive information within applications. Permanently masking data before copying it to non-production environments helps safeguard against potential threats such as a system breach or user compromise. Reducing the size of a data set through subsetting limits exposure and helps conserve resources. The real-time modification of data through data redaction enables safe display within an application without compromising stored data integrity.

These tools help enforce the principle of least privilege and serve as strategies for meeting anonymization and pseudonymization requirements of regulations like PCI-DSS and EU GDPR. In a world where data privacy and security are paramount, understanding and using these techniques is crucial for effectively handling sensitive data.

Chapter 11

Data Encryption and Key Management

Why is encryption important?

Even the most robust defense is useless if an attacker can go around it; an example from the physical world illustrates this point. When attempting to rob a building, thieves will bypass a front door protected by a deadbolt and look for a less secure back door or an unlocked window for entry. Similarly, the database authentication and authorization capabilities we've discussed so far secure the "front door," ensuring that data is only available to authorized users and not anyone else. However, if attackers cannot gain access via normal means, they might try to bypass database access controls and go after the data in another way.

Encryption protects data in these situations, rendering it unintelligible to anyone trying to access data without going through a database session. Encryption reduces the problem of securing a large amount of data to a more straightforward problem of securing a much smaller key used to encrypt the data. As long as the attackers do not have the key, any encrypted data they access is useless. Encryption is also frequently required to comply with regulations or security standards regarding sensitive or personally identifiable information.

Why should you focus on encryption?

Attackers might try to bypass the controls built into the database by intercepting data as it travels over the network, for example, between a database client and a database server. On many networks, it is relatively easy for an attacker to capture network traffic and extract whatever information was transmitted between the two systems in clear text.

Another way attackers might try to bypass the database controls is by accessing the system as a privileged operating system user and directly reading the database files. Attackers can also go after unencrypted database backup files, backup tapes or file copies stored in a remote location, and backups sent to cloud storage or a cloud service.

Because the data thieves aren't trying to create a session and connect to the database during these attacks, database access controls can't help. The solution is to encrypt the data so it can't be read outside of a valid database connection.

This chapter explains how encryption protects data in motion and at rest for the database. It also discusses considerations for securely storing and managing the encryption keys that ultimately protect the encrypted data.

Encrypting data in motion – database connections

The ability to encrypt data on the network is built into the Oracle Database, including enterprise and standard editions. Oracle Database offers two ways to encrypt data in motion – industry-standard transport layer security (TLS) and Oracle native network encryption (NNE). Both provide confidentiality (to prevent others from reading data sent over the network) and integrity protection (to prevent others from modifying or replaying the data). TLS adds server authentication and optionally provides client authentication. When we discussed authentication in Chapter Five, one of the options was to authenticate by PKI certificate. This type of authentication depends on TLS. Most Oracle client software (thin and thick JDBC, instant client, SQL*Plus, SQL Developer, VSCode SQL Developer plug-in) supports both NNE and TLS network encryption.

Native network encryption encrypts data using a unique shared secret key created for each connection. When a client establishes a connection to an NNE-enabled database, the client and server negotiate to create a unique key used only for that session. Neither the database server nor the client side of the connection needs a certificate with NNE. Thus, you can enable NNE by adding a simple entry to the database's network configuration file. NNE requires no changes on the client. Because of the straightforward setup, without the need for additional external infrastructure or certificates, many organizations use NNE for network encryption.

Another option for encrypting data in motion is industry-standard transport layer security (TLS), which provides authentication and encryption. You can configure TLS in server-authenticated mode (where the server presents an identifying certificate) or in mutually authenticated mode (mTLS) where both the server and the client possess an identifying certificate to provide both encryption of data in motion and database client authentication. TLS certificates are stored in the system certificate store or an Oracle wallet.

Optionally, TLS and NNE can operate in "FIPS mode," which enforces even stricter control over the encryption libraries used.

For TLS encryption, the server and client typically negotiate the cipher suite and select the strongest mutually supported algorithm. However, if desired, you may explicitly require specific algorithms. Oracle Database 23ai and newer versions support TLS 1.3 (the latest version of the TLS standard as of the time I'm writing this), while older versions support TLS 1.2. If your database client or application supports TLS 1.3, then you should use that instead of TLS 1.2 because TLS 1.3 provides superior protection against attacks based on quantum computing.

Security and performance usually have an inverse relationship, the stronger the cipher, the more computing overhead is required to encrypt and decrypt data using that cipher. Oddly enough, this is NOT true when choosing between TLS 1.2 and TLS 1.3! The newer TLS 1.3 standard updates how the client and server negotiate the initial connection and eliminates one of the back-and-forth checks in the communications setup, speeding up the time to create the connection significantly. The cipher suites included in TLS 1.3 are also optimized, and even though they are cryptographically stronger than those in TLS 1.2, they perform slightly better.

For more information on configuring TLS encryption, refer to the Oracle Database Security Guide.

Encrypting data at rest – database files and backups

Oracle Database stores information in data files located in a local file system or some other form of storage like Oracle Automatic Storage Management (ASM). Encrypting the stored data helps ensure an attacker cannot read the information directly from those data files.

Encrypting data can be done at multiple points in your application stack: at the application layer, the database layer, or the underlying storage layer.

In general, the lower down the application stack encryption is implemented, the less intrusive and better performing the implementation will be. At the same time, encrypting data lower in the stack reduces the number and types of threats the encryption addresses.

At the highest level in the stack, application code drives application-layer encryption, which encrypts the data before storing it in the database. Each application must know how to encrypt and decrypt within different parts of the process flow. Application-layer encryption can impact database performance and the types of queries that databases can perform on encrypted columns. Common analytic queries that search for matches against data ranges or computed values on the server may not work or may be extremely slow because database indexes tend to be less useful with application-encrypted data. If multiple applications share the information in the same database, encrypting in the application requires all applications to share access to the same encryption key to encrypt and decrypt the data.

Encrypting data at the highest application layer provides excellent security but imposes a significant development and management burden because it limits performing meaningful relational computation on the data. Application-layer encryption can also cause severe performance problems because the database might no longer be able to use indexes. There are some limited use cases where application-level encryption may make sense. For example, you could use application-level encryption to store a secret string or a blob that you do not want to be visible to any other database user, including the administrators, where there is no requirement to perform database operations on that value.

At the database level, database encryption takes whatever data the application sends, encrypts it, and stores the data in encrypted form in the storage system. It doesn't matter if multiple applications are using the same database because the encryption is transparent to the applications. With database encryption, data is now protected against attacks that bypass the database and attempt to read data directly from storage, but encryption does not protect against someone who connects through the database and has authorization to read the data. This is where the access controls discussed in earlier chapters come into play. To revisit our earlier analogy, access controls guard the front door, while encryption protects the windows and back door.

At the storage level, file or volume layer encryption also protects the data at rest, but this time the threat being protected against is someone bypassing the operating system and directly (physically) accessing the storage. This type of encryption provides very limited risk mitigation. The biggest problem with file system encryption is that file or volume layer encryption software lacks database user context. If Oracle Database runs as OS user ‘oracle’ on the operating system, any other (possibly malicious) user logged in as ‘oracle’ can also access decrypted data. Since the purpose of database encryption is preventing database bypass (a user directly accessing the stored data), employing a security control that automatically decrypts data for OS commands like strings, move, or copy doesn’t make much sense.

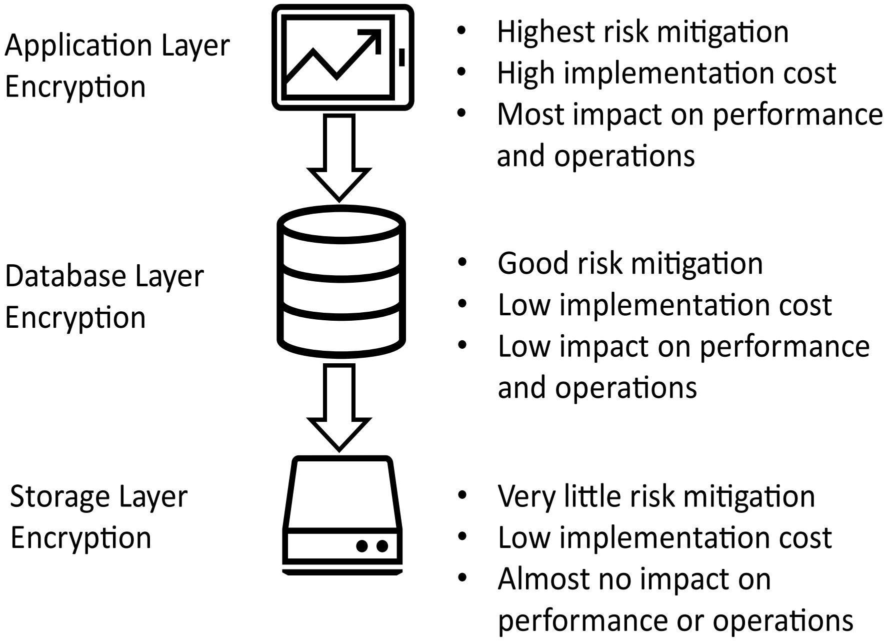

Application Layer Encryption
• Highest risk mitigation
• High implementation cost
• Most impact on performance and operations
Database Layer Encryption
• Good risk mitigation
• Low implementation cost
• Low impact on performance and operations
Storage Layer Encryption
• Very little risk mitigation
• Low implementation cost
• Almost no impact on performance or operations

Figure 11-1: Encryption options

Although storage layer encryption usually has almost no impact on system stability or operations, much depends on which storage layer encryption solution you choose. Encrypting file systems like ZFS or Microsoft BitLocker, or secure-wipe encryption (built into some high-end physical media) tends to be very stable. Third-party file system encryption solutions that interpose themselves between the operating system and the database add a possibility for introducing system instabilities and upgrade issues. Such third-party software will also complicate patching activity, preventing teams from applying database and operating system patches until a dependent patch from the file system encryption vendor is available. Because of these complications, Oracle is unable to support database issues arising from the use of third-party file-system encryption.

One size seldom fits all. Blended solutions using all or multiple encryption strategies are common. Here’s an example. Organization Z is a retail company that deals with financial data (including their customer’s credit cards, which they retain for reuse in their online storefront). The organization also retains personal data about their customers, including date of birth, physical address, email address, phone number, and records of past purchases. The encryption strategy that the organization adopts is:

1. Application-layer encryption for credit card numbers – the data is encrypted at point of collection and remains encrypted except when being retrieved to support customer purchases.

2. Database-layer encryption for all other application data – this protects the organization against bypass attacks like ransomware and lets them limit their exposure to data theft while still being able to use the data for analytics and (where their customers have opted in) for marketing purposes.
3. Storage-layer encryption – the organization's SAN uses media configured for secure wipe in the event of a failure where hardware needs to be sent back to the manufacturer.

Irrespective of which layer encrypts the data, the key considerations for selecting an encryption solution include:

- Which risks are mitigated by the encryption
- How the encryption impacts performance
- Simplicity of installation and use
- Transparency to existing applications
- Resources required to convert data between plaintext and encrypted form
- Patching
- Key management

Neither application layer nor storage layer encryption techniques are optimal for protecting data in Oracle Databases. We need a mechanism that improves security and reliability while minimizing the implementation and performance burden.

Transparent Data Encryption

Transparent Data Encryption (TDE), a feature of Oracle Advanced Security, mitigates the risk of bypass attacks by encrypting data within the database. The encryption is transparent to authorized operations performed through an authenticated database session because the database automatically encrypts data before it is written to storage and automatically decrypts it when reading from storage. Authorized users and applications that store and retrieve data in the database see only decrypted (or "plaintext") data. Attempts to go around the database and read the data directly through the operating system or from storage see only encrypted data. Because the database files are encrypted, backups are automatically encrypted, so attackers who steal backup copies cannot read the stolen data.

Authorized database users and applications do not need to do anything different from how they usually access the database. Instead, the database enforces access control policies described in the previous chapters and denies access if the user is not authorized to see the data.

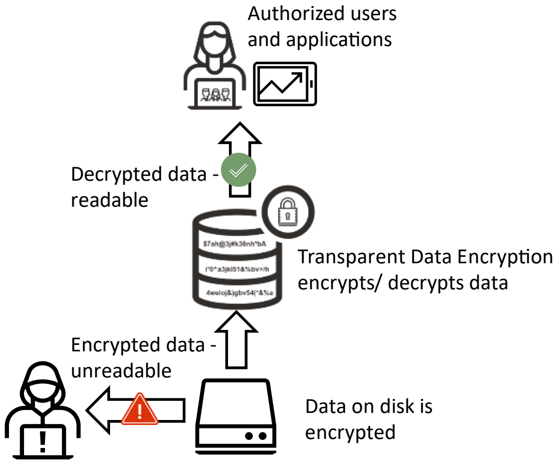


Figure 11-2: Users and applications work against decrypted data in database memory

TDE can encrypt whole tablespaces or individual columns. In almost all cases, tablespace encryption will be the optimal choice. With tablespace encryption, you do not need to track which columns to encrypt or worry about someone inserting sensitive data into a column that would not usually be considered sensitive. With tablespace encryption, you also don't need to worry about column characteristics such as indexes and constraints. Creating an encrypted tablespace is as easy as:

```
SQL> CREATE TABLESPACE investigations
     DATAFILE '$ORACLE_HOME/dbs/investigations.dbf' SIZE 50M
     ENCRYPTION  USING 'AES256' MODE 'XTS' ENCRYPT;
```

Figure 11-3: Creating an encrypted tablespace

Note: The default encryption algorithm in Oracle Database 23ai is AES256 using XEX-based tweaked-codebook mode with ciphertext stealing (XTS) encryption. Shorter key lengths (AES192 and AES128 are available, but not advised due to concerns about quantum computing resistance. Older databases implemented AES using cipher feedback (CFB) mode. Oracle Database 23ai still offers CFB as an option for those who wish to use it.

You may configure your database to encrypt new tablespaces automatically, even if an application does not create encrypted tablespaces as part of the application installation process.

When TDE is used to encrypt tablespaces, sensitive data remains encrypted throughout the database, whether in tablespace storage files, temporary or undo tablespaces, or other files such as redo logs. In addition, TDE can encrypt RMAN database backups and Data Pump exports.

TDE automatically leverages special instructions in Intel/AMD (AES-NI) and SPARC CPUs to accelerate cryptographic operations. In addition, TDE tablespace encryption integrates with the performance optimizations built into Oracle Engineered Systems such as Exadata and the Oracle Database Appliance. For example, TDE tablespace encryption works seamlessly with Exadata Hybrid Columnar Compression (EHCC) and Smart Scan technology.

TDE and the Database Configuration Assistant (DBCA)

Starting with Oracle Database 21c, DBCA can create encrypted databases so the database is encrypted from the very start. See the Transparent Data Encryption Admin Guide for details.

In addition to creating a new encrypted database, DBCA in 23ai allows you to:

- Setup wallet-based TDE and encrypt an unencrypted database
- Migrate a database from wallet-based TDE to Oracle Key Vault

Note: More about Oracle Key Vault later in this chapter

Migrating clear-text data to encrypted tablespaces

When encrypting tablespaces in an existing database, there are two choices – online and offline.

Online tablespace encryption, introduced with Oracle Database 12.2.0.1, enables zero-downtime migration from plaintext to encrypted data or converts from one encryption algorithm to another. Online encryption temporarily uses additional storage of the same size as the processed tablespace. In the sample command shown in Figure 11-4, datafile `hr_data01.dbf` will be copied and written in encrypted form into datafile `hr_data01_enc.dbf`. When the encryption operation completes, the original datafile `hr_data01.dbf` will be deleted and its space recovered.

```
ALTER TABLESPACE hr_data  ENCRYPTION ONLINE ENCRYPT
FILE_NAME_CONVERT=('hr_data01.dbf','hr_data01_enc.dbf');
```

Figure 11-4: Online tablespace encryption

TDE also supports two offline encryption methods that do not require extra storage but require the tablespace to be offline or the database to be in mount mode.

The first offline encryption method simply has you take the tablespace offline and issue an ALTER TABLESPACE command to encrypt all data files in the tablespace. The command can be as easy as shown in Figure 11-5 below if you wish to accept the default encryption algorithm and mode (AES256 using XTS mode in Oracle Database 23ai, AES128 using CFB mode in Oracle Database 19c). If desired, you can override the defaults specifying the algorithm and mode of your choice.

```
ALTER TABLESPACE hr_data ENCRYPTION OFFLINE ENCRYPT;
```

Figure 11-5: Offline tablespace encryption

The second offline encryption method makes use of Oracle Data Guard, where standby databases are already in mount mode. For this method, you use offline encryption to minimize downtime without needing additional temporary storage space. The procedure is to turn off the managed recovery process, encrypt the standby database's data files, resume managed recovery and wait for the recovery process to catch up. Then perform a switchover and encrypt the new standby (formerly primary) database. In this scenario, the downtime for encrypting any size database is as short as the duration of the two switchovers.

TDE two-tier key architecture

At the start of this chapter, I said, "*Encryption reduces the problem of securing a large amount of data to a more straightforward problem of securing a much smaller key used to encrypt the data.*" The security of encrypted data depends on maintaining the secrecy of the keys used for encryption. Proper key management is essential for preventing an attacker from discovering the encryption key and gaining access to data. Good key management also ensures that active keys are not lost, are rotated periodically, and that old keys are securely archived to provide continued access to encrypted backup data sets. Many regulations, such as PCI-DSS, require physical separation between encrypted data and encryption keys and periodic rotation of encryption keys to limit the exposure period if a key is compromised.

TDE uses a two-tier key architecture to create and manage the keys used for encryption. The first tier is a master encryption key (MEK) used throughout the database. The second tier is a series of data encrypting keys (DEK) that are unique for tablespaces or tables (depending on whether column or tablespace encryption is used).

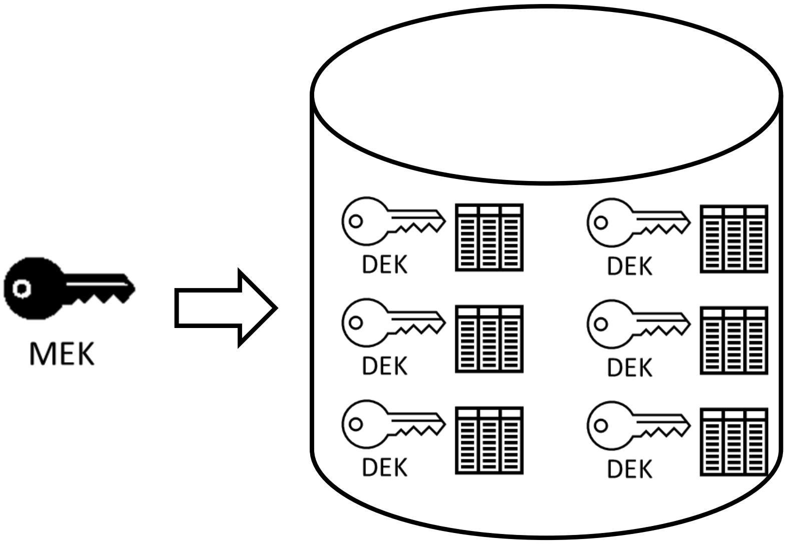


Figure 11-6: Two-tier key architecture

As shown in Figure 11-6, a single master encryption key (MEK) is outside of the database and is used to encrypt/decrypt the internally generated data encryption keys (DEK), which in turn encrypt tablespace data files or table columns. The database automatically manages DEKs behind the scenes. The two-tier key architecture simplifies key rotation. When rotating the MEK, there is no need to decrypt and re-encrypt all data, only the much smaller set of DEKs. You can use online tablespace encryption if there is ever a need to rotate the DEKs or switch to another encryption algorithm.

Administrators perform key management operations (create, open, rotate, backup, etc.) through SQL commands. These operations are also exposed in Oracle Enterprise Manager and (if the database is an Oracle Cloud database service) through the OCI console. Oracle Database 12c introduced the SYSKM administrative privilege to allow key administrators to perform key management operations without requiring the powerful SYSDBA privilege.

The MEK is always stored outside of the database, separated from encrypted data, and managed by the key administrator in the Oracle Wallet, Oracle Key Vault, or OCI Vault as shown below in Figure 11-7.

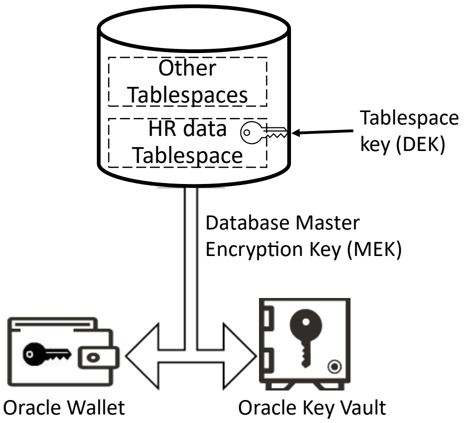


Figure 11-7: TDE encryption keys

The database master encryption key is created when you first enable TDE, and the master encryption key can be rotated with an `ADMINISTER KEY MANAGEMENT` command. Tablespace keys are automatically created when you encrypt a tablespace, and a tablespace key can be rotated with an `ALTER TABLESPACE` command. Each encrypted tablespace will have its own tablespace key that is stored within the database and encrypted using the master key.

With Oracle Database 18c and later, you can import externally generated MEKs into the key store, enabling “bring your own key (BYOK)” functionality if you wish to use an external key generator.

Key Management

Local key management with Oracle Wallets

The most important aspect of key management is storing the keys securely and restricting access to authorized entities. By default, TDE keeps the master encryption key in a PKCS#12 standard-based file called the Oracle Wallet. The wallet’s contents are encrypted using an AES256 key derived from the wallet passphrase. Note that if the wallet is somehow lost or corrupted or the password is forgotten, there is no way to recover the encrypted data – there are no “back doors” for TDE! That is why it is imperative to securely back up the password-protected wallet. Oracle Key Vault, described in the next section, provides an automated solution for backing up and managing wallets or to do away with them altogether and manage keys directly within Key Vault.

You may prefer to use local auto-login wallets to streamline day-to-day operations such that you do not require manual intervention to enter a wallet password. Local auto-login wallets are tied to the server where the auto-login wallet is created, reducing the risk of misuse of the wallet on another system.

Centralized key management with Oracle Key Vault

Encrypting databases is commonplace in today’s regulatory environment. With the increasing number of encrypted databases, managing hundreds or thousands of wallets becomes more of a burden. Lost or corrupted wallets and forgotten wallet passwords put data and even the entire database at risk. Furthermore, many regulations and most security best practices do not allow storing encryption keys on the same server as the encrypted data. Many organizations also require a clear separation of duties between DBAs and key management administrators. External key management solutions like Oracle Key Vault ease the management burden and reduce the risk of losing access to wallets.

Key Vault (OKV) is an enterprise-grade key manager that provides secure and reliable distribution for master keys, wallets, and secrets. Oracle Database connects to the Key Vault (or, more typically, the Key Vault cluster) through a PKCS#11 library installed on the database server as shown in Figure 11-8 below.

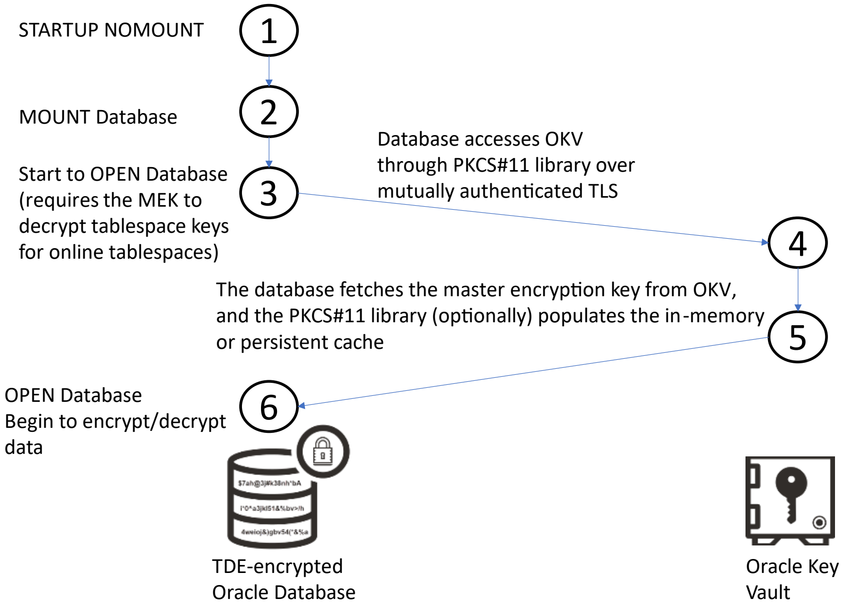

STARTUP NOMOUNT
1
MOUNT Database
2
Database accesses OKV through PKCS#11 library over mutually authenticated TLS
Start to OPEN Database (requires the MEK to decrypt tablespace keys for online tablespaces)
3
4
The database fetches the master encryption key from OKV, and the PKCS#11 library (optionally) populates the in-memory or persistent cache
5
OPEN Database Begin to encrypt/decrypt data
6
TDE-encrypted Oracle Database
Oracle Key Vault

Figure 11-8: Oracle Key Vault securely delivers MEKs to Oracle Databases on demand.

The master key is used for many different database operations. A partial list of operations that require the database to consult the MEK includes:

- When the database is started (opened)
- When a new tablespace is created
- When a tablespace or data file is brought online
- When a redo log group switch occurs
- When a new archive log is created
- When a parallel query process is begun
- During direct-path access to the database
- Initial access to an encrypted column
- GoldenGate and Data Guard operations

There is also a “heartbeat” check of the external keystore to ensure it remains available. If the database attempts to access the master encryption key and fails, it will immediately shut down to preserve data and prevent corruption. To prevent downtime, it is critical that Key Vault be highly available and respond to requests for the master key in a timely manner.

Key Vault delivers continuous availability and high scalability for read and write key operations while protecting against data loss through a multi-master cluster architecture. You can create Key Vault clusters with up to 16 nodes to form a multi-master cluster spanning geographically distributed data centers. So long as the database can access any of the nodes within the cluster, database operations continue. Key Vault nodes may be placed on-premises and within most major cloud providers (including Oracle, Amazon, Microsoft, and Google). Since all nodes in a cluster are active, each node has a copy of all keys in the cluster, and all nodes share the key management and distribution load.

Databases can connect to any Oracle Key Vault cluster node to retrieve master encryption keys. Updates to keys or changes to authorization rules are synchronously replicated to a partner node to guard against data loss in the event of an untimely node failure. If the Oracle Key Vault connection fails or a node goes down, database servers transparently failover to a nearby active node for read/ write operations without the need for downtime or administrator intervention.

As an additional guard against service interruptions, the default configurations enables a local persistent cache, where the Key Vault endpoints cache keys for the database server. The persistent cache is an encrypted software keystore on the database server managed by Oracle Key Vault, which provides keys in case of a lost network connection between databases and the OKV cluster nodes. Using the persistent cache provides the best performance because most key operations are served directly from the cache without requiring a call over the network to the Key Vault.

Keys can also be configured as non-extractable. Non-extractable keys never leave the OKV cryptographic boundary, meaning each time a DEK needs to be decrypted, the database will call OKV, and wait for a response from OKV before continuing.

Note: Before configuring a system to use non-extractable keys, Oracle recommends careful testing of the performance impact.

Key Vault supports TDE keys across enterprise database deployments using Multitenant, Real Application Cluster (RAC), Data Guard, Globally Distributed Databases, GoldenGate, and other Oracle products and technologies. In addition to TDE keys, Oracle Key Vault offer SSH key governance, and manages Java Keystores, MySQL and MongoDB encryption keys, Solaris Crypto keys, and ASM Cluster File System (ACFS) volume encryption keys. If local wallets are desired, Key Vault can periodically back up the local Oracle wallets as a form of key escrow service. Key Vault can also be used to backup Java Keystores and PKI certificate wallets.

Key Vault with DBMS_CRYPTO

Separating keys and secrets from application code increases security and simplifies development. Key Vault can provide key management for custom cryptographic applications using Oracle Database's `DBMS_CRYPTO` package.

Key Vault beyond encryption

Beyond encryption, Key Vault provides SSH key governance and remote server access control. Centralized management of SSH keys protects both private and public SSH keys from loss and abuse and gives you control over how SSH keys are used throughout your environment. For example, with a single click of a button or API call, you could disable all SSH connections during an ongoing security incident! Because there is now a central point for SSH key access, Key Vault will also provide you an audit trail showing who accessed your servers, when the access occurred, and where the request came from!

Key Vault is also a full-featured secrets manager and can manage a variety of secrets supporting everyday IT operations. For example, many sites use locally-managed Secure External Password Store (SEPS) wallets for maintenance scripts such as nightly RMAN backups, batch loading into a data warehouse, or refreshing materialized views. Because those wallets are resident on the database server, they are at risk of being stolen. Instead of a SEPS wallet, you can upload database account passwords into Key Vault and securely retrieve them via an API call at runtime. Centralizing credentials in Key Vault improves security for the credentials and eliminates the management burden of protecting hundreds or thousands of SEPS wallets for automated database operations. It simplifies strictly controlled sharing of those passwords between databases.

C and Java SDKs allow developers to integrate any Java, C, or C++ program with Key Vault to manage encryption keys, passwords, and other secrets and retrieve them on demand.

Key Vault's other capabilities include protecting public and non-extractable private SSH keys from loss and abuse. Centralized management of SSH keys helps achieve full key governance over your SSH keys, allowing you, for example, to turn off all SSH connections in case of an ongoing security incident.

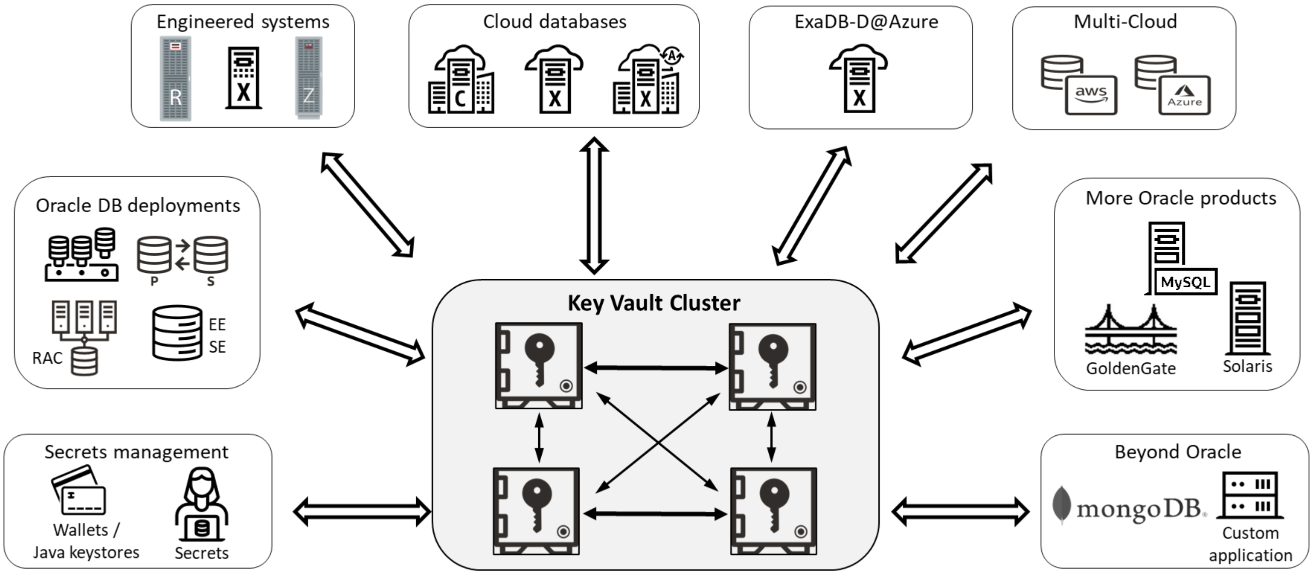


Figure 11-9: Oracle Key Vault manages Oracle TDE keys as well as other keys and security objects

Key Vault APIs for scripting and automation

Key Vault's RESTful APIs make it easy to monitor a Key Vault cluster, add or remove nodes from an existing cluster, or even create a new cluster. Onboarding new servers and managing encryption keys may also be automated through the APIs, with the process of distributing and enrolling Key Vault endpoints fully automated. Automation allows the DBAs to encrypt their databases while the security team securely manages the master keys within Oracle Key Vault.

Oracle LiveLabs

Oracle LiveLabs give you access to Oracle's solutions and technologies to run a wide variety of labs and workshops at your own pace. If you want to try the solutions discussed in this chapter, please go to:

- DB Security – Native Network Encryption
- Protect database communications using 1-way TLS
- DB Security – ASO (Transparent Data Encryption)
- Pluggables, Clones, and Containers – Securing Oracle Multitenant with TDE
- DB Security – Key Vault
- DB Security – Key Vault (SSH Key Management)

Summary

Rising security threats, expanding compliance requirements, and cloud computing are just a few reasons why encryption is critical for most organizations. Encrypting data-in-motion helps maintain the confidentiality and integrity of data as it travels across the network while encrypting data-at-rest blocks unauthorized access to sensitive data using methods that bypass the database. Important considerations for selecting encryption solutions are security, performance, simplicity of installation and use, transparency for existing applications, data migration from plaintext to encrypted form, patching, and key management.

TLS and NNE are standard features of Oracle Database for protecting data in motion. TDE safeguards sensitive data against unauthorized access from outside the database by encrypting data at rest. It prevents privileged operating system users from bypassing controls and directly inspecting the contents of database files. It also protects against theft, loss, or improper decommissioning of database storage media and backups.

Oracle Key Vault eases the management overhead of mass deployment of TDE by providing a centralized key management platform and delivering key and secrets management to the Oracle enterprise. Encryption and centralized key management work together to help address privacy regulations and standards such as the PCI-DSS, HIPAA, and EU GDPR.

Chapter Twelve

Database Activity Monitoring

Why is database activity monitoring important?

Monitoring database activity helps you detect potentially malicious behavior, comply with regulatory requirements, and support investigations following incidents. Audit trails (the data collected during monitoring) provide a record of what happened. The audit trail records actions performed within the database and is crucial for maintaining transparency, accountability, and security.

The data collected by auditing can have many uses, including:

- Detecting and understanding malicious activity
- Assisting in investigations of data breaches or other suspicious activity
- Deterring inappropriate or malicious activity – just the knowledge that auditing may reveal their activity may stop someone from trying to do the wrong thing
- Non-repudiation – providing proof that an event occurred with data that indicates who did it when it was done, and where the command to execute the event was issued from
- Supporting regulatory compliance audits (a different use of the word audit, and often confusing). Compliance auditors may require a record of activity, especially related to changes or viewing of financial data
- Monitor activities of privileged users. Most databases have at least a few users, some of whom may be privileged, that directly access the database. For example, database administrators (DBAs) are usually considered privileged users because of their broad access to the database. Accounts logging in directly to perform data analysis are another example of privileged users. Privileged user accounts are often soft targets for hackers accessing critical systems and data.
- Assist in troubleshooting problems. Imagine a database with hundreds of application servers connected to it. One of those application servers is consistently trying to login with the wrong credentials. Auditing of failed logins lets you know the issue exists and pinpoints which of those application servers is the problem
- Recovering Data - In case of data loss or corruption, audit trails can help recover previous versions of data

Why should you focus on activity monitoring?

Much like encryption (which we discussed in Chapter 11), monitoring database activity is simple conceptually. However, if you don't understand the database's capabilities, the different options available to you, and the implications of choosing between those options, you can make choices that have a serious impact on database performance and the amount of time you'll spend working with activity monitoring.

While organizational security guidelines vary, they invariably require auditing privileged user activity, login events, sensitive data access, and other security-related activities. Your goal is to capture the data you need to support organizational policy while minimizing the impact on database users and operations.

This chapter helps you evaluate the pros and cons of these different approaches and discusses which types of database activity monitoring are right for different security goals.

What is database activity monitoring?

Database activity monitoring examines operations being performed on the database and makes decisions about those operations. The operation might be logged (written to an audit trail), alerted upon (generate some sort of message to interested parties that the operation occurred), or passed with no action taken at all. There are a few ways to monitor activity:

- Database auditing. Auditing uses the native capability of the database to capture information about operations. Database auditing provides the most accurate record of database activity. Data captured during auditing is written to an audit trail defined by the database, usually a database table. Auditing has a broader contextual view of user activity than external activity monitoring techniques. Auditing also tends to have more of an impact on database resource consumption (primarily CPU, I/O, and storage), which, if the database server is resource-restricted, can result in lower query throughput.

- External monitoring. Monitoring uses capabilities from something outside of the database to capture information about operations. Monitoring usually collects data at the network layer (by sniffing network traffic) or by examining database memory and process activity. Data captured by external monitoring is written to something outside of the database – that can be to another database, to a cloud service, as log files to a file system, or to a special-purpose appliance. External monitoring, especially network-based monitoring, may consume little or no resources on the database server. One challenge with external monitoring can be that direct local logins (that do not operate over the network) usually cannot be monitored. Another challenge is that external monitoring lacks a contextual view of the session, and will not be able to understand recursive SQL, dynamic SQL, or stored procedures. Yet another challenge is that external monitoring lacks awareness of database objects and structures and can be fooled by something as simple as table synonyms or views. Attackers may use these techniques to circumvent external monitoring.

This chapter focuses on auditing. We will discuss external monitoring in Chapter Thirteen. Effective auditing requires that audit policies capture essential details about significant auditable events and minimize the collection of useless data. Targeted auditing reduces the cost of monitoring database activity by reducing the performance impact and storage costs. Targeted auditing can also lead to quicker and more reliable threat detection.

Oracle Database with unified auditing

Oracle Database auditing provides robust and highly customizable auditing that can be fine-tuned to address the organization's specific security requirements. Auditing was introduced in Oracle Database 7 (released in 1992). Oracle Database's audit features evolved over decades of development, leading to the introduction of a new unified audit framework in Oracle Database 12.1 (released in 2012). The legacy audit facility is desupported in Oracle Database 23ai and the unified audit framework is now the only audit framework. In addition to unifying several disparate audit trails that evolved during decades of development into a single normalized audit trail, the newer unified audit facility:

- improved performance (overhead is usually 3% or less for audit generation - even under heavy load of as much as 200 audit records per second)
- tightened down security (the unified audit trail is in its own dedicated AUDSYS schema and is protected against DML operations including update, delete, and truncate)
- added greater flexibility with conditional auditing and an extensible audit trail

Note: Legacy traditional auditing is desupported starting in Oracle Database 23ai. If you are still using traditional auditing when upgrading to Oracle Database 23ai the existing traditional audit settings will continue to be honored, and audit records will continue to collect into their respective audit trails. This continuation of the legacy audit facility is intended to allow those still using the old facility to bridge the gap between when they upgrade to 23ai and when they can migrate to the newer audit facility. Database 23ai will not allow you to create new legacy audit policies or update existing policies. You can only delete the existing old legacy audit policies.

What should you audit?

One way to detect inappropriate or malicious database activity is to audit ALL database activity. However, doing that would incur a significant trade-off in storage costs, performance impact, and efficiency. Most of the audit records would be for benign activity with no security relevance, and the volume of audit data might obscure malicious activity, introducing the "needle in the haystack" problem.

A more practical approach is to audit significant events using selective audit policies. Effective audit policies capture essential details about significant events and avoid capturing information on routine events that have no security value. Reducing the "noise level" of unnecessary audit records can lead to quicker and more reliable threat detection. In most cases, an effective approach focuses your audit policies on three types of database events:

1. Security-relevant events
2. Privileged user activity
3. Sensitive data access

while ignoring routine activity generated from known sources.

Oracle Database comes pre-loaded with several audit policies that cover the most common requirements, making it easy to start auditing quickly. Using the predefined policies is easy – once you know which policy you want to enable, all it takes is running the appropriate command. Here is an example of enabling one of the most commonly used policies:

```
AUDIT POLICY ORA_SECURECONFIG;
```

Figure 12-1: Enabling an audit policy

Let’s dig into those three recommended types of events, and the policies that can help you get up and running.

Security-relevant events auditing

Some database actions should be monitored because they have a broader impact than just one table or schema and could indicate potential abuse or hiding of events. Security-relevant events that should be audited include:

- failed login attempts – you may want to consider auditing ALL logins with exceptions for service accounts logging in from known locations running known programs
- schema structural changes – create/alter/drop table, view, stored procedures, etc.
- account changes – create/alter/drop of users, privilege and role grants, password changes, etc.
- security policy changes, especially relating to schemas that contain sensitive data
- activities using system privileges
- mass-data extraction using components like Oracle Data Pump
- activities from dormant database accounts

Three of the predefined unified audit policies will help you capture the security-relevant activity listed above:

- ORA_SECURECONFIG
- ORA_LOGIN_LOGOUT (named ORA_LOGON_FAILURES in older database versions)
- ORA_ACCOUNT_MGMT

These three policies address the most common security-relevant events. You can enable them with a simple PL/SQL call as shown in Figure 12-1 above or using the audit management consoles in Data Safe or AVDF. You can also provision the added-value “Database Schema Changes” policy through either the Data Safe or AVDF management console.

Many other security-relevant events are always audited as part of the *mandatory audit* policy. These are database activities that can potentially cause significant issues. A complete list of auditable events covered by the mandatory audit policy is here.

Privileged user activity auditing

We discussed privileged users in Chapter Seven. Monitoring privileged users allows you to identify anomalous behavior and detect sensitive data leaks quickly if these accounts are compromised or misused. For the purposes of auditing, you should consider database users granted system privileges, such as SELECT ANY TABLE or ALTER USER, as privileged and monitor them to detect abuse or misuse.

Top-level statements by administrative users (e.g., SYSDBA, SYSKM) issued while the database is in the closed or mount state are already included in the mandatory audit policy (which is enabled by default and cannot be disabled).

Note: the phrase “top-level” used in this context means the statement that was entered by the user. Auditing top-level statements captures only those statements, not internal or recursive SQL that is generated as a result of the statements. For example, if top-level statements by a database administrator are audited, and a DBA executes DBMS_STATS.GATHER_DATABASE_STATS, only the command to execute the PL/SQL package would be audited. Top-level ignores the SQL statements not directly issued by the client such as those SQLs called from named procedures and functions that can potentially generate thousands of audit records. Auditing at the top level significantly reduces the quantity of audit records generated for routine administrator activity.

To capture top-level actions of administrative user accounts, such as SYS during normal database operations you need to enable additional audit policies. If you are using Data Safe or AVDF to monitor database activity, you can provision an “Admin Activity Auditing” policy to audit all activities by privileged administrators, including SYS.

Database connections may be made through the database listener (remote connections) or locally from an account logged into the database server. Auditing local sessions is crucial because those sessions can bypass network monitoring and may be able to connect to the database without any authentication other than to the database server’s operating system account that initiates the session. Oracle recommends auditing all top-level operations originating from a local session. Create an audit policy to audit all top-level actions with the appropriate audit conditions as given below and consider enabling them on interactive users granted direct access to the database.

```
CREATE AUDIT POLICY local_db_access
ACTIONS ALL
WHEN '(SYS_CONTEXT (''USERENV'', ''IP_ADDRESS'') IS NULL)'
EVALUATE PER SESSION
ONLY TOPLEVEL;
```

Figure 12-2: Auditing local access to the database

In some cases, you will want to monitor all activity by privileged accounts that have access to sensitive data. If you use Oracle Data Safe or Oracle AVDF, you can provision the “User Activity Auditing” policy, which lets you audit all activities by a given set of users. You can also create your own custom policy to audit everything a user does. Figure 12-3 shows an example of this. The first statement creates a policy that audits all actions. The second statement enables that policy for the user DBA_DEBRA.

```
CREATE AUDIT POLICY actions_all_pol
ACTIONS ALL
ONLY TOPLEVEL;


AUDIT POLICY actions_all_pol BY DBA_DEBRA;
```

Figure 12-3: Auditing all activity by a user

Keep in mind that auditing all activities by a user can significantly increase the number of audit records generated, with accompanying storage costs and performance overhead.

Sensitive data access auditing

We discussed sensitive data in Chapter Four. Auditing access and updates to sensitive data may reduce the time to detect inappropriate activity and can deter those who do not have a business reason to access or modify the data.

Selective auditing of sensitive data access provides an effective way to monitor access and updates to sensitive data. Scan your database for sensitive data using any of Oracle Data Safe, Oracle Audit Vault and Database Firewall, Oracle Database Security Assessment Tool, and Oracle Enterprise Manager. See Chapter Three for more information on sensitive data discovery.

Once you’ve located sensitive data, create a custom audit policy to audit access to sensitive data outside of normal application behavior. Figure 12-4 shows an example of auditing all activity against the employees table in the HR schema (which we presume has sensitive data in it). The first statement creates the policy and defines which object is going to be monitored. The second statement enables the policy and applies it to all users.

```
CREATE AUDIT POLICY actions_on_hr_emp_pol
ACTIONS ALL
ON hr.employees;


AUDIT POLICY actions_on_hr_emp_pol;
```

Figure 12-4: Auditing activity on sensitive data

The problem with the policy in Figure 12-4 is that there is a good chance the HR application accesses that table frequently, so you would generate a lot of useless audit data—with implications for performance and storage costs. A better way to craft this policy would be to audit all access to the table outside of the approved application.

Modifying our example from above to what you see in Figure 12-5 eliminates most of the normal activity against the table.

```
CREATE AUDIT POLICY actions_on_hr_emp_pol
ACTIONS ALL
ON hr.employees
WHEN 'SYS_CONTEXT (''USERENV'', ''HOST'') NOT IN (''APPSERVER1'',''APPSERVER2'')'
EVALUATE PER SESSION;


AUDIT POLICY actions_on_hr_emp_pol;
```

Figure 12-5: Auditing activity from outside of an application

In the modified policy, we add a condition (WHEN clause) that defines a trusted path as accessing data through the application server – the policy checks a session context value HOST to verify that the connection was made through the application server. To improve performance, we evaluate the condition only once per session, and reuse the results of that evaluation throughout the rest of the session. Any activity on the table from outside of the application server generates audit records, but normal activity (from the application server) does not. The WHEN condition you specify can be complex, including and/or statements, but you are limited to 4000 bytes. If needed, you can overcome that limitation using a custom PL/SQL function. For complex conditions, the `EVALUATE PER SESSION` clause can significantly reduce overhead from the auditing.

Focusing audit policies on privileged user activity, security-relevant events, and sensitive data access helps build better audit policies – policies that capture data on the activities that matter, selective enough to reduce the creation of unnecessary audit records, and practical enough to let you meet your audit goals.

As you can see, auditing policies are designed to be very flexible. They allow you to target your auditing to capture the important information and not the “noise” that hurts performance, costs you money in terms of storage, and can make it harder to detect anomalous or malicious activity.

Effective audit policies

Here are a few general-purpose guidelines to follow when designing effective audit policies:

1. Do not duplicate mandatory audits - creating policies that duplicate the audit information already captured with the mandatory audit has a slight but measurable impact on performance since all audit policies must be evaluated for every command. Refer to the Oracle Database Security Guide for a complete list of mandatory auditable events corresponding to your database version. For instance, avoid duplicating mandatory audit policies in your own custom policies.

2. Leverage predefined audit policies - Oracle Database provides several predefined “best practice” unified audit policies that cover common security-relevant audit settings. These only need to be enabled to start collecting audit data. The available policies vary by database version; therefore, you should check the Oracle Database Security Guide that corresponds to your version. For instance, you’ll find a description of the predefined audit policies that ship with Oracle Database 23ai here.

   If you use Oracle Data Safe or Oracle Audit Vault and Database Firewall (AVDF), you can enable many of the recommended audit configurations and predefined Oracle Database audit policies using a single click or via a RESTful API call.

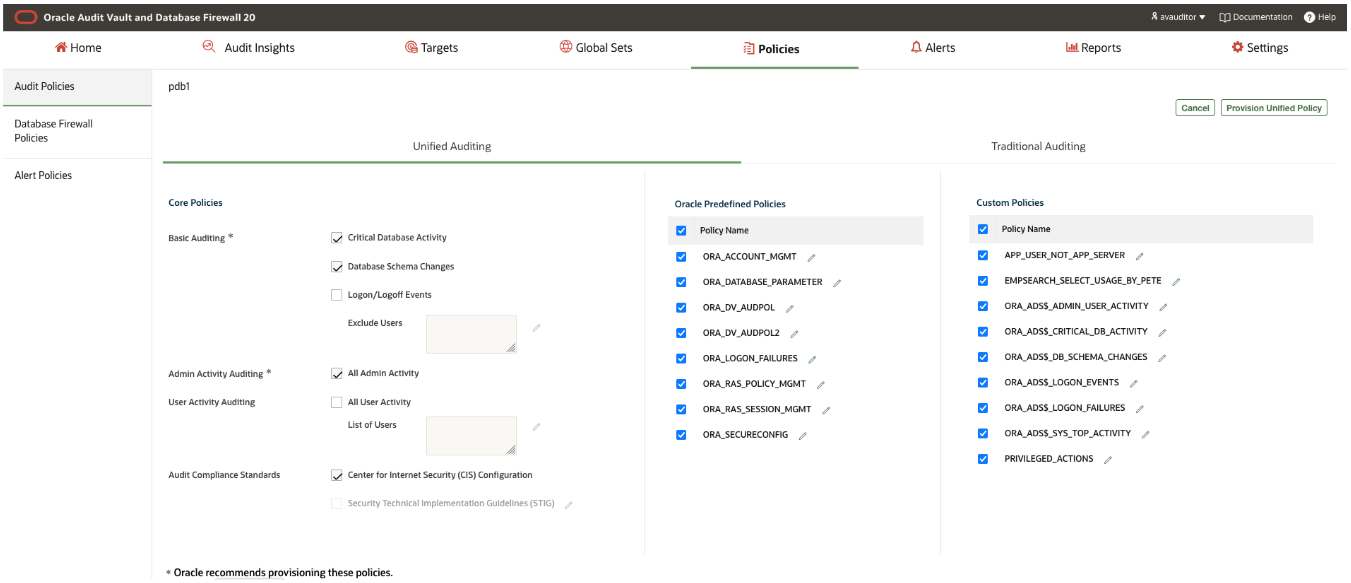
3. Create custom audit policies for context-dependent scenarios - some of the audit configurations will be unique to your system (e.g., sensitive data access), and we recommend you create custom audit policies in Oracle Database for such scenarios.

Audit policy provisioning from Data Safe or AVDF

Oracle Audit Vault, Database Firewall, and Oracle Data Safe both provide flexibility to provision audit policies with a single click. Policies are broadly classified into the following categories.

1. Basic auditing policies
   Critical Database Activity
   Login Events
   Database Schema Changes
2. Administrator and user activity auditing policies
   Admin Activity Auditing
   User Activity Auditing
3. Audit Compliance Standards
4. Oracle Predefined Policies
5. Custom Policies

You can also retrieve custom unified audit policies from Oracle Database and enable/disable them from the console. The audit policy provisioning in the AVDF console is accessible from its auditor’s console, as shown here.



pdb1

Core Policies

Basic Auditing *
☑ Critical Database Activity
☑ Database Schema Changes
□ Logon/Logoff Events
Exclude Users

Admin Activity Auditing *
☑ All Admin Activity

User Activity Auditing
□ All User Activity
List of Users

Audit Compliance Standards
☑ Center for Internet Security (CIS) Configuration
□ Security Technical Implementation Guidelines (STIG)

Oracle Predefined Policies

☑	Policy Name
☑	ORA_ACCOUNT_MGMT
☑	ORA_DATABASE_PARAMETER
☑	ORA_DV_AUDPOL
☑	ORA_DV_AUDPOL2
☑	ORA_LOGON_FAILURES
☑	ORA_RAS_POLICY_MGMT
☑	ORA_RAS_SESSION_MGMT
☑	ORA_SECURECONFIG

Custom Policies

☑	Policy Name
☑	APP_USER_NOT_APP_SERVER
☑	EMPSEARCH_SELECT_USAGE_BY_PETE
☑	ORA_ADS$_ADMIN_USER_ACTIVITY
☑	ORA_ADS$_CRITICAL_DB_ACTIVITY
☑	ORA_ADS$_DB_SCHEMA_CHANGES
☑	ORA_ADS$_LOGON_EVENTS
☑	ORA_ADS$_LOGON_FAILURES
☑	ORA_ADS$_SYS_TOP_ACTIVITY
☑	PRIVILEGED_ACTIONS

* Oracle recommends provisioning these policies.

Figure 12-6: Audit policy provisioning from the AVDF Console

Oracle Data Safe offers similar single-click provisioning capabilities.

Transitioning from legacy to unified auditing

Legacy auditing (sometimes called traditional auditing) is desupported in Oracle Database 23ai. If you create a new Oracle Database 23ai, the legacy audit facility will not exist. If you upgrade an older database to 23ai, the result depends on the database you upgrade. If the older database was already using the new audit facility (no existing legacy policies remain), then the upgraded database will continue to use the new audit facility. But, if you upgrade a database that has not switched over to the newer unified audit facility, then the upgraded database will continue to apply legacy audit policies and capture them in the older-version audit trail until you disable those policies with a `NOAUDIT` command. You will not be able to create new legacy audit policies. If you do still use the old legacy auditing facility, plan your transition to unified auditing following these guidelines before or after the upgrade.

- Oracle Database provides a set of predefined unified audit policies to help you get started. If you have upgraded your Oracle database installation from release 11g, then at a minimum, you should enable the following predefined policies, which address the most common security and compliance needs:
  - Security-relevant activity (ORA_SECURECONFIG), such as the use of the CREATE USER system privilege
  - Logon failures (ORA_LOGON_FAILURES)

  All new Oracle Databases, created from release 12.2 and later, automatically enable ORA_SECURECONFIG and ORA_LOGON_FAILURES policies.
- If you have highly customized legacy audit settings, then you have the following choices to transition them to unified audit policies in order of preference:
  - Create custom unified audit policies using the rich features of unified audit to make your policies more conditional, selective, and focused (recommended – the new audit facility has much more capability for fine grained, conditional, and extendible auditing).
  - If you are unfamiliar with the syntax of creating unified audit policies, you can use the syntax converter script available in My Oracle Support note 2909718.1. The converter converts legacy audit configuration settings into syntactically correct unified audit policies, writing them into a script for your review. Oracle strongly recommends that you examine the policies and incorporate the various features of unified audit, such as creating conditions or auditing application context values before you enable your policies.

After converting your legacy audit settings to unified audit policies, you should `NOAUDIT` your traditional audit policies and delete the traditional audit parameter settings. Ensure you appropriately change any purge jobs created with the DBMS_AUDIT_MGMT PL/SQL package.

Oracle LiveLabs

Oracle LiveLabs gives you access to Oracle's solutions and technologies to run a wide variety of labs and workshops at your own pace. If you want to try the solutions discussed in this chapter, please go to:

- DB Security – Unified Auditing

Summary

Oracle Database provides the industry's most comprehensive auditing capabilities, collecting detailed information on critical events and letting you meet both security and regulatory compliance goals. With practical auditing principles focused on privileged user activity, security-relevant events, and sensitive data access, administrators can provision specific, context-aware, unified audit policies. Oracle Database, Oracle Audit Vault and Database Firewall, and Oracle Data Safe cloud services provide several predefined unified audit policies, simplifying the provisioning and management process. If you still use traditional auditing, plan your transition to unified auditing soon.

Chapter Thirteen

Using and Managing Audit Data

Why is managing and using audit data important?

Simply collecting audit records is only the first part of the equation. That collected data is worthless unless it's used! You should actively review and monitor database activity because accounts are always at risk of being compromised or misused, and early detection can help reduce the damage of a compromise.

This chapter describes two approaches to managing, analyzing, and reporting from audit data and gives you the information you need to choose the best solution based on your requirements.

Why should you focus on managing and using audit data?

Organizations frequently have hundreds or thousands of databases where user and administrator activities must be audited and monitored to comply with regulations and detect security breaches. Monitoring database activity is important, but it can generate audit trails with large amounts of data. Those audit trails are scattered across many different databases, making analysis and reporting on the audit data difficult. If you have more than a few databases, you need to transfer the audit data into a consolidated repository for safekeeping, analysis, alerting, and report generation.

Database auditing is frequently augmented with solutions that collect and store audit data from your fleet of databases. Both Oracle Data Safe and Oracle Audit Vault and Database Firewall (AVDF) provide this capability. In both cases, you get audit collection across your target databases, providing a centralized audit repository. Both offer built-in information lifecycle management to control the retention and archiving of the audit data, a broad list of audit reports with flexible filter capabilities to fulfill your requirements, and alerting capabilities to notify you of specific user activities or unusual behavior. Both Data Safe and AVDF take a comprehensive approach, offering numerous benefits, including:

- Consolidation of audit data in a dedicated repository that maintains the integrity and security of audit data
- Near real-time monitoring of all database activity by first capturing audit data and then analyzing and alerting on policy violations
- Out-of-the-box and customizable reports for security and compliance
- Ability to aggregate and search audit records to support forensic analysis
- Assist with investigations of data breaches or other suspicious activity. Forensic analysis requires the ability to collect and store vast amounts of audit and network event records and search through them efficiently.
- Support storing and retrieving historical data
- Track user privilege changes
- Provide proof of monitoring critical systems and data to auditors. It's rare to encounter an organization that is NOT subject to one or more regulations requiring data protection. Most of these regulations require tracking access to sensitive data. In almost all cases, demonstrating compliance requires analysis of the audit trail and multiple reports for the auditors to review.

What is audit data management?

Database auditing is a critical control for database security and regulatory compliance. Completely aside from managing audit policy, which we discussed in Chapter 12, you'll also need to manage audit data collected as a result of those audit policies. Audit data management includes:

- Controlling the amount of space that audit data consumes in your databases
- Protecting the audit data from alteration or destruction
- Retaining audit data as needed to meet your security goals and satisfy regulatory requirements

The best way to manage your audit data is to adopt a solution that collects it into a centralized repository. Once that is done, you can remove the data from the source database and return that space to the rest of the system.

Within that centralized repository, you can protect the data from accidental or malicious deletion or alteration, further separating the audit data from those who might have access to it in the originating database.

You can also enforce data retention and lifecycle policies, archiving audit data to low-cost storage when it is not needed for analysis or reporting, and eventually deleting old audit data when it is no longer required.

How do you effectively use audit data?

Just collecting the audit data isn't enough—you need to do something with that data. Remember from Chapter Twelve that the reasons we audit database activity include:

- Identify anomalous or suspicious activity
- Support regulatory compliance audits
- Assist in troubleshooting operations

That's where analysis, reporting, and alerting come into the picture. That same repository you created to manage your audit data is effectively a data warehouse of information about database activity throughout your database fleet. That warehouse is designed to facilitate mining audit data to support post-incident forensic analysis, generate alerts on unusual or suspicious activity, provide reports on activity to those who need it, and assist in troubleshooting problems.

If your organization processes sensitive data, you are probably subject to regulations like the General Data Protection Regulation (GDPR), California Consumer Privacy Act (CCPA), Payment Card Industry Data Security Standard (PCI-DSS), Health Insurance Portability and Accountability Act (HIPAA), IRS Publication 1075, Sarbanes-Oxley Act, and the UK Data Protection Act. Almost all regulations require that you know who has accessed sensitive data and be able to demonstrate that there is a record of access. Many regulations or corporate policies require keeping these records for several years. Again, this is where your audit data warehouse comes into the picture with automation to manage the retention of data and reports.

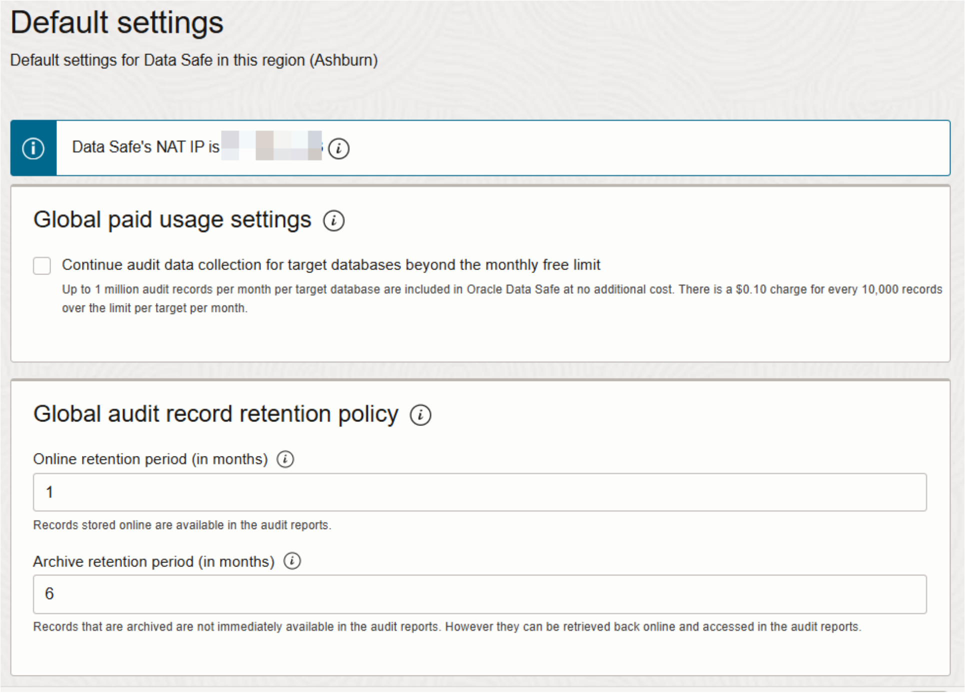
Oracle offers two solutions for managing and using audit data, both of which we've discussed earlier in this book:

- **Oracle Data Safe –** a database security cloud service running in the Oracle Cloud Infrastructure (OCI). It offers a complete solution for activity auditing, including managing the audit policies and retention periods for your audit records, a central audit collection across your Oracle Databases, providing a broad list of audit reports with flexible filter capabilities to fulfill your requirements, and alert capabilities to notify you of specific user activities or unusual behavior. These capabilities help manage the day-to-day security and compliance requirements of Oracle Databases, both on-premises and in the cloud.
- **Oracle Audit Vault and Database Firewall** (AVDF) – may be deployed on-premises or in the cloud. AVDF is a complete database auditing and network activity monitoring solution that combines native audit data collection with network-based SQL traffic capture for most of the commonly used databases, including Oracle, Microsoft, IBM, SAP, and others. It includes a highly scalable enterprise-quality data warehouse, audit data collection agents, Database Firewall, SQL Firewall, robust reporting and analysis tools, and an alert framework. AVDF's recommended auditing policies can be enabled with a single click.

This chapter discusses how you can manage the volume of audit data and create a centralized repository of audit data with accompanying analysis, reporting, and alerting.

Managing and using audit data with Oracle Data Safe

Data Safe collects and stores audit records (including SQL Firewall violations – see Chapter Eight) from your Oracle Databases. Once collected, audit data can be safely deleted from the databases that produce them, freeing up that space for use by the database. In Figure 13-1 below, Data Safe lets you configure a data retention period for your entire fleet of databases. You can specify a total time to keep audit records of up to seven years (up to one year online and up to six years in archive storage).



Default settings

Default settings for Data Safe in this region (Ashburn)

Data Safe's NAT IP is [redacted]

Global paid usage settings

□ Continue audit data collection for target databases beyond the monthly free limit

Up to 1 million audit records per month per target database are included in Oracle Data Safe at no additional cost. There is a $0.10 charge for every 10,000 records over the limit per target per month.

Global audit record retention policy

Online retention period (in months): 1

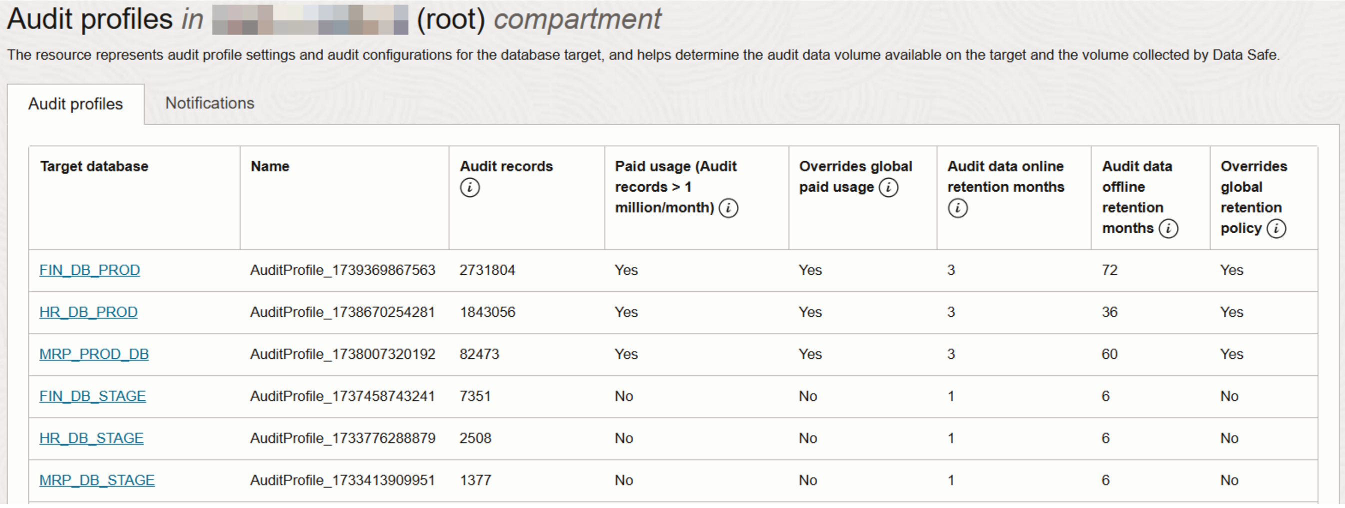
Records stored online are available in the audit reports.

Archive retention period (in months): 6

Records that are archived are not immediately available in the audit reports. However they can be retrieved back online and accessed in the audit reports.

Figure 13-1: Data Safe audit data global retention settings

As you can see in Figure 13-2, you can override the fleet setting for individual databases that may require different retention periods.



Audit profiles *in* [redacted] (root) *compartment*

The resource represents audit profile settings and audit configurations for the database target, and helps determine the audit data volume available on the target and the volume collected by Data Safe.

Target database	Name	Audit records	Paid usage (Audit records > 1 million/month)	Overrides global paid usage	Audit data online retention months	Audit data offline retention months	Overrides global retention policy
FIN_DB_PROD	AuditProfile_1739369867563	2731804	Yes	Yes	3	72	Yes
HR_DB_PROD	AuditProfile_1738670254281	1843056	Yes	Yes	3	36	Yes
MRP_PROD_DB	AuditProfile_1738007320192	82473	Yes	Yes	3	60	Yes
FIN_DB_STAGE	AuditProfile_1737458743241	7351	No	No	1	6	No
HR_DB_STAGE	AuditProfile_1733776288879	2508	No	No	1	6	No
MRP_DB_STAGE	AuditProfile_1733413909951	1377	No	No	1	6	No

Figure 13-2: Data Safe individual database retention settings

Data Safe provides interactive reports that let you track specific users, objects, or actions. It also supports forensic investigation and provides summary reports for compliance and activity reporting. These reports can be scheduled and downloaded as PDFs. Scheduled reports are useful for documenting compliance reviews. You can choose from several predefined alert policies to receive notifications of unusual activities that may indicate compromise or create your own custom alert policies.

Data Safe supports all flavors of Oracle Database as target databases, whether running in the Oracle Cloud Infrastructure, on Cloud@Customer, on-premises, or in other cloud environments like AWS or Azure.

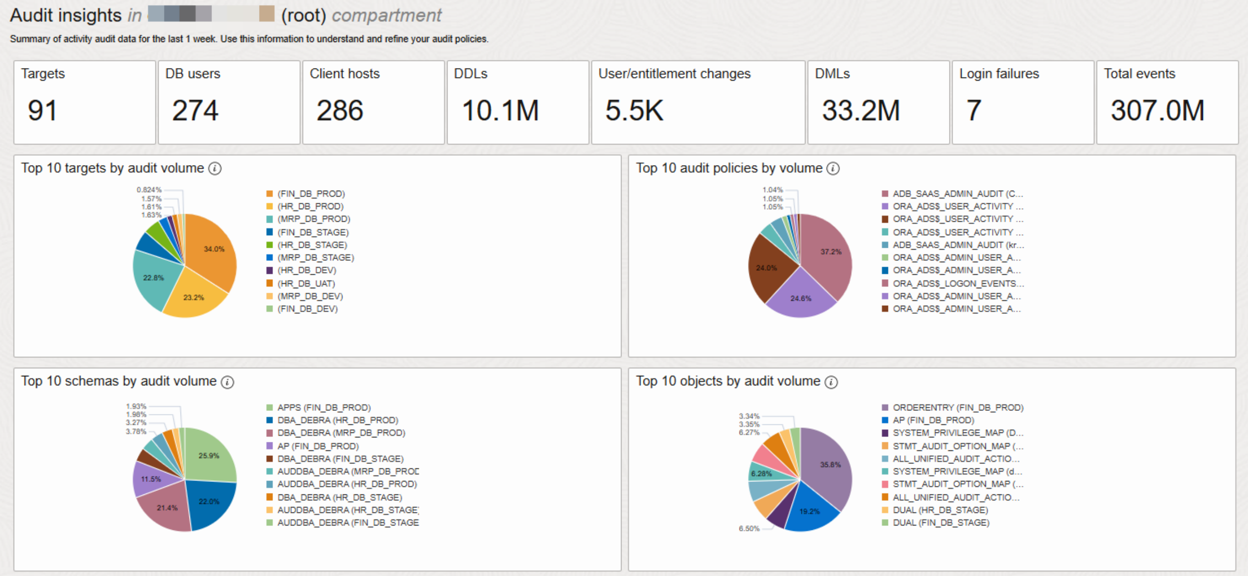
Once Data Safe provisions and enables audit policies for the target database, Data Safe collects audit records for audited activities within the target database. You can limit the volume of audit data retained in the target database using Data Safe's optional auto-purge feature (disabled by default) to delete audit data after collection.

Audit insights

Audit Insights in Data Safe provides quick insight into all the collected audit data from your target databases.

With Audit Insights, you can quickly:

- Identify increases in the volume of Data Definition Language or Data Manipulation Language statements
- View trends in login failure and user/entitlement changes
- Identify the which databases, database schemas, and objects contribute the most records to the audit volume
- Identify which audit policies generate the most audit volume
- Identify client connections and database users generating the most audit volume
- Identify if your audit policies capture enough database activity on your intended schemas and objects.
- Analyze audit data using different filters, including time period and target scope.


Audit insights *in* [redacted] (root) *compartment*

Summary of activity audit data for the last 1 week. Use this information to understand and refine your audit policies.

Targets	DB users	Client hosts	DDLs	User/entitlement changes	DMLs	Login failures	Total events
91	274	286	10.1M	5.5K	33.2M	7	307.0M

Figure 13-3: Data Safe audit insights

You can use the metrics and charts to examine your audit record volume and refine your auditing policies. Analyzing your top items by audit volume helps you identify audit policies that might be adjusted to improve the overall security of your target databases.

Activity auditing dashboard

Data Safe's activity auditing dashboard shows charts and tables summarizing audit events for the last week across all target databases. The default dashboard display gives a broad overview of the audit events for all target databases that Data Safe monitors. You can easily add filters to select different time periods, all databases in a specified compartment or even a single target database.

Audit reports

With Data Safe, you collect audit records from target databases and store the audit data centrally within the Data Safe repository. Once in the repository, the audit data is available for analysis and reporting. Data Safe includes predefined audit reports with detailed information on database activity. You can access these interactive reports for user activity tracking or forensics and summary reports for routine collection and reporting. The reports cover a wide range of activities, including privileged user activity, login activities, schema changes, entitlement changes, and sensitive data access, to name a few. Table 13-1 lists examples of predefined audit reports provided in Data Safe:

Report	Description
All activity	All audit activities
Admin activity	Database activities of admin users (as identified in Data Safe's user assessment feature)
User/entitlement changes	User creation and deletion, privilege and role changes
Audit policy changes	All changes in audit policies
Login activity	Database login attempts
Data access	Database query operations
Data modification	Data modification activities (DMLs)
Database schema changes	Database schema changes (DDLs)
Common user activity	Database activities of common users in pluggable databases
Database errors	Errors reported in the database for audited activities
Data extraction	Data Pump and RMAN activities in the database
Sensitive data activity	Database activity on sensitive objects (as identified in Data Safe's data discovery feature)

Table 13-1: Data Safe audit reports

You can set filters and modify columns within the reports to meet your needs. If you want to reuse the report with your customizations, save the modified report as a custom audit report. You can schedule audit reports (predefined and custom) to run at specific times then download the reports as PDF or XLS files.

Audit retention periods

You may need to comply with organizational policies or regulatory requirements that require you to keep your audit records for a specific time. Mandatory retention periods are frequently measured in years. The Data Safe repository for audit records consists of online storage (available for immediate reporting and analysis of the audit records) and offline storage (for long-term retention). You can configure the online and offline retention periods at the global level for all databases or at the individual database level for those databases that may need a different retention policy than most of your databases. Individual database settings override global settings.

The minimum online retention period is one month, and the maximum is twelve months. The minimum offline retention period you can set is zero months, and the maximum is 72 months for a total of up to 84 months (online+offline) – that's up to seven years! If you need to retain your audit records for more than seven years, you can submit a request with Oracle Support.

Audit data is automatically moved from online storage to offline when it reaches the end of the online retention period and then permanently deleted when it reaches the end of the offline retention period.

When you need to access audit records from the offline archive storage, you can retrieve the audit data back into the Data Safe online repository for analysis and reporting. The retrieved archives are available for up to a month before they are automatically returned to the archive, but you can manually release them back to the archive at any time.

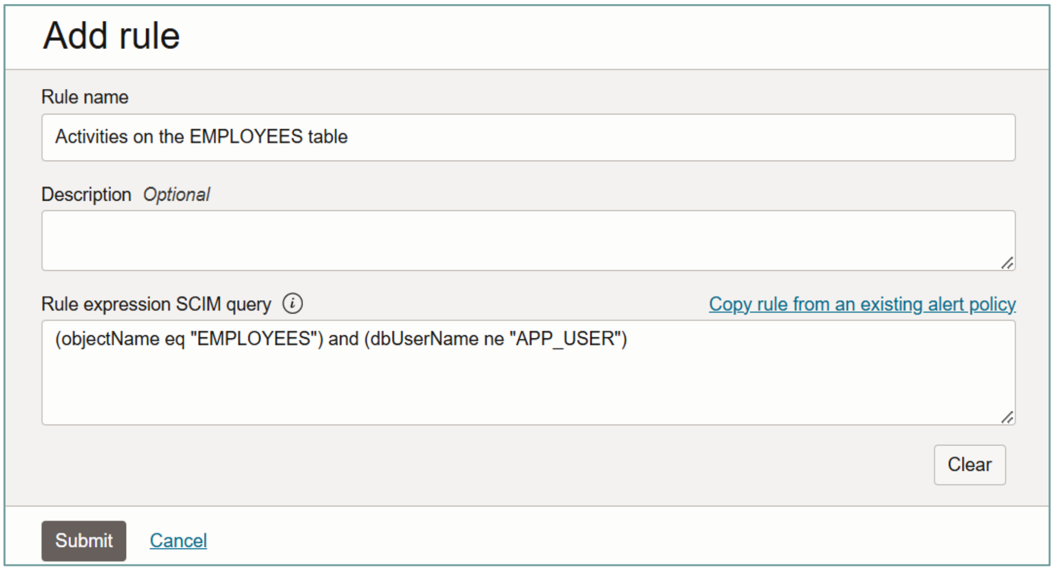
Alerts

You can enable alerts on your target databases to track and receive notifications of activities you are especially interested in. An alert message proactively notifies you when a specific audit event occurs on a target database. The alerts you receive depend on the alert policies that you enable in Data Safe. Data Safe has several preconfigured alerts that just need to be enabled if you want to be notified when these conditions occur – a list of those alerts is in Table 13-2.

Audit policy	Severity level
Failed logins by admin users	Critical
Profile changes	Critical
Audit policy changes	High
Database parameter changes	High
Database schema changes	Medium
User creation/modification	Medium
User entitlement changes	Medium

Table 13-2: Alert policies in Oracle Data Safe

You can create custom alerts that will monitor the incoming stream of audit data and apply a filter to spot alert conditions. Figure 13-4 shows an example of an alert generated when an audit record involving the EMPLOYEES table where the user performing the action is not the application user.



Add rule

Rule name: Activities on the EMPLOYEES table

Description *Optional*:

Rule expression SCIM query: (objectName eq "EMPLOYEES") and (dbUserName ne "APP_USER")

Figure 13-4: Data Safe example of a custom alert policy

Data Safe implements the flexible and widely used System for Cross-domain Identity Management (SCIM) query language for alert definition and report filters.

Data Safe also includes a built-in alert dashboard to view and manage alerts. You configure notifications by leveraging the corresponding Data Safe event in OCI's Rules and Notifications.

Auditing with Audit Vault and Database Firewall (AVDF)

Oracle Audit Vault and Database Firewall (AVDF) is a scalable and flexible database auditing and network activity monitoring solution that consolidates audit data from databases, operating systems, directories, file systems, and applications into a single repository for analysis, alerting, and reporting. AVDF is a full-stack software appliance you can deploy on dedicated hardware or virtual machines.

AVDF supports the most common database types: Oracle Database, MySQL, Microsoft SQL Server, SAP Sybase, IBM Db2 LUW, PostgreSQL, and MongoDB. Almost anything that produces audit data in a standard format is supported using audit collection plugins that can collect audit data from database tables and CSV, XML, or JSON files. An included software development kit (SDK) accommodates targets that cannot be accessed using the quick collector framework format.

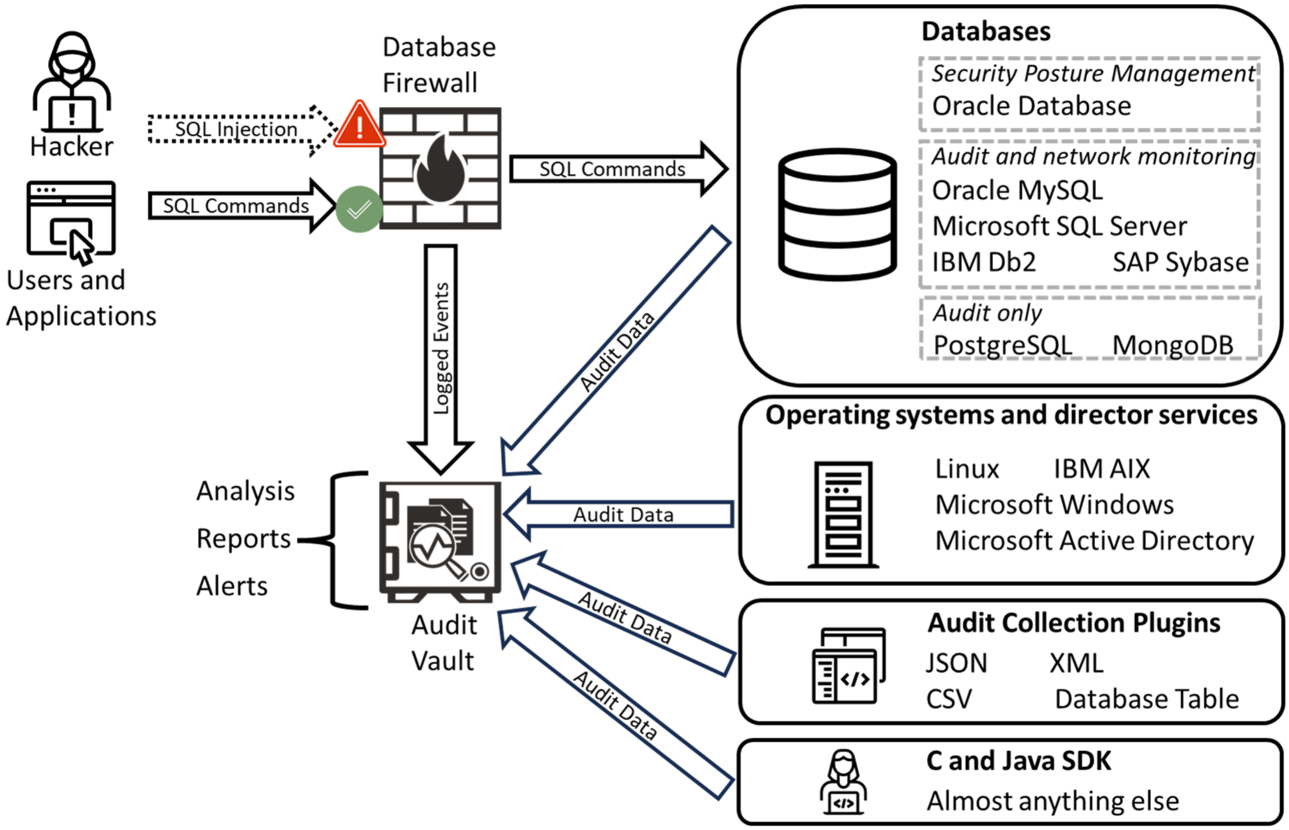


Figure 13-5: Oracle Audit Vault and Database Firewall

External monitoring

In Chapter Twelve, I briefly discussed external monitoring and pointed out some of the disadvantages of external monitoring when compared with using the database's native audit capability. Despite those disadvantages, external monitoring still has an important place in your security portfolio. Auditing is very good at capturing the information you tell it to capture. We've seen that audit policies are versatile and can be carefully crafted to capture only information you are interested in, reducing the "noise" in your audit trail. But suppose your security goal is to collect data on clients connecting to your database that have never connected before? New SQL commands that the database has never seen before? It's difficult to create an audit policy that does that, but if we leverage the same security control we discussed in Chapter Eight we can easily have our Database Firewall examine and log anything that is not explicitly included in its policy allow-list. As mentioned earlier, the collected information will lack session context – but that isn't a problem for this particular use case since we just want to detect the anomaly.

Activity data collection

The Audit Vault Server collects data from multiple sources, including audit data from databases, operating systems, and application tables or files. Events logged by the Database Firewall and policy violations detected by the SQL Firewall are also captured.

The Audit Vault server maps collected data into a normalized format and stores it in the Audit Vault repository – a specially configured Oracle Database. This normalized format allows you to produce a single report across different types of databases and even non-database targets like operating systems, applications, and directories.

For Oracle and Microsoft SQL Server databases, AVDF can also capture before/after values, supporting data governance with historical records of who changed data and both the old and new data values. For Oracle Database, AVDF also assesses security configuration (including drift detection), tracks user entitlements, and watches for stored procedure changes (we discussed the importance of monitoring for stored procedure changes in Chapter Three). If you're unsure what to audit, AVDF includes recommended audit policies for Oracle Database and can provision those policies with a few simple mouse clicks.

The Audit Vault agent retrieves audit data from audit trail files for various targets, such as databases and operating systems, and sends the audit data to the Audit Vault server. The Audit Vault agent periodically connects to the target to see if new audit records have been generated (since the last retrieval) and sends those records to the Audit Vault server. A single Audit Vault agent can support audit collection from multiple targets of different types. Audit Vault also supports agentless collection of audit data from Oracle Databases and Microsoft SQL Server databases. With agentless collection, Audit Vault collects data directly from the targets instead of deploying an Audit Vault agent on the target host machines.

Audit reports

Audit Vault provides dozens of out-of-the-box reports on database activity, privileged user activity, schema changes, entitlement changes, stored procedure modifications, before/after data value changes, and login failures. There are also predefined compliance reports for GDPR, PCI-DSS, SOX, HIPAA, Gramm-Leach-Bliley Act (GLBA), DPA, and IRS Publication 1075.

AVDF reports provide detailed information on database activity events gathered from audit data and the Database Firewall (see Chapter Eight for more details on the Database Firewall). The reports cover a wide range of activities, including all audit and network activity, privileged user activity, schema changes, entitlement changes, stored procedure modifications, before/after data value changes, sensitive data access, and login failures. Table 13-3 summarizes the different types of reports in AVDF.

Report type	Description
Activity	These reports track database access activities such as audited SQL statements, application access, and user login activities. Some typical activities are: • All activity • Data modifications • Before and after values of modified data • DDL activity • Failed logins
Alerts	Alert reports display the raised alerts and their status.
Stored procedure audit	This report tracks changes made to stored procedures, such as create, replace, and drop.
Oracle Database Firewall	These reports give detailed event information about the SQL traffic Database Firewall monitors. For example, you can see details of statements that had warnings or were blocked according to the database firewall policy.
User entitlements	These reports show information on user privileges for Oracle Databases. Using AVDF, a baseline can be created, and it's then easy to report on drift away from the baseline. Drift detection and reporting making it easy to monitor and investigate changes.
User correlation	For Oracle Database targets running on Linux, these reports let you correlate events on the database with the original Linux OS user. This is useful in cases where the organization is following the best practice of having users login to the operating system using a named account and then becoming special purpose users like the database owner (usually ORACLE) by using su or sudo.
Database Vault activity	The Database Vault Activity report shows Database Vault events, which capture policy or rule violations and unauthorized access attempts.
Anomalous activity	These reports pertain to new or previously dormant users accessing the system, or users accessing the system from IP addresses not seen before.
Assessment Reports	These reports contain security assessment data from Oracle Databases and provide recommendations that help secure your Oracle Database system. Assessment reports also include drift reporting against a security baseline.

Table 13-3: Built-in reports in AVDF

Auditors access AVDF reports interactively through a web interface. They may save reports in PDF or XLS format and schedule AVDF to run and deliver reports via email automatically. The AVDF repository (an included Oracle Database) uses an open and documented schema so that you can use the reporting tools of your choice directly against the AVDF repository. The open schema also makes integrating AVDF with tools like Splunk or your SIEM easy.

Audit Insights

The Audit Insight dashboard provides a bird's-eye view of the top user activities captured through auditing or network-based event collection across one or multiple databases, with the option to drill down for further analysis. You can drill down from the summarized view for more detailed information. The snapshot of Audit Insights below shows the clickable summarized view of all events with different contexts, including database targets, users, privileged users, event types, etc.

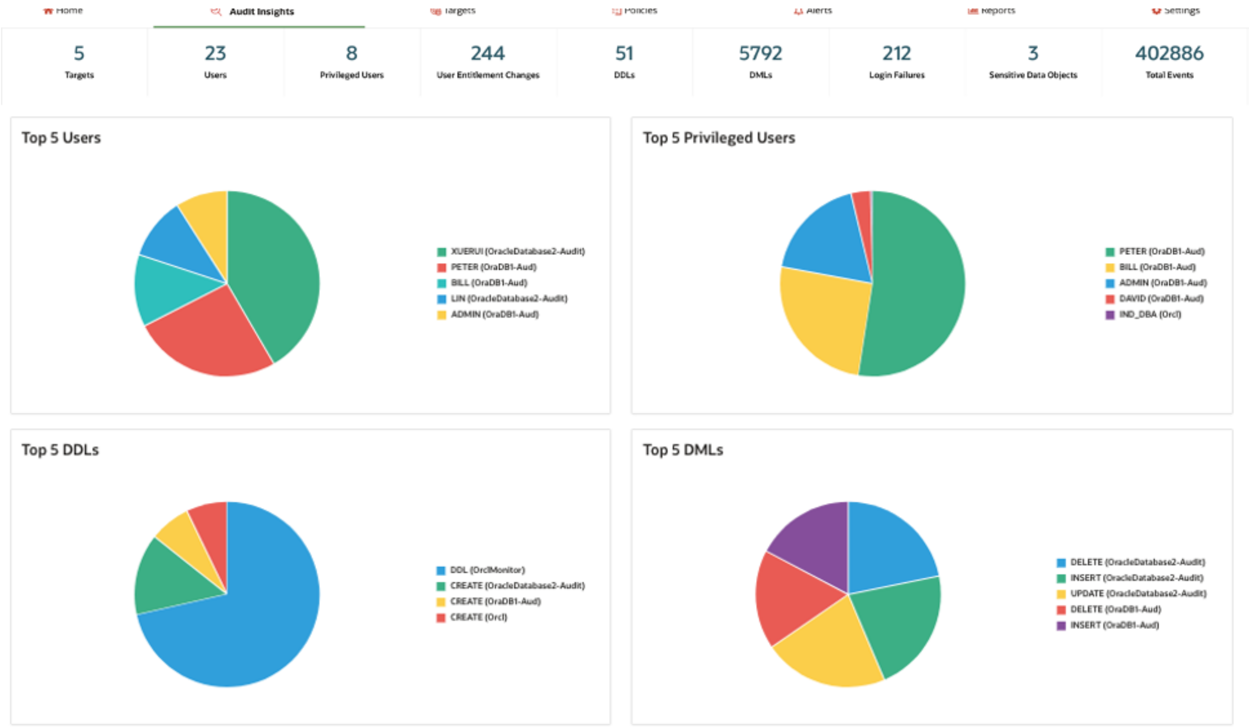


Figure 13-6: AVDF Audit Insights

Before/after data value changes

One of the unique capabilities of Audit Vault and Database Firewall is the ability to track changes to data, recording both the original and new values. If a data value is changed, AVDF records the old value (before the change) and the new value (after the change), along with the user and time. Before/after value tracking is extensively used in the healthcare and financial services industry and many other regulated industries where protected data attributes require a high degree of data governance. Some regulatory bodies (including India's Ministry of Corporate Affairs) require before/after value tracking for financial data. Auditors can use before/after value tracking to validate the lifecycle of individual data attribute changes.

Before/after value tracking is available for Oracle and Microsoft SQL Server databases.

AVDF special support for Oracle Database

As you've seen, audit collection works for almost anything that produces an audit trail. Network event collection is possible for most common database types. However, there are a few things that AVDF does only for Oracle Database. These fall under the heading of Database Security Posture Management (DSPM).

Provisioning audit policies

Deciding what to audit depends upon your organization's policies. Typically, you will want to audit privileged user activity, security-relevant events like user creation, privilege or role grants, and changes to the database structure. You may also want to audit access to sensitive data, especially if that access is from outside of the application.

Creating audit policies is a balancing act – you need to collect enough information to meet regulatory requirements and support incident investigations. At the same time, you want your audit policies to be selective enough to reduce unnecessary audit records, minimize performance impact, and control storage costs.

AVDF makes it easy to provision audit policies by providing a baseline of recommended audit policies for Oracle Database. You can enable those recommended policies with a single click.

In addition to the recommended predefined policies, AVDF makes it easy to enable pre-configured compliance-related policies for standards like CIS and STIG.

Monitor database security configuration and detect configuration drift

In Chapter Three, we discussed using AVDF to assess database security and detect configuration drift. AVDF also provides fleet-level reporting of configuration issues, making it easy for you to detect security configuration concerns and identify databases that need attention.

Monitoring and drift detection for user entitlements

In Chapter Three, we discussed using AVDF to monitor user entitlements and track drift in privilege and role assignments. AVDF produces reports on what privileges or roles a database user has and can compare one report against a baseline or earlier report and list what's changed. Being able to certify entitlements once and then shift to monitoring changes reduces the workload (and accompanying chance for human error) in privilege reviews.

Drift detection for stored procedures

Stored procedures often contain complex business and data access logic. Attackers sometimes modify stored procedures to create backdoors to the database. Monitoring stored procedures to detect changes can be a lifesaver, especially if you suspect someone has gained access to the system. AVDF can periodically check the Oracle Databases for any new stored procedures or updates to existing stored procedures, and the reports can be reviewed for any changes.

Sensitive data discovery

In Chapter Four we discussed using AVDF to discover and classify sensitive data, including reporting on the volume of different types of sensitive data contained within the database.

Oracle LiveLabs

Oracle LiveLabs give you access to Oracle's solutions and technologies to run a wide variety of labs and workshops at your own pace. If you want to try the solutions discussed in this chapter, please go to:

- Get started with Oracle Data Safe fundamentals → Lab 6: Audit database activity
- Integrate Oracle Data Safe with Applications and Services → Labs 3, 4 and 5
- DB Security – Audit Vault and DB Firewall

Summary

Database auditing is essential, but if you do nothing with the information you collect, you'll miss the chance to identify problems early on. Oracle Data Safe, and Oracle Audit Vault and Database Firewall help you manage the volume of audit data, creating a centralized repository for analysis, reporting, and alerting.

Oracle Data Safe provides a unified control center for database security for all your Oracle Databases. It offers a complete solution for activity auditing, including managing the audit policies and retention periods for your audit records, a central audit collection across your target databases, providing a broad list of audit reports with flexible filter capabilities to fulfill your requirements, and alert capabilities to notify you of specific user activities or unusual behavior.

Oracle Audit Vault and Database Firewall (AVDF) is a complete database auditing and network activity monitoring solution that combines native audit data collection with network-based SQL traffic capture. It includes a highly scalable enterprise-quality data warehouse, audit data collection agents, Database Firewall, robust reporting and analysis tools, and an alert framework. Users can leverage AVDF's recommended policies to enable database auditing quickly with a single click.

Chapter Fourteen

Database Security in a Multicloud Environment

Why is multicloud important?

A multicloud cloud computing strategy leverages cloud computing services from two or more cloud service providers. You may have also encountered another term, hybrid cloud. A hybrid cloud strategy mixes on-premises resources with cloud services. Mixing hybrid and multicloud strategies is common, running some services on-premises and others scattered across different cloud providers. Although the proper term for this would be *hybrid multicloud,* most people just shorten it to *multicloud*.

You probably work in a multicloud environment. Almost all organizations have at least some systems in the cloud. Some organizations have moved completely to the cloud, but most large organizations still retain some systems on-premises and have no current plans to move those systems to the cloud. There can be lots of reasons for this – you might be trying to reduce costs, take advantage of best-in-class services, avoid cloud provider lock-in, improve redundancy, optimize performance, and comply with data sovereignty requirements. By leveraging the strengths of multiple cloud providers, you can tailor your cloud environments to meet your specific requirements and maximize the benefits of cloud computing.

This chapter reviews the security and management of Oracle Database in a multicloud environment.

Why should you focus on multicloud?

Multicloud strategies lead to scenarios where one cloud manages identity, another manages applications, the third has security tools, and yet another contains the data. This is a common scenario, with different cloud providers selected based on how well and at what cost they provide a service.

For example, you may have databases running within Oracle Cloud Infrastructure (OCI), applications running within Amazon Web Services (AWS), and your business intelligence services running in Microsoft Azure. This leads to scenarios like the one shown in Figure 14-1 where identity is managed in one cloud, your applications are in a second cloud, data is stored in a third cloud, and your security and analytics tools could be in a fourth and even fifth cloud.

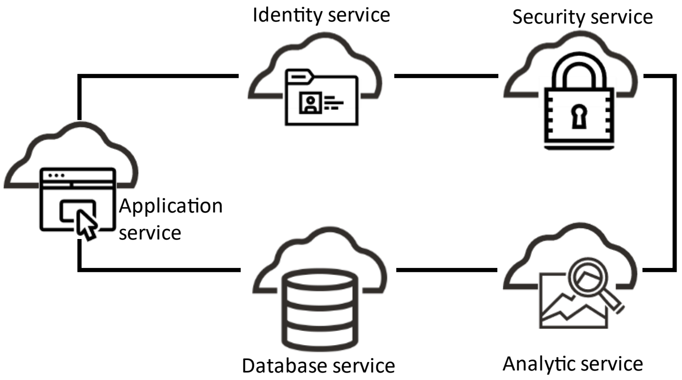


Figure 14-1: Multicloud environment

One of the great things about working with Oracle Database security is that, for the most part, Oracle Database security works the same in any cloud, and the tools and services that support Oracle Database security are also cloud agnostic.

Identity cloud service

Although your services may span multiple clouds, you'll usually want one cloud provider to handle your enterprise identities. As we discussed in Chapter Five, Oracle Database includes native support for Microsoft Entra ID and Oracle

Cloud Infrastructure IAM. Other cloud identity services can usually work with Oracle Database using one of the supported authentication protocols. For example, cloud identity service provider Okta supports Oracle Database using RADIUS. Your multicloud applications and tools will either accept identity tokens issued by your identity provider of choice or perhaps allow federation between your identity provider and their cloud identity service. Having a single identity source of truth lets you centrally manage authentication and authorization for cloud users as you may already do for users in your on-premises applications and databases. Your users are probably accustomed to single sign-on (SSO) and don't want to use multiple identities with different credentials to access their applications. There are different approaches to cloud identity integration, such as:

- Design your cloud network to be an extension of your on-premises network and continue using your existing on-premises identity solution (e.g., Microsoft Active Directory) for cloud solutions. While this has the advantage of leveraging existing skills and knowledge, it doesn't take advantage of cloud identity features like forwardable tokens and cross-platform authentication.
- Synchronize your on-premises identity solution into the cloud for cloud resources. Your users will use the cloud identity solution to access cloud resources and the on-premises identity solution to access on-premises resources. Users are typically managed in the on-premises or cloud environments and synced to the other environment.
- Migrate your on-premises identity solution into the cloud. As part of this migration, your on-premises resources will connect with the cloud identity solution and any new cloud resources.

As part of the criteria for selection of an identity cloud service, you'll need to evaluate whether your users and resources will directly interact with the identity service or if it will federate with another identity service – or maybe multiple identity services depending on where your applications, tools, and data reside.

Managing security across clouds

Managing the security of your Oracle Databases was more straightforward when they were all within the boundary of your on-premises data center. In the multicloud, your Oracle Databases can be in the Oracle Cloud Infrastructure (OCI), a non-OCI cloud like Azure, GCP, or AWS, and on-premises. Managing the security of your Oracle Databases across cloud boundaries shouldn't require you to work with multiple tools requiring different skill sets just because the databases are in different environments.

Oracle helps you manage security for your Oracle Databases across the multicloud. First, security features you're familiar with in your on-premises Oracle Databases also work on your Oracle Databases in the cloud. These capabilities include Transparent Data Encryption (TDE), Database Vault, Label Security, Data Masking and Subsetting, Real Application Security, Virtual Private Database, and many more.

Note: Managed database services, like Amazon RDS, may restrict the use of some features. For example, RDS does not allow Database Vault.

Second, operating in a multicloud environment enables you to take advantage of data security services that work across clouds. We've discussed several of these throughout this book:

- Data Safe – runs in Oracle Cloud Infrastructure but can be used with Oracle Databases anywhere, including in OCI, AWS, Azure, GCP, and on-premises. Data Safe is included with all Oracle Database cloud services, including both Autonomous Database and Exadata Cloud Services running in Azure, AWS, and GCP. Chapters Three, Four, Eight, Ten, and Thirteen discuss Data Safe in more detail.
- Key Vault – available in the OCI marketplace. May also be installed on-premises, in AWS, GCP, or Azure. If there is reliable network connectivity, a Key Vault node installed in one location can service databases in other locations. Chapter Eleven discusses Key Vault in more detail.
- Audit Vault and Database Firewall – available in the OCI marketplace. May also be installed on-premises or in AWS. Again, if there is network connectivity, Audit Vault and Database Firewall installed in one location can

service databases in others. Chapters Three, Four, Eight, Twelve, and Thirteen discuss Audit Vault and Database Firewall in more detail.

Oracle Database is available in most clouds

Managing resources separately in every cloud is demanding, can be expensive, and increases security risk due to the complexity introduced by different tools and capabilities. Oracle recognizes this and has a solution to ease this management.

Oracle partners with most major cloud service providers, including AWS, Azure, and Google Cloud (GCP), offering Oracle Autonomous Database and Oracle Exadata Database Service (both of which include Oracle Data Safe as part of the service) as native services within those clouds. Oracle Database Services on OCI running inside another hyperscaler's data centers let you run the entire application stack in one cloud and eliminate the network latency and cost of moving large amounts of data across clouds. Oracle manages the private network connection between an OCI child site running on hardware installed within the hyperscaler's data center and an Oracle data center in the region, allowing other OCI services direct access to the Oracle Database Services in the respective hyperscaler for integration.

Multicloud service access

Services need the ability to connect securely across clouds without the overhead of using, storing, and rotating passwords. Creating a trust relationship between clouds and using service principals to connect one cloud service to another cloud service provides high security without the work of managing the credentials separately in each cloud. Databases also need to connect to external data sources like object store to represent the data in a structured way and allow database tools to analyze the data. If a database cannot connect to an object store in another cloud, you must move the data between clouds to access it, adding cost and complexity. The Oracle Autonomous Database can use identity tokens from cloud authentication services offered by Azure, Google Cloud, and Amazon Web Services to connect to resources like object storage in those clouds, freeing you from the need to transfer data from one cloud to another. Multicloud service access capabilities are one of the fastest evolving areas in database security – you should periodically check this blog for updates.

https://blogs.oracle.com/database/category/db-security

Multicloud identity management for Oracle Database

Enterprise users enjoy single sign-on integration with their various web applications, and their experience shouldn't be any different when they need to access services in other clouds, including Oracle Databases. To facilitate this, you can configure Oracle Database to accept tokens from Microsoft Entra ID or Oracle Cloud Infrastructure Identity and Access Management (IAM). We talked briefly about cloud-based authentication services in Chapter Four and will discuss it in more detail below.

Oracle Cloud Infrastructure Identity and Access Management (IAM)

IAM Users, OCI services, resources, and applications hosted on the OCI platform can use OCI-IAM credentials to get an IAM database token that lets them access an Oracle Database in OCI. IAM tokens are more secure than simple database passwords and can leverage IAM multifactor authentication when a user requests a token. If you work with an application or service using a different cloud identity provider, the other provider can be federated with IAM so users, applications, or services can still authenticate with the non-OCI identity service but can then use IAM database tokens to access the OCI databases.

Oracle Database clients can accept IAM tokens through the client API, from the file system, or they can directly get the token from IAM. Users need only log in to their favorite tool, and the properly configured client will retrieve the database token from IAM for the user. If the user doesn't have a current session token, IAM will have the user log in securely.

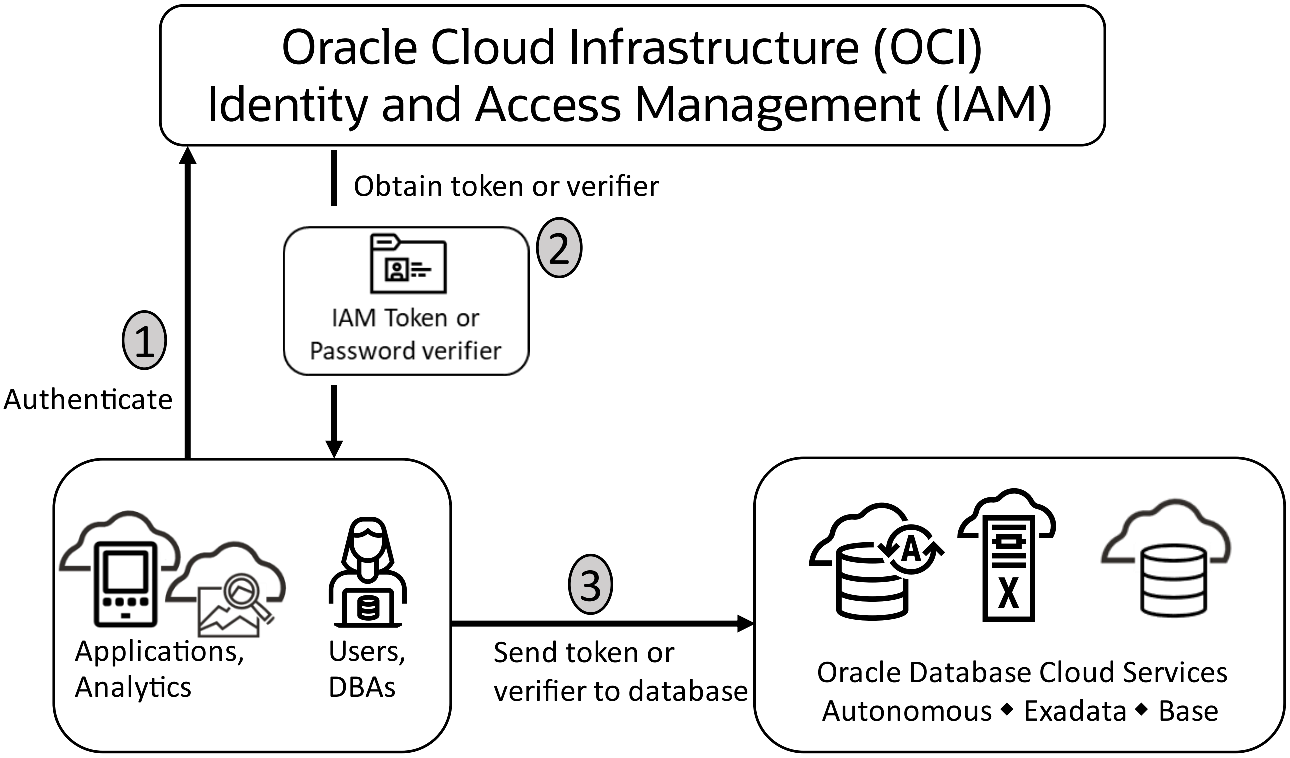


Figure 14-2: OCI database integration with IAM

Microsoft Entra ID Integration

You can use Entra ID OAuth2 access tokens to access Oracle Databases in OCI, AWS, Azure, GCP, or on-premises from applications hosted on Azure or Azure services. These services and applications can pass Entra ID managed identities (service principals) to the database via an OAuth2 token, so the service or application connects to the database as the service or application. Azure services and applications running in the Azure cloud can also request on-behalf-of (OBO) tokens for their logged-in user and then access Oracle Database as the application user. This end-to-end authentication enables enforcement of user access controls and activity auditing within the database.

Oracle Database clients get Entra ID tokens in much the same way as they do OCI IAM tokens. Entra ID access tokens can be passed through the client API, retrieved from the file system, or the database client can request the token directly from Entra ID using the Entra ID user's session token. The client can also direct the user to authenticate with Entra ID if the session token doesn't exist.

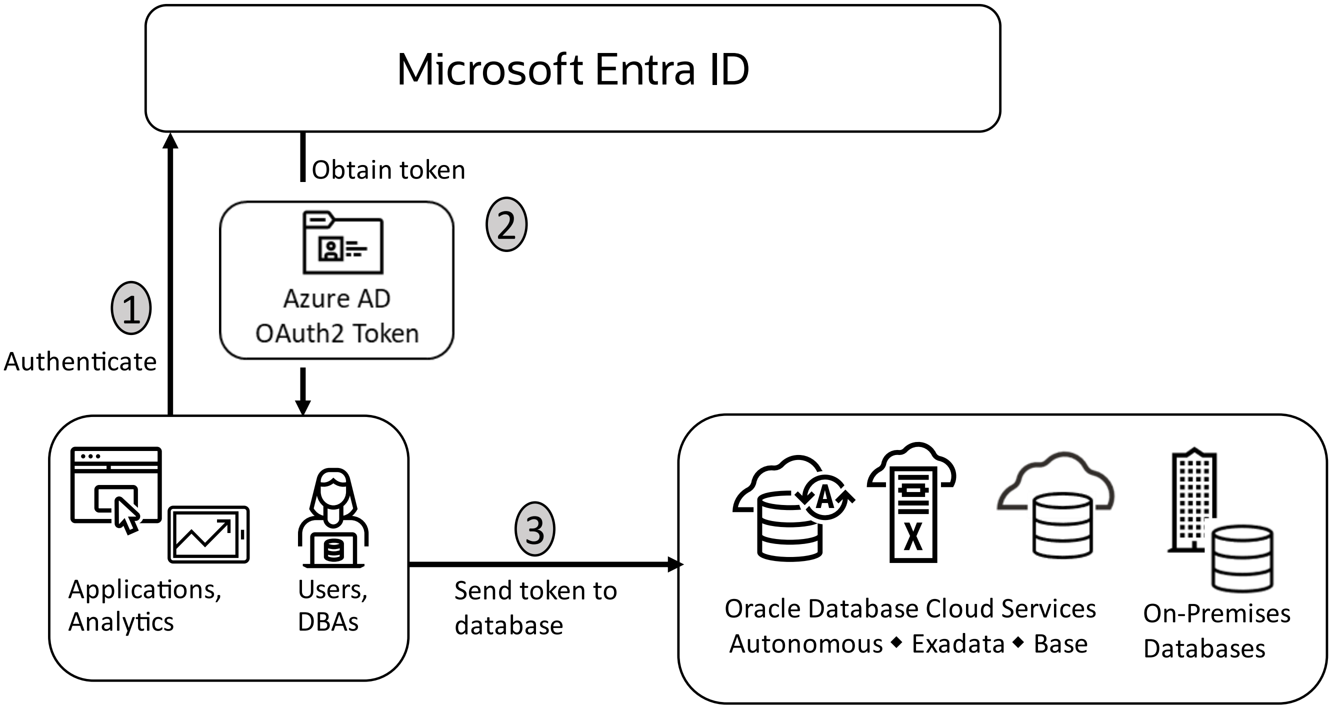


Figure 14-3: Oracle Database integration with Azure AD

Microsoft Power BI single sign-on

You may want to use Microsoft's Power BI tool to analyze application data stored in the Oracle Databases in OCI. You probably also want single sign-on access to the Power BI tool and the data. Before multicloud identity integration, you had to migrate their data into another cloud environment before they could analyze it. With Oracle's integration with Power BI, you can use Entra ID single sign-on and Power BI to access Oracle Database, whether you store your data in OCI or on-premises.

Oracle LiveLabs for multicloud integration

Oracle LiveLabs gives you access to Oracle's tools and technologies to run a wide variety of labs and workshops at your own pace. If you want to try some of the technologies discussed in this chapter, please go to:

Simplify User Management by integrating Oracle Autonomous Database and Oracle Cloud Identity

Summary

You have many reasons that drive you towards multicloud solutions. Oracle Database cloud services are available in most of the major hyperscaler clouds, and include security features and services like Data Safe and Transparent Data Encryption. Keeping your data safe and integrating Oracle Database in your multicloud helps you continue operating without forcing artificial security and data boundaries on your operation.

Chapter Fifteen

Ransomware and Zero Trust

Why is ransomware important?

Most cybersecurity advisory organizations, including the European Union's ENISA, the United State's CISA, and the United Kingdom's NCSC, rate ransomware as the most immediate and disruptive threat to organizations and infrastructure. Ransomware is the most common tactic used by criminals to generate income and is increasingly used by malicious nation-states to inflict damage on their enemies, distract attention from other attacks, and fund economies damaged by international sanctions.

Why should you focus on Zero Trust?

Zero trust is generally accepted as the most effective approach to preventing ransomware attacks. Zero trust assumes no implicit trust is given to any IT interaction, regardless of where that interaction originates from or the authenticated identity performing the action. Zero trust recognizes that any account can be compromised and implements controls to reduce reliance on any single point of trust.

What is ransomware?

The simplest definition of ransomware is a type of malicious software (malware) that encrypts a victim's files and offers to provide a decryption key in return for some form of compensation (usually monetary).

Ransomware has been around for a long time—the first known ransomware attack happened in 1989! In that attack, Dr. Joseph Popp, a biologist, attended a World Health Organization AIDS conference and gave malware-infected floppy disks to 20,000 fellow researchers in more than 90 countries. Back then, the attack was called the AIDS Trojan—it would encrypt a victim's files and demand a payment of $189 to decrypt them. This was long before the dark web – victims had to send money via snail-mail to a post-office box in Panama.

The term "ransomware" became widely used in 2005, a year after GPcode ransomware was released (believe it or not, GPcode is still in circulation today!). GPcode was primarily distributed through phishing schemes that tricked victims into thinking they were applying for a job. The original version of GPcode would encrypt a user's "My Documents" folder and demand a ransom in return for the decryption code.

The use of ransomware exploded around 2015, thanks to the increasing use of cryptocurrencies. Cryptocurrencies, such as Bitcoin, are difficult to trace, making them an ideal payment method for ransomware attacks. Furthermore, as the value of breached data declined and finding buyers became more challenging, ransomware emerged as a convenient means for cybercriminals to profit swiftly, inflicting immediate harm on victims to extort payment. In addition, the rise of ransomware-as-a-service (RaaS) platforms made it easier for cybercriminals to create and distribute ransomware, leading to a sharp increase in the number of ransomware attacks.

In recent years, ransomware attacks have become increasingly sophisticated, with attacks focused as much on servers as on users' workstations. Most ransomware variants now encrypt files and exfiltrate them to a remote server. This means that even if a victim pays the ransom, there is no guarantee that their data will not be misused or sold in underground markets. Further, paying ransom is no guarantee that victims can recover their files.

Most organizations identify ransomware as the most serious cybersecurity threat facing their organization.

Protecting databases from ransomware

In a perfect world, we would have 100% effective solutions to block ransomware – after all, it's been a problem for over 35 years! Unfortunately, ransomware continuously evolves to produce better outcomes (for the ransomware gangs). As American football coach Vince Lombardi is often quoted, "Offense only needs to succeed once. Defense has to succeed every time." That means that all too often, even the most prepared organizations experience a ransomware attack they are unable to block. That's why most organizations adopt a two-pronged strategy for dealing with ransomware:

1. Block attacks
2. Minimize the damage from attacks that are not blocked

Let's start by focusing on the second objective – minimizing the damage from a successful attack. For the most part, we start there because it's where Oracle can help the most – Oracle isn't in the end-point or network protection business. But Oracle IS the name you should think of when it comes to ensuring data isn't lost, protecting data from theft or misuse, and safeguarding against data corruption!

At the organizational level (not just the database), ransomware disrupts business, damages reputations, increases security costs, exposes the organization to legal actions and regulatory fines, results in lost opportunities – the list goes on and on. At the database level, the ramifications of a ransomware attack are a bit easier to focus on:

- Destruction of data (denial of service)
- Theft of data (data breach)

Mitigating the risk of data destruction

One common risk to the database from ransomware is "simple" denial of service. The ransomware renders your databases unusable (most often by encrypting the underlying data files), and to get your data back, you need to either pay the ransom and hope you get a decryption key that works or restore from backup. "Simple" is in double quotes because the post-ransomware scenario is anything but simple! If the ransomware was effective, you're dealing with a situation where the database and servers have been rendered useless. And it usually won't be one database server, it will be many of them – maybe even ALL of your database servers. So, your recovery plan needs to go far beyond the database to include how you are going to remove the malware from your network, reinstall operating systems, reconnect/restore storage, and only THEN can you start thinking about restoring the databases.

Once you get around to restoring the databases, unless you are one of the rare organizations using monolithic systems with no references to data in other databases, you'll need to worry about data consistency across those databases. The backups were almost certainly taken at different times, perhaps even on different days, meaning that transactions will be in some systems and missing from others.

The need to recover from data destruction, combined with the need to synchronize recovered data across different databases, drives three requirements:

1. Immutable backups: Ransomware has become proficient at seeking out and destroying backups. After all, the ransomware developers are doing their best to influence you to pay the ransom, and one way to do that is to remove your ability to restore your systems from backup. That means ransomware is now fiendishly effective at finding and destroying traditional backups. The answer to this is an *immutable* backup—a backup that cannot be deleted or corrupted. Most of the larger backup vendors now offer some variation on the concept of an immutable backup, and Oracle is no exception. Immutable backups are a capability of Oracle's ZFS Storage Appliance (ZFS appliance), Zero Data Loss Recovery Appliance (ZDLRA), and the Oracle Database Zero Data Loss Autonomous Recovery Service (ZRCV). Check with your backup system vendor if you are not sure about it's capabilities.
2. Synchronized recovery point objectives: When multiple databases are destroyed, restoring business operations (not just the individual databases) means that all of those databases need to be recovered and have transactions synchronized across them so that they are recovered to the same point in time — preferably to the last committed transaction. Unlike immutable backups, recovering databases with zero data loss is a specialized capability only present in ZDLRA and its cloud equivalent, the ZRCV.
3. Infrastructure to host the recovered systems: It's helpful to think of ransomware recovery as a disaster recovery scenario, not purely a database recovery effort. Do you have a warm recovery site with servers standing by to restore operations? Recovery sites are not a new requirement – the idea of hot/warm/cold disaster recovery sites dates back decades. What is new are some of the options available in our new cloud-

centric computing environment. Cloud service providers, including Oracle Cloud Infrastructure, now offer secure enclaves – separated from production systems by network air gaps and running continuous incremental restore operations to help you recover quickly in the event of a catastrophe. Recovery clean rooms are another common approach. Clean room used in this context refers to highly isolated environments dedicated to data and application recovery. The idea behind a clean room is to have a minimally viable IT infrastructure up and running very quickly after a ransomware incident.

Mitigating the risk of data theft

Data theft (and the threat of exposing that stolen data) is one of the most significant advances in ransomware in the past few years. Many recent ransomware attacks no longer involve data destruction at all – the attack is pure data theft differing from a traditional data breach only in how the criminals monetize the stolen data. Instead of selling the data on the dark web, the ransomware gangs extort the victim, forcing them to pay to avoid having the data released.

Ransomware gangs employ a variety of tactics to monetize data collected in ransomware attacks:

- Simple extortion – threaten the victim with publishing the data if the ransom is not paid.
- High-pressure publicity – contact data subjects of the stolen data (customers, employees, partners, and suppliers) and inform them that the data has been stolen. Instruct them to urge the victim to pay in order to keep the data private. In some cases, the gangs have been observed shifting their extortion focus to the data subjects.
- Traditional sales of stolen data – data, especially identity-related data, can still be sold on the dark web.

The solution for mitigating this risk lies in the method used by ransomware to collect the data. As of the time I'm writing this, no known ransomware variant directly targets databases. The ransomware collects data from the servers it infects by simply scraping data files from storage. That means encrypting the database (using Transparent Data Encryption) is usually effective in mitigating this threat. A word of caution though – encrypting the database doesn't do much good if the same attack that scrapes the data files from the server also captures the database encryption keys! You should also protect your keys by storing them off-server in Oracle Key Vault or OCI Vault. Refer to Chapter Eleven for more information on encryption and key management.

Note: If you do not properly protect the database's encryption keys, you might open yourself up to one of the simplest ransomware attacks! The attacker just needs to remove the keys from your system, and you will not be able to access data that you encrypted using those keys.

Database encryption is commonplace enough that your databases are probably already protected from this type of out-of-band attack.

Note: Out-of-band attacks circumvent database session controls – attacking the underlying data files or backups of the database or sniffing data as it travels over the network. Encryption is the primary strategy used to mitigate the risk of out-of-band attacks.

Plan for success

You've already covered the failure scenario with a tactical combination of encryption and zero data-loss recovery. Now, let's turn to the much more challenging and strategic issue of blocking the ransomware from succeeding. This requires keeping the ransomware from infecting your database servers in the first place and will involve a holistic approach to security. The current state-of-the-art approach to holistic information security is known as zero trust.

What is Zero Trust

There is a really good chance your organization is already engaged in a zero-trust project. The concept of zero trust has been around since around 2010 when then-Forrester research analyst John Kindervag coined the term to describe an environment where data breaches are minimized or eliminated by removing implicit trust from the IT landscape. Adoption of zero-trust principles is now commonplace in security-conscious organizations.

Here is a simple example of zero-trust thinking – should you trust an attempt to connect to your database simply because that attempt originates from within your firewall? Of course you shouldn't! Today, most of us are painfully aware that networks can be penetrated, and firewalls can't block every attack. But in 2010, the security industry was still struggling to educate stakeholders that defense needed to go beyond anti-virus and firewalls. John's research in zero trust was the start of today's zero trust movement, and zero trust is so popular now that it is the second-most common (after ransomware) security topic we deal with.

What is zero trust?

"Zero Trust is a strategic initiative that helps prevent successful data breaches by eliminating the concept of trust" – John Kindervag, SVP, Cybersecurity Strategy, ON2IT Cybersecurity

The US National Cybersecurity Center of Excellence says, *"Zero Trust … focuses on accessing resources in a secure manner, regardless of network location, subject, and asset, and enforcing risk-based access controls while continually inspecting, monitoring, and logging interactions."*

Yet another definition from Bill Harrod, SVP, Public Sector Chief Technology Officer, Ivanti, is perhaps easier to think about in the context of database security because it is more actionable: *"The Zero Trust model enforces that only the right people or resources have the right access to the right data and services, from the right device, under the right circumstances."* Bill's definition is how we usually talk about database security – controlling the conditions and context under which someone can access data.

Typical Zero Trust projects

The US Cybersecurity and Infrastructure Agency (CISA) offers a graphical model of the technology components involved in Zero Trust:

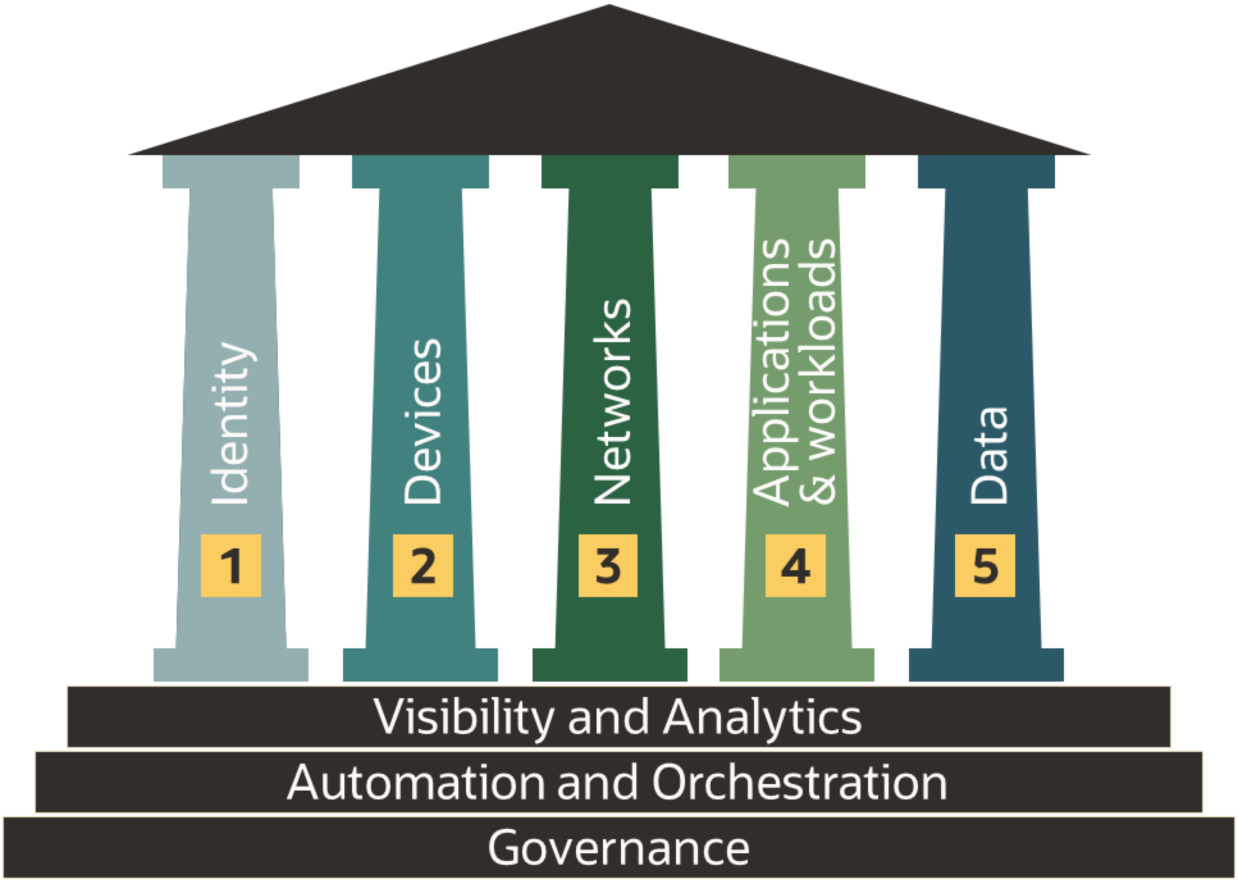


Figure 15-1: CISA Zero Trust Maturity Model Pillars

As you can see from the model, zero trust impacts almost all areas in IT. Zero trust implementations almost always include an increased focus on fundamental security practices. Some common zero trust projects that impact database operations include:

- Increased network segmentation enforced by newer and more capable firewalls
- Mandates to route administrator connections to databases and database servers through bastion hosts or jump servers
- Adoption of privileged account managers, particularly for operating system accounts like ROOT or ORACLE and database accounts like SYS and SYSTEM.

Applying Zero Trust to the database

Applying zero trust to the database can usually be divided into six projects:

1. Assess your configuration and monitor for configuration drift
2. Minimize the attack surface
3. Encrypt data at rest and in motion
4. Strengthen authentication
5. Control access to data
6. Monitor database activity

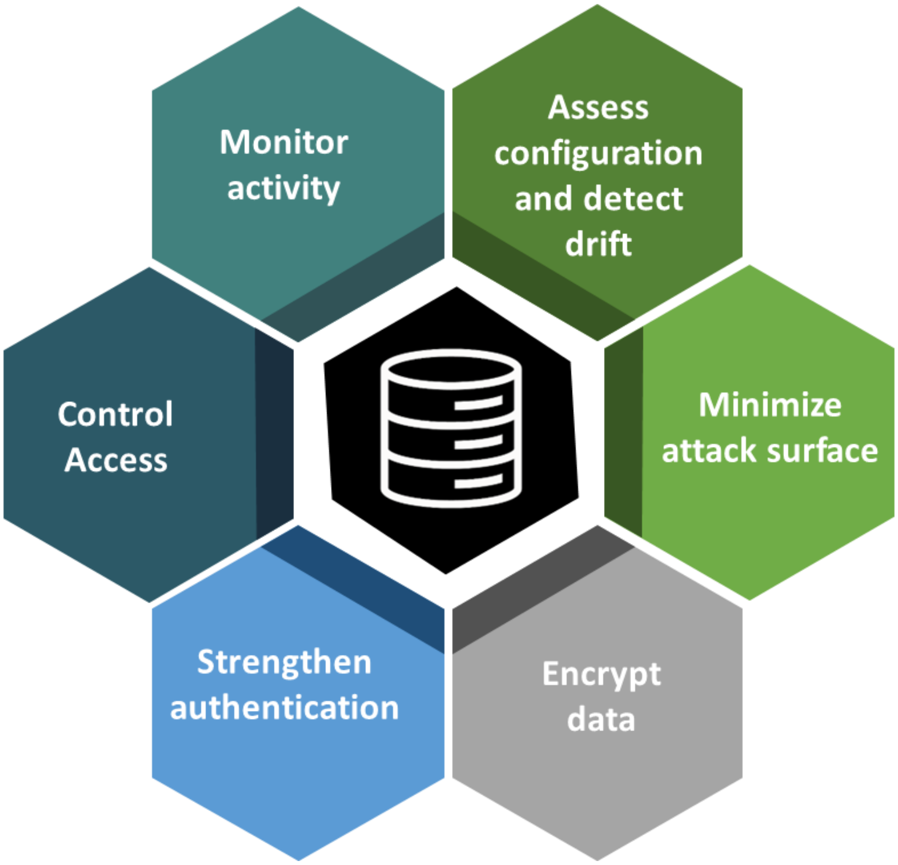


Figure 15-2: Typical zero trust projects for Oracle Database

Assess your configuration and monitor for configuration drift

Are your databases configured to minimize risk? Do they follow your standards for configuration? Can you trust that they will continue to remain configured properly?

Hint: We are discussing zero trust, so if you answered "yes" to that last question and you don't already have some form of configuration drift detection implemented you might want to reconsider.

Databases are complex systems with dozens of parameters that impact security. Configuring them properly and ensuring they REMAIN configured properly is important to a zero-trust environment. Assess your databases to check their configuration, then periodically reassess to detect configuration drift.

Relevant Oracle utilities, features, services, and products:

- Data Safe for security assessment and drift detection
- Audit Vault and Database Firewall for security assessment and drift detection
- Database Security Assessment Tool for one-time assessment
- Enterprise Manager Database Lifecycle Management pack for one-time assessment

Review Chapter Three for more information on assessing database security posture and risk.

Minimize the attack surface

Do not trust your network to keep the database secure! Assume that at some point, attackers will directly attack the database and reduce their chances of success. There are a few key areas to focus on here:

First, every additional database that contains sensitive data increases risk – so wherever possible, minimize the amount of data you're maintaining. Especially in test and development databases you should try to remove sensitive data so that even if an attack against the system succeeds, there is little or no damage.

Relevant Oracle utilities, features, services, and products:

- Data Safe Masking
- Enterprise Manager Data Masking and Subsetting Pack

To learn more about masking data in non-production systems, review Chapter Ten of this book.

Second, assume that most database breaches use compromised accounts during the breach – the bad guys just login to the database and steal whatever data they can access. So, limit that access! Every unnecessary privilege or role increases the blast radius of a compromised account. Remove unnecessary privileges and roles and block administrator access to application data.

Relevant Oracle utilities, features, services, and products:

- Database Vault
- Data Safe *User Assessment*
- Audit Vault and Database Firewall *entitlement monitoring/reporting*
- Database *Privilege Analysis*
- Database Security Assessment Tool
- Data driven controls like Virtual Private Database, Real Application Security, and Label Security

To learn more about controlling access to data, review Chapters Three (the portion on privilege analysis), Six, Seven, and Nine in this book.

Encrypt data at rest and in motion

Do not trust your network – there is a chance you've already been penetrated, and the bad guys are scouting out your infrastructure while capturing data as it travels over the network. Encrypt data as it travels over the network to block this type of attack.

Encryption is a fundamental security control and should be applied both for data in motion AND for data at rest. If you're storing sensitive data, there is a good chance you are already encrypting that data, either for regulatory reasons or because you're worried about ransomware. If you aren't encrypting yet, now is the time!

As we discussed earlier during this chapter's ransomware portion, you should also ensure the encryption keys are protected and not stored on the same server as the encrypted data.

Relevant Oracle utilities, features, services, and products:

- Database *Native Network Encryption* (for encrypting data in motion)
- Database *Transport Layer Security* (for encrypting data in motion)
- Advanced Security *Transparent Data Encryption* (for encrypting data at rest)
- Key Vault (for protecting encryption keys)

Review Chapter Eleven in this book to learn more about encrypting data and managing keys.

Strengthen authentication

Don't trust passwords! There are times when you have no choice but to use them but recognize that you are accepting risk and do what you can to mitigate that risk.

Like encryption, authentication is a fundamental security control – most access controls are based to some degree on the identity of whoever is accessing the data. In general, you should use the strongest authentication mechanism that is mutually supported by the client and database.

It is often helpful to divide database accounts into categories (we discussed categories of users in Chapter Six) and establish policies that define acceptable authentication methods and risk mitigation strategies for each category:

- Superuser accounts (e.g., SYSDBA, SYSKM) – secure with a Privileged Account Manager (PAM) and use these accounts only when absolutely necessary. SYSDBA is seldom required for day-to-day database administration.
- Administrator accounts (DBAs, security administrators, application administrators) – require multi-factor authentication (MFA) if possible. You may want to manage them centrally (e.g., Active Directory). You may also want to secure these accounts with a PAM.
- User accounts (Data analysts, Developers, testers, business intelligence users) – most databases have few (if any) users that connect directly. If yours does, require strong authentication (Kerberos, certificate, MFA).
- Application service accounts – Your choices are frequently limited by application design – the client may not allow for strong authentication. Use strong authentication wherever possible but recognize that many applications cannot handle strong authentication and will force you to use a simple username and password. Consider using multi-factor *authorization* with Database Vault or SQL Firewall to mitigate the risk of compromised accounts. Monitor logins for unusual patterns.

For cases where you are forced to rely on passwords, make the passwords as strong as you reasonably can. Enforce password complexity rules and limit the number of times someone can try a password without locking the account.

You should audit logins from those accounts and watch for indicators of compromise – like new IP addresses connecting to the database, new programs suddenly in use, connections at odd times of the day or during non-working hours, or multiple sessions from different geographical locations.

You may also want to enforce a *trusted path* for those accounts relying on passwords. Define the conditions under which they can connect or access sensitive data.

Relevant Oracle utilities, features, services, and products:

- Database *strong authentication* (Kerberos, certificate, RADIUS, OCI IAM token, and Entra ID OAuth2 tokens)

- Database *Gradual Database Password Rollover* (letting you safely change an application's password without requiring application downtime)
- Database *Centrally Managed Users* (integrating database authentication and authorization with Active Directory)
- Oracle Radius Adapter (part of Oracle Access Manager, connects the database to Oracle Access Manager to enable MFA)
- Database *Unified Audit* (to audit logins by users authenticated via password)
- Database Vault *Trusted Path* (to lockdown accounts with weaker authentication so they can only be used under certain conditions)

To learn more about authentication, review Chapter Five in this book.

Control access to data

Don't trust users to do the right thing! When choosing between hoping users will only do those things they should be doing or using technology to block them from doing anything except what they should be doing, go for the technical control!

Lock down your data so it can't be accessed outside of valid business needs. That means (in most cases) DBAs don't have the ability to view application data, the application's service account can't be used except by the application, and users who DO have a valid business need to see data can only see the data they need to see for their job function. Developers cannot access production, and sensitive data is masked in test/development environments.

Relevant Oracle utilities, features, services, and products:

- Database *Privilege and role grants, including secure application roles*
- Database Vault *Privileged User Controls, Trusted Path Enforcement, SQL Firewall*
- Database *Blockchain and immutable tables*
- Database *Virtual Private Database, Real Application Security*
- Label Security
- Advanced Security *Data Redaction*

To learn more about controlling access to data, review Chapters Six and Seven.

Monitor database activity

Don't trust your preventive controls to be 100% effective. A truly secure system is usually impractical – you'll need to allow SOME access to data. Locking down everything could impede business activity to the point that your data loses its value to the organization. When that happens, fall back to a detective control so you can at least KNOW that someone did something they shouldn't have done.

Use a combination of auditing and network-based monitoring to identify anomalies that might indicate malicious or unauthorized activity. Be ready to support incident investigations.

Ensure your database's audit trail collects information on data definition and control language activities. If someone creates a new user, grants a role or privilege, replaces a stored procedure, or creates a new table by copying an existing table, you want a record of that activity.

If someone is accessing sensitive data from outside an application, you want a record of that activity.

And if your privileged users are accessing data, you most definitely want a record of that activity.

Relevant Oracle utilities, features, services, and products:

- Database *Unified Auditing* (to capture information on security-relevant activity)

- Audit Vault and Database Firewall's *Database Firewall* or *SQL Firewall* (to examine ALL database commands and identify anomalies) – *Note: SQL Firewall is a new feature in Oracle Database 23ai. We discussed SQL Firewall in Chapter Eight.*
- Data Safe *auditing* or Audit Vault and Database Firewall (to analyze audit data, create reports, and generate alerts)

Review Chapters Twelve and Thirteen of this book to learn more about monitoring database activity.

Is that all there is to zero trust?

No, there is always more you can do to secure a system – but if you do everything we discussed above, you've elevated your security posture to world-class and reduced trust in your environment to what is probably an acceptable level.

You may have noticed that after each topic, we've included a "to learn more about…" that points you to other locations in this book. If you've been paying attention, you'll realize that zero trust involved concepts and features from almost everything we discussed in the book. If you find yourself in the planning stages of a zero-trust project, keep the scope in mind. You don't "do" zero trust in a few weeks and be done with it – a zero trust project can take a long time, and many of them never complete; they just continually tweak things to make security a little bit better.

Use a two-pronged approach to address ransomware risk

Protecting against ransomware requires vigilance and diligence, but the technology and infrastructure improve daily. Zero trust is your most effective approach to blocking ransomware. In the event that a ransomware attack against your organization succeeds, every law enforcement agency advises victims NOT to pay the ransom. The combination of encryption and an immutable, zero-data-loss recovery plan makes following law enforcement's advice easier.

Oracle LiveLabs

Oracle LiveLabs give you access to Oracle's tools and technologies to run a wide variety of labs and workshops at your own pace. If you want to try out the technologies discussed in this chapter, please go to:

- Tales from the Dark Side: Hacking the Database

Chapter Sixteen

Securing the Autonomous Database

Why is Autonomous Database security important?

As adoption of Oracle Autonomous Database continues to grow, it's important for administrators to understand the differences between Autonomous Database and other Oracle Database services and editions. Understanding these differences lets you focus on risk mitigation projects and better protect data within Autonomous Database.

Autonomous Database provides standardized, hardened security configurations that reduce the time and money expended in managing configurations across your databases. Security patches and updates are applied automatically, so you don't spend time keeping security up to date. These capabilities protect your databases and data from costly and potentially disastrous security vulnerabilities and breaches. Oracle Autonomous Database automatically encrypts your data at rest and in motion, using industry-standard cryptographic algorithms while providing you with the tools to further restrict or isolate sensitive data from privileged users, developers, data analysts, and application administrators.

Why should you focus on Autonomous Database security?

There are some things Autonomous Database cannot do for you. It has no way to know if the users you grant access to the database behave according to your organization's policies. Nor does the database know what type of sensitive data you may have added to the database. That is why it's so important you read this chapter to understand how security responsibilities are shared between you and your database operations team. It's also why you need to know what tools are at your disposal to help you control risk and better secure your system.

Why Autonomous Database?

Oracle Autonomous Database minimizes and eliminates unnecessary human labor – and associated human error. The basic premise is to free up database administrators' and application developers' time to focus on higher-value activities instead of spending time on mundane, time-consuming tasks.

Broadly speaking, Autonomous Databases handle many routine tasks and provide these features:

- Automates database and infrastructure provisioning, management, monitoring, backup, recovery, and tuning.
- Provides preventative protection against all unplanned and planned downtime – and rapid, automatic recovery from outages without downtime. Autonomous Database availability and performance management are taken to the next level using AI-based autonomy that integrates multiple areas of diagnostics and enables analysis and action to be taken at runtime to minimize or eliminate operational disruption.
- Protects itself from many vulnerabilities and attacks. The Oracle Cloud provides continuous threat detection, while Autonomous Database automatically applies all security updates online and provides "always on" end-to-end encryption. This preventative approach is critical because most security breaches that leverage a known vulnerability occur after the patch that mitigates that vulnerability is already available!

The security benefits of Autonomous Databases

It's unlikely you've gotten this far in the book without developing an appreciation for the risks to data. Mitigating those risks is one of the top concerns in application development and deployment. But as we've seen in previous chapters, there is a LOT to consider when it comes to securing a database. The idea here is that the Autonomous Database does as much of that work for you as possible, freeing you up to focus on those security tasks that it isn't practical to automate.

Autonomous Database leverages years of security development effort and experience running critical workloads for some of the world's most demanding customers. Oracle has been a leader in database security for many decades. We've applied that experience while developing Autonomous Database, locking down areas of risk that most of our customers agree should “always” be done. Very few things are universally true across over 400,000 customers in 175 countries serving every industry vertical! So, while there is no such thing as “always should be done,” – most of us can agree on many areas of security. Everything that we CAN automate in that category is automated.

The security capabilities of Autonomous Databases

The security capabilities we’ve discussed so far are integral to Oracle Autonomous Database. They provide a baseline security posture that is already superior to most on-premises environments and is extensible enough to easily meet the most stringent security requirements. Most of the Oracle Maximum Security Architecture (MSA) technologies are de facto industry standards for protecting and monitoring Oracle Database environments. These security capabilities include:

- **Assessment and Configuration –** You’ve probably picked up from earlier chapters just how important it is to configure your systems securely. The Autonomous Database starts with a secure configuration. We’ve isolated application data to well-defined tablespaces, applied sensible password policies that should meet most customer’s requirements, and configured security-relevant initialization parameters with the CIS Benchmarks and DISA STIG in mind. Because Data Safe is included with the Autonomous Database service, you can easily monitor the database for configuration drift.
- **Encryption for data in motion –** Autonomous Database is automatically configured to use industry-standard Transport Layer Security to encrypt data in transit between the database service and clients or applications. Required client certificates and networking information are automatically packaged for the service consumer when the service is provisioned**.**
- **Encryption for data at rest –** Data in the Autonomous Database is automatically encrypted using Oracle Transparent Data Encryption**.** Because the data is encrypted, backups of the data are also encrypted.
- **Flexible key management –** Key management for Autonomous Database can be Oracle-managed (Autonomous Database creates and manages the encryption keys) or customer-managed. With customer-managed keys, the encryption keys are stored in OCI Vault for Autonomous Databases running in OCI, or in Key Vault for Autonomous Database running in third-party clouds or on Exadata Cloud@Customer.
- **Automated separation of duties –** The Autonomous Database eliminates direct access to the database node and local file system. Further isolation between the service administrators and service consumers is provided through Operations Control, a feature of Oracle Database Vault. This separation of duties – a key Oracle Cloud differentiator – not only reduces the risk of administrator malfeasance, but it also eliminates the ability of the service administrators to view or modify data stored in the Autonomous Database. As with Transparent Data Encryption, Database Vault has been continuously enhanced and improved - with new features like Operations Control added explicitly to support the Autonomous Database.
- **Database auditing configured by default and customizable to meet your needs –** Autonomous Database comes preconfigured with audit policies that monitor privileged user activity and logon failures and, optional pre-configured policies as per the Center for Internet Security (CIS) audit benchmarks, account management, and much more.
- **Assessing the security of your database and its data –** Data Safe is included **with** Autonomous Database to help you assess and secure your databases. Data Safe helps you understand your data's sensitivity, evaluate data risks, mask sensitive data, implement and monitor security controls, assess user security, monitor user activity, and address data security compliance requirements in your databases. You can use Data Safe to identify and protect sensitive and regulated data in your Autonomous Database by registering your database with Data Safe.
- **Masking sensitive data for non-production use -** Often, a lot of time and attention is paid to protecting data in production databases, but then copies of the production databases are made available to developers in mostly insecure "non-production" environments. In reality, if the database contains production data, it

must still be protected as a production database. To provide developers with an environment free of high-risk production data, Data Safe allows you to identify sensitive data, create masking templates, and mask data for non-production Oracle Databases, including Oracle Autonomous Database.

- **Reduced opportunity for human error –** Human error plays a significant role in many data breaches and is one of the most difficult threat vectors to eliminate. The Autonomous Database minimizes the chances of human error by automating a significant portion of database administration. Opportunities for human error are further reduced by restricting the range of commands that an Autonomous Database user is allowed to run.
- **Automated patching, upgrades, and maintenance –** One of the biggest advantages of the Autonomous Database is its ability to automatically apply security patches and upgrade them without downtime. Autonomous Database is automatically patched for you when new security patches become available – and the patch frequency is at least monthly (more than most customers could reasonably do on their own). Much of this capability builds on well-tested, mature Oracle Database technologies like Real Application Clusters (for rolling online RAC patches) and cloud service process automation.

That's only the basics! Autonomous Database also locks down access to sensitive packages like UTL_HTTP, UTL_TCP, and UTL_SMTP, carefully controlling interactions between the database and the surrounding network. OCI administrator access to the databases is restricted by Database Vault *operations control*.

Together, these capabilities within the Autonomous Database provide a security framework that covers the core security requirements for most organizations out of the box, freeing up operations and security teams to elevate enterprise security posture to the next level.

Shared responsibility

There are some things that Autonomous Database's self-securing capabilities can't take care of for you. Oracle implements data privacy controls to ensure that no one at Oracle can see the data you place into an Autonomous Database. That means we don't know if the data in your database is sensitive or not, nor do we know if there are regulatory requirements related to your data.

Oracle doesn't know anything about your database users – their job function, your policies on which data they should or should not see, or any contextual information about them such as working hours, client programs, and normal work locations. That means we can't tell if you are applying additional protection that might be required by regulations, laws, or simply the risk profile for the type of data you store in the database. It also means we can't tell if you've granted your database users appropriate privileges or if those users are accessing data in accordance with your policies.

To help you with the security requirements that we cannot do for you autonomously, Oracle provides a full suite of security tools and services with the Autonomous Database.

Foremost among those tools is Data Safe. Data Safe reports on users, their privileges, and their activity to help you identify over-privileged users or users who are not acting according to your policy. Data Safe also scans your database for sensitive data, helping you understand where your data risk resides and your potential exposure based on the types and amount of data stored in the database. Data Safe enables you to deal with the proliferation of sensitive data by allowing you to mask data for use in non-production environments.

And, of course, Autonomous Database has the same security capabilities that Oracle Database provides in other deployments, including Database Vault, Data Redaction, and Label Security. They are ready to use and just need to be configured to meet your requirements.

Oracle LiveLabs

Oracle LiveLabs give you access to Oracle's solutions and technologies to run a wide variety of labs and workshops at your own pace. If you want to try the solutions discussed in this chapter, please go to:

- Autonomous Database for Security Administrators
- Prevent unauthorized access in Autonomous Database with Database Vault
- Securing a legacy application using Database Vault on Autonomous Database
- Simplify user management by integrating Autonomous Database and Oracle Cloud Identity
- How do I create a database user using Data Tools

Summary

Autonomous Databases leverage both the Oracle Maximum Security Architecture (MSA) and Oracle operations best practices to ensure strong security out-of-the-box. Oracle MSA combines advanced technologies, best practices, and autonomous functions to proactively protect against common attack vectors and frees up scarce security resources to focus on higher-value activity. Autonomous Databases include many securing capabilities such as encryption, access control, automated patching, and auditing. This collection of built-in security capabilities offered by the Oracle Autonomous Database is unmatched by any other cloud (or on-premises) database in the industry.

Chapter Seventeen

Putting it all together

We hope you have enjoyed this journey into Securing the Oracle Database and that it has provided you with some insight into the variety of defense-in-depth strategies available for keeping your data secure. We've covered a lot of ground and many different types of security controls. Putting all of this together creates Oracle's *Maximum Security Architecture* – a blueprint for securing data and minimizing risk.

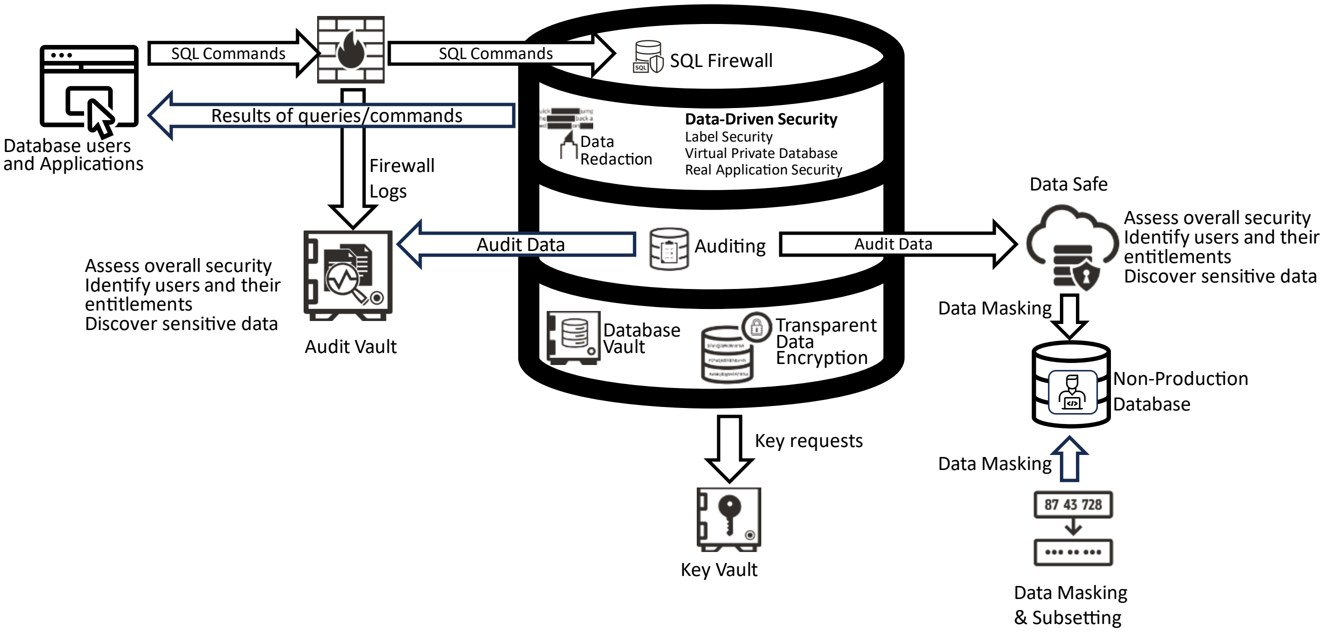


Figure 17-1: Maximum Security Architecture

With so many solutions available, you may wonder how you would approach implementing the various data security controls. Which are essential? Which should I prioritize for sensitive, mission-critical systems? Should I use a phased approach to deploying controls? And how do I leverage these solutions to provide the best return on my security investment?

"If we guard our toothbrushes and diamonds with equal zeal, we will lose fewer toothbrushes and more diamonds."

McGeorge Bundy - US National Security Advisor to President John F. Kennedy

The best advice we can give is to plan to implement a set of security controls appropriate for the sensitivity of the data, the criticality of the data to your business, and your threat environment. For example, one approach would be to inventory your systems and categorize them into different levels according to sensitivity. We can name these categories: Bronze, Silver, Gold, and Platinum. Bronze systems might include internal portals, employee directories, and wikis. Silver systems include business transaction systems, supplier information, and parts catalogs. Gold systems might consist of data subject to regulatory compliance, whether EU GDPR, CCPA, PII, PCI, HIPAA, or SOX. Platinum systems could include highly sensitive and restricted data, including quarterly sales numbers, sales forecasts, M&A activities, and intellectual property such as source code.

The next step would be to specify and implement controls appropriate for each category. For example, an organization might identify the controls shown in Figure 17-2 that would build on each other as they progress through the different levels. Databases at the Bronze level and above must be securely configured and current with security patches. If not, it would be easy for hackers to break into an unpatched system and use it as a command-and-control base for further attacks or a staging area for all the data they discover. In addition, we want to monitor and audit all the activities privileged users perform on this machine to track any significant changes in configuration or user access privileges.

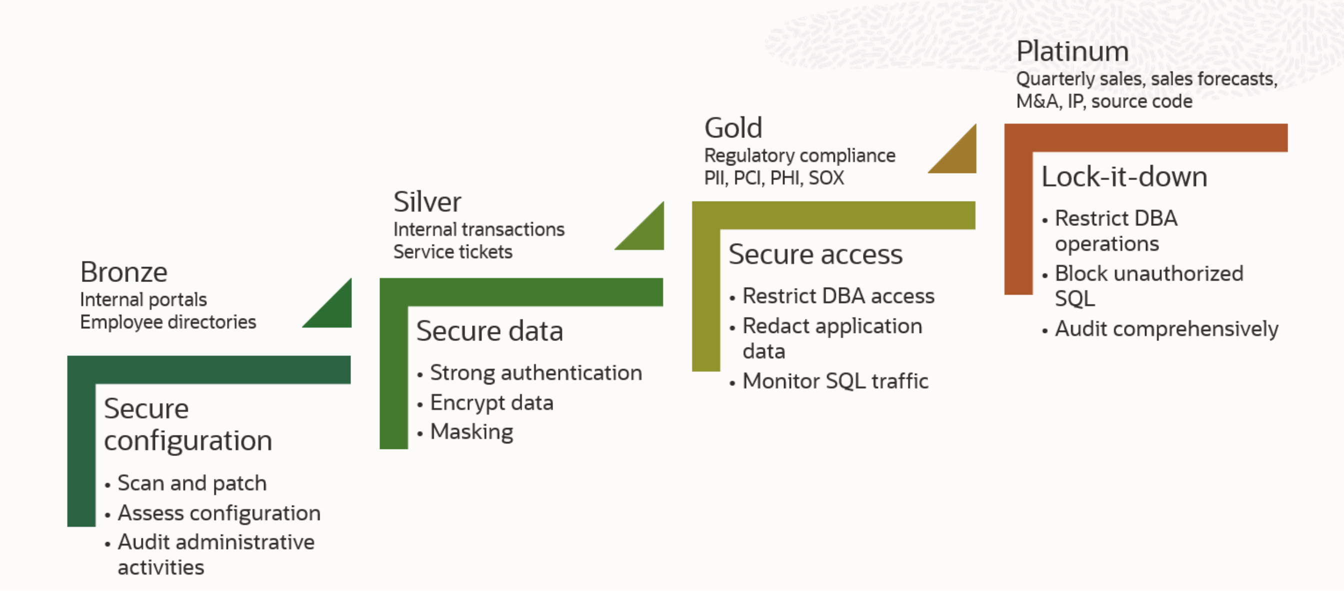


Figure 17-2: Example of implementing security controls based on system sensitivity

You could consider implementing additional measures to protect data from unauthorized users for Silver and above databases. These could include protecting data so it is not visible as it travels over the network, can't be viewed directly from the operating system, and is not present in test and development machines. These systems may require that privileged users authenticate with strong passwords or use PKI or Kerberos-based authentication. The primary security controls used for Silver would include encryption (both in motion and at rest), masking, and strong authentication.

For Gold and above databases, additional controls would protect data from privileged users and from other users who do not have a business need to access the data. We need to restrict privileged users while enabling them to perform their duties and monitor SQL activities over the network to identify malicious attempts to exploit application vulnerabilities quickly. To address basic compliance requirements, we must protect all PII, PCI, and PHI data.

Platinum databases should have all the security controls used for Bronze, Silver, and Gold. In addition, Platinum databases should have additional controls to effectively lock them down because they contain the most sensitive data in the organization. These could include restricting who can log on to the database server, monitoring SQL operations in real-time, guarding against potential SQL injection attacks, and implementing comprehensive audit policies so that forensics teams could determine the impact and method of attack in case of a breach.

Depending upon the priorities and the organization's security strategy, deployment of these controls could start from either edge of the spectrum. You might take a controls-first approach and begin by assessing and securing all your databases' configurations, then move on to configuring transparent data encryption, and so on—implementing one control at a time across your environment. Or, you might start with your most critical and sensitive systems and implement all necessary controls on individual systems one at a time. Both approaches are valid; in practice, we usually see a combination of the two methods.

Either way, you need to have a proper strategy that considers the overall business objectives, people, resources, and time available. In this way, we can protect both "toothbrushes" and "diamonds" with the appropriate level of security, getting maximum return for our security investment.

Index

Access controls... 7, 17, 23, 30, 31, 35, 61, 69, 71, 77, 79, 80, 86, 90, 92, 100, 101, 108, 109, 110, 113, 114, 115, 116, 126, 135, 137, 138, 172, 178, 181, 188

Active Directory 70, 73, 74, 83, 89, 170, 181, 182

Administrative privileges 78, 84, 88, 89, 90

Advanced Security 62, 119, 120, 127, 138, 181, 182

ASM 136, 144

assessment........... 3, 16, 27, 30, 34, 36, 38, 39, 40, 42, 43, 44, 45, 46, 47, 50, 51, 52, 54, 57, 58, 81, 93, 161, 165, 180

assessments 5, 27, 34, 36, 38, 39, 40, 44, 49, 50

Audit.........7, 16, 18, 19, 25, 27, 30, 40, 42, 44, 47, 60, 61, 71, 75, 80, 84, 85, 89, 93, 97, 105, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 181, 182, 183, 186, 191

audit trail 89, 105, 144, 148, 149, 154, 156, 163, 164, 166, 182

Audit Vault 16, 18, 27, 35, 44, 45, 47, 50, 54, 56, 57, 61, 65, 66, 75, 80, 82, 86, 93, 101, 102, 103, 104, 105, 106, 151, 153, 154, 156, 157, 163, 164, 166, 167, 170, 180, 183, 202

auditing 7, 16, 23, 25, 30, 31, 35, 38, 39, 42, 47, 61, 66, 94, 102, 103, 148, 149, 150, 151, 152, 153, 154, 156, 157, 159, 163, 165, 167, 172, 182, 183, 186, 188, 201

Authentication.....6, 19, 23, 24, 25, 30, 68, 69, 70, 71, 72, 73, 74, 75, 80, 84, 85, 86, 89, 100, 135, 151, 170, 171, 172, 179, 181, 182, 191, 201

Automatic Storage Management 136

Autonomous Database 8, 84, 170, 171, 173, 184, 185, 186, 187, 188, 201

AVDF 16, 17, 18, 19, 35, 44, 45, 46, 47, 50, 51, 54, 57, 61, 62, 66, 83, 98, 101, 102, 105, 106, 150, 151, 153, 156, 157, 163, 164, 165, 166, 167

AWS 159, 169, 170, 171, 172

Azure 18, 159, 169, 170, 171, 172, 173

baseline 25, 38, 44, 49, 165, 166, 167, 186

break-glass 96

brute-force 34, 73

CCPA 21, 119, 157, 190

centralized user management 73

Centrally Managed Users 17, 83, 84, 182

CIS 5, 26, 37, 38, 39, 48, 167, 186

CISA 178

CMU 17, 83, 84

command rules 95, 97, 98, 104

Common Access Card 73

Compound masking 123

Conditional masking 123

configuration drift 23, 167, 179, 180, 186

containerized databases 94

Cybersecurity 6, 21, 88, 178

Data discovery 16, 60, 61, 62, 63, 64, 65, 125

data fiduciary 34

Data masking 7, 17, 38, 60, 61, 64, 118, 119, 120, 121, 122, 124, 125, 126, 133, 201

Data Masking and Subsetting 18, 62, 65, 119, 120, 124, 125, 126, 131, 132, 133, 170, 180, 201

data privacy 7, 30, 31, 34, 118, 119, 133, 187

data redaction 7, 118, 119, 120, 126, 127, 128, 129, 133

data retention 157

Data Safe 16, 17, 18, 19, 27, 35, 38, 39, 40, 41, 42, 43, 44, 47, 50, 51, 52, 54, 56, 57, 59, 60, 61, 62, 65, 66, 75, 80, 81, 82, 86, 93, 98, 105, 106, 119, 120, 122, 124, 125, 126, 133, 150, 151, 153, 154, 156, 157, 158, 159, 160, 161, 162, 167, 170, 171, 173, 180, 183, 186, 187, 201, 202

data scraping 24

data subsetting 131, 132, 133

Database activity monitoring 148

Database Configuration Assistant 139

Database Firewall 6, 16, 17, 18, 19, 27, 35, 44, 45, 47, 50, 54, 56, 57, 61, 62, 65, 66, 75, 80, 82, 86, 93, 100, 101, 102, 103, 104, 105, 106, 151, 153, 154, 156, 157, 163, 164, 165, 166, 167, 170, 180, 183, 202

database privilege 77

Database Security Posture Management 166

Database Vault 17, 61, 62, 65, 71, 74, 84, 85, 90, 91, 92, 93, 94, 95, 96, 97, 98, 104, 106, 165, 170, 180, 181, 182, 186, 187, 188, 202

DBCA 139

DBLM 35, 47, 48, 49, 50, 54

DBSAT 16, 26, 27, 35, 36, 37, 38, 39, 49, 52, 54, 56, 57, 58, 61, 66, 75, 86, 202

defense-in-depth 21, 22, 24, 32, 54, 190

Deterministic masking 123

Dirty dozen 16, 19, 22, 23, 24, 26

DORA 29, 31

DSPM 35, 166

encrypted 7, 18, 23, 70, 104, 105, 135, 136, 138, 139, 140, 141, 142, 143, 144, 145, 177, 181, 186 encryption....... 7, 18, 23, 30, 35, 39, 70, 75, 78, 79, 80, 85, 88, 118, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 148, 177, 181, 183, 185, 186, 188, 191, 192

Enterprise User Security........ 83

entitlements 47, 75, 86, 164, 165, 167

Entra ID........ 69, 70, 73, 84, 169, 171, 172, 173, 181

EUS 83, 84

FAILED_LOGIN_ATTEMPTS........ 72, 73

GCP........ 170, 171, 172

GDPR........ 5, 7, 21, 29, 34, 37, 39, 66, 113, 118, 119, 121, 133, 146, 157, 164, 190, 202

Google 143, 171

HIPAA........ 66, 146, 157, 164, 190

hybrid cloud 169

IBM 27, 102, 157, 163

Immutable backups........ 176

input validation........ 101

Kerberos........ 69, 73, 74, 75, 84, 181, 191

Key Vault........ 18, 70, 74, 139, 141, 142, 143, 144, 145, 146, 170, 177, 181, 186, 202

Label Security 17, 62, 71, 109, 110, 111, 112, 113, 116, 128, 170, 180, 182, 187

latency 105, 171

LiveLabs 54, 65, 75, 86, 98, 106, 116, 133, 145, 154, 167, 173, 183, 188

Mandatory realms 93

Microsoft........ 69, 70, 73, 83, 84, 102, 137, 143, 157, 163, 164, 166, 169, 170, 171, 172, 173, 202

multicloud........ 8, 50, 73, 169, 170, 173

multifactor authentication........ 24, 25, 171

MySQL 102, 144, 163

native network encryption........ 135

NIS2........ 29

NNE 135, 136, 146

OAuth2 .... 172, 181

Object privileges........ 77, 81

offline encryption........ 140

Okta 73, 170

OLS 109, 110, 111, 114, 128

Online tablespace encryption 140

Operations control........ 94

OWASP 22, 100

Password. 17, 23, 34, 42, 43, 46, 68, 69, 70, 71, 72, 73, 74, 75, 80, 83, 84, 93, 142, 144, 150, 171, 181, 182, 186, 191

PASSWORD_ROLLOVER_TIME 72, 74

PASSWORD_VERIFY_FUCTION 72

PCI-DSS 7, 66, 118, 119, 121, 133, 140, 146, 157, 164

Performance 105

persistent cache 144

PKI 69, 73, 74, 75, 84, 135, 144, 191

Privilege Analysis 51, 52, 54, 75, 81, 86, 89, 180

Privileged users 25, 61, 88, 90, 93, 148, 150, 165, 182, 185, 187, 191

process control 26, 27

Proxy authentication 71

RADIUS 69, 73, 170, 181

ransomware 5, 8, 21, 24, 31, 138, 175, 176, 177, 178, 181, 183

RAS 17, 109, 113, 114, 115, 116, 127, 128

Real Application Security 17, 71, 109, 110, 113, 114, 115, 116, 119, 120, 127, 128, 170, 180, 182

regulatory compliance 5, 8, 119, 148, 154, 156, 157, 190, 201

Reversible masking 119, 123

roles 4, 19, 34, 35, 37, 42, 51, 52, 69, 71, 73, 74, 75, 77, 79, 81, 83, 84, 85, 86, 89, 93, 94, 97, 110, 114, 115, 128, 167, 180, 182, 202

SAP 98, 102, 157, 163, 201

Schema privileges 78

SCIM 162

Secrets manager 70, 74, 80, 146

Secure External Password Store 70, 144

Sensitive data. 3, 5, 6, 7, 16, 18, 23, 25, 34, 35, 36, 42, 44, 47, 56, 57, 58, 59, 60, 61, 62, 63, 66, 68, 78, 79, 85, 88, 90, 93, 94, 97, 98, 100, 108, 114, 115, 118, 119, 120, 121, 122, 123, 124, 126, 127, 128, 130, 133, 139, 145, 146, 148, 150, 151, 152, 153, 154, 156, 157, 160, 161, 164, 166, 167, 180, 181, 182, 185, 186, 187, 191

Separation of duties 6, 35, 75, 88, 89, 90, 92, 93, 94, 95, 97, 142, 186

SEPS 70, 144

Service accounts 23, 35, 51, 68, 71, 73, 74, 80, 85, 89, 90, 150, 181

Shared accounts 23, 25, 71, 74

simulation mode 97

SQL Firewall 6, 17, 74, 85, 86, 100, 101, 102, 104, 105, 106, 157, 164, 181, 183

SQL Injection 6, 17, 23, 34, 44, 47, 85, 90, 99, 100, 101, 102, 103, 104, 106, 191

SQL Server 102, 163, 164, 166

SSH 144, 145

STIG 16, 37, 38, 39, 48, 49, 72, 167, 186

Sybase 102, 163

SYSDBA 74, 78, 84, 88, 89, 141, 151, 181

SYSKM 78, 79, 84, 88, 141, 151, 181

System privileges 17, 51, 78, 81, 154

tablespace encryption 139, 140, 141

TDE 18, 138, 139, 140, 142, 144, 145, 146, 170

technical control 26, 27, 30, 182

TLS 135, 136, 145, 146

Transparent Data Encryption 138, 139, 145, 170, 173, 177, 181, 186, 202

transport layer security 135

Trusted path 35, 89, 91, 104, 152, 181

unencrypted 23, 24, 135, 139

User profiles 43, 72

Virtual Private Database 109, 110, 113, 115, 116, 119, 120, 127, 170, 180, 182

VPD 17, 109, 110, 113, 114, 115, 116, 127

WAF 100, 101

Wallet 70, 141, 142

web application firewalls 100, 101

ZDLRA 176

Zero trust 175, 179, 183

zero-day 101, 106

Appendix: Tools – Features, Options, Products, and Packs

Any discussion of this nature can easily devolve into a listing of products and features – I've tried to minimize that in the main portion of this paper. Below is a list of the different features, options, products, and packs mentioned in the paper, along with links to documentation. The features are listed in the order in which they were mentioned above. One thing to consider about those documentation links – they are current as of the time I write this, but white papers tend to linger for a long time, while database versions and documentation are fluid and frequently updated. Check docs.oracle.com to find the latest version of the documentation – the links provided here will at least let you know which manual (and usually chapter or appendix) to look for.

Database Security Assessment Tool (DBSAT)

DBSAT is a standalone utility included with your database support. DBSAT helps with security assessment, user assessment, and sensitive data discovery. There is no additional fee for the use of DBSAT. DBSAT works with all supported versions of the Oracle Database, on all supported operating systems, and can be run for databases on-premises or in the cloud – including non-Oracle clouds. The utility may be downloaded from My Oracle Support – check MOS note 2138254.1 for more information on downloading the tool. DBSAT documentation is available here.

Oracle Data Safe

Data Safe is an Oracle Cloud service, included with Oracle Database as a Service offerings and available for use with on-premises databases and Oracle Databases running on Oracle Cloud Compute. Data Safe includes several database security capabilities, including security assessment, user assessment, sensitive data discovery, sensitive data masking, unified audit policy control, audit data retrieval, reporting, and alert generation. Data Safe is Oracle's newest database security service and is rapidly evolving (two-week development sprints) new features and capabilities. Click "What's New" in the documentation for updates on recent changes. Documentation for Data Safe is included here.

Enterprise Manager Database Lifecycle Management

Database Lifecycle Management is a management pack for Oracle Enterprise Manager Cloud Control. Database Lifecycle Management provides numerous functions for managing the lifecycle of your database, including configuration management, and can play a valuable part in security assessments. Information about configuration management using Enterprise Manager Database Lifecycle Management is available here.

Privilege Analysis

Privilege analysis is a database feature included with all databases and database services except for standard edition. It helps with user assessment, particularly with determining which privileges a user account has, but is not using. Privilege Analysis was introduced in Oracle Database 12c Release 1. Privilege Analysis documentation is available here.

Native Network Encryption

Native Network Encryption (NNE) is a database feature included with all databases and database services with the exception of Autonomous Database (Autonomous Database uses TLS instead of NNE). NNE encrypts data as it travels between the database and database client or application. NNE documentation is available here.

Transport Layer Security

Transport Layer Security (TLS) is a database feature included with all databases and database services. It is configured by default for Autonomous Databases. TLS encrypts data as it travels between database and database client or application. TLS documentation is available here.

Centrally Managed Users

Centrally Managed Users (CMU) is a database feature included with Oracle Database Enterprise Edition. CMU was introduced with Oracle Database 18c. CMU allows Oracle Databases to connect directly to Microsoft Active Directory. With CMU, users are created in Active Directory and mapped to database schemas. Optionally, database roles can be associated with Active Directory groups, and database role membership controlled by Active Directory group membership. CMU documentation is available here.

Enterprise User Security

Enterprise User Security (EUS) is a database feature included with Oracle Database Enterprise Edition. EUS was introduced with Oracle Database 8.1 (Oracle 8i). EUS allows Oracle Databases to connect to an Oracle directory service. With EUS, users are created in Internet Directory and database schemas are mapped to users or to collections of users within an LDAP organizational unit. Database roles can be associated with Internet Directory groups, and database role membership controlled by LDAP group membership. EUS documentation is available here.

Traditional Auditing

Traditional auditing was first introduced in Oracle Database 7 and was the primary auditing mechanism for Oracle Database until the release of Oracle Database 12c. Traditional auditing is being replaced by unified audit. Traditional auditing is desupported starting with Oracle Database 23ai. Traditional audit documentation is available here.

Fine-Grained Auditing

Fine-grained auditing was introduced in Oracle Database 9.0 (9i Release 1). As the name suggests, Fine-grained auditing allows audit policies to be focused more narrowly than traditional auditing, allowing audit policies based on columns. Fine-grained auditing also introduced the concept of conditional auditing, where audit records would only be generated if a certain condition evaluated to true. Documentation for fine-grained auditing is available here.

Unified Auditing

Unified auditing was introduced in Oracle Database 12.1 (12c Release 1). Unified auditing consolidates audit records into a single location, combining audit data from Database Vault, Label Security, Data Pump, SQL*Loader, Recovery Manager (RMAN), fine-grained auditing, and audit records generated from unified audit policies. Unlike traditional auditing, unified audit policies may be conditional, may choose to audit only top-level statements, and are extensible to include context information not in the default audit trail. Documentation for unified auditing is available here.

Enterprise Manager Data Discovery

Data Discovery is available in Enterprise Manager. Data discovery scans databases to locate sensitive data and helps drive Data Masking and Subsetting (DMS), Audit Vault and Database Firewall (AVDF) sensitive data audit reporting, and Transparent Sensitive Data Protection (TSDP) policies. Enterprise Manager Data Discovery stores the list of applications, tables, and relationships between table columns that are either declared in the data dictionary, imported from application metadata, or user specified. Data Discovery maintains sensitive data types and their associated columns, and is used by test data operations, such as data subsetting and data masking, to securely produce test data. Like Data Safe, Data Discovery scans data contained within tables to find sensitive data.

Data Discovery is included at no additional cost with Oracle Advanced Security, Oracle Database Vault, Oracle Label Security, Oracle Data Masking and Subsetting, and Oracle Audit Vault and Database Firewall. Use Data Discovery to understand a database schema or schemas, including how columns relate to one another and which columns contain sensitive data. Documentation for Data Discovery is available here.

Oracle Advanced Security

Oracle Advanced Security (ASO) is a database option that includes Transparent Data Encryption, RMAN backup encryption, Data Pump export encryption, encrypted Database File System (DBFS), encrypted SecureFile LOBs, and Data Redaction. Advanced Security is one of the oldest database options, tracing its roots to the Advanced Networking Option introduced in Oracle 7. Documentation for Advanced Security's Transparent Data Encryption is available here. Documentation for Advanced Security's Data Redaction is available here.

Oracle Key Vault

Oracle Key Vault is a key management system supporting the Oracle infrastructure, optimized for use with Transparent Data Encryption. Key Vault provides continuous access to encryption keys with a multi-master fault-tolerant cluster architecture. In addition to protecting encryption keys, Key Vault also provides complete SSH key governance (an important capability for protecting all of your Linux servers, not just the database servers), secrets management, DBMS_CRYPTO key management, and much more. Documentation for Key Vault is available here.

SQL Firewall

SQL Firewall is a new component of Oracle Database 23ai. It is designed to detect and block SQL injection and other anomalies. SQL Firewall is included with both Oracle Database Vault and Oracle Audit Vault and Database Firewall. Documentation for SQL Firewall is available here.

Oracle Database Vault

Oracle Database Vault is a database option that provides advanced access control capabilities. Database Vault is commonly used to enforce separation of duties, block administrator access to sensitive data, and enforce trusted path access to data. Database Vault includes SQL Firewall. Documentation for Database Vault is available here.

Oracle Audit Vault and Database Firewall

Oracle Audit Vault and Database Firewall (AVDF) is a database activity monitor with heterogeneous capabilities – covering Oracle Database, Oracle MySQL, Microsoft SQL Server, PostgreSQL, MongoDB, IBM DB2, and SAP Sybase. For Oracle Databases, AVDF also provides security assessment, tracks user entitlements, performs drift detection, discovers sensitive data, and includes SQL Firewall. Documentation for Audit Vault and Database Firewall is available here.

Oracle Data Masking and Subsetting

Oracle Data Masking and Subsetting (DMS) is a management pack for Oracle Enterprise Manager. DMS removes risk from databases by replacing sensitive data with artificial values. DMS can also be used to create subsets of a database – smaller copies with only a portion of the original data. Documentation for Data Masking and Subsetting is available here.

About the authors

Alan Williams is a product manager for Oracle Database security, responsible for authentication and authorization technologies. Prior to joining the Oracle Database security team, he was involved in government and military projects involving high-security architecture, design, and processes, along with ITIL implementation. Alan is a 40-year veteran of the IT industry and has certifications in ITIL v3 Foundation and DOD Architecture Foundation and is a United States Air Force veteran. He earned his bachelor's degree from the Massachusetts Institute of Technology and Master of Business Administration from the Rensselaer Polytechnic Institute.

Angeline Janet Dhanarani is a product manager for Oracle Database security, focusing on SQL firewall, auditing and activity monitoring. With over 20 years of experience in Oracle spanning multiple products, she now helps Oracle customers adopt comprehensive database security strategies and closely works with the engineering team to define the product roadmap for SQL firewall, auditing and activity monitoring.

Bettina Schäumer is a product manager for Oracle Data Safe. She has over 20 years of experience in product and solution management, go-to-market strategies, sales operations, sales enablement, program management and consulting. Prior to joining Oracle, Bettina worked at SAP with global responsibilities for end-to-end scenarios and key capabilities within the SAP HANA platform. Throughout her career, she covered a variety of solutions in enterprise software, business networks, business analytics, internet of things, technology and database systems. Bettina has a degree in Computer Science.

Ethan Shmargad is a product manager for Oracle Database security, focusing on field enablement. Ethan earned his bachelor's degree in Information Systems and Data Analytics from California Polytechnic State University.

Hakim Loumi is a product manager for Oracle Database security, focusing on artificial intelligence. Hakim also handles outbound product manager issues for the global francophone community. With over 25 years of experience in information technology, he started his career as a developer, then spent almost 15 years working with some of the largest European companies as a production DBA, Data Architect, and Data Analyst in a wide range of fields. Before joining Oracle, he led the data department of Europe's leading e-commerce company.

Kajal Singh is the product manager for Oracle Database security, focusing on Data Masking and Subsetting. Kajal concentrates on product quality and usability. Kajal earned her master's degree in engineering management from Dartmouth College. Her certifications include Certified Associate in Project Management (CAPM), Oracle Cloud Platform Enterprise Analytics, and Oracle Database Autonomous Database Specialist. She completed an executive program in Product Management with the India Institute of Management in Lucknow.

Manoj Shringarpure is a product manager for Oracle Database security, focusing on customer adoption and regulatory compliance in the Asia Pacific region. Manoj earned his Master of Business Administration degree from the Narsee Monjee Institute of Management Studies (NMIMS) in Mumbai.

Michael Mesaros is the Director of Product Management for Oracle Database security and has over 35 years of experience in various security areas. At Oracle, he has managed products for collaboration, networking, directory services, and identity management. In his career, Michael has also managed a variety of security products, including those for network behavior analytics, physical security information management, data masking, and firewall systems. Michael attended the

University of Michigan in Ann Arbor, receiving a BSE in Electrical Engineering, a BS in Cellular and Molecular Biology, and an MBA from the Ross School of Business. He also has a Master’s Degree in Electrical Engineering from San Diego State University and is a Certified Information Systems Security Professional (CISSP).

Nazia Zaidi is the product manager for Oracle Audit Vault and Database Firewall. She has two decades of experience in database, database security, and cloud security technologies. Nazia helps Oracle's customers strategize their information/cloud security posture to meet varying business and regulatory requirements. She advises across a broad range of security solutions and markets, including financial institutions, government, defense, technology, telecom, healthcare, and retail.

Pedro Lopes is a product manager for Oracle Database security. He covers Europe, Middle East, and Africa (EMEA), and Latin America regions for all Database Security features and products and manages the Security Assessment technologies (DBSAT, Data Safe). He played numerous roles from Consulting to Presales during the last 20 years at Oracle. Pedro is helping customers adopt Oracle Data Safe and to understand how Oracle Database Security solutions may help address EU GDPR and other regulatory requirements. Pedro’s certifications include CISSP, ITIL v3 Foundation, and International Project Management Association Level D (IPMA).

Peter Wahl is the product manager for Transparent Data Encryption and Oracle Key Vault and has over 20 years of experience in various security areas. Peter has also been a member of Oracle sales consulting organization, working with some of the largest Oracle Database customers in the US and Canada. Peter is a certified Oracle Cloud Infrastructure Architect Associate and earned his master’s degree in Electrical Engineering from the University of Applied Sciences in Ravensburg, Germany.

Russ Lowenthal is the Vice President of Database Security, helping thousands of customers understand how to mitigate risks and secure their databases using Oracle’s Maximum Security Architecture. Leveraging more than thirty years of experience in IT including database, UNIX systems, and network administration, he advises Oracle’s customers on database security strategy and implementations. Russ earned his master’s degree in Computer Science from Texas A&M university. His certifications include CISSP, Certified Information Systems Auditor (CISA), Certified Information Systems Manager (CISM), Oracle Certified Master (OCM), Microsoft Certified Systems Engineer (MCSE) and Certified Technical Trainer (CTT).

Richard C. Evans is the product manager for Oracle Database Vault and host of the monthly AskTom Oracle DB Security Office Hours webinars. Richard was an Oracle Certified Professional (OCP) DBA for 15 years and is a US Air Force veteran. He earned a master’s degree in Computer Information Systems from Boston University. Richard holds various IT, cloud, and security related certificates, including the CISSP and PMP.

Acknowledgments

The following individuals helped with the review and provided invaluable feedback for this sixth edition:

Gopal Mulagund

Sachin Deshmanya

Thanigai Nallathambi

We would especially like to thank Vipin Samar, Senior Vice President, Oracle Database Security, for helping and guiding us during the preparation and review of this book

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle