



# Oracle Database Security

## a technical primer

Fifth edition

September, 2023, Version 5.0  
Copyright © 2023, Oracle and/or its affiliates

## Foreword

Having been in the security space for over 25 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. While the global security spend is expected to exceed \$195 billion in 2025, hacks are becoming bigger and bolder, impacting everything from customer and citizen databases to vaccine data and Wi-Fi routers.

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 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 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  
September 2023

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 7 \* 24. 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 would 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.

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 180 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.

Angeline Dhanarani

Rich Evans

Hakim Loumi

Russ Lowenthal

Pedro Lopes

Michael Mesaros

Bettina Schaeumer

Peter Wahl

Alan Williams

Nazia Zaidi

## **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 book takes you through the various aspects of Oracle's defense-in-depth security for databases. It provides a high-level overview of how the different controls work and the types of protection they provide.

## **Chapter Two: Assess database security posture and risk**

Today's systems are complex, with many security configuration settings. As recent data breaches have demonstrated, it is critical to have properly configured and secure systems. Human errors could leave your database open to everyone, or an attacker could maliciously exploit configuration mistakes to gain unauthorized access to sensitive data.

Failing to implement basic security controls may risk exposing customer 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 the most critical systems, such as 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. It is critical to regularly assess database security posture, considering recommendations from different regulations, security frameworks, and vendor 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 Three: 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 Four: Authenticating database users**

A fundamental step in securing a database system is validating the identity of the users accessing the database (authentication) and controlling the operations they can perform (authorization).

This chapter discusses how a proper 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 Five: 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 what 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 the user account authorizations, whether locally within the database or with centralized directory or identity services.

### **Chapter Six: 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.

This chapter discusses how you can minimize the losses from compromised accounts 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 Seven: Minimize risk from SQL Injection**

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

This chapter discusses the two main approaches to mitigate the risk of SQL Injection: 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 Eight: Data-driven application 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 Nine: 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 dynamic data

masking prevents the proliferation of sensitive data to users 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 Ten: 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 Eleven: Database auditing**

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 monitoring network events.

This chapter helps you evaluate the pros and cons of these approaches and discusses Oracle's solutions for these approaches. Database auditing provides the most accurate record of database activity. Auditing has a broader contextual view of user activity than database activity monitoring techniques like network-based monitoring. One challenge with network-based monitoring is that direct local logins, recursive SQL, dynamic SQL, or stored procedures may circumvent the monitoring.

## **Chapter Twelve: Database auditing – reporting and alerts**

Database auditing provides the most accurate record of database activity, but you need to centralize the audit data, analyze it, and create 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 Thirteen: Ransomware and Zero Trust**

Today, most security conversations are driven by one of these topics: Regulatory compliance, Data breaches, Ransomware, and Zero-trust . In this book , so far, we have been discussing mainly data breaches and regulatory compliance. In this chapter, we'll discuss the last two – 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 improving zero trust.

## **Chapter Fourteen: Database security in the multicloud world**

Many 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: Securing the Autonomous Database**

The Oracle Autonomous Database provides standardized, hardened security configurations that reduce the time and money 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 solutions while also providing you with the tools you need to further restrict or isolate sensitive data from privileged users, developers, data analysts, and application administrators.

But there are some things the Autonomous Database cannot do for you. It has no way to know if the users you grant access to the database are behaving in accordance with 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 and why you need to know what tools are at your disposal to help you control risk and better secure your system.

# Table of Contents

---

Foreword .....	2
From the authors' desk .....	3
<b>Protecting data .....</b>	<b>13</b>
Introduction .....	14
Data is the new currency .....	14
The need to protect data has never been greater .....	14
Threat actors and the “Dirty Dozen” .....	15
Addressing the Dirty Dozen through data security controls .....	17
<b>Assess database security posture and risk .....</b>	<b>21</b>
Introduction .....	22
Evaluate and assess database configuration .....	22
Assessing configuration security of the Oracle Database with DBSAT .....	23
Assessing your database fleet using Oracle Data Safe .....	26
Assessment with Oracle Audit Vault and Database Firewall (AVDF) .....	30
Configuration Assessment with Enterprise Manager .....	31
Choosing the right database security assessment tool .....	32
Reducing the blast radius with Privilege Analysis .....	33
Security patch management .....	34
Oracle LiveLabs .....	36
Summary .....	37
<b>Discovering sensitive data .....</b>	<b>38</b>
Introduction .....	39
Why is sensitive data important? .....	39
Discovering sensitive data .....	40
Discovering sensitive data using DBSAT .....	40
Discovering sensitive data using Data Safe .....	42
Discovering sensitive data using Audit Vault and Database Firewall .....	44
Discovering sensitive data using Enterprise Manager .....	45
Oracle LiveLabs .....	46
Summary .....	47



**Authenticating database users..... 48**

- Introduction ..... 49
- Users: Your weakest link..... 49
- Database authentication methods..... 49
- Making users resistant to attacks..... 50
- Protecting your users from getting hacked ..... 53
- Oracle LiveLabs..... 54
- Summary ..... 55

**Controlling database access ..... 56**

- Introduction ..... 57
- Types of database users ..... 57
- Privileges ..... 58
- Roles ..... 58
- Who can do what in your database?..... 59
- Managing users centrally ..... 60
- Protecting database accounts ..... 62
- Oracle LiveLabs..... 63
- Summary ..... 63

**Enforcing separation of duties ..... 64**

- Introduction ..... 65
- Controlling powerful administrators..... 65
- Control privileged users with Oracle Database Vault ..... 66
- Enforce separation of duties with Oracle Database Vault ..... 67
- Control SQL database commands with Oracle Database Vault ..... 68
- Operationalizing Oracle Database Vault..... 69
- Oracle LiveLabs..... 70
- Summary ..... 70

**Minimize risk from SQL Injection..... 71**

- Introduction ..... 72
- SQL injection overview..... 72
- Approaches to address SQL injection attacks..... 72
- Network-based Database Firewall..... 73
- Database-resident SQL Firewall..... 75

- Deciding which to use: Database Firewall or SQL Firewall ..... 77
- Oracle LiveLabs for hands-on experience ..... 78
- Summary ..... 78
- Data-driven application authorization ..... 79**
- Introduction .....80
- Data-Driven, fine-grained authorization .....80
- Challenges with implementing data authorization in applications .....81
- Controlling data access using Virtual Private Database.....81
- Controlling data access using data labels..... 82
- Controlling data access using Real Application Security .....85
- Which data-driven access control should I use?..... 87
- Oracle LiveLabs..... 87
- Summary ..... 87
- Masking sensitive data..... 88**
- Introduction .....89
- Why mask Data? .....89
- Use cases for data masking .....90
- Static data masking.....90
- Dynamic data masking.....91
- Data subsetting.....91
- Masking formats.....91
- Masking techniques.....93
- Masking sensitive data using Oracle Data Safe .....95
- Masking sensitive data using Oracle Data Masking and Subsetting.....96
- Differences between Data Masking and Subsetting & Data Safe.....97
- Masking data dynamically.....97
- Oracle Data Redaction.....98
- Data Subsetting.....102
- Oracle LiveLabs.....103
- Summary .....103
- Data encryption and key management..... 104**
- Introduction .....105
- Why encrypt?.....105

Encrypting data in motion – database connections ..... 105

Encrypting data at rest – database files and backups ..... 106

Transparent Data Encryption ..... 107

Key Management ..... 110

Centralized key management with Oracle Key Vault ..... 110

Oracle LiveLabs..... 113

Summary ..... 113

**Database auditing ..... 114**

Introduction ..... 115

Why audit?..... 115

Oracle Database with unified auditing ..... 116

Effective database auditing ..... 117

Audit policy provisioning from Data Safe or AVDF..... 120

Transition from traditional to unified auditing ..... 121

Oracle LiveLabs..... 121

Summary ..... 121

**Database auditing – reporting and alerts ..... 122**

Introduction ..... 123

Auditing with Oracle Data Safe ..... 123

Auditing with Audit Vault and Database Firewall (AVDF)..... 127

Oracle LiveLabs..... 131

Summary ..... 131

**Ransomware and Zero Trust ..... 132**

Introduction ..... 133

Zero Trust..... 135

Applying Zero Trust to the database..... 137

Oracle LiveLabs..... 141

**Database security in the multicloud..... 142**

Introduction ..... 143

Why multicloud? ..... 143

Identity cloud service..... 144

Managing security across clouds ..... 144

Managing resources in multiple clouds..... 145

# ORACLE

Multicloud service access .....	145
Multicloud identity management for Oracle Database .....	145
Oracle LiveLabs for multicloud integration.....	147
Summary .....	147
<b>Securing the Autonomous Database.....</b>	<b>148</b>
Introduction .....	149
Why Autonomous Database? .....	149
The security benefits of Autonomous Databases .....	149
The security capabilities of Autonomous Databases .....	150
Shared responsibility .....	151
Oracle LiveLabs.....	152
Summary .....	152
<b>Putting it all Together .....</b>	<b>153</b>
About the authors .....	157
Acknowledgments.....	159

# Chapter One

## **Protecting data**

---

## Introduction

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 book takes you through the various aspects of Oracle's defense-in-depth security for databases and provides a high-level overview of how the different controls work and the types of protection they provide.

## Data is the new currency

Organizations worldwide are experiencing the impact of data breaches at an unprecedented rate. It seems like every day brings a news story about a service provider losing subscribers' personal information, an employer losing employee HR records, or a government contractor losing sensitive intellectual property. Data is the new currency, and bad actors can often leverage stolen data for financial or political advantage for years after a breach.



Figure 1-1: Personal Data

And where do they keep their sensitive data? At the end of the day, this data is stored and managed mostly in databases. Perimeter security solutions such as network firewalls were once considered sufficient for protecting internal systems and repositories such as databases from data theft. However, the threat environment for organizations has changed considerably in recent years. Tools vary widely depending upon the attackers, from exploiting unpatched systems to very advanced methods where hackers penetrate a network, search for vulnerabilities, and then covertly exfiltrate data from servers. These attacks can go undetected for weeks, months, or even years.

## The need to protect data has never been greater

You operate in an increasingly stringent and fast-evolving regulatory landscape. The United States has more than 20 national privacy and data security laws, with other laws enacted at the state level. The European Union (EU) harmonized data privacy laws across multiple member states with the General Data Protection Regulation (EU GDPR), and in the years since GDPR became law, over 130 countries have enacted similar privacy regulations.

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).

In addition to the stringent regulatory environment and data privacy issues, threats to data have grown to top-of-mind for most organizations. Ransomware drives much of this, accounting for about 20% of cybercrime, at an annual cost of tens of billions of dollars.

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.

## Threat actors and the “Dirty Dozen”

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 must examine the 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 include personal, financial, trade secrets, and regulated data.

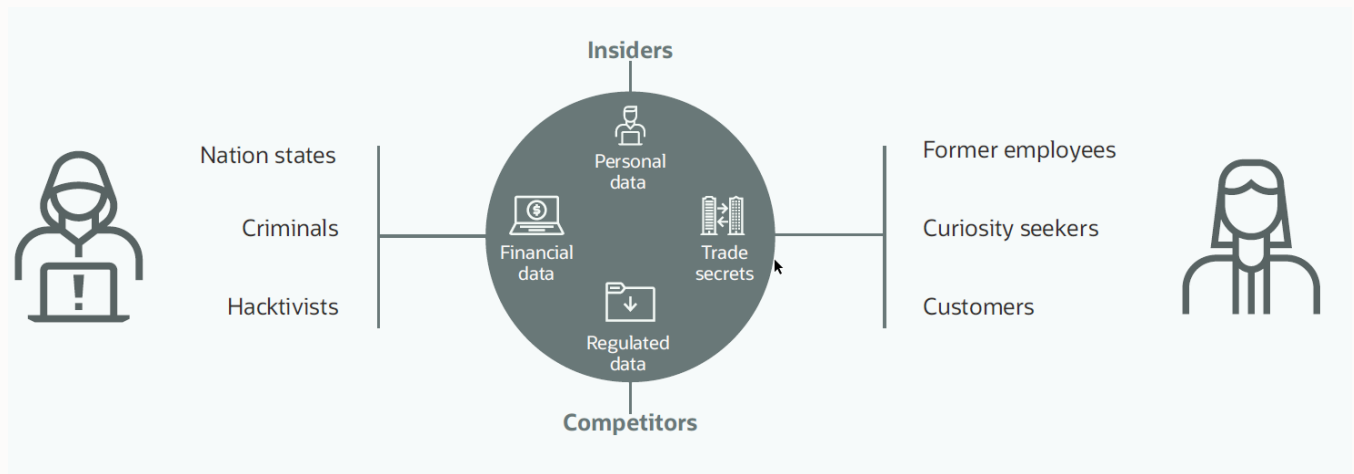


Figure 1-2: Threat actors

What tools or 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 proposed a list of the twelve most common database security risks we call the “Dirty Dozen.” The items on this list are a hacker’s “tool chest” of tactics, techniques and ways 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 through email-based phishing and other forms of social engineering or by using malware to sniff for credentials and data.
- Accessing accounts through password guessing or exploiting careless credential management.
- Exploiting application weaknesses with techniques like SQL injection, bypassing application layer security by embedding SQL code into a seemingly innocuous end-user-provided input.
- 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 encrypted data, rendering it inaccessible to users and demanding a ransom.

A complete list of the Dirty Dozen appears in Table 1.1

**A peek inside the Hacker’s tool chest:    The most common database security risks**

# THE DIRTY DOZEN

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



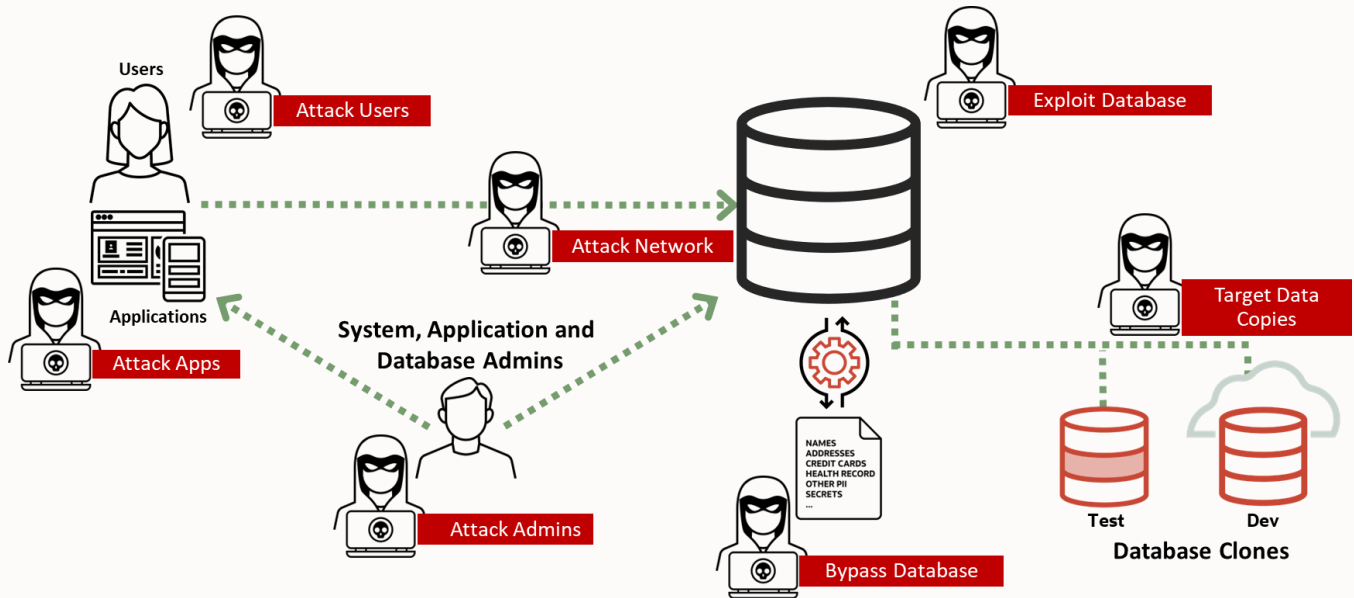


Figure 1-3: How hackers attack the database

## Addressing the Dirty Dozen through data security controls

A well-structured data security solution can help mitigate the risks from the Dirty Dozen. The best approach incorporates multiple layers of security controls to provide defense-in-depth protection from threats. We can group these controls into the following four categories:

- **Assessment controls** help assess the security posture of a database, including the ability to monitor and identify configuration changes. They also help you assess your users' security configuration, how much sensitive data you may have in the database, and where it resides.
- **Preventive controls** block access to data by unauthorized users with technologies such as encryption and database-level controls.
- **Detective controls** monitor user and application data access, allowing administrators to detect and block threats and support compliance reporting.
- **Exposure-limiting controls** selectively redact or obfuscate sensitive data to limit their opportunity for compromise or disclosure for various uses.

Finally, two additional categories of controls are fundamental to data security. These support the other controls and help provide the required defense-in-depth security for mitigating risks:

- **Data controls** enforce fine-grained access at the row and column level within the database, providing a consistent authorization model across multiple applications, reporting tools, and database clients.
- **User controls** enforce proper user authentication and authorization policies, ensuring only authenticated and authorized users can access their data.

Table 1.2 lists database security controls and how they map to Oracle products and technologies.

Control Category	Database Security Controls	Oracle Products and Technologies
<b>Assessment controls</b>	Security assessment	Database Security Assessment Tool (DBSAT), Data Safe, Audit Vault and Database Firewall (AVDF)
	User assessment	Data Safe
	Privilege analysis	Privilege Analysis
	Sensitive data discovery	DBSAT, Data Safe, Data Masking and Subsetting
<b>Preventive controls</b>	Network encryption	Native Network Encryption, TLS
	Data encryption	Transparent Data Encryption (TDE), RMAN encryption, Key Vault
	Privileged user controls	Database Vault
	SQL blocking	AVDF, SQL Firewall (23c)
	Automated database patching	Autonomous Database
<b>Detective controls</b>	Database activity auditing	Traditional Audit, Unified Audit
	Audit record collection and reporting	AVDF, Data Safe
	SQL monitoring	AVDF, SQL Firewall (23c)
<b>Exposure-limiting controls</b>	Dynamic masking	Data Redaction
	Static masking	Data Safe, Data Masking and Subsetting
<b>Data controls</b>	Fine-grained data access controls	Virtual Private Database (VPD), Real Application Security (RAS)
	Label-based access controls	Oracle Label Security
<b>User controls</b>	Password policies	User profiles
	Strong authentication	Kerberos, Certificate-based authentication, Multi-factor authentication, Token-based authentication (Cloud)
	Centralized user management	Centrally Managed Users in LDAP directories, Cloud-based identity integration

Table 1-2: Database security controls with applicable Oracle products and technologies.


With this comprehensive set of database security controls, we now begin to see how to deploy defense-in-depth security to address threats such as the Dirty Dozen listed in Table 1.1. Table 1.3 provides an example of connecting the Dirty Dozen with compensating controls that address these risks.

Dirty Dozen items	Representative security controls
<b>(DD #1) Insecure configuration and configuration drift</b>	Security assessment
<b>(DD #2) Unpatched and out-of-date systems</b>	Automated database patching
<b>(DD #3) Lack of a consistently enforced security policy</b>	Fine-grained data access controls
<b>(DD #4) Lack of visibility into sensitive data placement and quantity</b>	Sensitive data discovery

Dirty Dozen items	Representative security controls
<b>(DD #5) Overprivileged database users and administrators</b>	Privileged user controls
<b>(DD #6) Weak authentication and shared accounts</b>	Password policies Strong authentication Centralized user management
<b>(DD #7) SQL Injection vulnerabilities and insecure application design</b>	SQL monitoring SQL blocking
<b>(DD #8) Trusting vulnerable networks</b>	Network encryption
<b>(DD #9) Insufficient or inefficient monitoring and auditing</b>	Database activity auditing Audit record collection and reporting
<b>(DD #10) Sensitive data proliferation to non-production databases</b>	Static data masking
<b>(DD #11) Unprotected servers and database backups</b>	Data encryption
<b>(DD #12) Insecure encryption keys and secrets</b>	Key and secrets management

Table 1-3: Dirty Dozen security risks and compensating security controls.


*Insecure Configuration*  
*Unpatched systems*  
*Weakly protected 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


*DB users*  
*Trusted admin accounts*  
*Trusted networks*



**Control access to sensitive data**

- User profiles, CMU, SSO
- Privilege Analysis
- Database Vault
- Label Security
- Virtual Private Database
- Real Application Security


*SQL Injection*  
*Attack through application*  
*Escalate privileges*



**Monitor user activities**

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

*Servers and backups*  
*Encryption keys*  
*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: Combatting the Dirty Dozen

Initially, many organizations begin by implementing security controls on a project-by-project basis but then later expand the scope after realizing that hackers would target any unprotected system on the network and then use that as a launching point to attack other systems with sensitive data. Many organizations then move to centralized security management using tools and cloud services such as Oracle Data Safe, Oracle Audit Vault and Database Firewall, Oracle Key Vault, and Oracle Enterprise Manager.

Finally, since many organizations are migrating their workloads to the cloud and embracing new, agile deployment models, these controls need to scale and work seamlessly across on-premises, private cloud, public cloud, and hybrid cloud environments.

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 they work and the types of protection they provide. The following chapters cover different aspects of database security. Let's begin.

# Chapter Two

## **Assess database security posture and risk**

## Introduction

Today's systems are complex, with many security configuration settings. As recent data breaches have demonstrated, it is critical to have properly configured and secure systems. Human errors could leave your database open to everyone, or an attacker could maliciously exploit configuration mistakes to gain unauthorized access to sensitive data.

Failing to implement basic security controls may risk exposing customer 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 the most critical systems, such as 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. It is critical to regularly assess database security posture, considering recommendations from different regulations, security frameworks, and vendor best practices.

This chapter describes how Oracle database security solutions can help quickly evaluate your database security posture, categorize the findings, and recommend suitable action to develop a strategy to keep your databases secure.

## 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, open ports, and 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 they try to answer while probing your databases:

- Which version is the database? What are the known vulnerabilities? Have those been patched yet?
- Are there any known users with default or easy-to-guess passwords?
- Who are the privileged users on this database? Is there a way to escalate privileges from regular users?
- Which packaged applications are running on this database? Are those running with all-powerful privileges?
- Is auditing on? For whom? Which conditions? Can activities be tracked?
- Is the data encrypted? If not, can we access the underlying storage or a backup?

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 owners, you need to think like a hacker to harden your database's security posture.

Properly hardening and securing a database is a challenging task. Success requires understanding the 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. Because Oracle Database is highly customizable, assessing security requires understanding the impact of configuration choices on the overall security.

### Here are some key considerations for protecting your databases:

- Almost all databases hold sensitive data, but the level of importance 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 database vulnerabilities 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. These tools include the Database Security Assessment Tool (DBSAT), Oracle Data Safe (an Oracle cloud service), Audit Vault and Database Firewall (AVDF), and Oracle Database Life Cycle Management (a management pack for Oracle Enterprise Manager).

## **Assessing configuration security of the Oracle Database with DBSAT**

DBSAT identifies areas where your database 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 to assess the current security status of a database quickly and recommends best practices. Updated best practices rules are delivered periodically with new versions of the tool.

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

DBSAT is a free command line tool available to all Oracle customers to quickly find sensitive data, evaluate their database security posture, identify gaps, and help implement the recommended security best practices for their organization. DBSAT is also used worldwide by Oracle consultants and partners while performing database security assessments.

### **DBSAT security assessment reports**

DBSAT has three components: collector, reporter, and discoverer. The collector and reporter are used for generating database security risk assessments, and the discoverer discovers the different types of sensitive data in the database.

The DBSAT Collector first gathers security configuration information from the database and underlying OS. The DBSAT Reporter then analyzes the collected data and generates detailed findings and recommendations. The output reports are in HTML, spreadsheet, text, and JSON formats. The DBSAT Discoverer (described later in Chapter Three) helps to identify sensitive data by looking into table metadata (comments and column names), and classifies, and summarizes the findings in HTML and spreadsheet reports.

The HTML reports provide detailed assessment results in a format that is easy to navigate. 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.

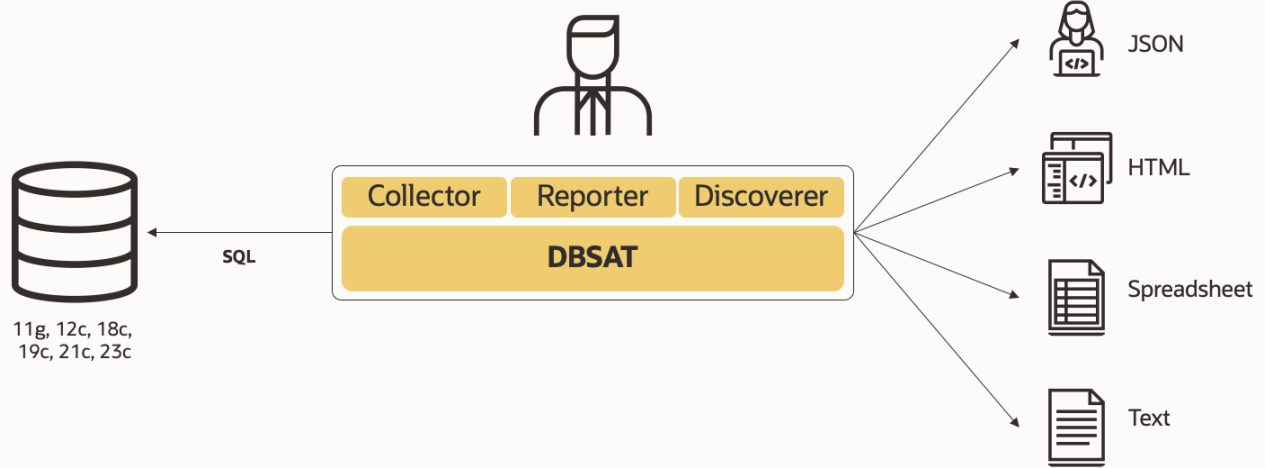


Figure 2-1: Database Security Assessment Tool

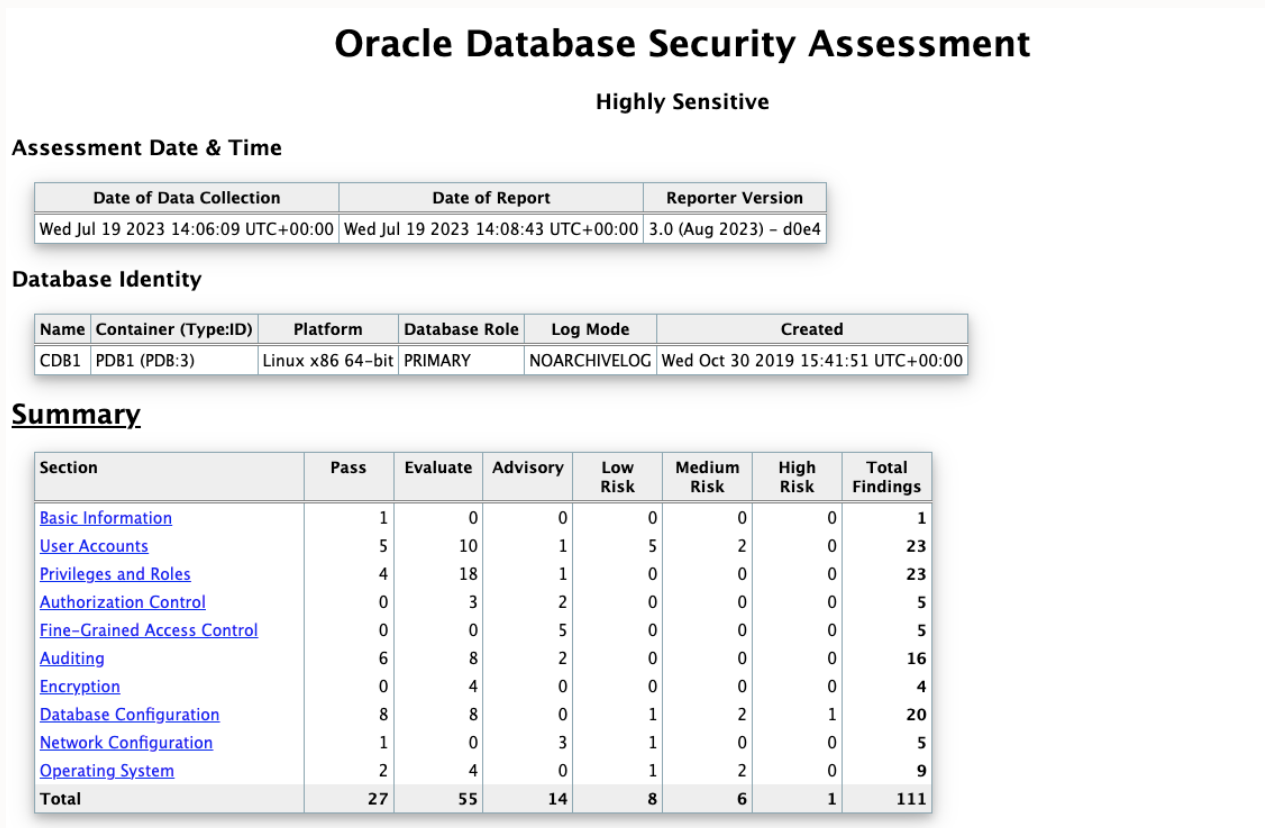


Figure 2-2: Database Security Assessment report summary

The resulting analysis is reported as findings that consist of the following:



- Rule ID: By convention, a rule ID contains a prefix that identifies the report section for the finding, followed by a period and a name to identify the finding uniquely.
- One-liner: A single sentence that describes what should be done.
- 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, followed by recommendations.
- Remarks: Explain the reason for the rule and recommended actions for remediation.
- References: When applicable, it will reference the corresponding CIS Oracle Database benchmark recommendation, Oracle Database STIG Rule, and the EU GDPR article/recital.

In the finding below, DBSAT has identified nine users with the DBA role, and that further analysis is needed (Status = Evaluate). The Remarks provide 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 12c Benchmark recommendation 4.4.4, DISA STIG rule V237710 and 237711, and an Oracle-recommended best practice.

**Users with DBA Role**

PRIV.DBA

CIS
OBP
STIG

Ensure DBA and PDB\_DBA roles are granted only to necessary users

<b>Status</b>	Evaluate
<b>Summary</b>	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.
<b>Details</b>	<p>Users with DBA/PDB_DBA role:</p> <p>DBA_DEBRA: DBA</p> <p>DBA_HARVEY: DBA</p> <p>DBA_NICOLE: DBA</p> <p>DMS_ADMIN: DBA</p> <p>EVIL_RICH: DBA</p> <p>JTAYLOR: DBA</p> <p>MASKING_ADMIN: DBA</p> <p>PDBADMIN: PDB_DBA(*)</p> <p>SCOTT &lt;- APPROLE1 &lt;- APPROLE2 &lt;- APPROLE3: DBA</p> <p>(*) = granted with admin option.</p>
<b>Remarks</b>	<p>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.</p> <p>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.</p>
<b>References</b>	<p>Oracle Best Practice</p> <p>CIS Benchmark: Recommendation 4.4.4</p> <p>DISA STIG: V-237710, V-237711</p>

Figure 2-3: Sample finding: Users with DBA role

## Assessing your database fleet using Oracle Data Safe

DBSAT is a great tool to assess a few databases quickly, but what if you have dozens, hundreds, or thousands of databases?

Enter Oracle Data Safe. Data Safe is an Oracle Cloud Infrastructure (OCI) service focused on Oracle Database security.

Customers can use Data Safe to gain visibility on their databases' security whether running on-premises, in the Oracle Cloud, or 3<sup>rd</sup> 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 for non-production environments. 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 employs a simple “click-and-secure” model designed for users with no special 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 to database administrators.

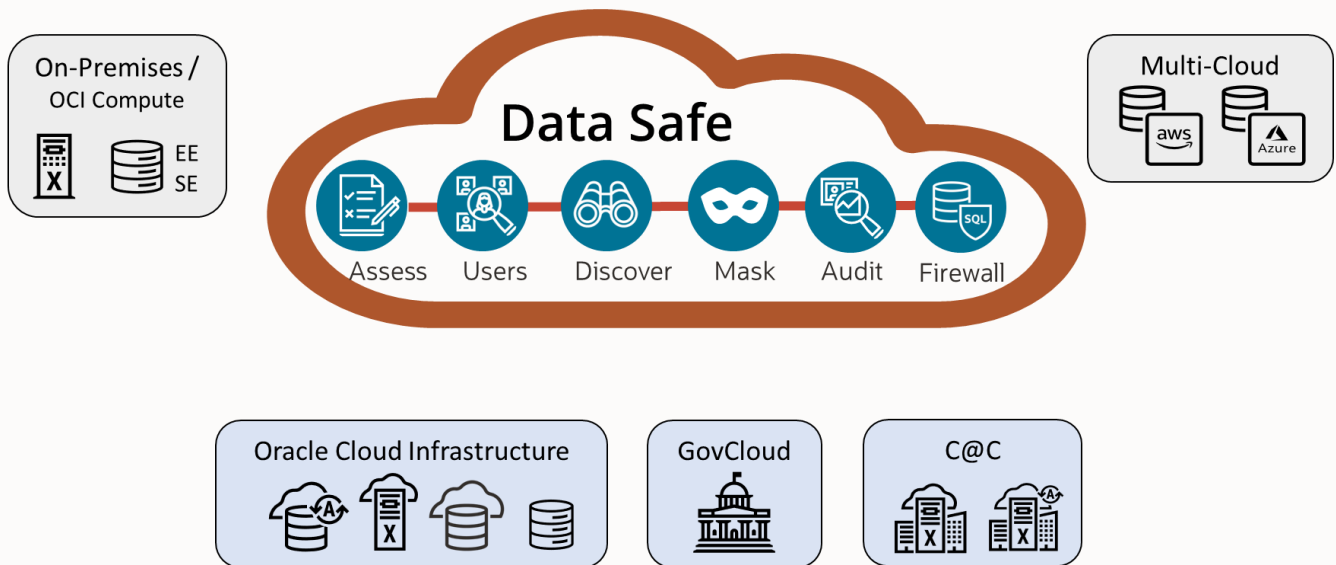


Figure 2-4; Oracle Data Safe essential security for Oracle databases, both in the cloud and on-premises

### Assessing database security with Data Safe

Data Safe’s security assessment helps identify configuration gaps that could represent a vulnerability. Data Safe performs a comprehensive check of your database configuration. It examines user accounts, privilege and role grants, authorization controls, fine-grained controls, auditing, encryption, and configuration parameters. Data Safe identifies gaps compared to organizational best practices and delivers actionable reports with prioritized recommendations and mappings to common compliance mandates like EU GDPR, DISA STIGs, and CIS benchmarks.

The following snippets show a sample security assessment report, including a high-risk finding and a finding that needs further analysis. 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.

Assessment summary
Assessment information
Tags

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
<b>Total risks</b>	<b>2</b>	<b>6</b>	<b>6</b>	<b>3</b>	<b>40</b>	<b>12</b>	<b>69</b>

Displaying 7 categories

---

### Assessment details

Expand all

- ∨ All Risks
- ∨ User Accounts (Note: Predefined Oracle accounts which are schema-only or locked are not included in this report)
  - > User Profiles
  - > User Details
  - > ■ Password Verification Functions
  - > ■ Account Locking after Failed Login Attempts
  - ∨ ■ User Schemas in SYSTEM or SYSAUX Tablespace

**Status:** MEDIUM

**Summary:** Found 4 users using SYSTEM or SYSAUX tablespace.

**Details:** Tablespace SYSTEM: DBA\_DEBORA, RUSS  
Tablespace SYSAUX: APP1\_DATA, PEDRO

**Remarks:** The SYSTEM and SYSAUX tablespaces are reserved for Oracle-supplied user accounts. To avoid a possible denial of service caused by exhausting these resources, regular user schemas should not use these tablespaces.

**References:** **STIG:** Rule SV-75949r2, SV-75951r3

- > ■ Sample Schemas

Figure 2-5: Oracle Data Safe: Security assessment

Data Safe also allows you to configure acceptable risks. For example, if the assessment lists a finding that needs to be evaluated (“Evaluate”; needs manual confirmation), you can mark the finding as “Pass” after successfully validating that there is no risk in your case. You can lower a finding’s risk if you have other compensating controls that reduce the risk. You can accept the risk if you do not plan to address it.

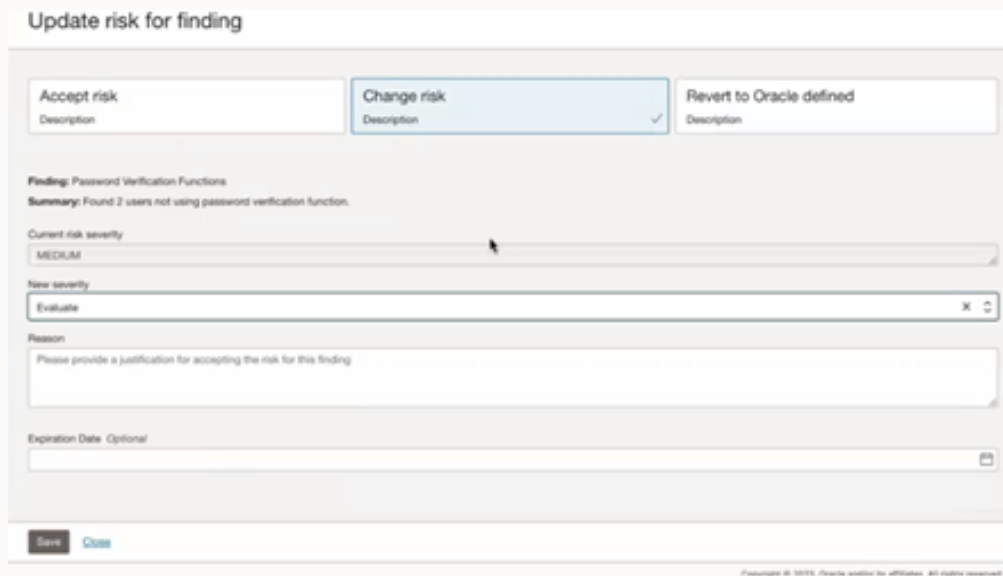


Figure 2-6: Oracle Data Safe: Configuring acceptable risks

### Understanding user risk with Data Safe

Data Safe offers a unique capability that allows you to evaluate the risk represented by various database users. User risk assessment evaluates database users, looking at static and dynamic user profile characteristics to identify the highest-risk users. User risk is presented alongside other user 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. 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 related audit records showing that user’s activity.

Stealing privileged user credentials is the most common method hackers use to access sensitive data. Data Safe helps you take steps to make your applications more secure by providing the ability to assess and visualize user risk.

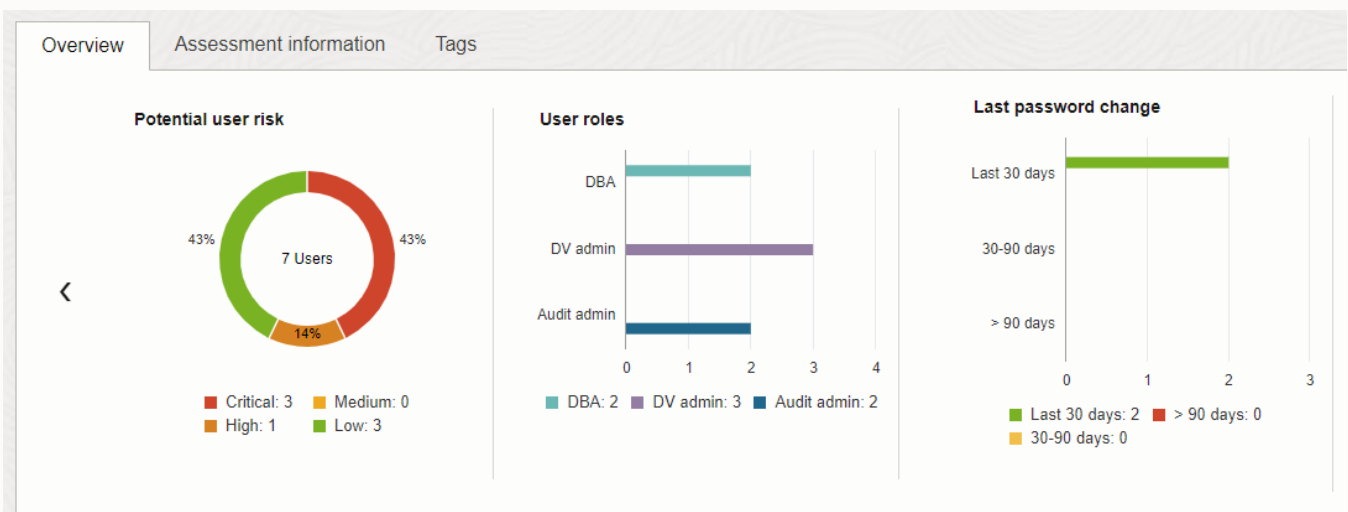


Figure 2-7: Oracle Data Safe: User risk assessment

Data Safe User Assessment also provides you with visibility over 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.

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
- 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

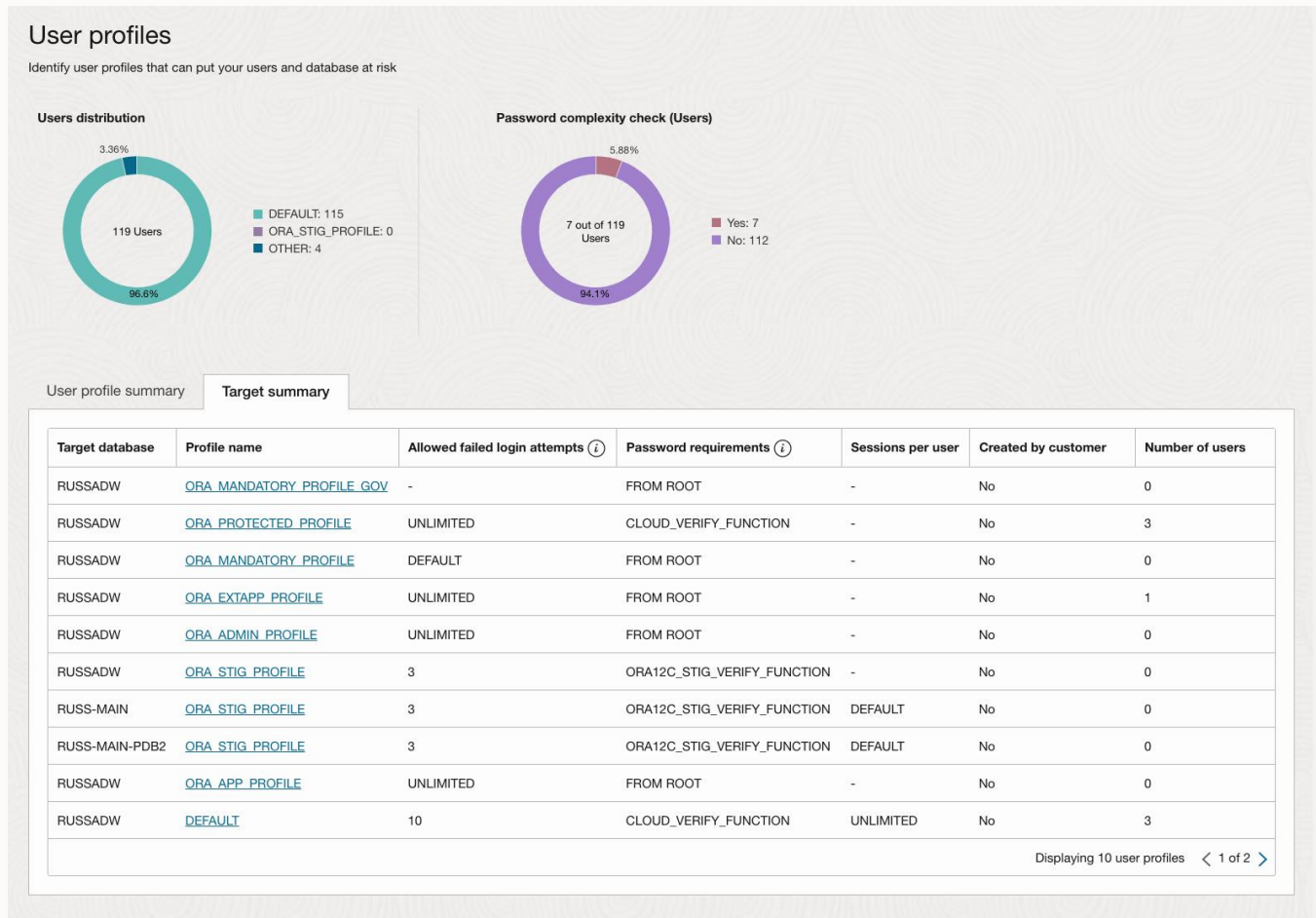


Figure 2-8: Oracle Data Safe: User profile insight

See how else Oracle Data Safe can help:

- To discover sensitive data, please see Chapter three.
- To detect or block SQL injection attacks, please see Chapter seven.
- 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 twelve.

## Assessment with Oracle Audit Vault and Database Firewall (AVDF)

Starting with release update (RU) 20.9, AVDF provides a fleet-wide simplified and centralized view of security configuration assessments for all your Oracle databases, along with the security findings and associated risks. Like Data Safe, AVDF also leverages the [Database Security Assessment Tool \(DBSAT\)](#) for Oracle Databases. 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.

Database Security Posture Management in AVDF adds to the existing capabilities of AVDF, including centralized audit monitoring, audit policy provisioning, reporting, and database firewall.

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.

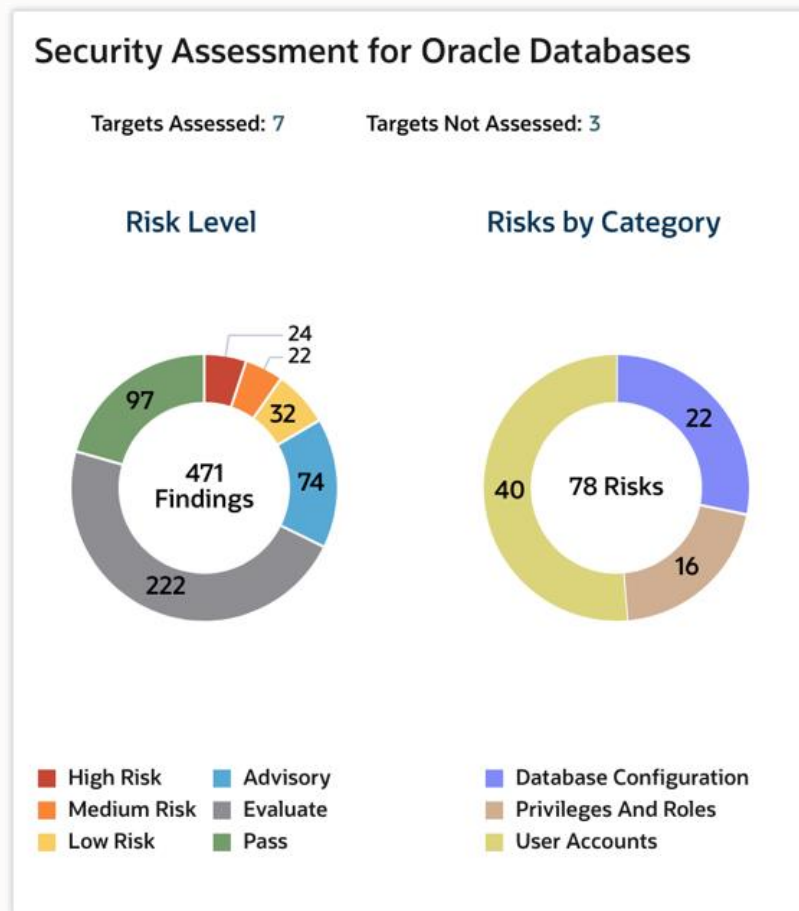


Figure 2-9: Audit Vault and Database Firewall security posture management

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

- To prevent SQL injection attacks, please see Chapter Seven.

- Please see Chapter Twelve for information on centrally managing audit records for analysis, alerting, and reporting.

## 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 provides numerous reports for security configuration checks and a comprehensive compliance framework. Reports include information on initialization parameters, operating system directory permissions, user account profiles, and sensitive object reports.

You can customize the compliance framework by adapting existing standards and rules or creating new ones. DBLM also ships with a DISA Security Technical Implementation Guide (STIG) compliance standard, including rules to validate STIG requirements.

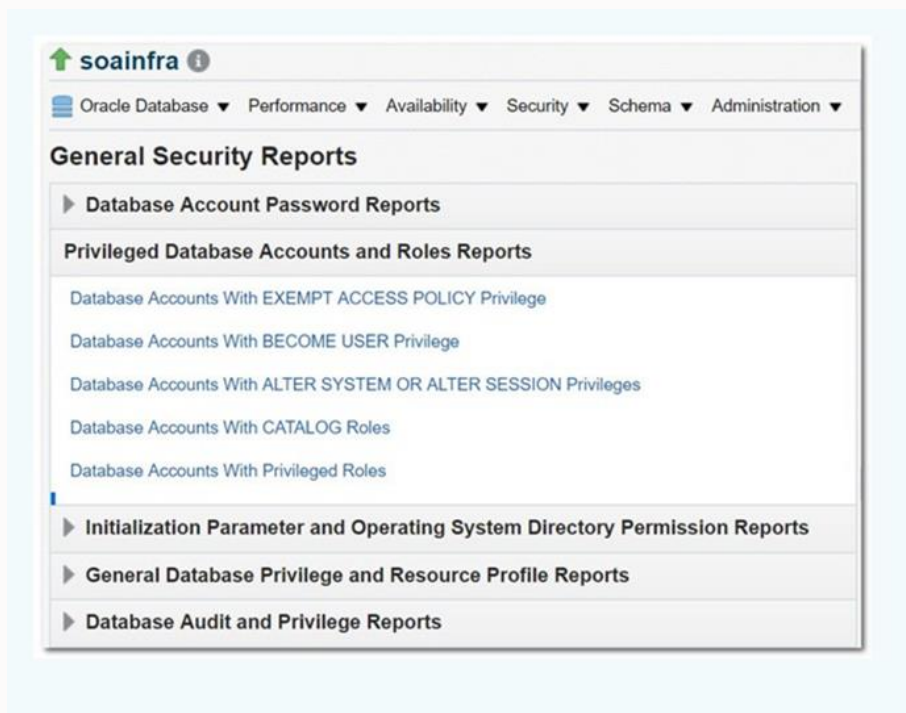


Figure 2-10: Oracle Enterprise Manager general security reports

DBLM ships over four dozen out-of-the-box compliance standards, including basic security configuration for Oracle Database, RAC nodes, and Oracle Listener. It also monitors configuration for Exadata Compute nodes and Security Linux packages. In addition, trend analysis allows fine-grained tracking of compliance scores over time.

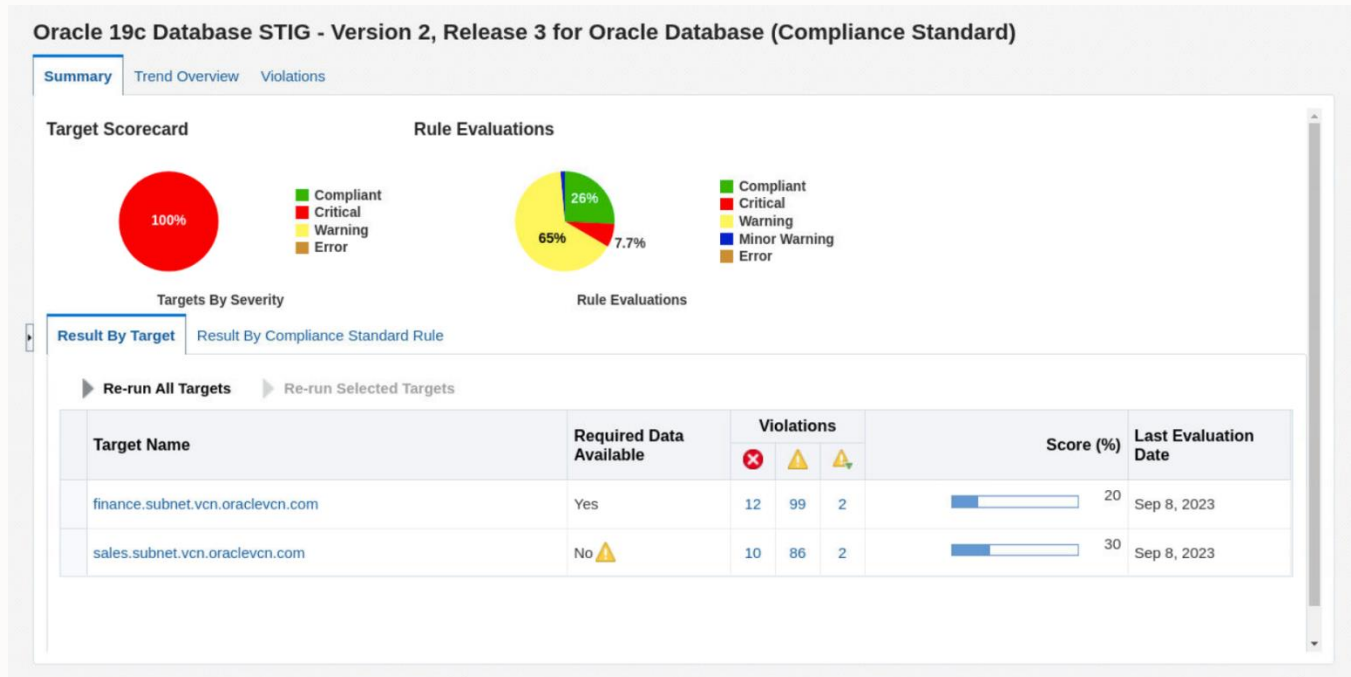


Figure 2-11: Enterprise Manager STIG compliance standard detail

### Asset discovery and grouping

DBLM eliminates the need to track IT assets, including databases, manually. It provides non-intrusive network scanning capabilities to discover servers, databases, and other applications. With the ever-growing number of systems and services that administrators are responsible for, administrators need a view that includes only those targets they need to monitor and manage.

Through such “Groups,” you can monitor and manage different targets collectively, efficiently perform administrative operations against the targets, and consolidate and monitor your distributed targets as one logical entity. For example, you can define a TEST group containing all applications, databases, and host targets within your test environment.

From a group’s home page, an administrator can:

- Determine the security configuration compliance of all the members in the group and outstanding alerts,
- Drill down and analyze the specifics of a particular target,
- Compare multiple targets and find out configuration divergence. For example, it could find out which database has not enabled auditing.

### 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.

- DBSAT 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.
- If corporate/regulatory requirements require you to keep all security configuration information on-premises and within your full control, the tool of choice is AVDF. Besides activity monitoring, AVDF provides fleet-wide security posture management of your databases. Functionality-wise, both AVDF and Data Safe are very similar from a security assessment perspective.
- If you already use Oracle Enterprise Manager and want to assess beyond the database or need to do customizations, the Database Lifecycle Management Pack is your answer.

## 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 login to 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. That approach is facilitated by tools like Data Safe's User Assessment or AVDF's entitlement reviews. However, managers may be reluctant to mark privileges, and DBAs may hesitate to revoke roles and privileges from user accounts as the potential impact is unknown.

Another 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 dynamically analyzes privilege and role usage for database users and application service accounts at run time. PA identifies unused privileges and roles based on the actual user or application usage of the roles and privileges during a period of time. Removing privileges you know are not being used is easier than asking a user's manager if they still need the privileges they already have.

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

The figure below shows that the APPS user has been granted 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 2-12: Enterprise Manager report on DBA\_UNUSED\_PRIVS

The following figure 2.13 also shows interesting details of used roles and privileges. Here, 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 DBA should revoke the SELECT ANY TABLE system privilege that allows the APPS user to select from any table in the database and grant an object privilege 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 23c.

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 2-13: Enterprise Manager report on DBA\_USED\_PRIVS

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, 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. Doing this minimizes the chance of encountering known bugs and security vulnerabilities. If you run your databases on Linux x86-64 platforms and have tested an application against an RU and want to go live, you should check for the latest MRP and apply it.

Quarterly patch 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 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 the critical content released up to six months prior. By waiting to take new RU content for up to six months, you can take a more conservative approach to Oracle Database software maintenance, but you still risk the chance of hitting known issues that are fixed in the most recent RU. The main benefit of this patching strategy is that if any regressions are reported on the base RU or succeeding MRP, they will be fixed in later MRPs.

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

Table 2-1: 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 compliance of the environments. 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 the database is 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) provide the means to help you identify the gaps in your Oracle database security configuration and the recommendations to overcome these gaps.

# Chapter Three

## **Discovering sensitive data**

---

## Introduction

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 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

## Why is sensitive data important?

The amount of data that organizations collect and manage is growing every day. In today’s world, data is the most valuable resource and a necessity for every organization. A significant percentage of data is sensitive or personal. Malicious actors monetize stolen data by committing identity theft and financial fraud, selling government and trade secrets, or using it for future attacks. Because of its value, data loss can impact companies’ finances, reputations, customer trust, and competitiveness. The importance of data and growing security threats make it necessary to protect sensitive information.

At the same time, data privacy laws and standards such as the European Union General Data Protection Regulation (EU GDPR), California Consumer Privacy Act (CCPA), Payment Card Industry Data Security Standard (PCI-DSS), and Health Insurance Portability and Accountability Act (HIPAA) mandate personal data protection.

What is considered sensitive? Sensitive data should not be made available to unauthorized people, whether they operate inside or outside the organization. The following figure shows a few examples of the 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 3-1: Examples of sensitive data categories and types

These sensitive types are arranged under sensitive categories such as identification, biographic, healthcare, financial, employment, and academic data.

## Discovering sensitive data

The most common way to discover sensitive data in a database is to search for column names using keywords or search patterns (regular expressions).

Data patterns can also be used to check values in a column. Combined with a column name or comment match, a data pattern helps raise confidence that a certain column has sensitive data.

Oracle provides multiple sensitive data discovery tools: Database Security Assessment Tool (DBSAT), Oracle Data Safe, Oracle Audit Vault and Database Firewall (AVDF), and Enterprise Manager (EM) Application Data Modeling (ADM). They can help discover over 150 common sensitive and personal data. Users can modify the predefined sensitive types and create new ones to meet specific requirements.

## Discovering sensitive data using DBSAT

Chapter Two introduced the Oracle Database Security Assessment Tool (DBSAT) to review database configurations. DBSAT can scan your database to identify sensitive data by inspecting column names and comments to determine if they hold sensitive data. DBSAT also 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 library of sensitive data definitions 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 3.1 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
<b>Total</b>	<b>29</b>	<b>132</b>	<b>8617644</b>

Table 3-1 Sensitive Data Report - Category Summary

Each sensitive category has a predetermined risk level (high, medium, or low). The report recommends protecting sensitive data based on the associated risk level. Figure 3.2 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**

<b>Risk Level</b>	High Risk
<b>Summary</b>	Found BIOGRAPHIC INFO – ADDRESS within 51 Column(s) in 17 Table(s)
<b>Location</b>	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_QUEUE_TABLE_H, IX.AQ\$ ORDERS_QUEUE_TABLE_L, IX.AQ\$ ORDERS_QUEUE_TABLE_S, IX.AQ\$ STREAMS_QUEUE_TABLE_S, LOOKUPS.LOOKUP_ADDRESSES, LOOKUPS.LOOKUP_PLACES, OE.CUSTOMERS, SH.COUNTRIES, SH.CUSTOMERS

Figure 3-2: 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 Two, 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 create your own custom sensitive types and categories.

Figure 3.3 shows a sample data discovery report generated by Data Safe.

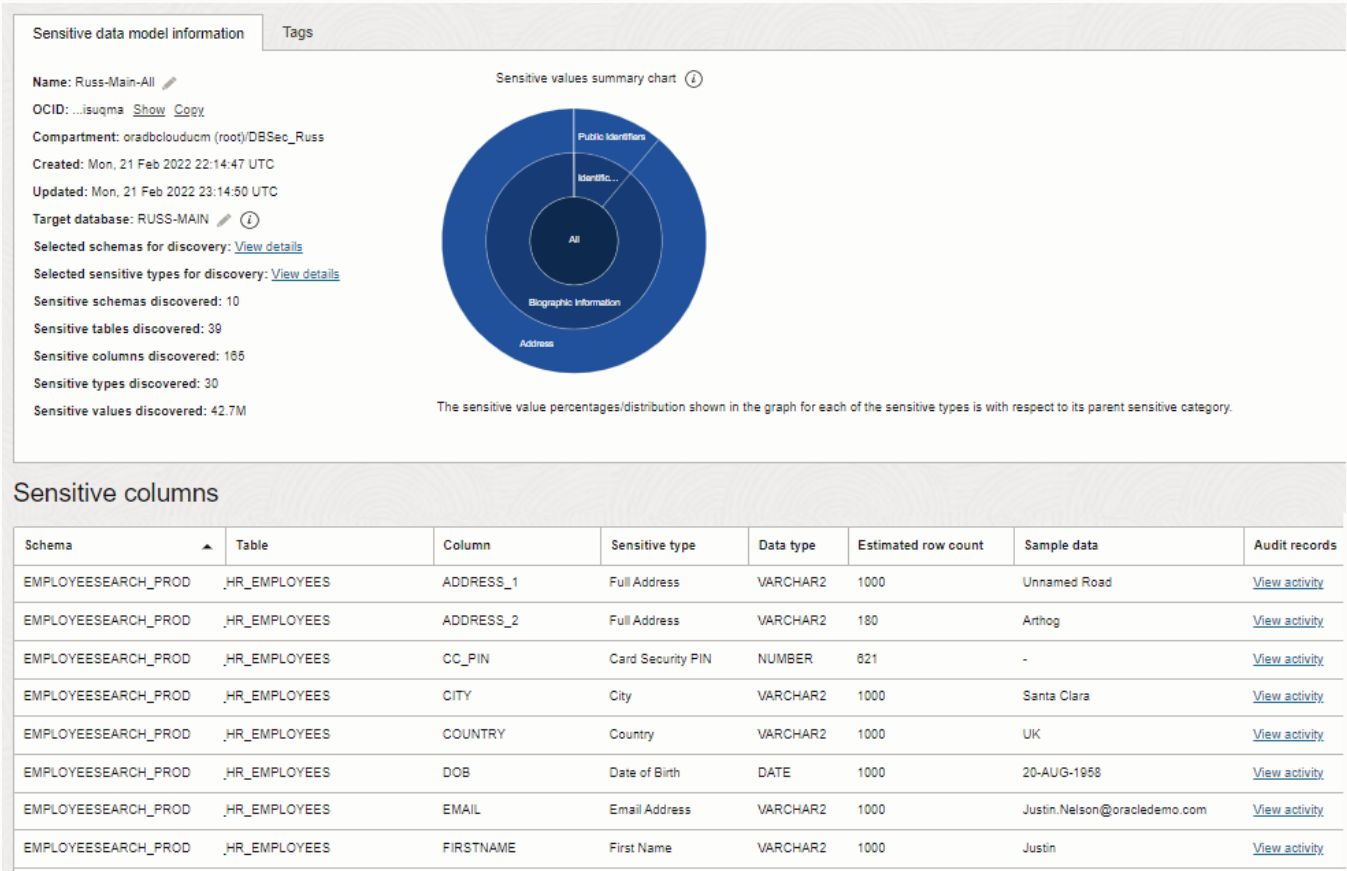


Figure 3-3: Data discovery report in Oracle Data Safe

Data Safe scans column names, comments, and data values. Data Safe further identifies relationships between primary and foreign key columns and includes those in the sensitive data model (SDM). Data Safe also allows you to use non-dictionary referential relationships defined in the application to find sensitive columns.

### Sensitive Data Models in Data Safe

Data Discovery reports provide 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 are stored as a sensitive data model (SDM), allowing it to be shared or reused. You can optionally store metadata in a sensitive data model, including sample data and estimated row counts -depicted in Figure 3.3. This information gives you a perspective on the quantity of the different types of sensitive data in your target databases.

When changes occur on 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.](#)

Name	State	Description	Created	Updated
<a href="#">SensitiveDataModel_20230508203</a>	● Active		Tue, 09 May 2023 03:04:28 UTC	Wed, 31 May 2023 15:57:44 UTC
<a href="#">SensitiveDataModel_202305171323</a>	● Active		Wed, 17 May 2023 20:24:17 UTC	Wed, 17 May 2023 20:25:06 UTC
<a href="#">SensitiveDataModel_202305031259</a>	● Active		Wed, 03 May 2023 20:00:21 UTC	Wed, 03 May 2023 20:01:02 UTC
<a href="#">SDM_Exa</a>	● Active		Wed, 19 Apr 2023 22:47:47 UTC	Wed, 19 Apr 2023 22:48:42 UTC
<a href="#">SensitiveDataModel_202304191450</a>	● Active		Wed, 19 Apr 2023 21:51:06 UTC	Wed, 19 Apr 2023 21:52:05 UTC
<a href="#">SensitiveDataModel_202304171612</a>	● Active		Mon, 17 Apr 2023 23:13:55 UTC	Mon, 17 Apr 2023 23:14:43 UTC
<a href="#">SensitiveDataModel_202304051053</a>	● Active		Wed, 05 Apr 2023 17:54:23 UTC	Wed, 05 Apr 2023 17:55:06 UTC
<a href="#">SensitiveDataModel_202303291250</a>	● Active		Wed, 29 Mar 2023 19:51:13 UTC	Thu, 30 Mar 2023 17:30:30 UTC
<a href="#">SDM-incr_Test</a>	● Active		Fri, 17 Mar 2023 18:57:32 UTC	Fri, 17 Mar 2023 19:15:36 UTC
<a href="#">SDM-Tina_202303161341</a>	● Active		Thu, 16 Mar 2023 20:42:49 UTC	Thu, 16 Mar 2023 20:46:44 UTC

Displaying 10 sensitive data models < 1 of 19 >

Figure 3-4: Data discovery sensitive data models

Sensitive data models help you design other security controls, such as data masking, auditing, and fine-grained access controls. For example, you can define a masking policy using a sensitive data model to mask the sensitive data on target databases. You can reuse a sensitive data model for multiple masking policies.

After discovering sensitive data for a target, you can also get visibility on who accessed sensitive data (if you also have audit policies on those objects) via the Activity Auditing reports. Oracle Data Safe lets you quickly drill down from the sensitive data model to the user activity report. You can also view the sensitive data activity report to review access to sensitive objects across your target databases and all sensitive data models.

## Discovering sensitive data using Audit Vault and Database Firewall

Oracle Audit Vault and Database Firewall (AVDF) assists you in discovering sensitive data and privileged users in Oracle Database. Like DBSAT, AVDF scans column names and column comments. Unlike Data Safe, AVDF does not scan column data. Sensitive data scans are done by scheduling the User Entitlements and Sensitive Object discovery jobs.

The sensitive data list could be used to determine how you will protect that data. Audit Vault and Database Firewall can also use the sensitive data list for multiple purposes. Once the privileged users and sensitive objects have been discovered, they can be added to the Database Firewall privileged user and sensitive object sets, respectively. These sets are global and can be used in multiple Database Firewall policies. As an example, a Database Firewall policy can:

- Monitor user access and their operations on sensitive data
- Block unauthorized access to sensitive data
- Monitor sensitive data exfiltration attempts

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 3-5: Sensitive data report

## Discovering sensitive data using Enterprise Manager

The Application Data Modeling (ADM) feature in Oracle Enterprise Manager (EM) can identify sensitive data present within an application and where it resides within the database schema. Like Oracle Data Safe, ADM examines column names, comments, and data to discover sensitive columns. This helps drive down false negative and false positive rates associated with the data discovery process. For example, ADM can help locate credit card and national identification numbers based on the column name, column comment, and data.

### Discovering sensitive columns and referential relationships

ADM uses sensitive column types to perform pattern matching and identify sensitive columns. Users can review the discovered sensitive columns and manually add additional columns to the list if required. Figure 3.7 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 3-6: Sensitive columns discovered using Enterprise Manager ADM

ADM 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 3.8 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 3-7: Referential relationships

Figure 3.9 puts this all together. ADM automatically discovers sensitive columns, database-defined referential relationships, and application-level referential relationships and stores them in the Enterprise Manager repository. This application data model can be used to implement security controls such as data masking and subsetting.

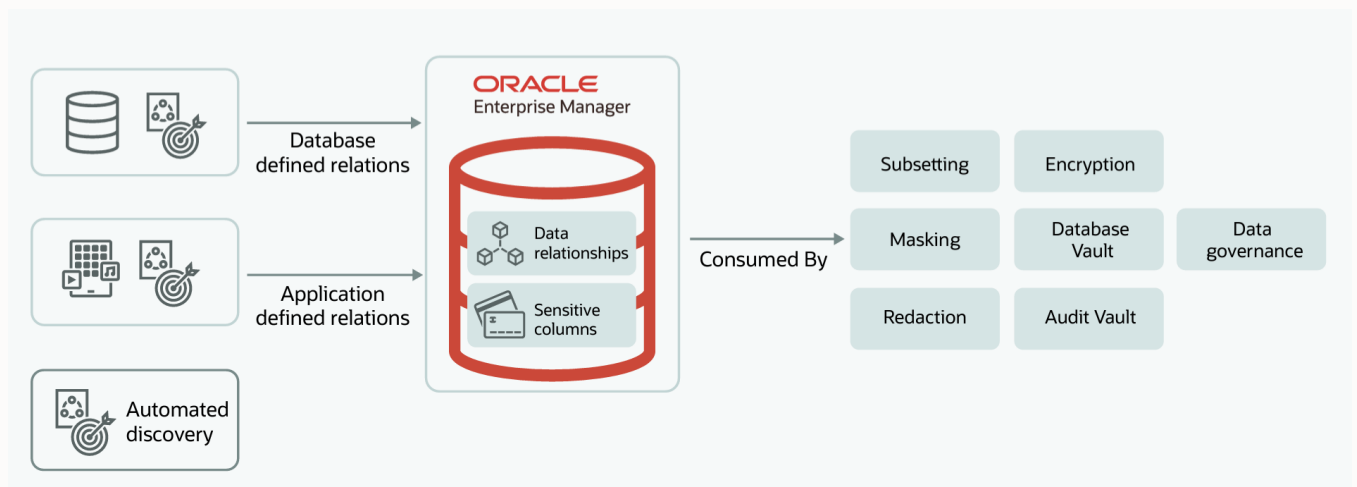


Figure 3-8: Overview of Application Data Model (ADM)

## 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 give it a try on 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 Application Data Model)

## 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, and Enterprise Manager (EM) Application Data Modeling.

- DBSAT is a lightweight, easy-to-use tool that helps quickly analyze the sensitive data in a database and understand the risk. It automatically identifies sensitive columns, classifies them into risk categories, and provides detailed reports.
- Oracle Data Safe offers a comprehensive data discovery feature and is recommended for discovering sensitive data for Oracle databases in the Oracle Cloud, 3rd 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. As Data Safe is a cloud service, customers do not need to manage infrastructure and can use Data Safe APIs to address complex use cases.
- AVDF leverages DBSAT for sensitive data discovery, integrates the discovered objects into Database Firewall policies, and provides data privacy reports.
- The Enterprise Manager (EM) Application Data Modeling (ADM) 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 users to implement security controls effectively, minimize security risk, and address requirements associated with regulations such as CCPA, EU GDPR, PCI-DSS, and HIPAA.

# Chapter Four

## **Authenticating database users**

---



## Introduction

A fundamental step in securing a database system is validating the identity of the users accessing the database (authentication) and controlling the operations they can perform (authorization).

This chapter discusses how a proper 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.

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

## Users: Your weakest link

The easiest (and unfortunately most common) way to hack into the database is to impersonate an authorized user on that database. Some of the common techniques include:

- Apply social engineering to capture account credentials: With targeted phishing attacks, hackers can target end users or database administrators (DBAs) in an organization (who are easy to find via social media channels such as LinkedIn) and steal their credentials. Generative AI (GenAI) makes this task even easier.
- Try passwords used on other compromised sites: Many users use the same password across multiple applications or websites, and if any of them get compromised, attackers can try those passwords to attack your database.
- Find hardcoded database connection information: Applications frequently connect to a database using embedded database usernames and passwords or store these credentials 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 trying combinations of known passwords and their variations, hackers can break into database accounts with weak passwords when there are no limits on password retries. Brute force attacks are much harder as the database implements exponential backoff on incorrect passwords, but the possibility still exists. Without enforcement of complex passwords, some users may use passwords that are easy to guess, such as 'password' and 'Oracle123'.

These attacks are not necessarily sophisticated and can be executed by “script kiddies.” Still, they give hackers at least as much access as that user, and perhaps more if the attacker is even minimally skilled. We will address most of these potential attacks later in this chapter.

## Database authentication methods

Oracle supports different means of authentication, including hashed passwords stored locally in the database or with centralized directory or identity services. Users can also be authenticated by the operating system or external authentication services, including the OCI IAM and Azure AD cloud identity providers, Kerberos, public key certificates, and RADIUS.

Passwords are used for one-way user authentication to the database. In contrast, Kerberos, public key certificates, and certain access tokens involving authentication via a third-party service may provide additional layers of protection beyond simple password checks. While passwords are convenient, it may be easier to compromise a user's

password than their Kerberos or PKI credentials. We will describe later how to increase password security through stronger password profiles.

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. The schema and the user are the same entity when using local database password authentication. When the user logs into the database, they are connected to their own schema. When authenticating users with a directory or identity service, users either get their own dedicated database schema (exclusive mapping) or get mapped to a shared schema (shared mapping).

Although we won't talk about authorization until Chapter Five, 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
<b>Database Password</b>	Schema is the same as user	Managed in the database
<b>OS Password</b>	Database maps the user to a schema	Managed in database or through OS groups
<b>Kerberos</b>	Database maps the user to a schema	Managed in database
<b>Public Key (PKI)</b>	Database maps the user to a schema	Managed in database
<b>RADIUS</b>	Database maps the user to a schema	Managed in database or through RADIUS server
<b>Oracle Directory Services</b>	Managed in database and directory service	Managed in database and directory service
<b>Active Directory</b>	Managed in database and directory service	Managed in database and directory service
<b>OCI IAM</b>	Managed in database and identity service	Managed in database and identity service
<b>Azure AD</b>	Managed in database and identity service	Managed in database and identity service

Table 4-1: Database user authentication and authorization methods

## Making users resistant to attacks

With password authentication, users are expected to remember and use complex passwords and enter them when needed. Different passwords should be used for different databases, and the organization should prohibit sharing database passwords. However, administrators and regular users are attracted to convenience and shortcuts, and hackers are ready to exploit such human behavior. Below are mechanisms that put constraints on this potential weakness.

### Problem: Storing 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 Stores (SEPS) attempted to solve this problem by storing various authentication credentials, including passwords, private keys, and certificates in an encrypted file form known as an Oracle Wallet. With SEPS, the users need 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 could use brute force tactics to eventually open the wallet and read the credentials.

Over the last few years, secrets management – a more modern and secure way to handle password storage and distribution – has become popular. Using a secrets manager like 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. 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.

### Problem: Sharing accounts and passwords

Application administrators often need to connect to an application schema for maintenance. If there are multiple application administrators, they all typically know and share the application username and password.

If there are multiple DBAs, they also sometimes share passwords. Sharing passwords may be convenient, but it provides no accountability and makes auditing and investigating issues difficult.

Proxy authentication solves the problem of allowing an administrator to act as the application account without the administrator knowing the account's password. With proxy authentication, a database user can act as another user working through their own account. 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 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 knowing the password for the application schema account. For example, `alice_appdba` connects using their password and assumes the identity and privileges of the `hrapp` schema by proxy as follows:

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

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).

You can use Oracle Key Vault to securely store and manage database account credentials for application use. Key Vault eliminates the overhead of managing, updating, and protecting Secure External Password Stores (SEPS). OKV allows you to centralize your credentials instead of having local SEPS wallets spread across all your application servers. OKV, 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 require the use of username/password for authentication, this is the recommended approach.

### Problem: Poor password hygiene

Sometimes, users use very weak and short passwords, making it easy for somebody to guess the password and access their account.

User profiles can be used to create a common policy of password and resource authorization parameters for user accounts. Each user account can be associated with a selected user profile to simplify the management of common policies across an organization.

Database users always have a profile assigned. The default profile is assigned if no profile is specified when a user is created. The sample profile below (`org_profile`) incorporates both password and resource authorization parameters.

```

SQL> CREATE PROFILE org_profile LIMIT
CONNECT_TIME 90 -- Limit connection time to 90 min
SESSIONS_PER_USER 2 -- Allow two sessions for each user
IDLE_TIME 30 -- Automatic logout after 30 min idle
FAILED_LOGIN_ATTEMPTS 6 -- Lock the account after 6 attempts
PASSWORD_LIFE_TIME 180 -- Force password change after 180 days
PASSWORD_VERIFY_FUNCTION ora12c_stig_verify_function; -- STIG password complexity rules

```

Figure 4-1: Sample password profile

One of the default password complexity functions included with the database, the `ora12c_stig_verify_function`, imposes several password requirements that correspond with the United States Department of Defense Security Technical Implementation Guide (STIG), including a minimum length, minimum number of alphanumeric characters, and at least one special character.

Each user should have an associated profile to ensure that a common baseline security policy is uniformly applied. Changes to the security policy can easily be done by changing a single profile instead of changing every user account.

### Problem: Downtime when changing application account passwords

The database accounts for applications 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). Most organizations would schedule downtime for their applications to eliminate the chance that the application might fail when it tries to log in during password rotation. Some organizations even forego password updates to minimize application downtime.

Starting with Oracle Database 19c, you can create a new password for an application while allowing other application clients to use the old password for a limited period. 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.

### Problem: Managing multiple accounts and passwords

Users with accounts on multiple databases tend to keep the same password for convenience. However, they then must update the password on every database when any single database requires a password update. It is difficult for administrators to manage unique accounts for each user in every database because each account needs to be provisioned separately. When an employee leaves the organization, administrators need to remove the user account from all the databases. However, in practice, those accounts often live on for a very long time after their legitimate owner has left, making them an attractive target for hackers. Hackers often exploit such dead/orphan accounts to gain unauthorized access to data.

Many organizations manage their users through a central directory or identity service. Oracle Database can work with different methods to authenticate users managed in these directories or services.

- Kerberos is a common mechanism used for user authentication. Windows systems natively use Kerberos for authentication and Linux databases and clients can be configured to work with Kerberos.
- PKI user certificates require some work to manage and maintain all the user certificates, but they can be used to create a mutual Transport Layer Security (TLS) connection to the database as well as authenticate the user to the database.
- RADIUS servers can be integrated with Oracle Database so users can use multi-factor authentication to access the database.

- Users can authenticate with Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) or Azure Active Directory (Azure AD) and send an access token to the database.

A central user management service enforces a consistent password policy, and lockout limits become cumulative across all integrated systems instead of per-system lockouts. Services like RADIUS and cloud identity providers can enforce multi-factor authentication for users.

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 regular users

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

- Impose a strong password profile for all the users, controlling the password strength, the inactivity period, and the number of password retries.
- Implement strong authentication mechanisms with Kerberos, PKI, or with external directories or identity services.
- Manage users centrally using an identity service like Active Directory to reduce the probability of orphan accounts when the employees leave or change their roles.
- Create users with shared schemas if they do not need a private schema for their objects.

### 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:

- Database administrators should use named accounts with strong authentication to provide accountability. Use of shared accounts or default accounts like SYSTEM should be prohibited; instead, they should use proxy authentication. Where practical, use strong authentication mechanisms such as PKI and Kerberos.
- Use a Privileged Account Management (PAM) system when operating system accounts for root and database owner are needed for SYSDBA access during upgrade.
- 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 the DBA is accessing the operating system for the database, 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.

### Protecting application accounts

Application DBA and 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 (PKI, Kerberos, or one of the newer cloud identity mechanisms) to authenticate to the database.
- Use a secrets manager like Oracle Key Vault to store credential information and have the application retrieve the credentials via API calls.
- 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.
- Use proxy authentication instead of sharing passwords across multiple administrators.
- 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 the past, 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 ensure 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.
- Assign them profiles, ensure they use strong passwords, and have a password rotation policy. If you use the `FAILED_LOGIN_ATTEMPTS` parameter, note that a hacker could launch a denial of service attack by trying too many failed login attempts. Also note that the database automatically enforces exponential delays on login failures, so it may not be easy for somebody to brute force a long, complex password.

## 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 give it a try on the technologies discussed in this chapter, please go to:

- [DB Security Basics](#)
- [Proxy authentication](#)

## Summary

As most attacks target the authorized users of the database, properly managing users is the first step in protecting the database. Oracle provides many options for user authentication to meet customer requirements with 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.

Chapter Two 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.

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

# Chapter Five

## **Controlling database access**

---



## Introduction

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 what 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 the user account authorizations, whether locally within the database or with centralized directory or identity services.

## Types of database users

Access to the database is through user accounts, whether administrative users, application accounts, analysts, or other users. Oracle supports different types of database users, each with different access rights to the database, including:

- **Regular database users:** They are typically restricted to their schema containing their tables, views, indexes, and stored procedures. If hackers break into their accounts, they can view/update data within the user schema and access objects in other schemas that the user may be authorized to access.
- **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, with changes made in lower-level environments and promoted through change management processes. In DevOps (or SecDevOps) environments, those traditional lines are often blurred, and it is common to find developers with production database access. These accounts normally have elevated privileges compared to regular users, and therefore pose a higher level of risk.
- **Application accounts:** These are the database accounts used for running your commercial and homegrown applications. They are similar to your regular database user accounts. Still, as applications must run 24/7, their passwords are often stored on multiple middle-tier servers (see the discussion of secrets management in Chapter Four for a better way to manage application account credentials). As these are typically machine accounts, their passwords are as long as the database can support. Any compromise in these database accounts can lead to loss of data for the entire application. *Note: Oracle Database 21c and earlier support passwords up to 30 bytes in length. Beginning with Oracle Database 23c, the maximum length of a password is increased to 1024 bytes.*
- **Application administrators:** These accounts are used to manage, patch, and upgrade your application and, hence, have full access to all the data and the stored procedures used for the application. Application administrators are similar to DBAs but don't need the extensive set of privileges used by a DBA.
- **Data analysts or business intelligence users:** They typically get unfettered read access to the application schema without going through the application-level access controls.
- **Database administrators (DBAs):** They are responsible for a wide variety of tasks for the database, including performance management, diagnostics and tuning, upgrade and patching, database startup and shutdown, and database backup. Their highly privileged database access may give them access to all sensitive data within the database, including personal, health, corporate finance records, etc., even if that access is not required to perform DBA tasks. DBAs hold the keys to your data kingdom and are typically fully trusted within your organization. Unfortunately, this makes them hackers' prime targets.
- **Security administrators:** Many organizations have specialized DBAs who perform the responsibilities of security administrators, including user account management, encryption key management, and audit management.

Hackers may target any or all of them depending on the data they are after. Targeting DBAs gives the broadest access to data within the database.

Once a user is authenticated, the user is granted privileges directly or through a role. In general, the user can connect to their schema and access, modify, or delete all the 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. Such rights can be granted to subjects, including individual users, roles, or programs acting on behalf of users.

## Privileges

A database privilege grants the ability to perform specific operations on data objects or execute certain statements. There are four types of privileges: object, schema, system, and administrative.

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

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.

Schema privileges (new in Oracle Database 23c) 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 and easier to maintain than individual object privileges.

```
GRANT SELECT ANY TABLE ON SCHEMA APP TO SCOTT;
```

Administrative privileges are 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`.

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. Any such use should also be fully audited.

## Roles

Roles are nothing but privileges grouped together, making it easier to grant or revoke them from multiple users. Direct 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.

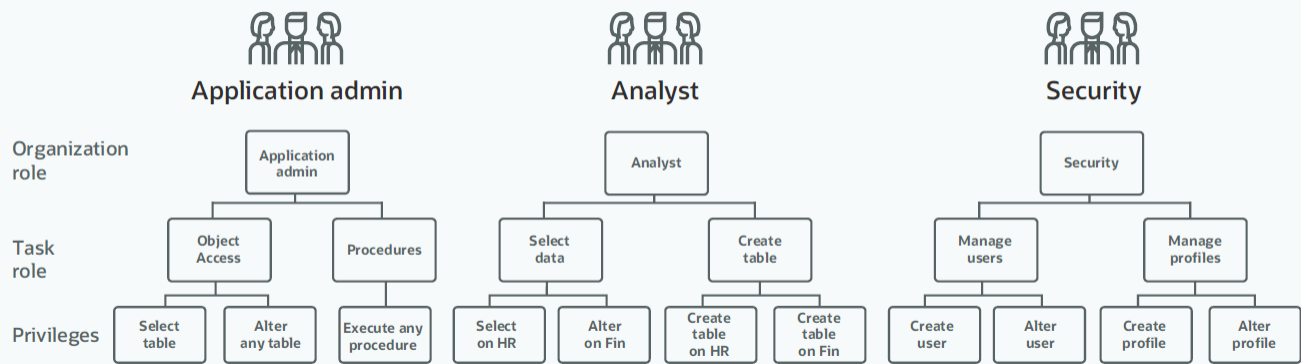


Figure 5-1: Sample role and privilege hierarchy

Multiple users can be granted the same role to simplify privilege management for users. 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 privileges needed and granting 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.

## 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 5.1 show the roles and privileges granted to users or roles.

Dictionary views	Contents
<b>DBA_TAB_PRIVS</b>	Object privilege grants to roles or users
<b>DBA_SCHEMA_PRIVS</b>	Schema privilege grants to roles or users
<b>DBA_SYS_PRIVS</b>	System privilege grants to roles or users
<b>DBA_ROLE_PRIVS</b>	Roles granted to all users and roles
<b>DBA_ROLES</b>	All defined roles
<b>ROLE_ROLE_PRIVS</b>	Roles granted to other roles
<b>ROLE_SYS_PRIVS</b>	System privileges granted to roles
<b>ROLE_TAB_PRIVS</b>	Object privileges granted to roles

Table 5-1: Data dictionary views for database roles and privileges

If you were to analyze the information in these tables, you would see that the out-of-the-box Oracle 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 their own sets of privileges.

The Oracle DBA role shouldn't be modified and, in most cases, should not be used. Instead of using the default 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.

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  
**Status:** OPEN

**Potential risk:** CRITICAL ⓘ

**Last login:** Fri, 31 Mar 2023 13:30:02 UTC  
**Created:** Thu, 07 Nov 2019 22:46:46 UTC  
**Last password change:** Thu, 07 Nov 2019 22:46:46 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**

Figure 5-2: Analyzing a user's privileges

## Managing users centrally

The previous chapter 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 de-provisioned when the user changes roles or teams or leaves the organization. Hackers often exploit such 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 (Oracle Internet Directory or Oracle Unified Directory), which 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. Such enterprise users can be assigned enterprise roles or groups that determine access privileges across multiple databases. An enterprise role maps to one or more roles on individual databases.

Centrally Managed Users (CMU) directly integrates the database with Microsoft Active Directory to manage user authentication and authorization. Active Directory users can have their own schema or share a schema through Active Directory groups. Active Directory groups can also map directly to database roles.

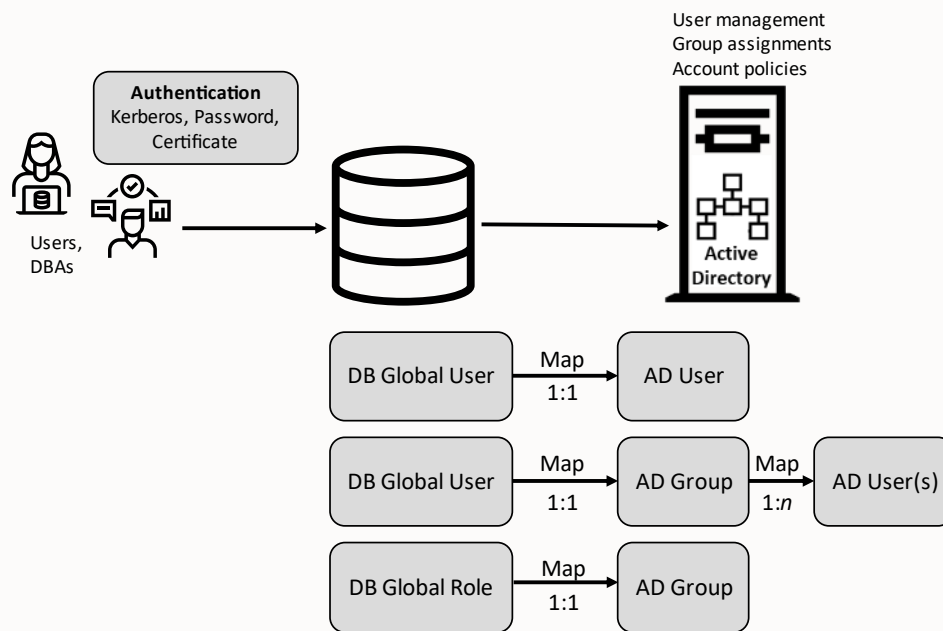


Figure 5-3: Centrally Managed Users (CMU)

EUS and CMU allow users and administrators 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 database 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 map 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 a different IAM group. IAM authentication and authorization is available for all OCI databases, including Autonomous Database, Oracle Base Database Service, and others.

Azure Active Directory users can access the database using Azure AD tokens and leverage Azure AD app roles for database authorization. App roles are included with the Azure AD access token and are used by the database to map Azure AD users to a database schema and, optionally, database roles. Azure AD authentication and authorization are available for OCI and on-premises databases.

## Protecting database accounts

### Protecting regular users

You should not expect regular database users to remember and follow all the security guidelines concerning roles/privileges granted to them. These security guidelines must be enforced at the database and directory/identity service levels to ensure users follow them. Policies such as the following help minimize your users from being hacked.

- Regularly examine if users have been granted any access to objects outside their schema.
- Track grants of system privileges like `SELECT ANY TABLE`. Encourage administrators to use object or schema privileges to reduce the user's access scope.
- Implement Privilege Analysis to ensure users only have the privileges they need for the job. We discussed this in detail in Chapter Two.

### 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 `SYSDBA`.
- Block access to user schemas or SQL commands using Database Vault—more on this in Chapter Six.
- Fully audit all DBA activities for accountability and tracking.

### Protecting application accounts

Application DBA 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 convenient. Chapter Six discusses a new feature (schema privileges) in Oracle Database 23c that makes this easier.
- Fully audit all their activities for accountability and tracking.

### Protecting security administrator accounts

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

- Limit roles granted to the security administrators such that they don't have wide access to the sensitive data in the database in addition to managing the security controls.
- All their activities should be fully audited for accountability and tracking.

## 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 give it a try on the technologies discussed in this chapter, please go to:

- [DB Security Basics](#)
- [DB Security Advanced](#)

## Summary

Since most attacks target the authorized users of the database, properly managing users and their authorization is an important step in protecting the database. Oracle provides options to manage user authorizations to meet customer 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.

Oracle Data Safe includes a User Assessment feature to analyze cloud and on-premises database user risk. User Assessment identifies users with DBA privileges to quickly identify user accounts that can pose a risk if account credentials are compromised.

Upcoming chapters will show 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. Finding the right set of privileges and roles can be done easily using Privilege Analysis.

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

# Chapter six

## **Enforcing separation of duties**

---



## Introduction

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.

This chapter discusses how you can minimize the losses from compromised accounts 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.

## Controlling powerful administrators

Some of the database users can become very powerful due to some of the system privileges that they may have. An example of a powerful system privilege is the SELECT ANY TABLE privilege granted to the default DBA role. With this privilege, a DBA can view any data on 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 get access to the sensitive data. They can now easily export and exfiltrate the data.

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

**Enforce separation of duties (SoD):** Most DBA operations use narrow and very specific privileges. 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. These administrative privileges are each associated with an operating system group (e.g., OSBACKUP or OSKM). Depending on one's organizational structure, one may assign all these administrative privileges to the same group (e.g., DBA) or assign each privilege to its own unique OS group.

Here, the administrative tasks are divided among multiple users instead of a single powerful individual with full access to all database administration, operations, and security controls. Dividing overall duties into separate administrative, operational, and security responsibilities makes it less likely for privileged users to abuse their privileges, as any single administrator will only have a portion of the privileges. Further, there is no reason for all these users to have access to the application schemas and data.

With these specific privileges, multiple administrators can perform all the normal operations to manage a database without the risk of having the all-powerful SYSDBA privilege. Using the different administrative privileges still does not prevent the administrators from accessing user-specific data. Read on to see how to enforce SoD further.

**Control the database owner account:** The SYSDBA administrative privilege provides full access to the SYS (database owner) account. This privilege and account should be limited to database upgrades and patching. Change management and privileged access management (PAM) systems should be used to control access to this privilege.

**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 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 the prime targets.

**Enforce least privilege:** Only grant your database users the minimum set of privileges needed to accomplish their intended tasks or functions and no more. 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 roles with the least privileges necessary for a particular function instead of using powerful roles like the built-in DBA role. Granting several 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 Four for more information.

**Safeguard the audit trail:** Audit records provide an irrefutable record of actions on a database, directory, or operating system. 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. 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 Eleven and Twelve.

## Control privileged users with Oracle Database Vault

Normally, only the schema owner and users with direct object grants are allowed to access their sensitive data. However, privileged users also have access to sensitive data through the `SELECT ANY TABLE` system privilege. Complex applications with multiple schemas also frequently use system privileges instead of object privileges for convenience, making it possible for SQL injection attacks to access data across schemas in the database.

You need a set of operational controls within the database to limit privileged users from accessing sensitive schemas, tables, and views.

Oracle Database Vault, available since Oracle 9i release 2, restricts privileged users, including database administrators, with system privileges like `SELECT ANY TABLE` from accessing sensitive data. Database Vault introduces the concept of a security realm, which is a collection of schemas or specific sensitive objects that only users authorized by Database Vault can access. With Database Vault, database administrators can manage the database but cannot access sensitive schemas/objects protected by realms.

The following illustration shows an example of a realm protecting sensitive data within the human resources (HR) schema. Oracle Database Vault prevents the powerful DBA from accessing data inside the HR realm yet still allows them to do their DBA tasks. The application administrator for the HR realm can manage the HR data but cannot access objects from the FIN Realm, even though the HR application administrator has broad system privileges. Thus, Database Vault realms control the use of system privileges, isolating the DBA from application data.

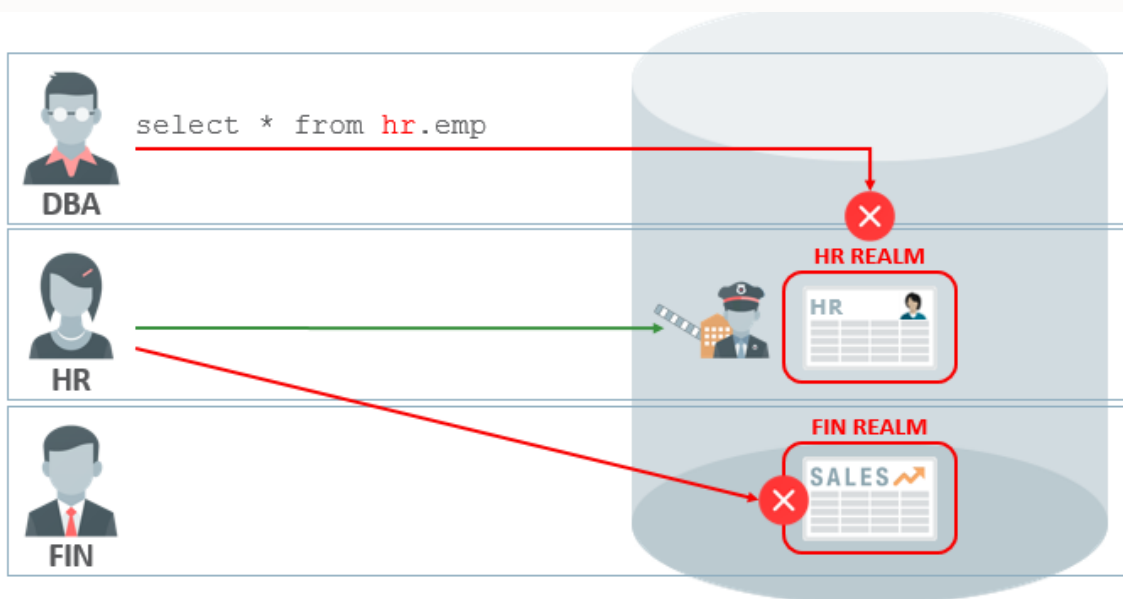


Figure 6-1: Oracle Database Vault realms protect sensitive data from privileged users

By default, realms are overridden by schema ownership and direct object grants. Still, there are situations where you want the realm to exercise control over ALL users, regardless of direct privilege grants. Having the realm override direct grants 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. Mandatory realms override both schema ownership and direct object grants. With a mandatory realm, no one is allowed access 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 with Oracle Database Vault

Separation of duties (SoD) within the database prevents an attacker from disabling security controls and stealing sensitive data using a single account. The attacker must compromise a minimum of TWO accounts to compromise the database fully.

Configuring 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 SoD for a security role to create and manage users in the database along with their password profiles.
- **Database Vault administrator and owner** are the only roles that allow a security DBA to create, modify, and manage Oracle Database Vault security controls, including the ability to manage realms and add realm participants. Database Vault Administrator controls the various PL/SQL packages associated with Database

Vault. Database Vault Owner includes the Database Vault Administrator role, but also allows you to manage and monitor the Database Vault configuration,

Normally, the DBA role cannot manage users or Oracle Database Vault unless that ability is explicitly granted to the DBA role. 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.

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. Using the legacy “traditional audit” facility, older database versions would also allow an attacker who misappropriated the DBA role to modify the audit records to hide activity (this is impossible with the more modern unified audit facility. See Chapter Eleven for details). Audit records can also contain sensitive data, such as queries identifying important objects or columns.

THE DBA TEAM CAN STILL DO common DBA tasks such as performance tuning, memory management, backup, and others. Certain DBA tasks may expose sensitive data to the DBA. These tasks include export and job scheduling, which will require additional authorization by the Database Vault security team.

While a single administrator may want to perform multiple functions for convenience, dividing these duties among multiple users allows for separating duties with Oracle Database Vault enforcing the SoD. These powerful roles shouldn't be granted to a single user. Otherwise, a single compromised account may allow the attacker to easily steal sensitive data.

## **Control SQL database commands with Oracle Database Vault**

Privileged users on production systems should not be dropping tables, changing system parameters, or modifying database objects outside of maintenance windows. Modifying data outside of application control may also be restricted. DBAs frequently have terminal windows open in various dev, test, and production systems and can potentially run damaging SQL on production systems accidentally.

Database Vault provides fine-grained SQL command rules to prevent accidental or malicious SQL changes to the data and the database. These fine-grained rules can prohibit certain commands from running or even dictate the time of day when commands can be run, along with the permissible client IP address and other context data.

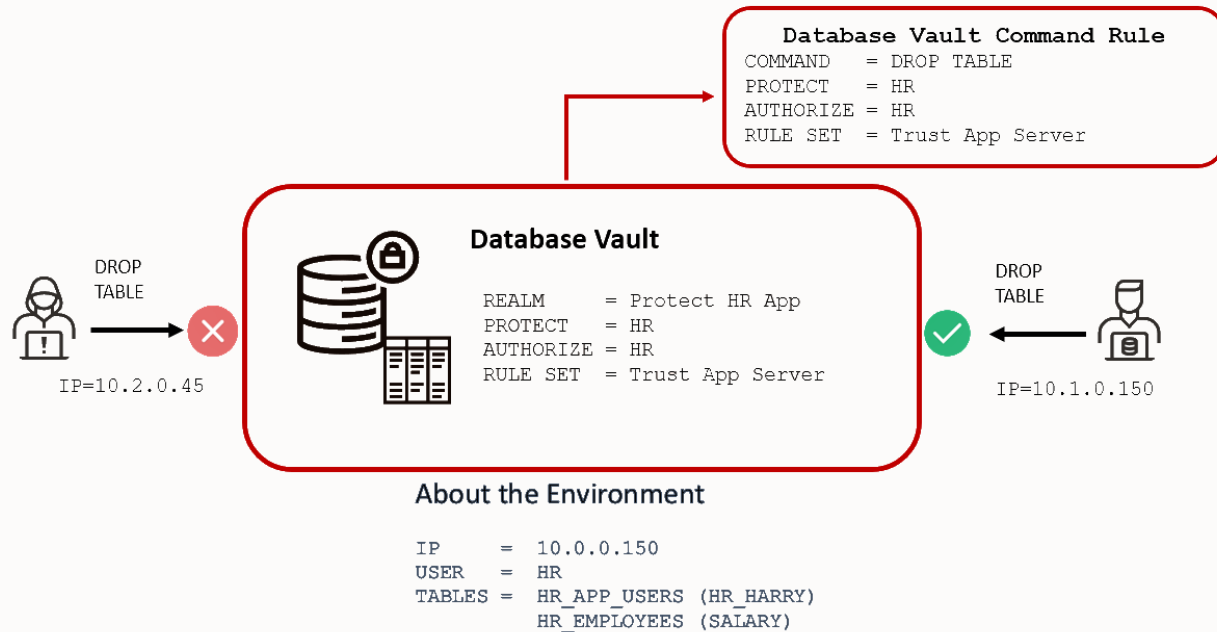


Figure 6-2: Database Vault Command Rule

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.

## 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. One could argue that all user data is sensitive and that a realm should be created on every user schema. Starting with Oracle Database 19c, Database Vault Operations Control can block common users (infrastructure DBAs, for example) by default 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 23c, which implements multitenant by default, Operations Control becomes more important than ever.

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 as it enforces the separation of duties outside of itself. Some organizations take multiple steps before they implement full SoD. For

example, since the new Database Vault roles enforce the SoD, the new roles can be granted to the same DBA(s) initially, and then steps can be taken towards separating duties. This allows some controls in Oracle Database Vault to be implemented quickly and for SoD changes to spread over a longer time frame.

Database administrative staff may hesitate to turn on preventive controls in case applications stop functioning or administrative scripts stop working. To minimize this impact, Oracle Database Vault provides a **simulation mode** to allow you to verify that realms and SQL command rules do not impact application run-time and operations. Instead of blocking, Simulation mode logs violations, allowing a full end-to-end regression test without interfering with existing application operations. Once complete, Oracle Database Vault controls can be updated based on the simulation mode log data.

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 Database Vault activity data.

## Oracle cloud and applications

Several Oracle applications on the Oracle cloud and on-premises have been certified to work with Oracle Database Vault, including Oracle Fusion Human Capital Management, Oracle Enterprise Resource Planning Cloud Services, Oracle E-Business Suite, and Oracle PeopleSoft. 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 give it a try on the technologies discussed in this chapter, please go to:

- [DB Security- Database Vault](#)

## Summary

Stealing sensitive data using compromised privileged user accounts is one of the most common attack vectors. 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 seven

## **Minimize risk from SQL Injection**

---

## Introduction

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

This chapter discusses the two main approaches to mitigate the risk of SQL Injection: 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.

## SQL injection overview

Most three-tier applications connect to the database as the powerful application account with access to the entire application schema and procedures. 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, which could do anything that the application could have done. Thus, they circumvent the authentication and authorization of a web application and retrieve the contents of the database - including unauthorized access to sensitive data: customer information, personal data, trade secrets, intellectual property, and more. Attackers may try to use SQL injection to add, modify, and delete records in the database.

There are many SQL injection vulnerabilities, attacks, and techniques. But all of them follow a similar attack pattern:

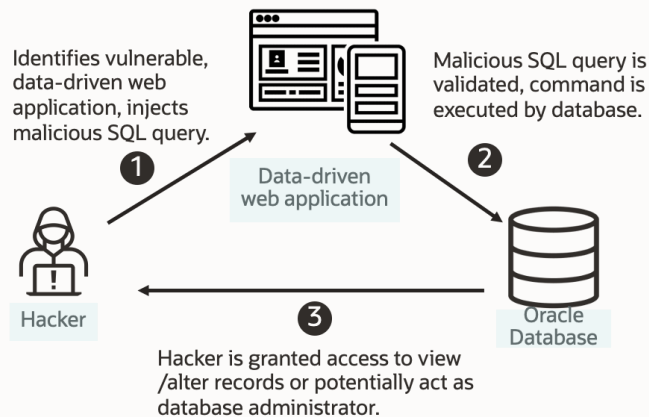


Figure 7-1: SQL Injection attack pattern

SQL injection is often the entry point for a more extensive compromise. In addition to exfiltrating data from the compromised database, attackers may move laterally to other hosts on the internal network.

SQL injection is one of the most popular attack patterns on the Open Web Application Security Project (OWASP) list of the [top ten threats](#) to application security and has been since 2017. Although the answer to the SQL injection problem is well known (developers using input validation to discard malicious inputs), SQL injection continues to plague data-driven applications. While user and developer education has helped, success remains limited.

## Approaches to address 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 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 slightly 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 block SQL injection attacks is to filter *database* traffic before the database processes it. This is where a database firewall comes into the picture.

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

Unlike other application-based firewalls, neither of the above approaches relies on regular expressions or looks for signature-based matches. Instead, the database firewall learns typical application SQL traffic and can reject or alert if the SQL statement does not match its trained model.

SQL Firewall in Oracle Database 23c moves that protection closer from the network to the database kernel, avoiding the complexity of routing database traffic through a separate Database Firewall. Web applications can leverage this powerful new database security feature to reduce the risk of SQL injection attacks. Let's look at both these solutions to understand how they mitigate the risk of SQL injection.

Both network and database-resident firewalls 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, an Oracle Audit Vault and Database Firewall component, 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.

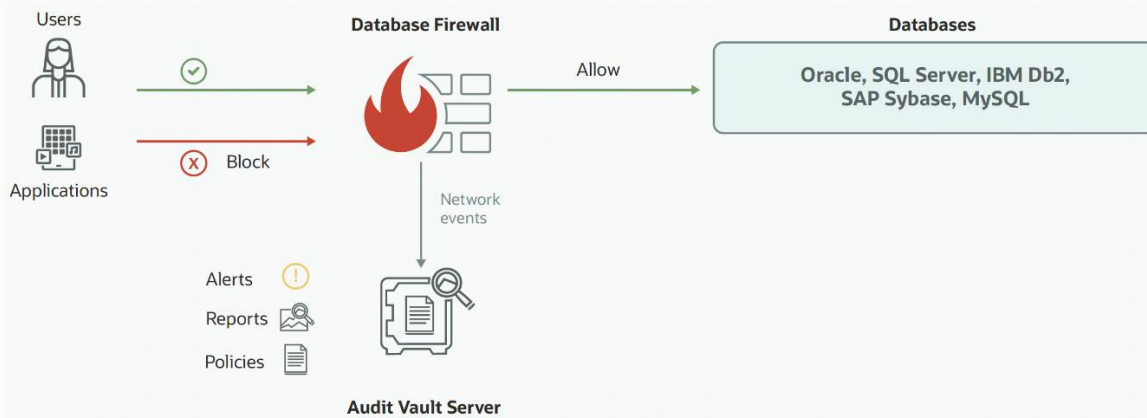


Figure 7-2: Oracle Audit Vault and Database Firewall

Database Firewalls can be deployed in various modes, as shown below.

MODE	Details	Monitoring or blocking
<b>Proxy</b>	All traffic to the database server routes through the Database Firewall, including return traffic.	<ul style="list-style-type: none"> <li>Monitoring</li> <li>Blocking</li> </ul>
<b>Host Monitor</b>	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.	<ul style="list-style-type: none"> <li>Monitoring</li> </ul>
<b>Out-of-band</b>	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.	<ul style="list-style-type: none"> <li>Monitoring</li> </ul>

Table 7-1: 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 in near-real time or if you also wish to enforce trusted access patterns to the database by blocking every unauthorized access.

If you only want to monitor for attacks (without blocking), then both host monitor or out-of-band modes are acceptable. Both modes work by “sniffing” network traffic and looking for SQL statements that violate the Database Firewall’s policy. Host monitor sniffs network activity from the database server, capturing incoming SQL traffic and relaying it to the Database Firewall for analysis. Out-of-band mode sniffs SQL traffic directly from the network and forwards it to the Database Firewall.

If you intend to block unauthorized SQL traffic, the firewall should be in line with the incoming SQL traffic using proxy deployment. Proxy enforcement can initially be set to monitor traffic (without blocking) and just alerting on unauthorized activity. Once you are confident with the SQL baseline, you can switch to the blocking mode to enforce the firewall policy and prevent SQL Injection and other unauthorized SQL.

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, alerted, allowed to go through, 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!



Figure 7-3: Monitoring SQL traffic with Database Firewall in proxy mode

For more information on Database Firewall policies, refer to the [Audit Vault and Database Firewall Auditor’s Guide](#).

### Database-resident SQL Firewall

Oracle SQL Firewall is a new feature built into Oracle Database 23c. 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 does not require additional product installation or network configuration. SQL Firewall is embedded within the Oracle Database kernel, operating closer to where data resides, thereby eliminating the possibility of bypassing the control. SQL Firewall inspects all incoming SQL statements and ensures the database executes only explicitly authorized SQL arriving via a trusted path.

Because Oracle Database 23c embeds SQL Firewall within its kernel, it can examine all SQL statements - whether local or over the network, encrypted or clear text. Unlike signature-based protection mechanisms, SQL Firewall cannot be bypassed by encoding the SQL statement, referencing synonyms, or using dynamically generated object names.

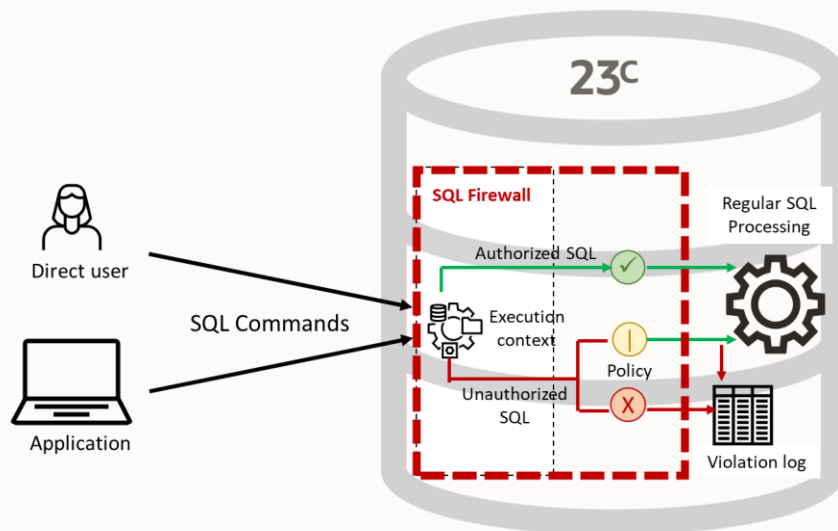


Figure 7-4: Oracle SQL Firewall built-in to the Oracle Database kernel

While other database security features embedded within Oracle Database provide different security controls to monitor/prevent web application attacks, the Oracle SQL Firewall feature is the only one that inspects ALL incoming

SQL statements and allows only authorized SQL statements. It logs and can block unauthorized SQL statements from executing in the database, offering best-in-class protection against SQL injection database attacks.

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 logs that as a violation. Unlike SQL Command rules, which require extensive training of the firewall to ensure that all possible SQL statements have been captured, these session context rules are easy to capture and, if desired, can even be entered manually.

Unlike Database Firewall, which is heterogeneous and works on many different types of databases, SQL Firewall is part of Oracle Database and only works for Oracle Database 23c and higher. SQL firewall is included with Oracle Database Vault option.

### Administering Oracle SQL Firewall with Data Safe

Oracle Data Safe provides fleet-wide administration capability of Oracle SQL Firewall. It gives you a comprehensive view of SQL Firewall violations across your Oracle databases. SQL Firewall administrators can use Data Safe to collect SQL activities with their connection context (IP address, program, OS user, DB user) and monitor the collection progress. Once you have captured the SQL statements, you can create firewall policies with allow-list rules (allowed contexts and/or allowed SQL statements), enable the firewall policies, collect logs, and monitor SQL Firewall violations.

Data Safe lets you monitor the progress of SQL collection using collection insights to ensure that SQL Firewall has fully learned the SQL application traffic. It is best to run the collection until the number of unique new SQL statements drops to and remains at zero. Session context attributes are also available to ensure that SQL Firewall has learned normal traffic patterns. Once you complete the review, you stop the collection and generate the Firewall policy.

Data Safe makes reviewing the contents of allow-lists (allowed contexts and/or allowed SQL statements) within an SQL Firewall policy easy. You can modify allowed contexts and generate a report of allowed SQL statements with corresponding database objects accessed if required.

Data Safe helps you enable the SQL Firewall policy on a target with appropriate values for enforcement, observation, blocking, and audit.



Figure 7-5: Oracle Data Safe: Collection insights to monitor unique SQL statements. The Y-axis represents SQL statements captured on a given date. The X-axis represents date.

Once the SQL Firewall policy is enabled, Data Safe automatically collects violations from the target’s violation log and stores them in Data Safe’s repository for up to six months. As shown below, SQL Firewall violations stored in Data Safe are available for online analysis and reporting across your database fleet.

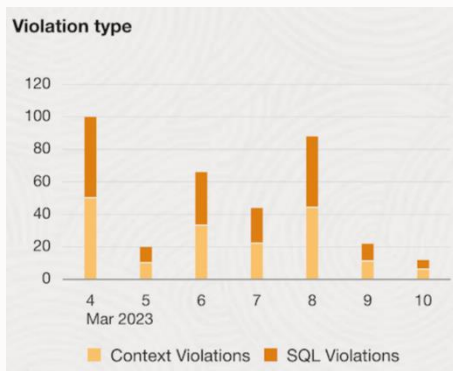


Figure 7-6: Data Safe: SQL Firewall violations (Fleet view)

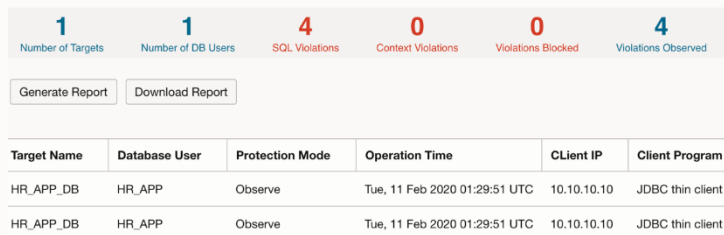


Figure 7-7: Data Safe: Detailed SQL Firewall violation reports

## 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 23c, AVDF is the only choice. For Oracle Database 23c 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 lacks visibility into SQL traffic over local BEQ connections and is unaware of internal jobs/ stored procedure executions. The SQL Firewall runs inside the database and sees ALL traffic, regardless of the origination point.
- 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.

## 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 give it a try on the technologies discussed in this chapter, please go to:

- [Oracle Audit Vault and Database Firewall](#)
- [Oracle SQL Firewall](#)

## 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 AVDF or Oracle SQL Firewall in Oracle Database. Neither relies on heuristic pattern matches typically used by most network firewall solutions. Instead, they rely on actual SQL signatures and construct an allowed list of SQL statements. They combine allowed SQL with session context to learn the normal application SQL traffic and detect and alert/block the violations in real time.

# Chapter eight

## **Data-driven application authorization**

---

## Introduction

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 require 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.

## 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 not 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.

The following graphic shows a table with both sensitive and non-sensitive HR data. Nancy should be able to see her own Social Security number (SSN) and her direct reports' salaries. She should see phone numbers for all employees, but all other sensitive data should be hidden.

Name	Manager	SSN	Salary	Phone number
Adam Fripp	Steven Stiles			650-123-4234
Neena Kochhar	Steven Stiles			650-124-8234
<b>Nancy Greenberg</b>	<b>Neena Kochhar</b>	<b>000-51-4569</b>	<b>120300</b>	<b>515-123-4567</b>
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 8-1 Example: Fine-grained access control applied to a table

Applications may manage and enforce their user authorizations in the mid-tier, but because application-enforced authorization does not apply to other applications accessing the same database tables or reporting tools with direct access to the database, security policies are enforced inconsistently, and users may be able to view data they have no business need to see. The solution is fine-grained access controls implemented centrally within the database.



This chapter discusses how applications can handle the problem of fine-grained authorization without coding these rules within the application. Centrally controlling fine-grained data access for application users within the database saves 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.

## 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 8.1. 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.
- Database audit frequently only sees that an application account has made an SQL query—not the actual end-user or which columns the user viewed.

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—at that time, it was the only database with this innovative capability. Oracle introduced Oracle Label Security (OLS) to automatically filter rows based on data and user labels. The most recent technology in this line-up is Real Application Security (RAS), introduced with Oracle Database 12c. RAS provides direct support for application privileges - making it much easier to develop secure applications. We will now look at each of these technologies in more detail.

## Controlling data access using Virtual Private Database

Virtual Private Database (VPD) enforces row and column-level security policies based on the end user context set by the application. Using VPD, an application-defined PL/SQL function is executed to generate the appropriate `WHERE` clause each time the table is accessed so that the SQL statement only returns the authorized rows and columns.

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 this example, 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. 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. Centralized enforcement of security policies significantly simplifies application development because the developer does not have to think about changing the query depending on the user.

```
SQL> select last name, email, department id, salary from
hr.employees;
```

LAST NAME	EMAIL	DEPARTMENT ID	SALARY
Hunold	AHUNOLD@EXAMPLE.COM	80	
Ernst	BERNST@EXAMPLE.COM	80	
Austin	DAUSTIN@EXAMPLE.COM	80	
Pataballa	VPATABAL@EXAMPLE.COM	80	
Lorentz	DLORENTZ@EXAMPLE.COM	80	

Figure 8-2: VPD fine-grained access control policy automatically keeps salary data hidden

## 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 expressive enough to accommodate virtually any data classification scheme in commercial or government organizations. Every label includes a level, ordered from lowest to highest, to indicate the overall sensitivity of the data. Within a level, 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.

The figure below shows an example of a worldwide retail store classification scheme using all three label components. Two levels are defined: HIGHLY SENSITIVE and SENSITIVE. 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.

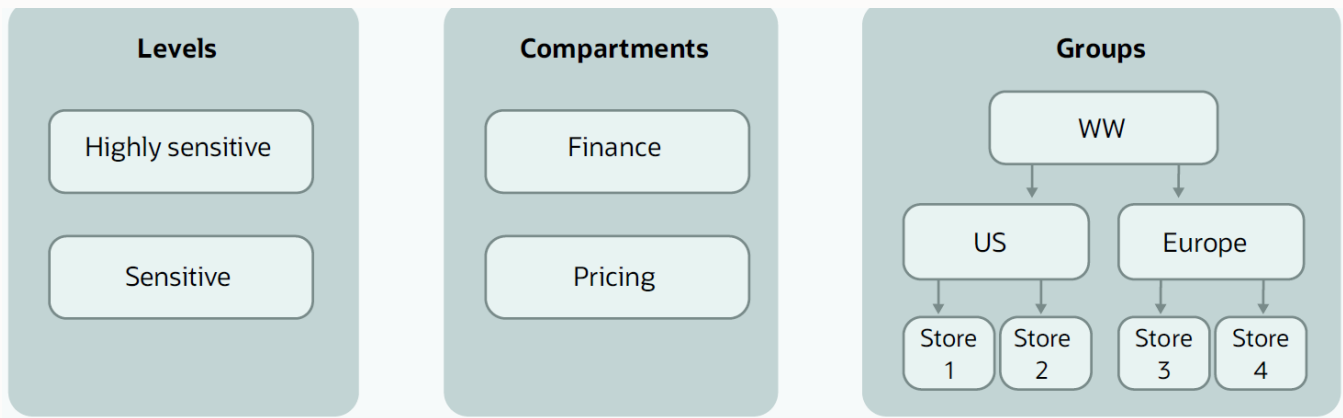


Figure 8-3: Label Security components: levels, compartments, groups

Labels are represented by the three components separated by a colon (:). A data label considered 'SENSITIVE,' with 'FINANCE' data for 'STORE1' would be represented as SENSITIVE:FINANCE:STORE1. Pricing data for store1 would be labeled SENSITIVE:PRICING:STORE1. For the store1 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.

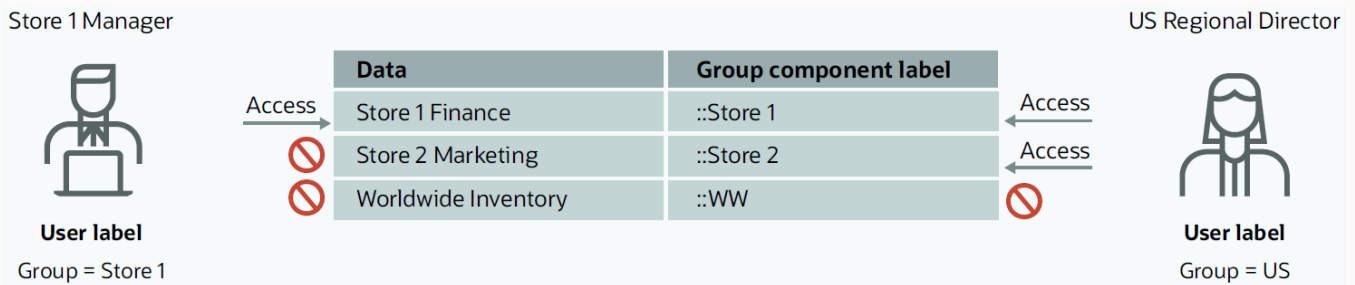


Figure 8-4: Using the group component in Label Security

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 US regional Director would have 'US' as their group label, allowing them to see all the elements that belong to US (STORE1, STORE2) since the US label is defined hierarchically over STORE1 and STORE2 in OLS.

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 store1 marketing manager needs to see pricing data to build sales material for the weekly store sales events, and the store1 manager needs to see both pricing and financial data for store1. But neither manager can see finance data for the US group.

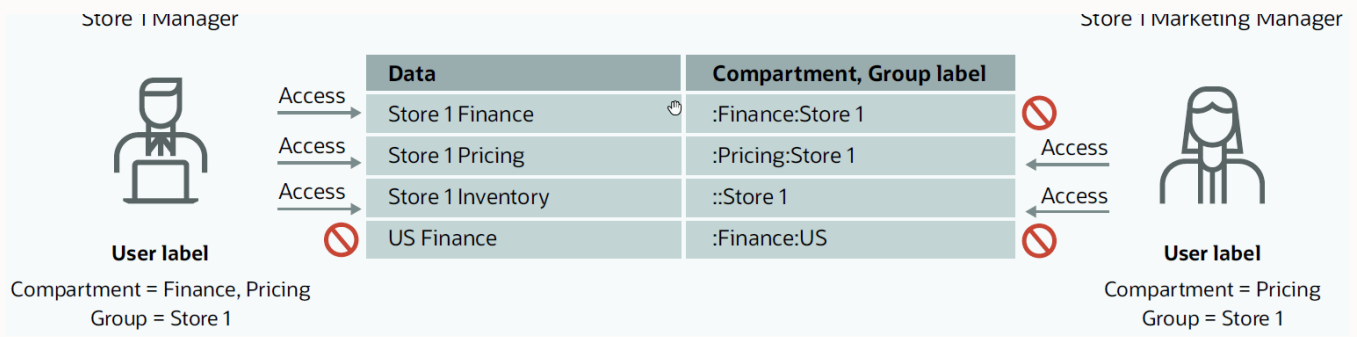


Figure 8-5: Using the compartment and group components

Since this is a public company, all non-released quarterly financial data and reports are highly sensitive. All data were considered ‘sensitive’ except for the quarterly financial data, which was ‘HIGHLY SENSITIVE’ with the ‘FINANCE’ compartment and ‘WW’ group. Only users with the level ‘HIGHLY SENSITIVE’ with the ‘FINANCE’ compartment and group ‘WW’ 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’ 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.
- 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 ‘STORE1,’ ‘US,’ or ‘WW’ group label could view it.



Figure 8-6: Using the level, compartment, and group components

While OLS has all the functionality to meet government, military, and intelligence agency requirements for labeling, 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.

The significant advantage of using Oracle Label Security is a complete authorization infrastructure with mediation and simplified administration. OLS leverages VPD to get the same type of row-level access control but implements 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 the PL/SQL policy function 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 the data in each table. OLS 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. Instead, a single database account representing the entire application issues queries and updates on behalf of individual end users.

The application must enforce per-user access control policies because the end user's identity is unknown to the database. Besides requiring additional software development in the application, this approach can lead to inconsistent enforcement, especially when there are system elements that can bypass the application and connect directly to the database.

Introduced in Oracle Database 12c, Real Application Security (RAS) provides the next generation of application access control frameworks within the database. It enables three-tier and two-tier applications to declaratively define, provision, and enforce access control requirements in the database layer.

RAS introduces 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. 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 each 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 9.7 below shows three different data realms. The 'public' data realm includes all employee records. The 'self' data realm is dynamic 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	SSN	Salary	Phone number	
Self Data-Realm →	Adam Fripp	Steven Stiles			650-123-4234	Public Data-Realm
	Neena Kochhar	Steven Stiles			650-124-8234	
	<b>Nancy Greenberg</b>	<b>Neena Kochhar</b>	<b>000-51-4569</b>	<b>120300</b>	<b>515-123-4567</b>	
Manager Data-Realm	Luis Popp	Nancy Greenberg		69000	515-123-4234	
	John Chen	Nancy Greenberg		82000	515-123-8181	
	Daniel Faviet	Nancy Greenberg		9000	515-123-7777	

↑ Select SSN Privilege      ↑ Select Salary Privilege

Figure 8-7: Controlling data access with RAS

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 Sa1ary data are protected and not included in the ACL for the 'public' data realm, those columns do 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 Sa1ary (Authorize Salary) data so an employee (role) can view their own sensitive data.



Figure 8-8: 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 Sa1ary data to be shown. The 'se1f' 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 user 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 users' sessions to the database, Oracle 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, Oracle 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?

VPD 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 predicate clause becomes complex and, therefore brittle. If the access control policy is simple and will remain that way, VPD is a great choice.

RAS is the most powerful data-driven access control. It is enterprise-ready with many enterprise patterns built in. However, the application code must be modified to set the RAS context and then attach and detach the RAS sessions.

OLS 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. Also, 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 give it a try on the technologies discussed in this chapter, please go to:

- [DB Security Basics](#)
- [DB Security Advanced](#)
- [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 can centralize authorization policies 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 nine

## **Masking sensitive data**

---



## Introduction

Most organizations create copies of their production databases for use in testing 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 dynamic data masking prevents the proliferation of sensitive data to users through various fine-grained access control mechanisms. These tools are integral to a comprehensive data privacy strategy, helping you meet compliance requirements such as PCI-DSS and EU-GDPR.

## Why mask Data?

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 need to see it is a challenge many organizations face. The problem is that there are many instances where you want to leverage your application data. For example, developing and testing new applications is much more efficient when the process can take advantage of actual application data. Likewise, marketing and customer service teams seek to mine application data to better understand customer needs and 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. In each of these cases, there are clear business benefits of using application data. However, how can you enable these use cases without compromising sensitive information?

Data masking modifies sensitive data so that it is of no or little value while still usable. 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, Social Security numbers, sales figures, and other personal or proprietary information. For example, testing a point-of-sale application may not require the actual credit card numbers contained in the application but can use fictitious but realistic credit card numbers instead.

Data masking may also be called redaction, scrambling, munging, or anonymization. Masking differs from encryption because there is usually no intention or ability to unmask the data. In most cases, the masked data must look real enough to remain and appear consistent across different tables, following referential integrity constraints.

Data masking is a security control that hides sensitive data by replacing the original data with altered data. Data masking may be “static” – where the stored data is actually 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 removed from non-production environments such as test, development, and training. Dynamic data masking prevents the proliferation of sensitive data to other systems by masking data in flight while keeping the original data intact.

Oracle supports both static and dynamic masking using a variety of tools, some integral to the database, others via external services:

- Static masking with Oracle Data Safe or Oracle Data Masking and Subsetting
- Dynamic masking with Oracle Data Redaction, Virtual Private Database, and Real Application Security

These tools are integral to a comprehensive data privacy strategy, helping you meet compliance requirements such as PCI-DSS and GDPR.

## Use cases for data masking

Common use cases include:

- Fine-grained control of access to sensitive data. Use masking to remove access to particularly sensitive columns of data.
- Reduce regulatory compliance risk by minimizing the amount of sensitive data you store: By masking sensitive data, you can help ensure that you comply with data 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.
- Saves 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
- Third-party outsourcing: When you outsource certain tasks to third-party vendors or partners, data masking ensures 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 more prevalent in healthcare and financial service fraud analytics.
- Training and education: Data masking in educational settings or training programs allows students or trainees to work with real-world data without exposing private information.

There are two main types of data masking: static and dynamic.

## 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 in a way that retains 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 9.1 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. Data masking helps minimize the proliferation of sensitive data and reduce the security and compliance perimeter of the organization while maintaining data usability.



Figure 9-1: Overview of 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.

## Dynamic data masking

Dynamic data masking 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 of a customer service application where a call center representative may only need access to the last four digits of a customer's Social Security Number to verify their identity. Although the system might require the entire Social Security Number for other purposes, such as credit reporting, displaying the whole number to the call representative through the application's user interface is unnecessary and may violate privacy regulations. Here, the database can use dynamic data masking to redact all but the last four digits of the Social Security number for specific users.

The database performs dynamic data masking 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. 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. Subsetting also minimizes storage costs because the subset database is smaller.

Together, data masking, redaction, and subsetting limit the avenues for hackers to steal sensitive data 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.

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

Masking format	Description
<b>Fixed value</b>	Replaces original values with a fixed value. e.g., real email address replaced with fake@no.co
<b>Random values</b>	Replaces original values with random values, usually within some specified range
<b>Random list</b>	Replaces values in a column using a list or array of values provided when the format is associated with the column
<b>Substitution</b>	Replaces values in a column with values selected from another table
<b>Encryption</b>	Applies a cryptographic algorithm to convert the original value into something that is unreadable. This type of masking is often used when there is a need to later reverse the data (e.g., a data set shared with a third party for analysis)
<b>Format preserving randomization</b>	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 can be applied to data such as national identifiers, zip codes, and license plate numbers.
<b>Shuffle</b>	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.
<b>SQL expression</b>	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. This masking technique may also be used to mask data contained in large objects (LOBs), including BLOBs, CLOBs, and NCLOBs.

Table 9-1: Simple masking formats

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 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 9.2 shows a few of the more than 50 predefined formats available in Oracle Data Safe. Predefined formats take some of the work out of designing your masking process.

## Masking formats

Create masking format

Name	Description	Oracle predefined
<a href="#">Age</a>	Replaces values with random numbers between 0 and 110	Yes
<a href="#">Bank Account Number</a>	Replaces values with random 9 to 16 digit numbers	Yes
<a href="#">Bank Routing Number</a>	Replaces values with random 9-digit numbers	Yes
<a href="#">Blood Type</a>	Replaces with values picked randomly from a list. Possible values are A+, A-, B+, B-, AB+, AB-, O+, and O-	Yes
<a href="#">Canada Social Insurance Number</a>	Replaces values with random Canada Social Insurance Numbers	Yes
<a href="#">Canada Social Insurance Number (Hyphenated)</a>	Replaces values with random Canada Social Insurance Numbers. Social Insurance Numbers are in 999-999-999 format, where 9 signifies a digit	Yes
<a href="#">Credit Card Number</a>	Replaces values with random credit card numbers. Card types covered are American Express, Diners Club, Discover, enRoute, JCB, Mastercard, and Visa	Yes
<a href="#">Credit Card Number (Hyphenated)</a>	Replaces values with random hyphenated credit card numbers. Card types covered are American Express, Diners Club, Discover, enRoute, JCB, Mastercard, and Visa	Yes
<a href="#">Credit Card Number (Type and Format Preserving)</a>	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 9-2: Predefined masking formats in Oracle Data Safe

Using the masking formats library, you can easily apply data masking and anonymization 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 may 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 – the predefined formats won't give you test account numbers that look realistic. 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 9.3 describes some masking techniques.

Masking technique	Description
<b>Conditional masking</b>	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.
<b>Compound masking</b>	Masks related data contained in multiple columns as a group. For example, mask city, state, and zip code together to correlate the values.
<b>Deterministic masking</b>	Masks data to the same consistent value across multiple databases or applications. This masks values like customer numbers to the same new value across different databases or masking runs. Deterministic masking maintains referential integrity in a multi-database test environment.
<b>Reversible masking</b>	Encrypts and decrypts sensitive data using a cryptographic key. This masking technique is helpful in scenarios where you want to be able 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 9-2: Masking techniques

Figure 9.3 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, shuffling disassociates health records from patient names, deterministic masking consistently applies masked values for employee names across databases, and masking runs, and random strings replace license plate numbers while preserving the original data format.

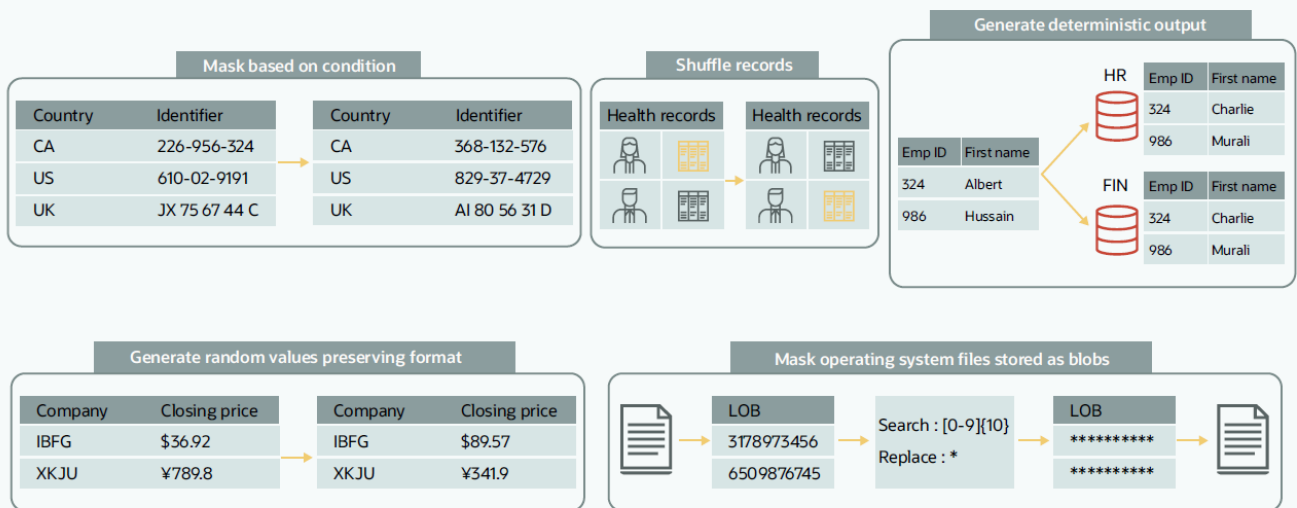


Figure 9-3: Masking formats and techniques in action

## Masking sensitive data using Oracle Data Safe

Data Safe provides an integrated set of security features for databases, enabling you to understand your data's sensitivity, evaluate data risks, and mask sensitive data.

### Simple, predefined, and custom masking formats

Data Safe's 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. Select the desired masking format from a drop-down list to mask a data column. When using the sensitive data discovery feature to analyze your application schema, 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 masked. This information can be downloaded and saved as a PDF file to assist with reporting and compliance.

**Masking report**

Generate report | Download report | Download masking logs

**Masking report information**

- Target database: VE61VXR74RV1G4H
- Masking policy: [MaskingPolicy\\_LL\\_202306141613](#)
- Masking report OCID: ...ab7gqg [Show](#) [Copy](#)
- Masking started: Wed, 14 Jun 2023 23:21:15 UTC
- Masking finished: Wed, 14 Jun 2023 23:22:36 UTC
- Masked sensitive types: 19
- Masked schemas: 1
- Masked tables: 8
- Masked columns: 27
- Masked values: 1.9K
- Masking options: [View details](#)

**Masked values summary chart**

The masked value percentages/distribution shown in the graph for each of the sensitive types is with respect to its parent sensitive category.

**Masked columns**

Schema	Table	Column	Masking format	Sensitive type	Parent column	Total masked values
HCM1	EMP_EXTENDED	PAYMENTACCOUNTNO	Credit Card Number	Card Number	-	107
HCM1	EMP_EXTENDED	TAXPAYERID	US Social Security Number	Tax ID Number (TIN)	-	107
HCM1	LOCATIONS	STATE_PROVINCE	Random Name	Province	-	17

Figure 9-4: Data Safe masking reports summarize key information from masking runs

### Process automation through RESTful APIs

All Oracle Data Safe features are available through restful APIs. Once administrators define their masking policy, the RESTful APIs make integrating data masking into a fully automated test data provisioning workflow easy.

## Masking sensitive data using Oracle Data Masking and Subsetting

Oracle Data Masking and Subsetting is an Enterprise Manager management pack. Data Masking and Subsetting is used to mask data and create subsets of data.

### Data masking

Masking data with Data Masking and Subsetting follows a similar path to Data Safe. The first step is creating an application data model (ADM) that identifies sensitive columns and referential relationships. This is a mandatory step with Data Masking and Subsetting. Only sensitive columns identified in the ADM are eligible for masking.

Next, 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.

### 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 more flexibility in executing masking jobs.

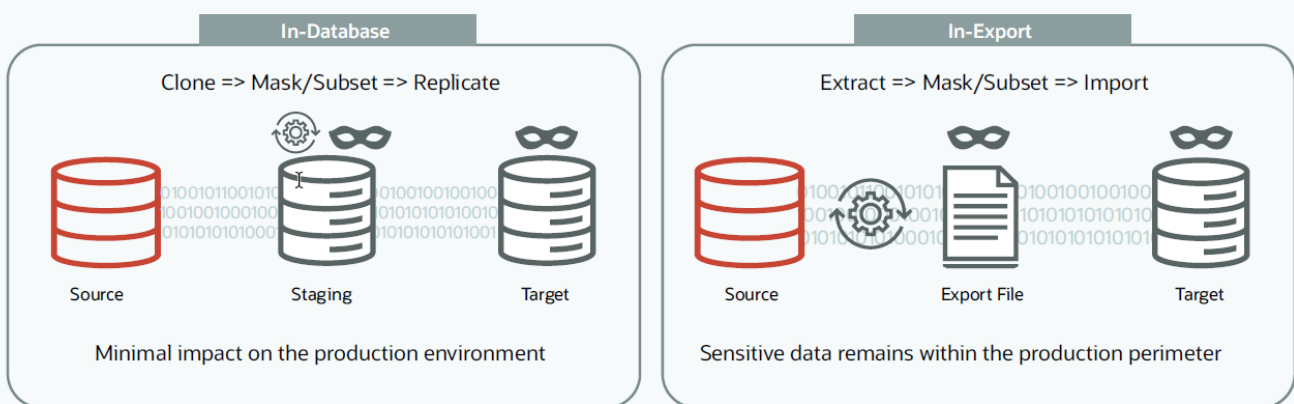


Figure 9-5: Masking modes for Oracle Data Masking and Subsetting

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 irreversibly modifies the data stored in the database and must not be performed on a production database instance.

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, leaving the original values untouched in the production database. The extracted and masked data are in standard Data Pump files, which can be imported into non-production databases. This mode eliminates the need to create a production database clone on a staging server and ensures that sensitive data remains within the production perimeter. With in-export masking, the 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 Data Masking and Subsetting, 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 dynamically mask data at runtime based on factors such as a database user's session context and the data being accessed. Unlike static data masking, dynamic masking does not change the underlying data.

### Use cases for dynamic masking

There are a few common use cases for dynamic masking

- Preventing the proliferation of sensitive data: Business intelligence 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 account numbers. Dynamic masking can mask the most sensitive columns from the report so that the analysts and report writers don't get access to that sensitive 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 dynamic data masking 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. Furthermore, 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, as may be the case for 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.

### Dynamic masking in Oracle Database

There are several ways to mask sensitive columns dynamically in 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.
- 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.
- Oracle Data Redaction: This provides the greatest flexibility in masking formats.

## Oracle Data Redaction

Oracle Data Redaction is one of the most flexible ways to implement dynamic masking. Data Redaction does not alter data stored in the database but instead masks the information when it's returned to the user. It supports frequently 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 traditional 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 Data Redaction is transparent to applications as it can preserve the original data type and formatting when sending the transformed data to the application.

Oracle Data Redaction is also transparent to the database as it doesn't alter the data in the database buffers, caches, or storage. For database transparency, Oracle 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 can be created with the DBMS\_REDACT PL/SQL package. Policies specify:

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

```
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 9-6: 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 as the expression 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 the context. 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.

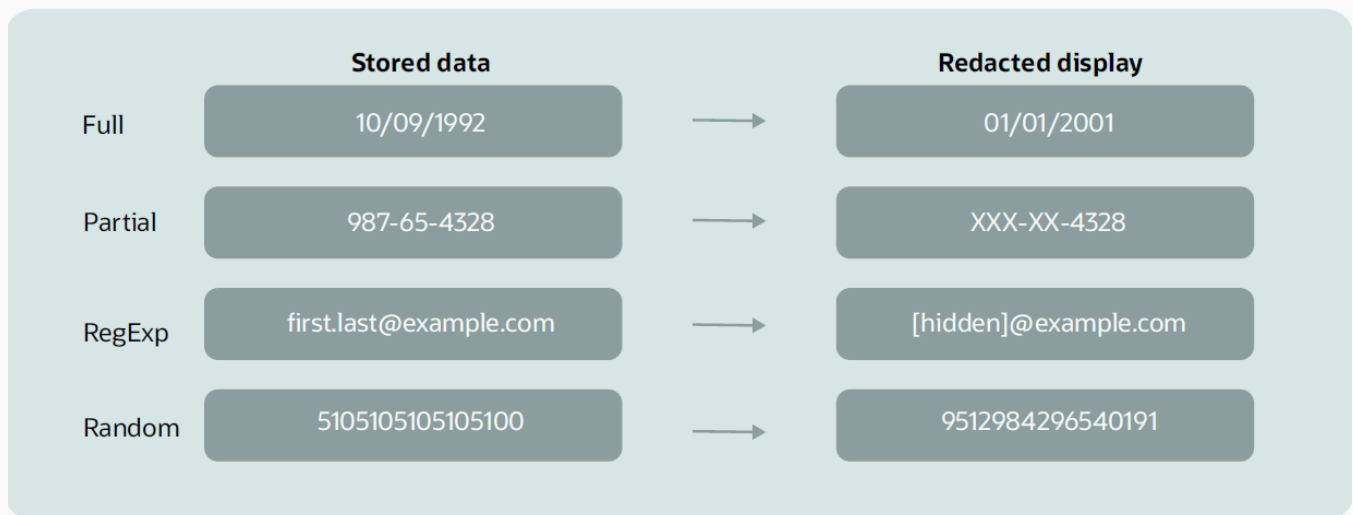


Figure 9-7: Data Redaction formats

Figures 9.8, 9.9, and 9.10 help demonstrate how application users see the final data after data redaction. Figure 9.9 shows an HR application's original data without any data redaction. Figure 9.10 shows the redacted data in the SSN, Corporate Card, Position, and Salary fields using partial, regex, random, and full transformations.

Home		Help		About		Logout	
<b>Employee Profile</b>							
Identity		Supplemental Data		Organization		Communication	
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	<input checked="" type="radio"/> Yes, Active						

Figure 9-8: Unredacted application data

Home		Help		About		Logout	
<b>Employee Profile</b>							
Identity		Supplemental Data		Organization		Communication	
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	<input checked="" type="radio"/> Yes, Active						

Figure 9-9: Redacted application data

In addition to simple formats like full redaction, nullify, partial redaction, and random values, Data Redaction supports several 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 9.4 shows some of the predefined redaction formats. Of course, you can also create new redaction formats for your specific requirements.

Format	Description
<b>US SSN last four</b>	Redacts the first 5 numbers of Social Security numbers when the column is a VARCHAR2 data type. For example, the number 987-65-4320 becomes XXX-XX-4320.
<b>US SSN first five (formatted)</b>	Redacts the last 4 numbers of Social Security numbers when the column is a VARCHAR2 data type. For example, the number 987-65-4320 becomes 987-65-XXXX.
<b>US SSN total (formatted)</b>	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.
<b>US SSN last four (formatted)</b>	Redacts the first 5 numbers of Social Security numbers when the column is a NUMBER data type. For example, the number 987654320 becomes XXXXX4320.
<b>US SSN first five</b>	Redacts the last 4 numbers of Social Security numbers when the column is a NUMBER data type. For example, the number 987654320 becomes 98765XXXX.
<b>US SSN total</b>	Redacts the entire Social Security number when the column is a NUMBER data type. For example, the number 987654320 becomes XXXXXXXXXX.
<b>CA SIN six nines plus last three</b>	Redacts the Canadian Social Insurance number by replacing the first 6 digits by 9 (number). For example, 123456789 is redacted to 999999789.
<b>CA SIN last three</b>	Redacts the Canadian Social Insurance number by replacing the first 6 digits by X (string). For example, 123456789 is redacted to XXXXXX789.
<b>CA SIN last three (formatted)</b>	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.
<b>UK NIN alpha (formatted)</b>	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.
<b>UK NIN alpha</b>	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.
<b>Credit Card last four (formatted)</b>	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.
<b>Credit card last four</b>	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.
<b>AMEX last five (formatted)</b>	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.
<b>AMEX zeros plus last five</b>	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.
<b>US Zip code (varchar)</b>	Redacts a 5-digit postal code when the column is a VARCHAR2 data type. For example, 95476 becomes XXXXX.
<b>US Zip code (number)</b>	Redacts a 5-digit postal code when the column is a NUMBER data type. For example, 95476 becomes XXXXX.
<b>Date to 1970</b>	Redacts dates to 01-JAN-1970.

<b>Date to millennium</b>	Redacts dates to 01-JAN-2000.
<b>North America phone number (formatted)</b>	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.
<b>North America phone number (zeros)</b>	Redacts the North American phone number by leaving the area code, but replacing everything else with 0. For example, 6505550100 gets redacted to 650000000.
<b>North America phone number</b>	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 9-3: 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 9.11, data subsetting supports many different use cases, 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.

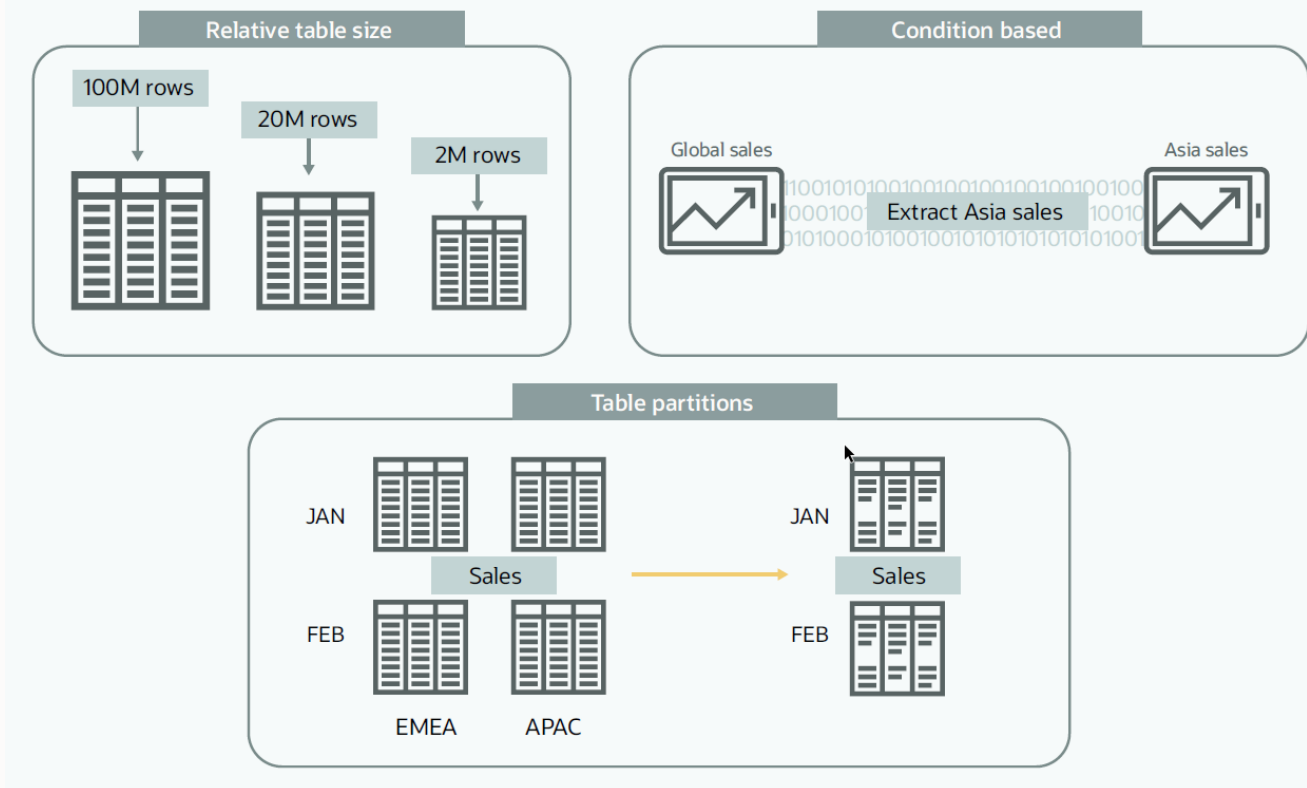


Figure 9-10: Data subsetting use cases

Table 9.4 lists different ways in which data can be subsetting.

Term	Description
<b>Random Sampling</b>	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.
<b>Stratified Sampling</b>	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.
<b>Time-based Subsetting</b>	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.
<b>Conditional Subsetting</b>	Selecting data based on certain conditions or criteria. For example, you might extract records where a particular attribute meets a certain threshold.
<b>Top-N Queries</b>	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 9-4: Subsetting conditions

Oracle Data Masking and Subsetting permits you to execute integrated data subsetting 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 provide 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, Oracle Data Masking and Subsetting offers the flexibility needed for your data subsetting 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 give it a try for the solutions discussed in this chapter, please go to:

- [Get Started with Oracle Data Safe Fundamentals](#)
- [DB Security ASO \(Data Redaction\)](#)
- [DB Security – 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 redaction and dynamic data masking 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 ten

## **Data encryption and key management**

---



## Introduction

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.

## Why encrypt?

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, database authentication and authorization 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 circumvent database access controls and go after the data in another way.

Attackers might try to access data by intercepting it 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 access data is by accessing the system as a privileged operating system user and directly reading the database files, bypassing database controls. Attackers can also go after unencrypted database backup files stored on removable physical media, tapes in a remote location, or the cloud.

Encryption is the best technique for protecting data in these situations, as it renders the data unintelligible to anyone trying to access it directly. 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 many regulations or security standards regarding sensitive or personally identifiable information.

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 a standard capability of all Oracle databases, 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. Most Oracle client software (thin and thick JDBC, instant client, SQL\*Plus) support 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 turn it ON 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 customers 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 at the endpoints and may be self-signed or issued by a recognized certificate authority.

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.

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

Security and performance often have an inverse relationship, which is generally true with TLS cipher suites. For example, ECDHE, the strongest default cipher suite, has the highest overhead. Although stronger from a security viewpoint, these algorithms will be less performant than algorithms using static keys (RSA, ECDH, or DHE), and within that family, ECDHE\_RSA is less performant than ECDHE\_ECDSA and so on. When choosing your cipher suite, you should consider encryption overhead if throughput degradation is a concern. Note that either the database client OR server can override the default list, so if throughput concerns are only applicable to a limited number of clients, users have the flexibility to specify the stronger algorithms only for specific clients. 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 a local file system or some other form of storage (ASM, for example). Encrypting this data helps ensure an attacker cannot read the information directly from those files.

One could encrypt data at rest in multiple ways: at the application layer, the file or volume layer, or the database layer. Irrespective of which layer the data is encrypted, the key considerations for selecting an encryption solution include:

- Security
- Performance
- Simplicity of installation and use
- Transparency to existing applications
- Time to convert data from plaintext to encrypted form
- Patching
- Key management

You should consider all these topics before selecting an encryption strategy, as it can have long-term effects on the manageability and maintenance of the applications. In general, the lower in the application stack encryption is implemented, the less intrusive the implementation will be. At the same time, encrypting data lower in the stack reduces the number and types of threats the encryption addresses.

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. Applications also need a way to manage encryption keys and securely store them somewhere. Application-layer encryption can also impact database performance and the types of queries other than equivalency searches 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. If multiple applications share the information in the same database, encrypting in the application requires they share access to the same encryption key to encrypt and decrypt the same data in the same way.

Essentially, encrypting data at the highest application layer imposes a significant development and management burden because it limits performing meaningful relational computation on the data. It 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.

File or volume-layer encryption is another option for protecting data at rest, but this type of encryption provides very limited risk mitigation. You could use an encrypting file system (such as ZFS), which seems simple but has many disadvantages when used to protect tablespace data in Oracle databases. 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 data store), employing a security control that automatically decrypts data for OS commands like strings, move, or copy doesn't make much sense.

When using third-party file-system encryption solutions, there is always a possibility for introducing system instabilities and upgrade issues through invasive operating system or file system modules. Such third-party software can also disrupt patching policies, preventing teams from applying operating system patches until a dependent patch is available.

Neither application-layer nor volume-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) encrypts data within the database. The encryption is transparent to authorized applications and database users 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 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.

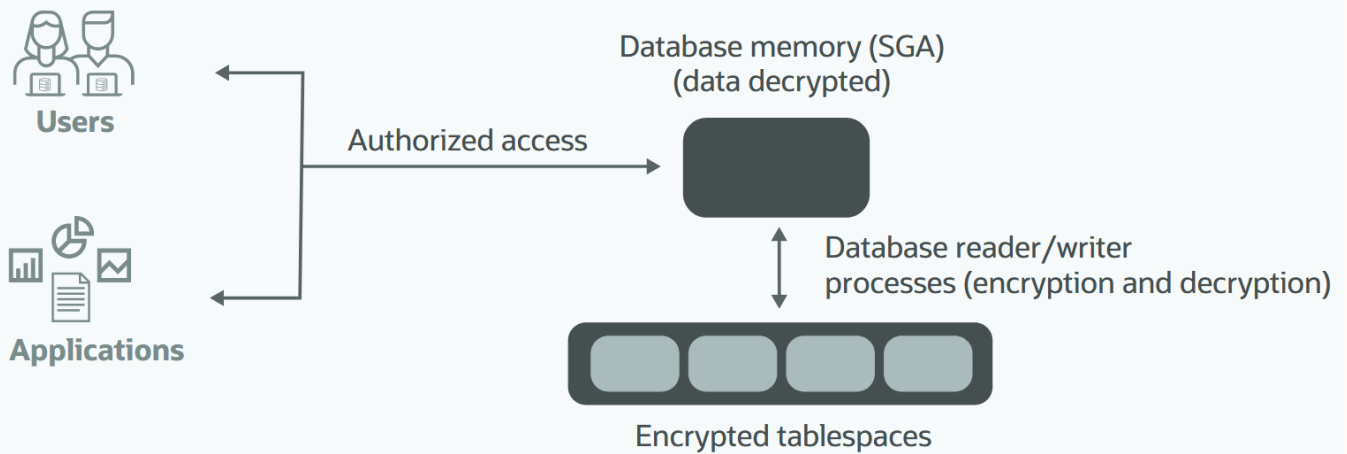


Figure 10-1: Users and applications work against clear-text decrypted data in database memory

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 'AES128' MODE 'XTS' ENCRYPT;
```

Note: The default encryption algorithm in Oracle Database 23c is AES256, and it allows you to choose between “cipher feedback” (CFB) and the modern “XEX-based tweaked-codebook mode with ciphertext stealing” (XTS) encryption modes.

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 [Advanced Security Admin Guide](#) for details.

In addition to this, dbca in 23c 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 the same size as the processed tablespace.

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.

In Oracle Data Guard, where standby databases are already in mount mode, use offline encryption to minimize downtime without additional temporary storage space. The procedure is to turn off the managed recovery process, encrypt the standby database's data files, perform a switchover, and then encrypt the new standby database. In this scenario, the downtime for encrypting any size database is as short as the duration of one or two switchovers.

## **TDE two-tier key architecture**

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 somehow gets exposed.

Oracle 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). The master encryption key (MEK) encrypts the internally generated data encryption keys (DEK), which in turn encrypt tablespace data files or table columns. The database 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.

Through SQL commands, administrators perform key management operations (create, open, rotate, backup, etc.). Key management 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. With Oracle Database 18c and later, you can import externally generated MEKs into the key store, enabling “bring your own key (BYOK)” functionality for those organizations who wish to use an external key generator.

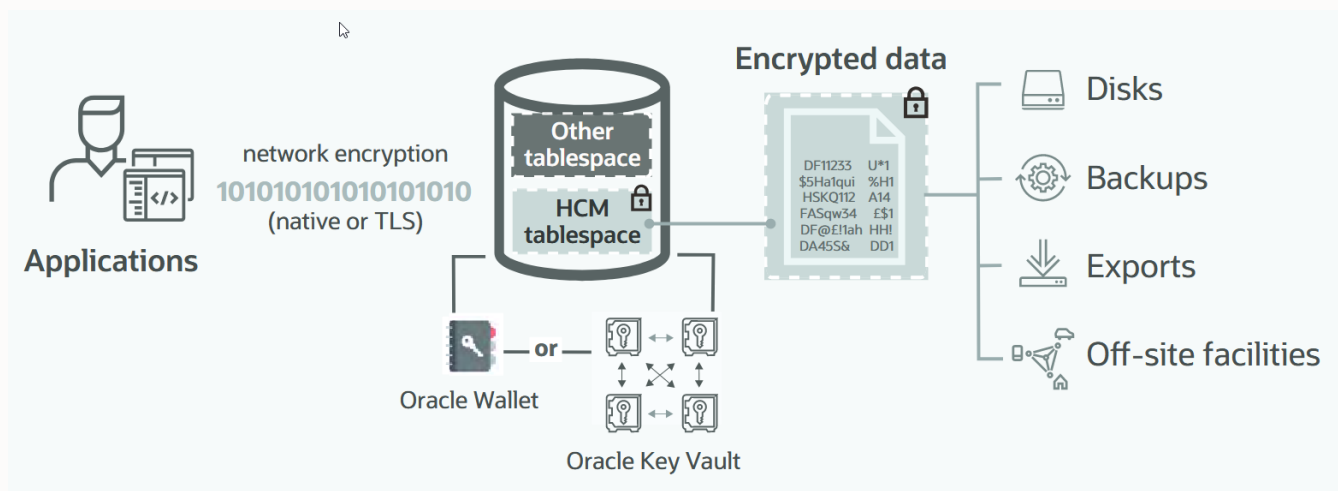


Figure 10-2: Oracle TDE encryption eco-system

## 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 Oracle Wallet. The wallet’s contents are encrypted using a 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 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 ease the management burden and reduce the risk of losing access to wallets.

Oracle Key Vault (OKV) is an enterprise-grade centralized key manager for your Oracle databases that provides secure and reliable distribution for master keys, wallets, and secrets. Oracle Key Vault delivers continuous availability and high scalability for read and write key operations while protecting against data loss. You can create Key Vault clusters with up to 16 nodes to form a multi-master cluster spanning geographically distributed data centers. Key Vault nodes may be placed within most major cloud providers (including Oracle, Amazon, and Microsoft). 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.

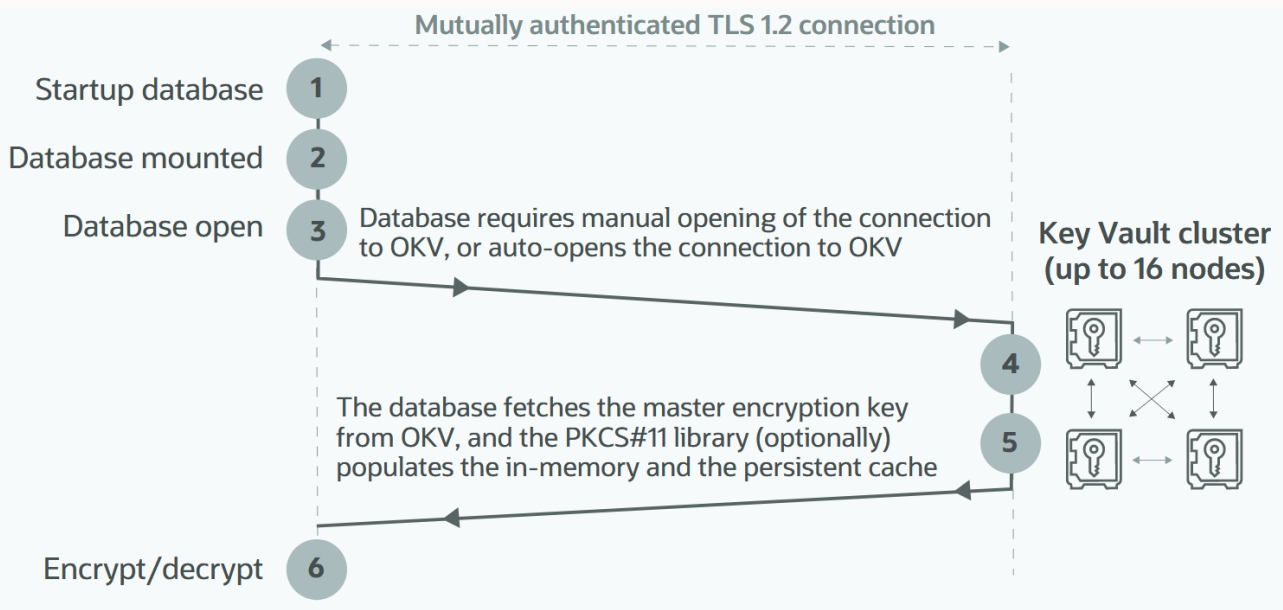


Figure 10-3: Oracle Key Vault securely delivers MEKs to Oracle databases on demand.

Databases can connect to any Oracle Key Vault cluster node to retrieve master encryption keys. Any updates to keys or changes to authorization rules are synchronously replicated to a partner node to guard against data loss in the event of a node failure. If the Oracle Key Vault connection fails or a node goes down, the database servers transparently failover to a nearby active node for read/ write operations without downtime or user intervention. As an additional guard against service interruptions, sites can optionally configure a local persistent cache, 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.

In its default configuration, Key Vault endpoints cache keys for the database server. This 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.

*Note: Before configuring a system to use non-extractable keys, Oracle recommends careful testing of the performance impact.*

Oracle Key Vault supports TDE keys across enterprise database deployments using Multitenant, Real Application Cluster (RAC), Data Guard, Sharding, GoldenGate, and other Oracle products and technologies. In addition to TDE keys, Oracle Key Vault 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.

Beyond encryption, Oracle Key Vault is a full-featured secrets manager. Key Vault 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 Oracle Key Vault and securely retrieve them at runtime. Centralizing credentials in Oracle Key Vault 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.

A recent addition to Key Vault’s capabilities is 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.

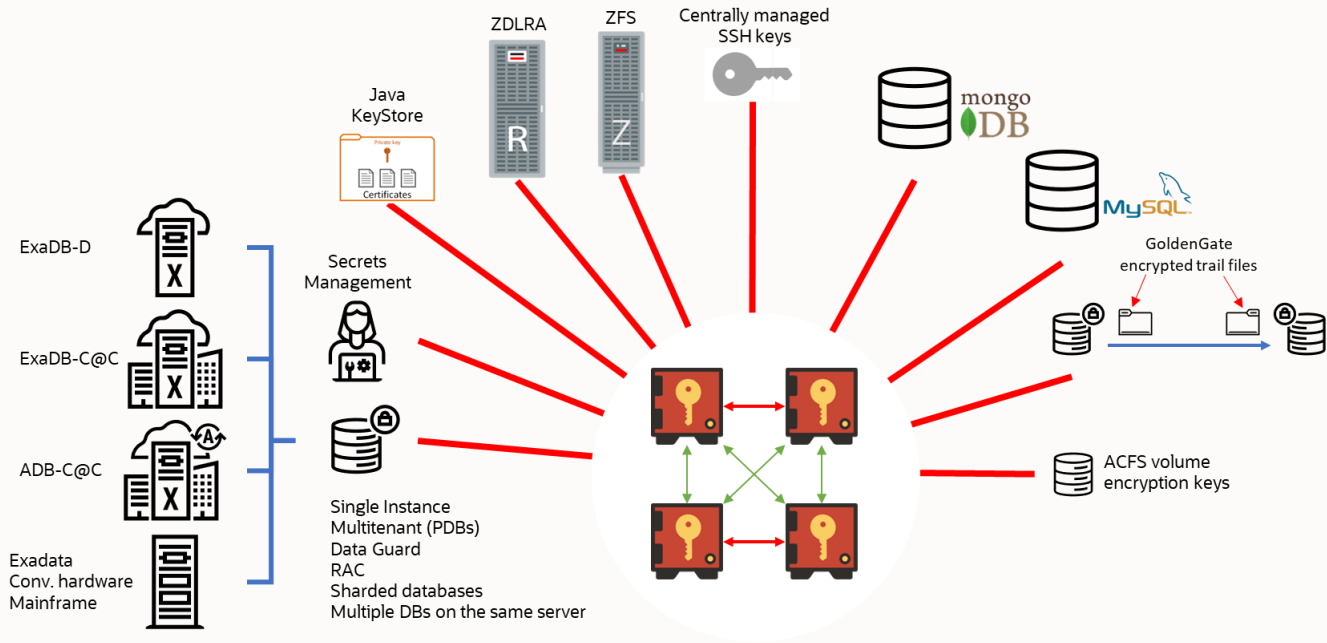


Figure 10-4: Oracle Key Vault manages Oracle TDE MEKs and other keys and security objects

### 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.

### Key Vault extensibility

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. Separating keys and secrets from application code increases security and simplifies development. For example, Key Vault can provide key management for Oracle Database’s DBMS\_CRYPTO package.

### Key Vault user management

Key Vault is designed for use in demanding enterprise environments. The console supports integration with Microsoft Active Directory, allowing centralized management of administrative user accounts. Key Vault also supports SAML-based single sign-on, enabling multi-factor authentication and integration with enterprise identity services.



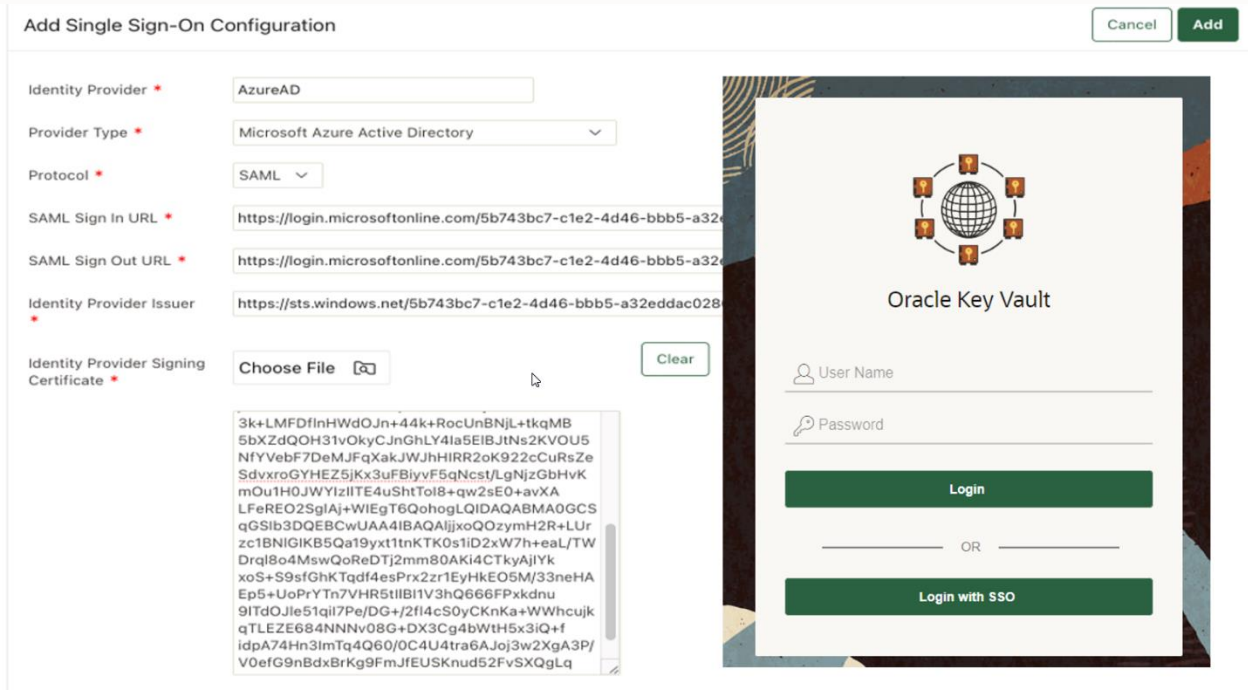


Figure 10-5: SAML-based Single Sign-On in OKV 21.6

## 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 give it a try for the solutions discussed in this chapter, please go to:

- [DB Security – ASO \(Transparent Data Encryption\)](#)
- [DB Security – Key Vault](#)

## Summary

Rising security threats, expanding compliance requirements, and cloud computing are just a few reasons why encryption has become critical for enterprises. 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 circumvent 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 Eleven

## **Database auditing**

---

## Introduction

Monitoring database activity to support incident investigation, detect potentially malicious behavior, and fulfill regulatory requirements is important. Activity monitoring can be done through database auditing and by monitoring incoming SQL commands as they travel over the network.

This chapter helps you evaluate the pros and cons of these approaches and discusses Oracle's solutions. Database auditing provides the most accurate record of database activity. Auditing has a broader contextual view of user activity than database activity monitoring techniques like network-based monitoring. One challenge with network-based monitoring is that direct local logins, recursive SQL, dynamic SQL, or stored procedures may circumvent the monitoring.

## Why audit?

Attackers that target your databases try multiple techniques to break in. They may bypass perimeter security, take advantage of trusted middle tiers, or masquerade as privileged insiders. Monitoring database activity is essential – both to know what is happening and so that you can take appropriate action.

There are two ways to track and monitor database activity: create audit logs for all critical activity inside the database or a network-based firewall that fully understands the database traffic.

Database auditing provides the most accurate record of database activity. Auditing tracks the creation or changes to database users (including password updates and privilege grants), use of privileges, access to specific objects or data, actions performed on database objects, and modifications to database settings. Auditing has a broader contextual view of user activity, but the policies need to be managed on when to create audit records. Network-based monitoring tracks all the SQL events that go to the database. However, as this happens on the network, it isn't easy to monitor direct local logins, recursive SQL inside the database, dynamic SQL, or stored procedures. Database auditing is frequently augmented with network-based activity monitoring.

Database auditing is more important now than ever and is mandatory in most organizations. You need to audit not only to detect malicious activity or unauthorized use but also to ensure that they comply with different regulations, including GDPR, PCI, CCPA, and other privacy regulations across the globe. Database auditing:

- Monitors activities of privileged database administrators. 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. While corporate security guidelines vary, they invariably require auditing privileged user activity, login events, sensitive data access, and other security-related activities.
- Detects unauthorized activity on sensitive assets. Organizations control access to their sensitive data for data privacy reasons, and any unauthorized activity must be quickly detected and reported.
- Assists with investigations of data breaches or other suspicious activity. Forensic analysis requires the ability to not only collect and store vast amounts of audit and network event records but also be able to search through them efficiently. In addition, the audit solution should support storing and retrieving historical data, tracking user privilege changes, and stored procedure changes to aid in analysis.
- Provides 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.

- Provides reports on changes to the database environment to auditors. You need to support forensic analysis during a breach or when you observe suspicious changes.

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. This oversight requires continuous collection and analysis of massive amounts of activity data to run reports and generate alerts on anomalous activities.

Database auditing is frequently augmented with Database Activity Monitoring (DAM) solutions that collect and store audit data from a fleet of databases for alert generation, analysis, and reporting. Oracle Database security products offering DAM capability include Oracle Data Safe and Oracle Audit Vault and Database Firewall (AVDF). Both 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
- Providing out-of-the-box and customizable reports for security and compliance
- Ability to aggregate and search through audit records for forensic analysis

This chapter focuses on unified audit, an audit framework introduced in Oracle 12.1 (over a decade ago), and how to audit effectively. Effective auditing requires that audit policies capture essential details about significant auditable events and minimize the collection of useless data. Targeted auditing leads 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 specific security requirements of the organization. Oracle Database's audit features grew over more than three decades of development. Auditing was introduced in Oracle 7 (1992), and continuous development and improvements paved the way for today's unified auditing.

**Unified auditing** was introduced over a decade ago in Oracle Database 12c, where the auditing functionality of Oracle Database underwent a significant redesign. Some of the noteworthy improvements in unified auditing compared to traditional database auditing are:

- **Consolidation of audit trails:** Traditional auditing over time had multiple audit trails to support new security features – standard audit, fine-grained audit, Database Vault audit, and SYSDBA audit. Each audit trail had a slightly different format, making it challenging to get a consolidated view of database activity. With unified audit, the separate audit trails are now unified into a single audit trail with a standard format. The consolidated and normalized unified audit trail simplifies the collection, analysis, and management of audit records generated by different audit sources. Consistent formatting simplifies reporting and analysis of the audit data.
- **Improved manageability and simplicity of audit policy configuration:** Unified audit provides named audit policies that can be created once and enforced in multiple dimensions (e.g., on users and roles), giving more flexibility and simplicity.
- **Improved precision and selectivity to audit:** With conditional auditing capabilities, you can create precise, highly selective, and context-aware audit policies, which makes it easier to audit specific actions and reduce the number of irrelevant audit records. Selective auditing lowers storage costs, reduces performance overhead, and provides high-value audit records useful for auditors, forensic investigations, or regulatory compliance needs. Conditions can be based on application contexts, session contexts, and built-in functions.

- **Higher degree of integrity of audit data:** Unified audit trail is stored in the AUDSYS schema, which no one can log into. Further, the unified audit table, AUD\$UNIFIED, is a specialized table that does not allow external operations other than SELECT by a user with the AUDIT\_VIEWER role. Any attempt to directly truncate, delete, or update contents of AUD\$UNIFIED fails and generates audit records. You can manage audit data with the built-in audit data management package DBMS\_AUDIT\_MGMT. If desired, you can encrypt the audit tablespace with Oracle Transparent Data Encryption (TDE) and further protect it with an Oracle Database Vault realm.
- **Improved performance:** For typical use cases of auditing privileged users or auditing critical database operations, the performance impact is so low that it usually cannot even be measured due to low audit volume spread throughout the week. Internal performance tests using a TPC-C mixed application workload show that with a unified audit, you may see a CPU overhead in mid-single digits when auditing up to 360,000 audit records/hour. The additional overhead is still in the single-digit range for extreme audit loads up to 1,800,000 audit records/hour. As auditing is a transactional activity with typical ACID properties to guarantee a record of database activities, fine-tuning audit policies is recommended to reduce database performance impact to negligible.
- **Extensible audit trail:** You can extend the unified audit trail to include additional attributes for your application context values.

**Note:** Legacy traditional auditing is desupported starting in Oracle Database 23c. Suppose you still use traditional auditing when upgrading to Oracle Database 23c. In that case, 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 customers still on the old system to bridge the gap between when they upgrade to 23c and when they can migrate to the unified audit facility. Database 23c will not allow you to create new traditional audit settings or update existing ones. You can only delete the existing traditional audit settings.

## Effective database auditing

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 – the “needle in the haystack” problem.

A more practical approach is to only audit significant events using selective audit policies. Effective auditing requires that audit policies capture the essential details about significant events while reducing irrelevant audit records. Reducing the “noise level” of unnecessary audit records can lead to quicker and more reliable threat detection. In most cases, this approach will have you focus your audit configuration on capturing privileged user activity, security-relevant events, and sensitive data access while ignoring routine activity generated from known sources. 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. Continuous privileged user activity monitoring allows security teams to identify anomalous behavior and detect sensitive data leaks quickly.

Database users granted system privileges, such as SELECT ANY TABLE or ALTER USER, should be monitored to detect abuse or misuse. Other noteworthy events include failed login attempts, schema structural changes, privilege grants, and so on. Such security-relevant events warrant greater scrutiny and constant monitoring to prevent abuse.

Databases usually contain sensitive data whose access should be controlled and monitored. Examples of sensitive data might include financial data, credit card numbers, email addresses, and other personal data that describes an employee or customer. Sensitive data access auditing is a powerful monitoring mechanism providing visibility into access and changes to sensitive data. It may serve as a primary deterrence to those who do not have a business reason to access or modify them.

Focusing audit configuration on privileged user activity, security-relevant events, and sensitive data access helps build better audit policies – policies focused 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.

Follow these rules for provisioning effective audit policies:

1. Do not duplicate mandatory audits - Certain security-sensitive database activities are always audited in Oracle Database and cannot be disabled. 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, refer [here](#) for mandatory auditable events in Oracle Database 23c. 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 the database version; therefore, you should check the Oracle Database Security Guide that corresponds to your version. For instance, refer [here](#) for predefined audit policies in Oracle Database 23c.

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.

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.

Following these recommendations, let us examine audit policy configuration for these use cases:

- Privileged user activity auditing
- Security-relevant events auditing
- Sensitive data access auditing

## Privileged user activity auditing

Certain database user accounts are privileged due to their job responsibility to work directly with infrastructure. Typical privileged database user accounts fall into three categories:

- Accounts with database administration privileges
- Accounts with direct database access
- Accounts that pose a higher risk due to exposure

You can identify database user accounts with administrative privileges from different sources, like the User Assessment report in Data Safe or the Database Security Assessment (DBSAT) report. Top-level statements by administrative users (e.g., SYSDBA, SYSKM) are mandatorily audited when the database is in the closed or mount state. You need to configure audit policies to capture all top-level actions of administrative user accounts, such as SYS, during normal database operations. 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 they bypass network monitoring. 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 direct_db_access
ACTIONS ALL
WHEN '(SYS_CONTEXT (''USERENV'', ''IP_ADDRESS'')) IS NULL)'
EVALUATE PER SESSION
ONLY TOPLEVEL;
```

In some cases, it will be necessary to closely monitor all user-initiated activities of individual database accounts as they may 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. 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.

## 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. Database-management events like backup/restores, cloning, and patching operations are always captured as part of mandatory auditing. Changes to Database Vault policies are also captured as part of mandatory audit. Additional actions you should audit include:

- Database-wide security policy changes
- Account-management events related to users, roles, privileges, grants, and revokes
- Security policy changes around protected objects or schemas
- Database schema structural modification events
- Activities using system privileges
- Components with data transfer, such as Data pump
- Suspicious events like multiple failed login attempts, non-business hour activities, and activities from dormant accounts.

Consider enabling these predefined unified audit policies to 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. If you are using Data Safe or AVDF, you can also provision the “Database Schema Changes” policy.

## Sensitive data access auditing

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.

A sample policy that audits updates on the `salary` and `bonus` columns in the HR schema’s `employees` table from outside of the trusted application path looks like this:

```
CREATE AUDIT POLICY USER_ACTIVITY_NOT_IN_TRUSTED_PATH
ACTIONS UPDATE(SALARY, BONUS) ON HR.EMPLOYEES
WHEN 'SYS_CONTEXT(''APPUSER_CONTEXT'', ''APP_USER'') NOT IN (''EMPLOYEE_USER'',
''HR_USER'', ''HR_MANAGER'')'
EVALUATE PER STATEMENT;
```

AUDIT POLICY USER\_ACTIVITY\_NOT\_IN\_TRUSTED\_PATH BY USERS WITH GRANTED ROLES EMP\_ROLE ;

In this policy, we define a trusted path as accessing data through the application – the policy checks an application context value APP\_USER to verify that the application has set this context to an approved value.

## Audit policy provisioning from Data Safe or AVDF

Oracle Audit Vault and Database Firewall, and Oracle Data Safe provide flexibility to provision audit policies in 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.

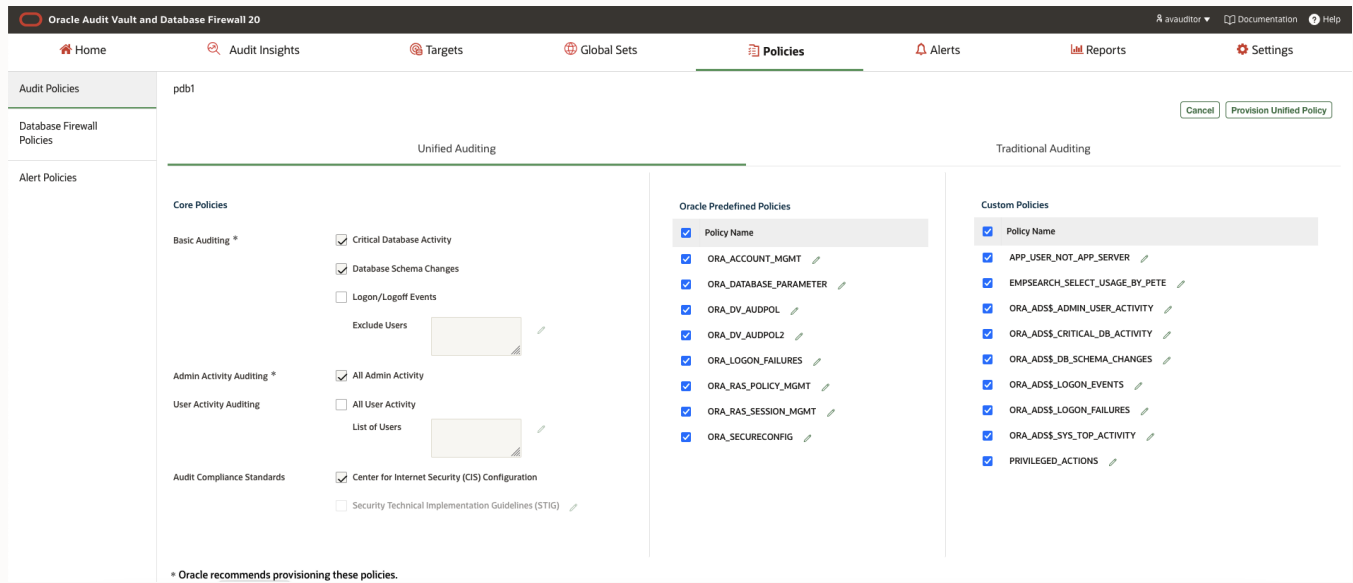


Figure 11-1: Audit policies provisioning from AVDF Console

Oracle Data Safe offers similar single-click provisioning capabilities.



## Transition from traditional to unified auditing

Traditional auditing is desupported in Oracle Database 23c. If you still use traditional auditing, 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 traditional 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.
  - If you are unfamiliar with the syntax of creating unified audit policies, use the syntax converter script available in My Oracle Support note [2909718.1](#). The converter converts traditional 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 traditional 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 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 give it a try for 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 Twelve

## **Database auditing – reporting and alerts**

## Introduction

As we have seen in Chapter 11, database auditing is a critical control for database security and regulatory compliance. Database user activity auditing facilitates a “trust but verify” approach to managing users and their privileges. But just collecting the audit data isn’t enough – you need to DO something with that data.

You should actively review and monitor database activity because accounts are always at risk of being compromised or misused. There are at least three reasons to implement a process for user activity auditing and reporting for your databases:

1. implementing corporate security guidelines
2. demonstrating regulatory compliance
3. conducting forensic analysis

While corporate security guidelines vary, most require auditing privileged user activity, login events, sensitive data access, unusual events such as SQL injection attempts, and other security-related activities. Aligning with these policies requires a solution that supports collecting audit data and generating alerts for administrators when unusual or suspicious activities are detected.

If your organization processes sensitive data, you may be 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. You need a solution that provides a rich set of pre-built compliance reports with configurable data retention periods to support you when demonstrating compliance with these regulations.

You probably need to support forensic analysis when investigating a breach or suspicious events. Forensic analysis requires the ability to not only collect and store vast amounts of audit records but also be able to search through them efficiently.

To support you with these use cases, Oracle offers two solutions, 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 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. 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. 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.

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.

## Auditing with Oracle Data Safe

Data Safe’s user activity auditing feature allows you to select from various predefined database audit policies and enable them on the database. Data Safe collects and stores audit records from the databases. Data Safe provides

interactive reports that let you track specific users, objects, or actions. Data Safe also supports forensic investigation and provides summary reports for compliance and activity reporting. These reports can be scheduled and downloaded as PDFs to document compliance programs. You can choose from several predefined alert policies to receive notifications of unusual activities that may indicate compromise.

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 then collects audit records for activities within the target database that match the audit policies. 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.

You can manage the audit data volume stored within Oracle Data Safe using the audit data retention settings, which allow you to specify a total time to keep audit records of up to seven years (up to one year online, up to six years in archive storage).

**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) ⓘ

12

Records stored online are available in the audit reports.

Archive retention period (in months) ⓘ

72

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.

Save Reset

Figure 12-1: Global audit settings in Data Safe

## 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:

- Spot Increases in Data Definition Language or Data Manipulation Language statements
- View trends in Login failure and user/entitlement changes
- Identify the top databases, database schemas, and objects contributing the most records to the audit volume
- Identify audit policies generating 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.

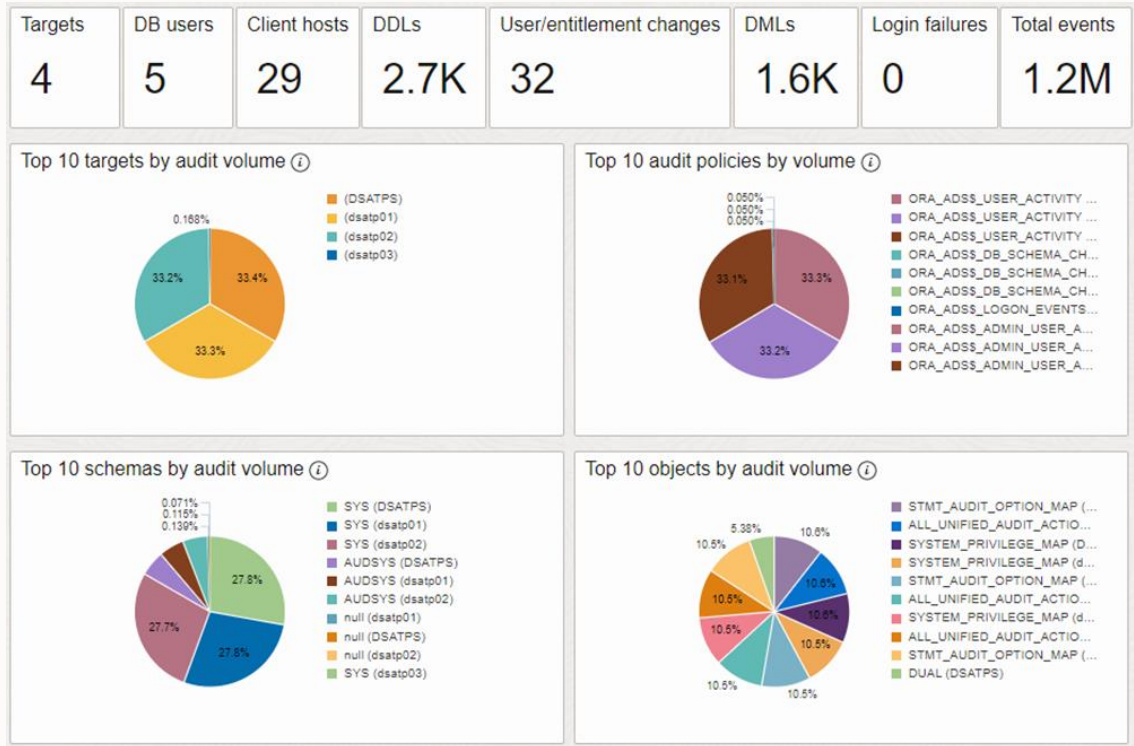


Figure 12-2: Data Safe audit insights

You can also 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 you charts and tables with a summary of 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.

Below are 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	DataPump and RMAN activities in the database
Sensitive data activity	Database activity on sensitive objects (as identified in Data Safe’s data discovery feature)

Table 12-1: 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 might have to comply with organizational policies or regulatory requirements that require you to keep your audit records for a specific time, which might be 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 online retention period and then deleted when it reaches 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 archive at any time.

## Alerts

You can enable alerts on your target databases to track and receive notifications of particular user activities and unusual behavior. 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. The following table shows a few of the predefined alert policies and their severity levels in Data Safe.

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 12-2: Alert policies in Oracle Data Safe

Data Safe includes a built-in alert dashboard to view and manage alerts. You can also set up 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, and PostgreSQL. Almost anything that produces audit data in a standard format is supported using the custom collector framework. The collector framework collects audit data in tables via JDBC or REST API. Audit collection is also possible from systems that write audit data to CSV, XML, or JSON files with AVDF's custom collector framework. An included software development kit (SDK) accommodates targets that cannot be accessed using the custom collector framework format.

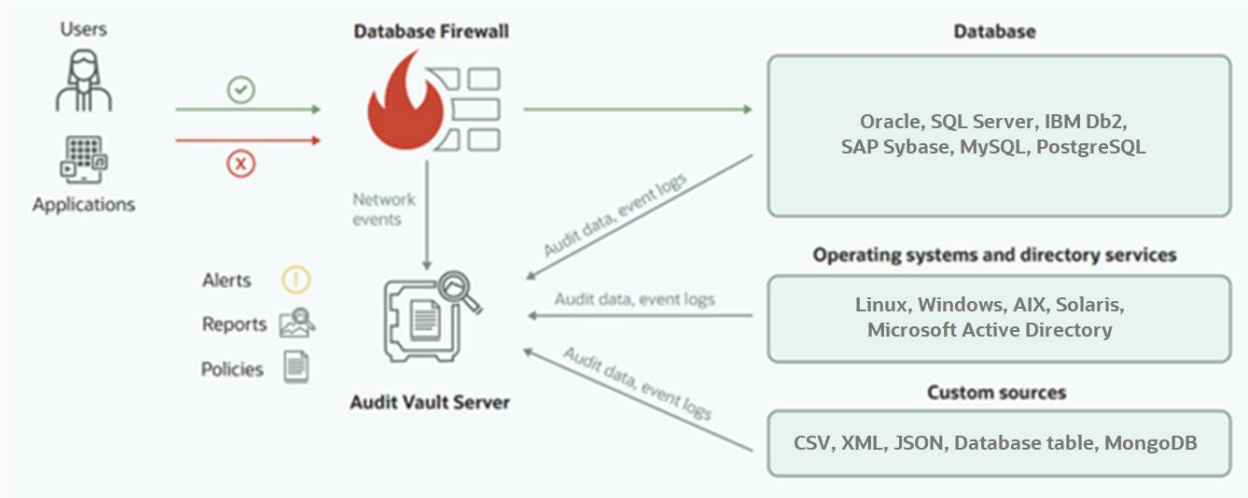


Figure 12-3: Oracle Audit Vault and Database Firewall

## Audit collection

The Audit Vault Server captures audit and event logs from various sources, including audit data from application tables or files. The Audit Vault server maps collected audit data and event logs into a normalized format and stores them in the Audit Vault repository. The normalized data format allows you to produce a single report across different databases and even non-database targets.

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 tracks user entitlements and stored procedure changes. 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. Based on the type of audit trail (database table-based audit trail, directory trail, transaction log trail, etc.), the Audit Vault Agents are deployed on a separate machine or on the same machine where the audit trail resides. A single Audit Vault Agent can support audit collection from multiple targets of different types. AVDF 20.9 introduces an "agentless" collection of audit data in Oracle Databases and Microsoft SQL Server databases. With the agentless collection, you use the agentless collection service that comes with the Audit Vault Server instead of deploying the 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. The schema for the AVDF repository schema is documented, enabling the use of third-party tools for specific reporting purposes.

AVDF reports provide detailed information on database activity events gathered from audit data and the Database Firewall (see Chapter Seven 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. The table below 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: <ul style="list-style-type: none"> <li>• All audit and network activity</li> <li>• Data modifications</li> <li>• Before and after values of the modified data</li> <li>• DDL activity</li> <li>• Failed logins</li> </ul>
Alerts	Alert reports display the raised alerts and their status.
Stored procedure audit	This report tracks the changes made to the stored procedures, such as creation, modification, and deletion.
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 capture user privileges for Oracle databases. Using AVDF, a baseline report can be created, and changes from the baseline can be analyzed, making it easier for security and DBA personnel to monitor 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 this user executes a command on the database as another user 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 users, dormant users accessing the system, or users accessing the system from IP addresses not seen before.
Assessment Reports	These reports capture security assessment data from Oracle Databases and provide recommendations that help secure your Oracle Database system. It also includes drift reports against the baseline.

Table 12-3: Built-in reports in AVDF

Auditors access reports interactively through a web interface. They may save reports in PDF or XLS format and configure 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. Additionally, 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.

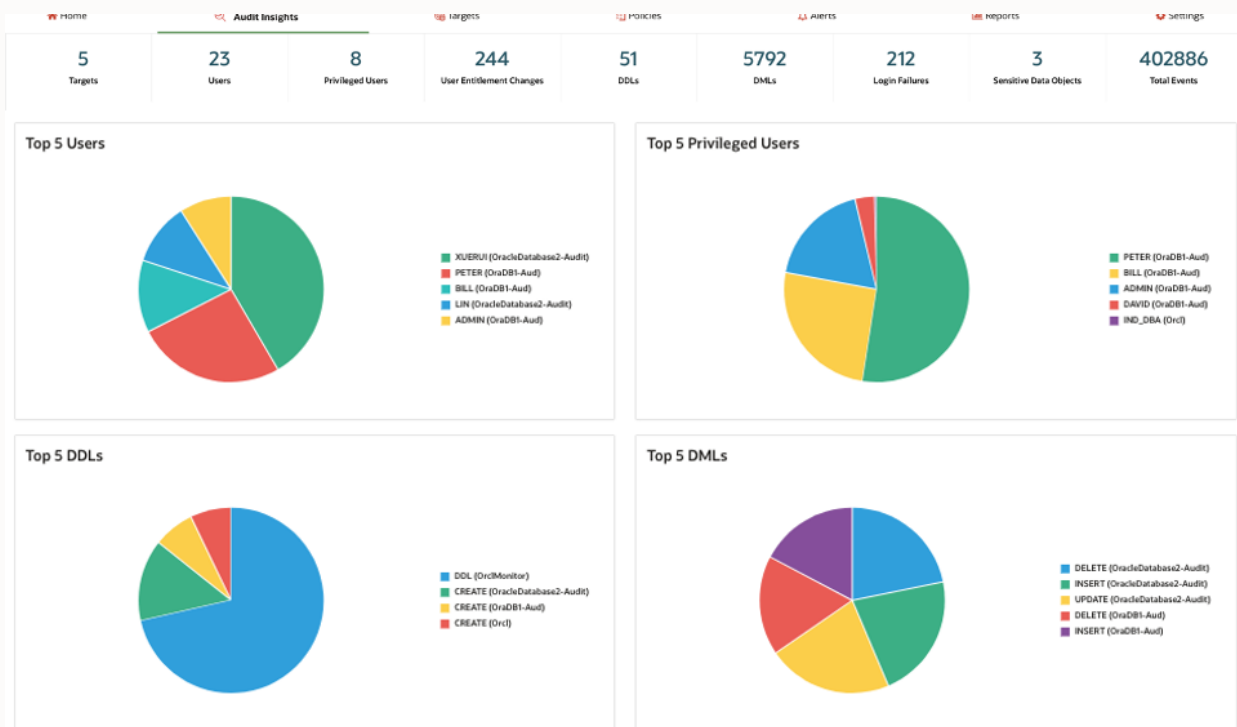


Figure 12-4: AVDF Audit Insights

## 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. 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

There are a few things that AVDF does uniquely for Oracle Database.

### 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 like CIS and STIG.

## Monitoring and drift detection for user entitlements

In Chapter Two, 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 procedure

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 any updates to stored procedures, and the reports can be reviewed for any changes.

## 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 give it a try for the solutions discussed in this chapter, please go to:

- [Get started with Oracle Data Safe fundamentals](#) → Lab 2: Audit database activity
- [Integrate Oracle Data Safe with Applications and Services](#) → Lab 3 and Lab 4
- [DB Security – Audit Vault and DB Firewall](#)

## Summary

Database auditing is essential, but if you do nothing with the information you collect, you're missing out on 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 Thirteen

## **Ransomware and Zero Trust**

---

## Introduction

Today, most security conversations are driven by one of these topics:

- Regulatory compliance
- Data breaches
- Ransomware
- Zero-trust

So far, we have been discussing mainly data breaches and regulatory compliance. In this chapter, we'll discuss the last two – 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 improving zero trust.

### A brief history of 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 payment (usually monetary).

Ransomware has been with us for a long time - the first known ransomware attack happened in 1989! Dr. Joseph Popp, an AIDS researcher, sent floppy disks to 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.

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. 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 30 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

First, let's focus on the second objective – minimizing the damage from a successful attack. For the most part, that focus is because it's where Oracle can help – 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

The 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. 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 which 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 (ZDRLAS).
2. **Synchronized recovery point objectives:** With multiple databases destroyed, restoring business operations (not just the individual databases) means that all of those databases need to be both recovered and have transactions synchronized across them so that all of the databases 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 ZDLRAS.
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.

Protecting against Ransomware requires vigilance and diligence, but the technology and infrastructure improve daily. Every law enforcement agency advises victims NOT to pay the ransom. The combination of encryption and an immutable, zero-data-loss recovery plan is much easier advice to follow.

## 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. Ransomware gangs employ a variety of tactics to leverage data that is now routinely 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.
- Traditional sales of stolen data – data, especially identity-related data, can 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 particular 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 in Oracle Key Vault. Refer to Chapter Ten for more information on encryption and key management.

*Note: If you do not properly protect the database's encryption keys, you might open 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 the data that you encrypted in the first place.*

Database encryption is commonplace enough that your databases are probably already protected from this type of out-of-band attack.

*Note: An out-of-band attack circumvents the normal database session – 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.*

## Zero Trust

If you're reading this chapter, there is a really good chance your organization is 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.

Here is a simple example – 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, Chief Technology Officer, MobileIron, 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:

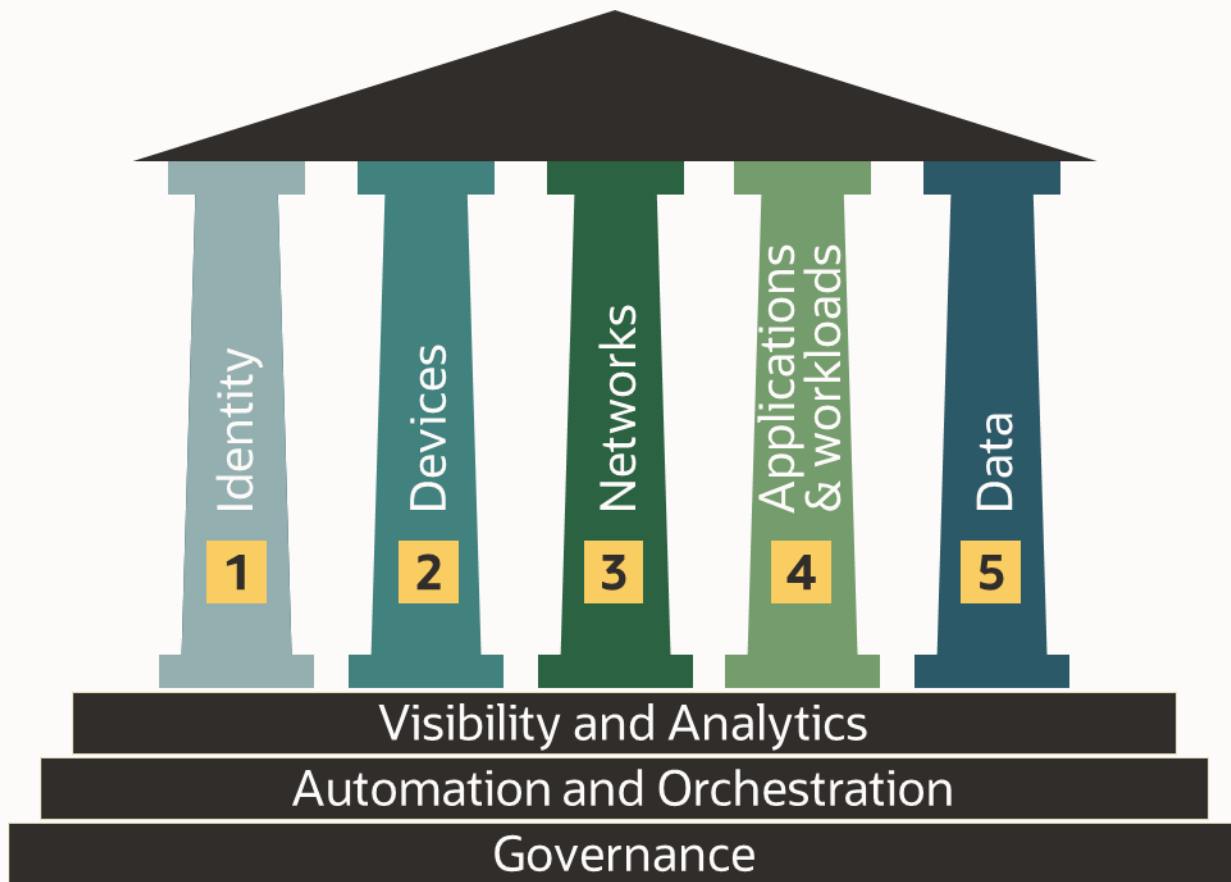


Figure 13-1: CISA Zero Trust Maturity Model Pillars

As you can see from the model, very few IT areas are not impacted by zero trust! 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

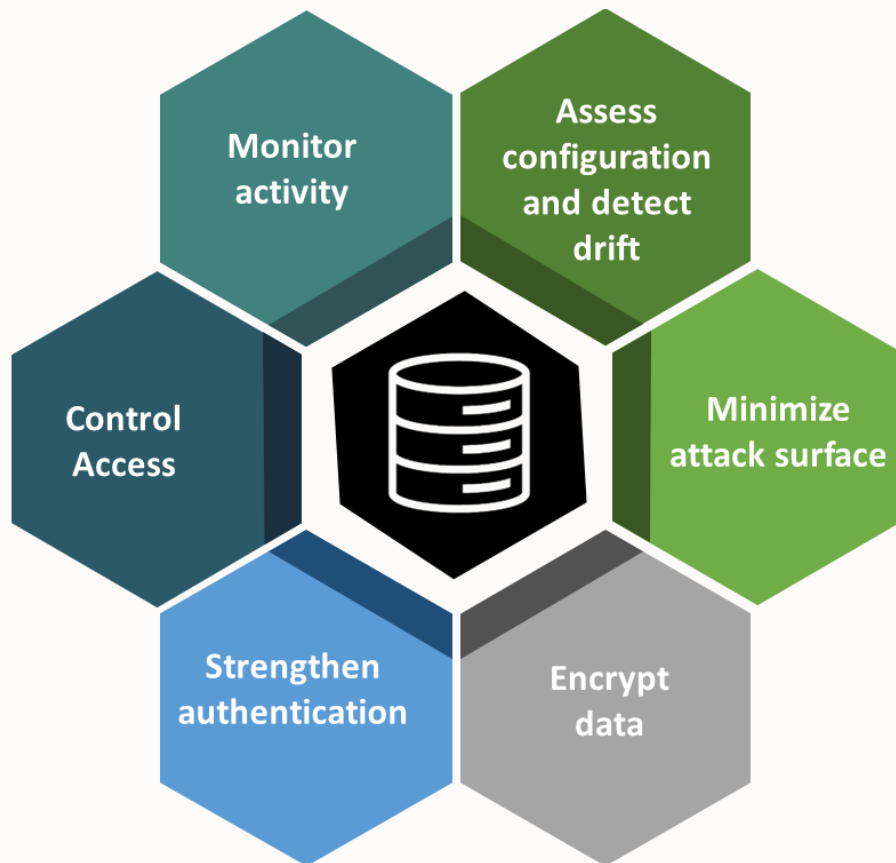


Figure 13-2: Typical zero trust projects for Oracle Database

### 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.*

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 Two 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 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 Nine 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

To learn more about controlling access to data, review Chapters Two (the portion on privilege analysis), Five, and Six 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 Ten 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 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 them infrequently.
- Administration/DBA accounts – 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 PAM.
- End users – Most databases have few (if any) end 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 Azure AD 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 Four 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*
- 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 Five and Six. .

## 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 23c*
- Data Safe *auditing* or Audit Vault and Database Firewall (to analyze audit data, create reports, and generate alerts)

Review Chapters Eleven and Twelve 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.

### **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 give it a try on the technologies discussed in this chapter, please go to:

- [Tales from the Dark Side: Hacking the Database](#)

# Chapter Fourteen

## **Database security in the multcloud**

---

## Introduction

Many organizations work with multiple cloud providers to avoid vendor lock-in, improve redundancy, optimize performance, cost, and services, 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. 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.

## Why multicloud?

Multicloud is a cloud computing strategy that leverages cloud computing services from two or more cloud service providers. 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 is a fairly common scenario, with different cloud providers selected based on how well and at what cost they provide a service.

You may have also encountered another term, hybrid cloud. A hybrid cloud is a strategy that 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*.

Several factors drive a multicloud strategy, including optimizing workloads at the cloud service provider that does the best job at the least expensive cost. Other reasons include avoiding vendor lock-in, improving redundancy, and complying with data sovereignty regulations. 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 leads to scenarios like the one below where identity is managed in one cloud, your applications are in another cloud, data is stored in another cloud, and your security tools could be in yet another cloud.

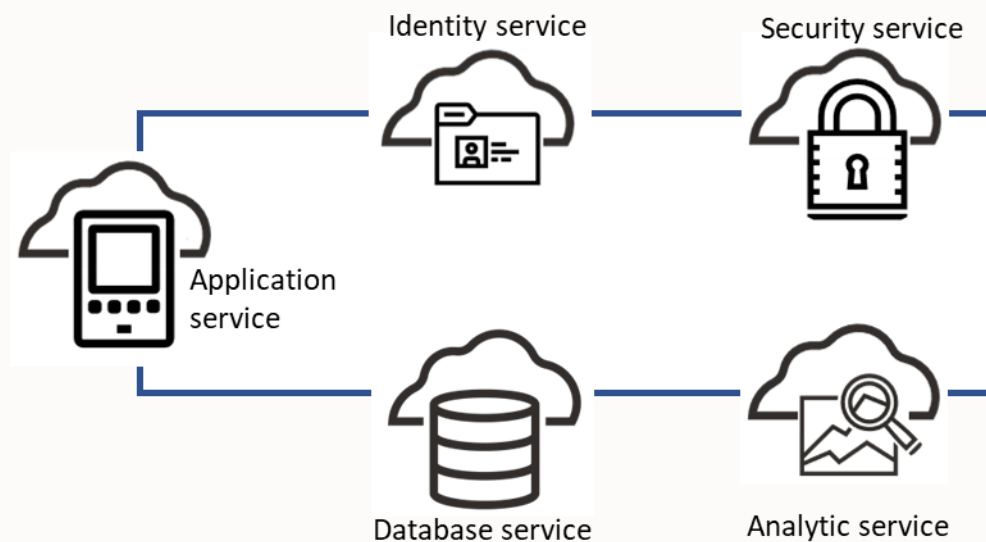


Figure 14-1: Multicloud environment

While you can try to fit all your cloud computing resources into one cloud, in most cases, you'll end up with multiple clouds. That may be because your cloud adoption happened organically, with different business units opting to use

services in different clouds. Or it might be because you are consciously trying to avoid lock-in with any single cloud vendor. The mix of clouds in your organization depends on your selection of enterprise applications, your cloud identity provider, your data storage cloud, and other requirements. In general, your application service should be extensible, with the ability to build new applications while reusing the same databases and tools to analyze the data.

## Identity cloud service

Although your services may span multiple clouds, you'll usually want one cloud provider to handle your enterprise identities. Other cloud services will either accept identity tokens issued by your 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 (generally LDAP directory-based) for cloud solutions. While this has the advantage of leveraging existing skills and knowledge base, 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 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 in OCI, AWS, Azure, and on-premises. Chapters Two, Three, Seven, Nine, and Twelve discuss Data Safe in more detail.



- Key Vault – available in the OCI marketplace, may be installed on-premises, in AWS, or Azure. As long as there is reliable network connectivity, a Key Vault node installed in one location can service databases in other locations. Chapter ten discusses Key Vault in more detail.
- Audit Vault and Database Firewall – available in the OCI marketplace, may be installed on-premises. Again, if there is network connectivity, Audit Vault and Database Firewall installed in one location can service databases in others. Chapters two, three, eleven, and twelve discuss Audit Vault and Database Firewall in more detail.

## Managing resources in multiple clouds

Multicloud is our reality, but managing resources separately in every cloud is demanding and increases security risk due to different tools and capabilities. Oracle recognizes this and has a solution to ease this management.

Oracle Database Service for Microsoft Azure (ODSA) is an Oracle-managed service for Azure customers to quickly provision, access, and operate enterprise-grade Oracle Database services in Oracle Cloud Infrastructure (OCI) with a familiar Azure-like experience. With ODSA, you can seamlessly build Azure applications with the high performance, high availability, and automated management of Oracle Database services, such as Autonomous Database, running on OCI. Each database can be monitored using a dashboard. Database metrics are streamed to Azure Application Insights, and database events may be provided to Log Analytics.

Management isn't the only concern; you must also address connectivity between services so performance doesn't suffer when crossing cloud boundaries. Oracle Interconnect for Microsoft Azure provides less than two milliseconds of latency with no intermediaries required. The service leverages private, redundant network links.

Additional information for ODSA is available [online in the Oracle documentation](#).

## 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 connect to the OCI object store and use identity tokens from Azure, Google Cloud, and Amazon Web Services to access resources in those clouds, such as object storage, so you don't need to transfer data from one cloud to another.

## 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 Azure Active Directory and 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. Suppose you work with an application or service using a different cloud identity provider. In that case, 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.

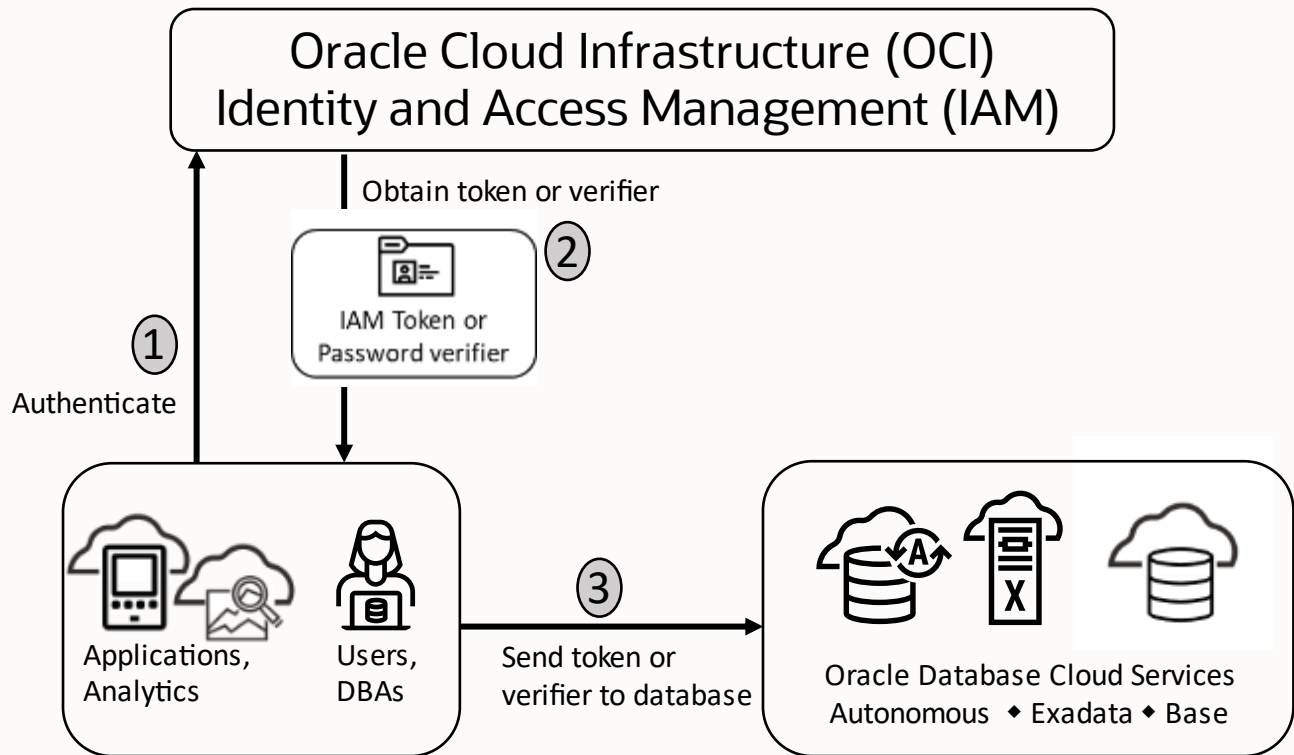


Figure 14-2: OCI database integration with IAM

### Microsoft Azure AD Integration

You can use Azure AD OAuth2 access tokens to access Oracle Databases in OCI or on-premises from applications hosted on Azure or Azure services. Federation won't work for these scenarios since the Azure service and Azure-hosted applications can only obtain an Azure AD access token for the database. These services and applications can pass Azure AD managed identities (service principals) to the database, 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 Azure AD tokens in much the same way as they do OCI IAM tokens. Azure AD access tokens can be passed through the client API, retrieved from the file system, or the database client can request the token directly from Azure AD using the Azure AD user's session token. The client can also direct the user to authenticate with Azure AD if the session token doesn't exist.

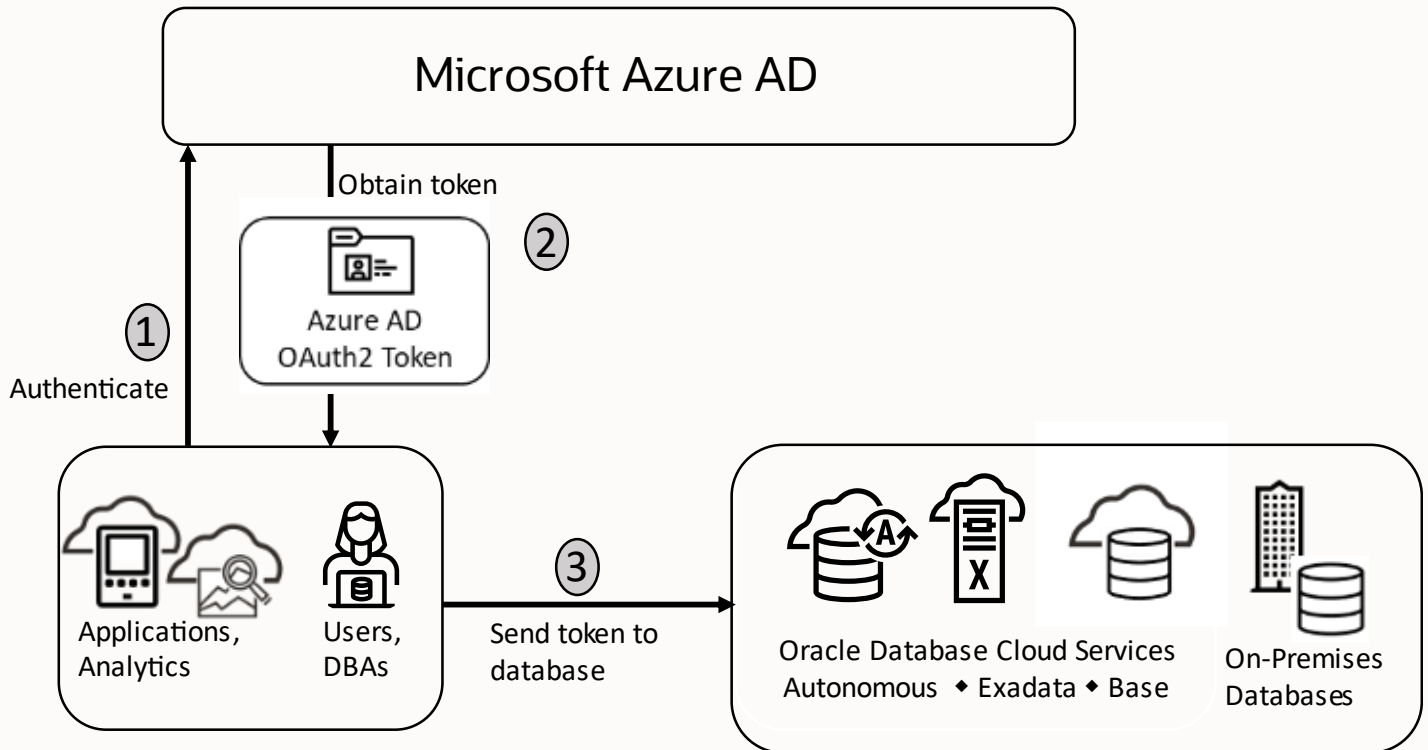


Figure 14-3: Oracle Database integration with Azure AD

### 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 Azure AD 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 give it a try on the technologies discussed in this chapter, please go to:

- Oracle Autonomous Database integration with OCI IAM workshop

### Summary

You have many reasons that drive you towards multicloud solutions. 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

## **Securing the Autonomous Database**

---

## Introduction

The Oracle Autonomous Database provides standardized, hardened security configurations that reduce the time and money 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 solutions while providing you with the tools to further restrict or isolate sensitive data from privileged users, developers, data analysts, and application administrators.

But there are some things the 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 and 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 is designed to minimize or eliminate 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.

Oracle Autonomous Database uses standardized, hardened security configurations that reduce the time and money spent managing database configurations. Security patches and updates are applied automatically, so you don't spend time, money, or attention keeping the software 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 solutions while also providing you with the tools you need to further restrict or isolate sensitive data from privileged users, developers, data analysts, and application administrators.

Broadly speaking, Autonomous Databases handle many 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 the Autonomous Database automatically applies all security updates online and provides “always on” end-to-end encryption. This preventative approach is critical because 85% of security breaches today occur after issuing a CVE (common vulnerability and exposure) alert.

## 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 in the 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 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 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 Oracle Unified Audit. This feature includes automated auditing for 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** – Autonomous Database easily integrates with Oracle Data Safe to help you assess and secure your databases. Oracle 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 Oracle Data Safe to identify and protect sensitive and regulated data in your Oracle 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 production data, Oracle 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 consumer is allowed to run.
- **Automated patching, upgrades, and maintenance** – One of the most significant 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. Oracle also 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 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, but they need to be configured as per 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 give it a try for 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](#)

## 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.



# Putting it all Together

---

We hope you have enjoyed this journey into Securing the Oracle Database and that it has provided you with some insights into the variety of defense-in-depth strategies available for keeping your data secure.

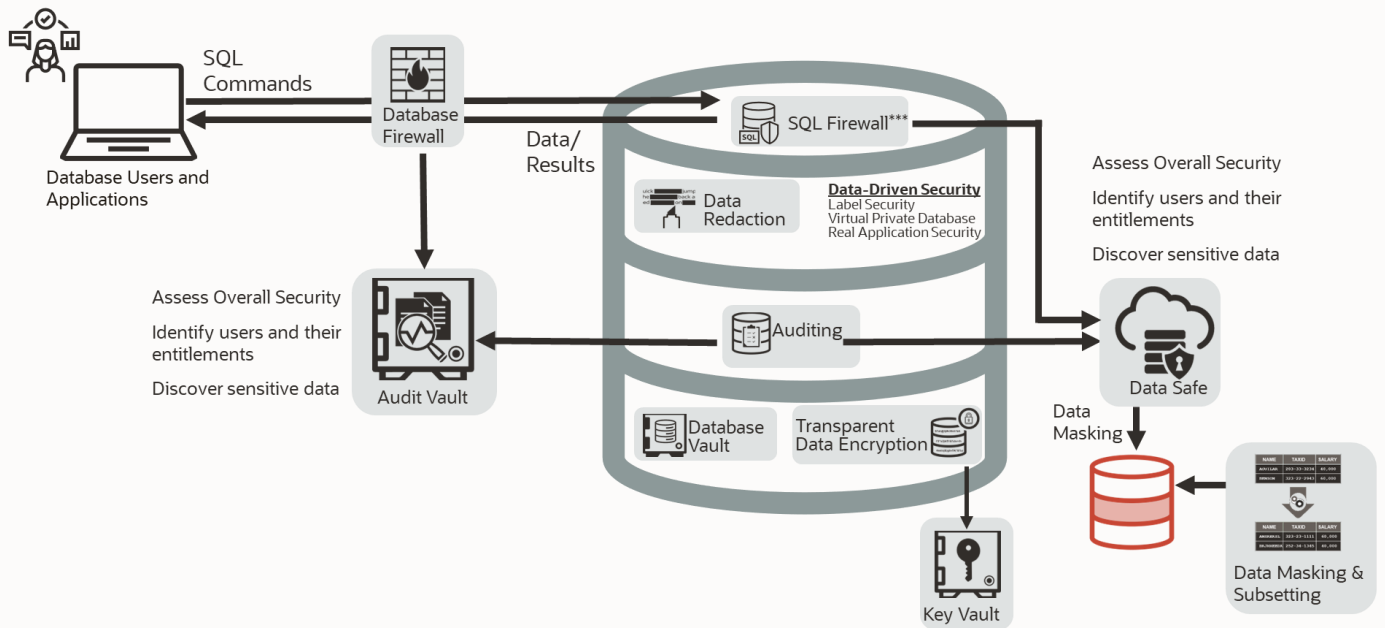


Figure 15-1: Maximum Security Architecture

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 17-1 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.

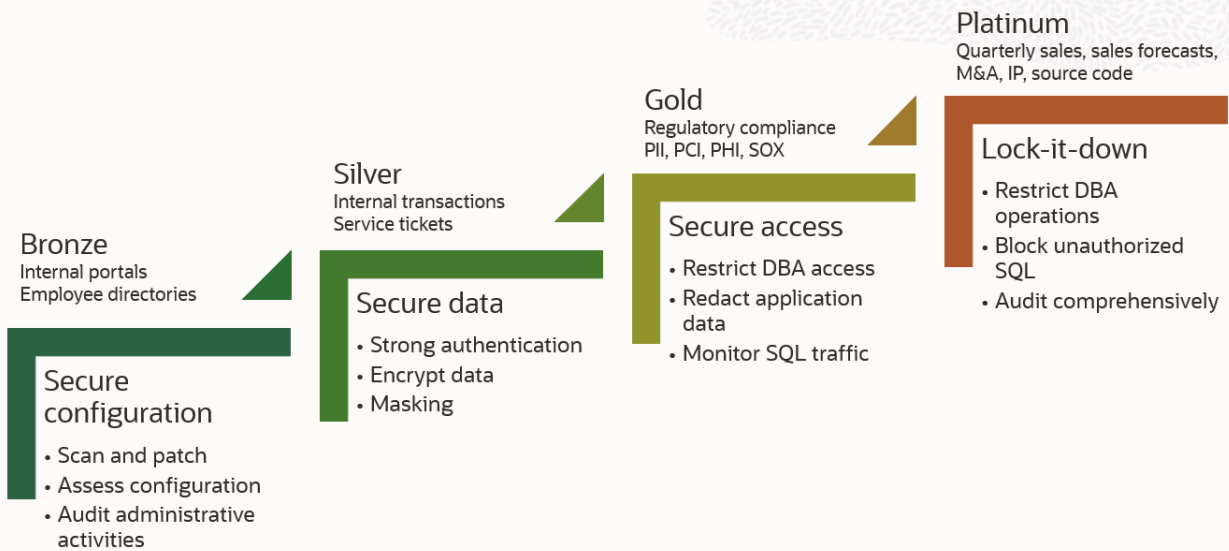


Figure 15-2: Example of implementing security controls based on system sensitivity

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.

## About the authors

**Alan Williams** is the Product Manager responsible for authentication and authorization technologies in the Oracle Database group. 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 30-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 auditing and activity monitoring. With close to 16 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 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.

**Michael Mesaros** is a Director of Product Management for Oracle Database Security and has over 25 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).

**Pedro Lopes** is a Product Manager in the Oracle Database Security group. 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 to 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 Oracle 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 holds a Master's Degree in Electrical Engineering from the University of Applied Sciences in Ravensburg, Germany.

**Russ Lowenthal** is Vice President of Database Security, helping over 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' 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 holds a bachelor's degree from the University of Texas, San Antonio, and 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.

**Hakim Loumi** is the Product Manager for Database Security field enablement and Oracle Data Masking and Subsetting. 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. For several years now, he's been in charge of the Oracle Database Security Livelabs, laser focused on giving the best user experience to discover, test, and understand (for free) the complete Oracle Database security portfolio.

**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.

## Acknowledgments

The following individuals helped with the review and provided invaluable feedback for this fifth edition:

Tanvir Ahmed	Sanjay Kulhari	Vikram Pesati
Anant Bhasu	Chao (Charles) Liang	Archana Rao
Ji-Won Byun	David Lin	Ajay Srivastava
Nishant Chaudhary	Peter Knaggs	Luna Tan
Yucechen Chen	Venkat Medam	Ruchi Tayal
Chi Ching Chui	Rahil Mir	Jinglei Xie
Marek Dulko	Hariprasath Mohankumar	Frank Xiong
William Howard-Jones	Gopal Mulagund	Deepak Yadav.
Ashwani Kadian	Thanigai Nallathambi	
Srinidhi Kayoor	Anita Patel	

The authors wish to gratefully acknowledge Michelle Malcher, Paul Needham, Scott Rotondo, George Csaba, Saikat Saha, Ashok Swaminathan, and Manish Choudhary, who contributed to previous editions of this Oracle Database Security primer.

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](https://blogs.oracle.com)

 [facebook.com/oracle](https://facebook.com/oracle)

 [twitter.com/oracle](https://twitter.com/oracle)

Copyright © 2023, Oracle and/or its affiliates. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

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