THIRD EDITION

Securing the Oracle Database

A Technical Primer

ORACLE

Table of Contents

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 (mainly encryption) while everybody else trusted the administrators, users, and computing environment to keep their data secure. In late 90s, when browsers opened up new possibilities for commerce over the net, companies began to understand the strong need for security. This new perspective led to SSL, network firewalls, and stronger 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 x 7. To address this new reality, we see many different security technologies protecting various layers of the stack all the way from the applications down to the chipsets. While the global security spend is expected to exceed $124 billion in 2019, hacks are becoming bigger and bolder, and impacting everything from customer and citizen databases to oil rigs 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 its hundreds of millions of records. The hackers try multiple techniques including attacks on the network, operating system, database, applications, and IT infrastructure. They also – many would even say primarily – target the users who have legitimate access to those systems. Sometimes it's 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 that they don't know where their sensitive data is, where the holes are, 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 still rely only on their perimeter security to keep them safe from attack, 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 the databases that contain most of the sensitive data assets. They also acknowledge that, while they would never be able to find and 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 are going 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, minimizing the attack surface, preventing the attacks, and detecting/alerting malicious behavior in databases. Industry analysts and security professionals recognize that the Oracle Database provides industry's most comprehensive security.

This book, authored by the Database Security Product Management team at Oracle, explains in simple terms the adversaries of today, how they exploit weaknesses, and how they get access to 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 possibly imagine, and it is crucial that you are prepared! Your data is your asset, but unless you protect it well, it could fall in wrong hands, and become a liability. Let's start by securing the source!

Vipin Samar
Senior Vice President, Oracle Database Security Development
Redwood Shores, California
September, 2019

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 Masters of Business Administration from the Rensselaer Institute of Technology.

Angeline Janet Dhanarani is Product Manager for Oracle Database Security, focusing on auditing and activity monitoring. She has been involved in many customer engagement activities over the years. 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.

Ashok Swaminathan is a Sr. Director of Product Management with overall responsibilities for the auditing area in the Oracle Database Security team. He has over 20 years of experience in the enterprise software market covering inbound and outbound product management. Prior to Oracle, he worked at SAP/Sybase, where he led PM activities for SAP HANA machine learning, Sybase ASE database, Database Auditing and other products. Ashok started his career as a software engineer in the hardware test automation field. Ashok has an MBA from The Wharton School, M.S in Computer Engineering from Univ. of Massachusetts, Amherst and a Bachelor's degree in Electrical Engineering from Indian Institute of Technology, Madras.

Chetan Vithlani is the Product Manager responsible for security assessment technologies in the Oracle Database group. Prior to joining the Oracle Database Security team, he was Product Manager for Identity and Security Operations Center. Chetan has 23 years of experience and started his career as Oracle Database Administrator with multiple Fortune 500 companies. He worked as Solution Architect with focus on security and supported multiple enterprise customers. Chetan received a BS in Statistics and Master's degree in Computer Science.

Manish Choudhary is a Product Manager for Oracle Database Security, currently focused on sensitive data discovery and masking technologies. He has eight years of experience in the security domain. Prior to joining the Oracle Database Security team, he worked with the Oracle Solaris group as a product manager for security and compliance and also as a security engineer on various security technologies. While with McAfee Labs, he worked on Host Intrusion Prevention System and Database Activity Monitoring. Manish has an MS in Security from Georgia Institute of Technology. His interest areas are data security and privacy, cryptography, cloud security, cyber-physical systems (IoT), and machine learning in security.

Michael Mesaros is Director of Product Management for Oracle Database Security Products and has over 20 years of experience in a variety of security areas. He is a 16 year veteran of Oracle, where he has managed products for collaboration, networking, directory services and identity management. In his career, Michael has also managed a wide 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 where he received 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.

Peter Wahl is the Product Manager for Oracle Transparent Data Encryption and Key Vault. Between 2004 and 2015 he helped to introduce Transparent Data Encryption to a world-wide audience and wrote numerous security articles as well as enterprise-grade tools that helped customers of prior database versions to migrate their data to encrypted tablespaces without downtime or data loss. 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.

Russ Lowenthal is Senior Director of Outbound Product Management for Database Security, focused on database encryption, access control, audit, and monitoring. Leveraging more than thirty years of experience in IT including database, UNIX systems, and network administration, he now 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).

Acknowledgements

The authors wish to gratefully acknowledge Michelle Malcher, Paul Needham, Scott Rotondo, Manish Choudhary, George Csaba, Pedro Lopes, Russ Lowenthal, Michael Mesaros, Saikat Saha, and Alan Williams who were the authors of the first and second edition of this book, Securing Oracle Database 12c: A Technical Primer.

The following individuals helped with the review and provided invaluable feedback: Pedro Lopes, Rich Evans, Sean Cahill, Phil March, Chi Ching Chui, Marek Dulko, William Howard-Jones, Chao Liang, Rahil Mir, Gopal Mulagund, Abhishek Munnolimath, Vikram Pesati, Sreekumar Kodunthirapully Seshadri and Rajesh Tammana.

The authors would like to especially thank Vipin Samar, Senior Vice President, Oracle Database Security, for helping and guiding them during the preparation and review of this manuscript.

1

Protecting Data

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 are often able to leverage stolen data for financial or political advantage for years after a breach has occurred.

Figure 1.1: Categories of sensitive data.

And where do organizations keep their sensitive data? At the end of the day, this data is stored and managed in databases. At one point, perimeter security solutions such as network firewalls were considered sufficient for protecting internal systems and repositories 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. In addition to the monetary and reputational losses arising from data breaches, organizations today operate in an increasingly stringent and fast-evolving regulatory landscape. The United States alone has nearly 20 national privacy and data security laws, with additional laws enacted at the state level. The European Union (EU) has harmonized data privacy laws across multiple member states with the General Data Protection Regulation (EU GDPR). Under this regulation, data breaches can lead to fines of up to four percent of a company's global annual turnover or €20 million, whichever is greater. The EU GDPR not only applies to organizations located within the European Union, but also organizations located outside of the European Union if they provide goods or services to EU residents, a fact that illustrates how new regulations are cutting across industries and geographies. Similar regulations are now cropping up across the world whether in Japan, Australia, India or Brazil.

THREAT ACTORS

To understand why a defense-in-depth approach to database security is important, it is necessary to understand the various actors who want your data and how they try to get it.

Threat actors can be broadly divided into two groups. We refer to them as "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. However, one class of threat actors often overlooked in media reports are insiders. These can include current or former employees, curiosity seekers, and customers or partners who take advantage of their position of trust to steal data. The prize for both of these groups includes personal data, financial data, trade secrets and regulated data.

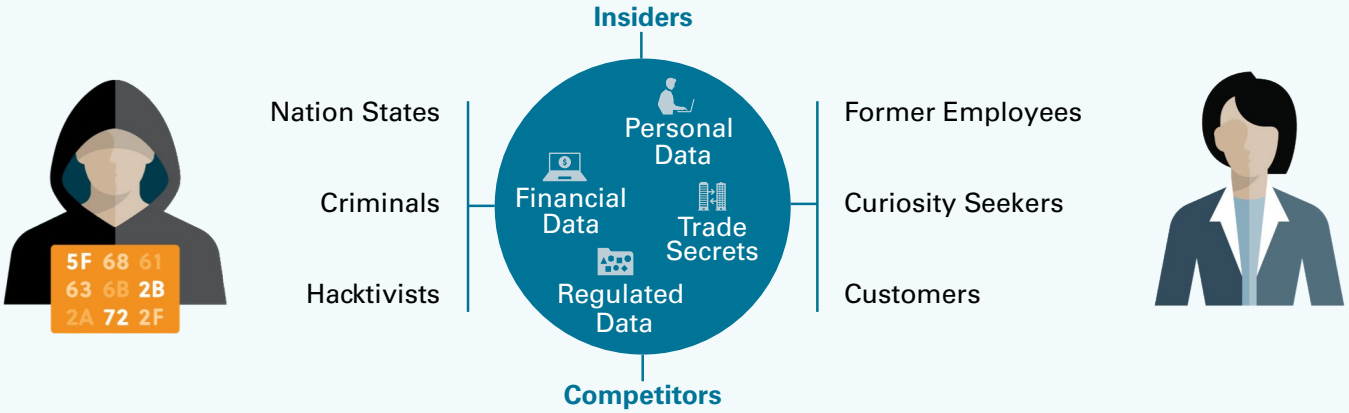


Figure 1.2: Threat actors

With a modern multi-tier application, threat actors have many approaches they can try in their quest for data including:

- 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.
- Exploiting weaknesses in applications with techniques such as SQL injection and bypassing application layer security by embedding SQL code into a seemingly innocuous end-user provided input.
- Escalating run-time privileges by exploiting vulnerable applications.
- Accessing database files that are unencrypted on the disk.
- Stealing archive tapes and media containing database backups.
- Copying live data from development and test systems where the data is typically not as well protected as in the production systems.

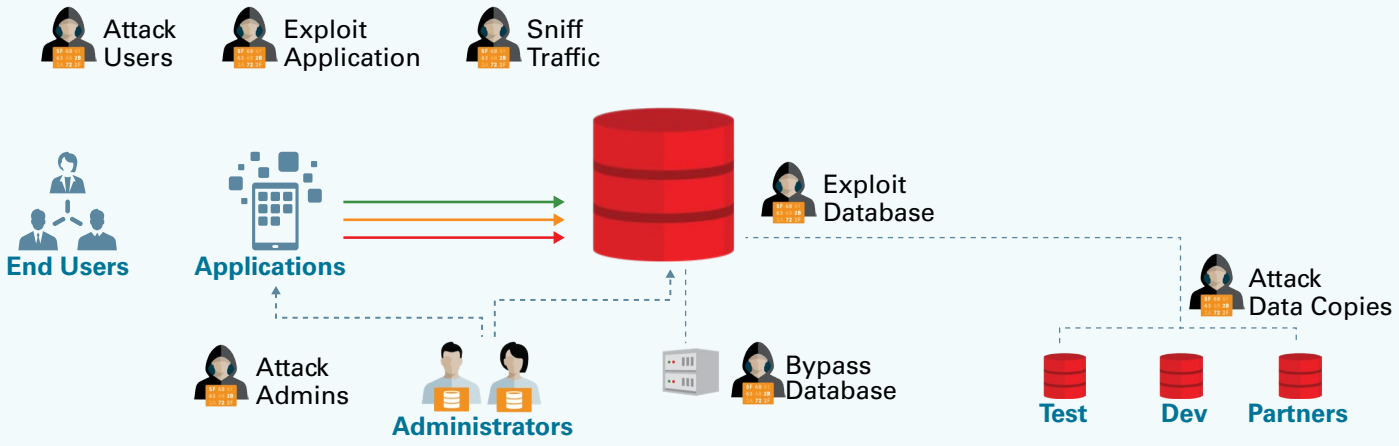


Figure 1.3: How threat actors exploit databases.

- Viewing sensitive data through applications that inadvertently expose sensitive data that exceeds what users should require to complete their tasks.
- Exploiting unpatched systems or misconfigured databases to bypass access controls.

RINGS OF SECURITY CONTROL

A well-structured data security solution must provide rings of controls to mitigate these various threat vectors and more. The best approach is a built-in framework of security controls which can be deployed easily to apply appropriate levels of security. Here are some of the more commonly used controls for securing database deployments:

- **Assessment Controls** help assess the security posture of a database, including the ability to monitor and identify configuration changes. They also help you assess how much sensitive data you may have in the database, and where.
- **Preventive Controls** block unauthorized access to data by encryption, redaction, masking, and restricting access to data by unauthorized users.
- **Detective Controls** monitor user and application data access, allowing administrators to detect and block threats and support compliance reporting.
- **Data Controls** enforce application-level access 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 that only authenticated and authorized users have access to their data.

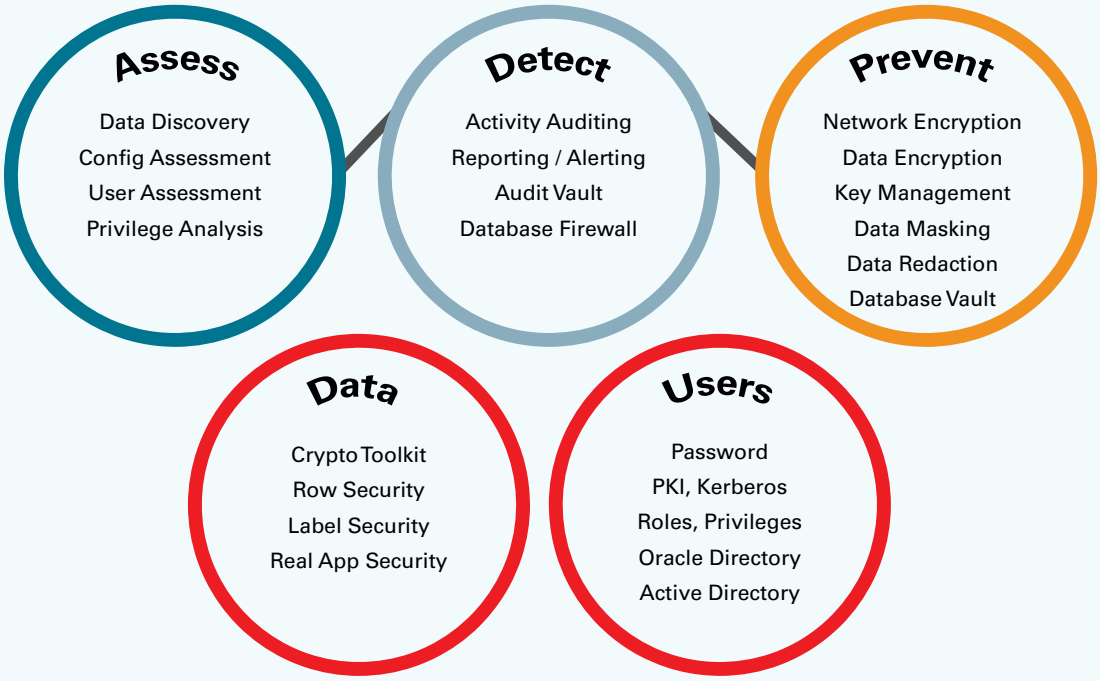


Figure 1-4: Rings of Security Controls for Oracle Database

Many organizations today are migrating their workloads to the cloud and embracing new, agile deployment models. While cloud deployments can provide a higher degree of infrastructure security, they also change trust boundaries and can expose data to new threats. As a result, data security controls need to scale and work seamlessly across on-premises, private cloud, public cloud and hybrid cloud environments.

ROADMAP TO DEFENSE IN-DEPTH

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.

Chapter 2 – Database Authentication and Authorization provides an overview of how users are authenticated and managed in the database, along with how roles and privileges are used to control access to data in the Oracle Database.

Chapter 3 – Enforcing Separation of Duties describes how to protect databases from a variety of inside and outside threats, and how Oracle Database Vault enforces this by limiting access to data by privileged users.

Chapter 4 – Data Encryption and Key Management shows how Oracle encrypts both data at rest and data in motion, and provides scalable key management.

Chapter 5 – Discovering Sensitive Data explains why understanding your sensitive data, and it's location, are critical to protecting it.

Chapter 6 – Masking Sensitive Data shows how to limit exposure to sensitive data in test, development and production environments.

Chapter 7 – Auditing Database Activity explains the need for tracking activities for all key database users using the audit capabilities built into the database. It describes how Oracle provides a scalable audit management and reporting solution for databases and operating systems.

Chapter 8 – Database Activity Monitoring with Database Firewall explains why organizations need to defend their production databases from SQL based attacks using database firewalls. It describes how Oracle provides a comprehensive solution for monitoring and protecting Oracle and non-Oracle databases.

Chapter 9 – Data-Driven Application Authorization shows how developers can leverage a range of authorization mechanisms attached to the data right within the database ensuring strong consistent access controls for application data.

Chapter 10 – Evaluating Security Posture discusses why it is important to know the security profile of your users, data, and configuration before hackers do. It describes various Oracle tools to evaluate and monitor database configuration for potential security issues.

Chapter 11 – EU GDPR and Database Security describes how Oracle Database Security can help address requirements related to EU GDPR and other emerging data privacy regulations.

Chapter 12 – Securing Databases in the Cloud discusses differences between on-premises and cloud security, as well as how you can apply the controls discussed earlier to databases in the cloud.

Chapter 13 – Keeping Data Safe in the Cloud describes Oracle Data Safe, which combines a number of the most essential security controls into a single, unified cloud-based service. With Oracle Data Safe, organizations can perform security and user assessments, identify sensitive data, mask data for test, development and analytics, and collect and manage their database audit information.

Finally, we conclude by bringing all of these technologies together to show how they can be applied to provide an appropriate level of defense-in-depth security according to an organization's risk management and compliance requirements.

2

Database Authentication and Authorization

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

This chapter discusses how a strong authentication and authorization strategy helps protect the users of databases from attackers. It also explains how to manage the user accounts whether the accounts are managed locally within the database or with corporate directory services.

FLAVORS OF DATABASE USERS

All access to the database is through database user accounts, whether these are administrative users, application accounts, or regular users. Oracle supports different flavors of database users, each with different access rights to the database including:

- **Regular database users:** They are restricted to their schema containing their tables, views, indexes, and stored procedures. If hackers hack into their accounts, they would not only be able to view/update data within the user schema, but also access objects in other schemas that the user may be authorized to access.
- **Application accounts:** These are the database accounts used for running your applications, both commercial and homegrown. These accounts are similar to your regular database user accounts, but as applications need to run 24 x 7, their passwords are often stored on multiple middle tier servers. Any compromise in these database accounts can lead to loss of data for the entire application including data about the end-users.
- **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 analysts or business intelligence users:** These users 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 also gives them access to any sensitive data contained within the database (personal, health, corporate finance records, etc.) even though that access is not required to perform DBA tasks. DBAs essentially hold the keys to your data kingdom, and hence are typically fully trusted within their organizations. Unfortunately, this makes them hackers' prime target.
- **Security Administrators:** Many organizations have specialized DBAs perform the responsibilities of security administrators, including user account management, encryption key management, and audit management.

Given broad access to data within the database, hackers typically target the database users.

USERS – YOUR WEAKEST LINK

The easiest 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 DBAs in an organization (which are easy to find via social media channels such as LinkedIn), and steal their credentials.
- **Try passwords used on other compromised sites** – Many users use the same password across multiple applications or multiple web sites, and if any of them get compromised, attackers can try those passwords to attack your database.
- **Find hardcoded database connection information** – Applications or users frequently connect to a database using embedded database usernames and passwords or store these credentials in a clear text configuration file. Compromising these accounts allows hackers to exfiltrate, modify, or delete any data that the account can access.
- **Use default or published passwords** – Hackers try common default passwords to connect as the user and use their privileges to access sensitive data.
- **Run brute force password attacks** – By trying various combinations of known passwords and their variations, hackers can break into database accounts with weak passwords when there are no limits on password retries. Without enforcement of complex passwords, some users may use easy to guess passwords such as 'password' and 'oracle123'.

These attacks are not necessarily sophisticated, and can be executed by "script kiddies", but they give hackers at least as much access as that particular user, and then some more. We will address most of these potential attacks later in this chapter.

DATABASE AUTHENTICATION METHODS

Oracle supports different means of authentication including passwords stored locally within the database or in centralized directory services. Users can also be authenticated by the operating system, using the `IDENTIFIED EXTERNALLY` clause when creating the user, or by various external authentication services including Kerberos, public key certificates, and RADIUS.

Passwords are used for one-way authentication of the user to the database, while Kerberos and public key certificates support mutual authentication, ensuring the user is indeed connecting to the proper database. While passwords are convenient to use, it is easier to compromise a user's password compared to their Kerberos or PKI credentials. We will describe later how to increase the security associated with passwords through stronger password profiles.

Once the user is authenticated, the user is then mapped to a schema on the database consisting of tables, views, indexes, and procedures, and then granted appropriate authorization through roles. When authenticating users with a directory service, users either get their own database schema (exclusive mapping) or get mapped to a shared schema (shared mapping).

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

Authentication Method	Schema Mapping	Role Mapping
Password	Schema is the same as user	Managed in database
OS	User mapped to a schema	Managed in the database or through OS groups
Kerberos	User mapped to a schema	Managed in database
Public key (PKI)	User mapped to a schema	Managed in database
RADIUS	User mapped to a schema	Managed in the database, or through RADIUS server
Oracle Directory Services	Managed in database and directory service	Managed in database and directory service
Active Directory	Managed in database and directory service	Managed in database and directory service

Table 2.1: Database User Authentication and Authorization Methods

MAKING USERS RESISTANT TO ATTACKS

With password authentication, users are expected to remember and use strong, long and complex passwords, and enter them when needed. Different passwords are supposed to be used for different databases and the sharing of database passwords is prohibited. However, both administrators and regular users are attracted to convenience and shortcuts, and hackers are ready to exploit such human behavior. We need to provide mechanisms that handle such behavior.

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 the password. In the past, a common way to provide passwords was to embed user names and passwords in the code or scripts. However, this increased the attack surface against the database and people had to make sure the scripts were not exposed. Also, if passwords were ever changed, changes to the scripts were required.

Oracle Wallets were introduced to solve this problem by storing various authentication credentials including passwords, private keys, and certificates in an encrypted file form. With wallets, the users need to remember only the wallet password, which then unlocks all remaining user credentials for multiple databases. Applications and database servers can also use the auto-login form of the wallet for 7x24 access to credentials.

With wallets, the database passwords are no longer exposed on command-line history or in clear text configuration files. Further, application code does not have to be changed whenever user names or passwords change. Wallets are also used for storing the master key for transparent data encryption. More on this in Chapter 4.

To configure use of the password stored in an Oracle wallet, set the `WALLET_LOCATION` parameter in the client's `sqlnet.ora.file`. Applications and users can then connect to the database without directly providing login credentials on the command line or hiding them in a configuration file.

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 share the same application username and password. If there are multiple DBAs, they also sometimes share passwords. Sharing passwords can be convenient but provides no accountability and makes it difficult to audit and investigate issues.

Proxy authentication can be used to hold individual administrators accountable. When authorized for proxy authentication to the application account, administrators first authenticate to the database with their own credentials, and then proxy to the application schema without having to know the password for the application schema account. For example, `alice_appdba` connects using her own password and then later 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 (discussed later in this book).

Problem: Poor Password Hygiene

Sometimes users use very weak and short passwords, making it easy for somebody to guess the passwords and break-in.

User profiles are used to create a common policy of password and resource authorization parameters for user accounts.

Each user account can be associated with a profile to simplify management of common policies across an organization. If no policy is specified when the user is created, the `default` policy is assigned. The sample policy 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;
```

The `ora12c_stig_verify_function` imposes a number of password requirements that correspond with the United States Department of Defense Security Technical Implementation Guide including a minimum length, minimum number of alphanumeric characters and that at least one special character is used (among other requirements).

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 policy instead of changing every user account.

Problem: Managing Multiple Accounts and Passwords

If users have accounts on multiple databases, they tend to either keep the same password, but then have to update the password on every database when any single database requires a password update. When an existing employee leaves the organization, the administrators need to remove the user account from all the databases. However, in practice those accounts live for a very long time making a very attractive target for hackers.

Through the use of global users with either Oracle Enterprise User Security (EUS) or Oracle Centrally Managed Users (CMU), the administrators can centrally manage users and roles across multiple databases within the organization's directory services. This also means users need to change their passwords only once in accordance to the password policy of the directory. More on this later in this chapter.

DATABASE AUTHORIZATION

Once a user is authenticated, the database then needs to decide what this particular user can do. In general, the user is permitted to connect to their own schema and access, modify, or delete all of the objects in their own schema. Through privileges and roles, user can get permission to perform a specific operation, such as updating a particular object in some other schema, or certain database-specific right. Such rights can be granted to subjects which may include individual users, roles, or programs acting on behalf of users.

Privileges

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

- Object privileges are fine-grained privileges to perform actions on specific data objects or execute specific 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 are typically used by database administrators for application or database maintenance. System privileges like `SELECT ANY TABLE` allows 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.
- Administrative privileges are used for specific tasks like database backup, encryption key management and database operations (e.g. startup, shutdown).

Certain maintenance activities like database upgrade and patching can only 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 absolutely necessary and be safeguarded.

ROLES

Roles are nothing but privileges grouped together, making it easier to grant or revoke them from multiple users. Direct object and system privileges can be grouped into task based roles. Roles can also be hierarchically grouped under other roles. This allows privileges to be grouped together in a task role and multiple task roles to be grouped together for an organizational role.

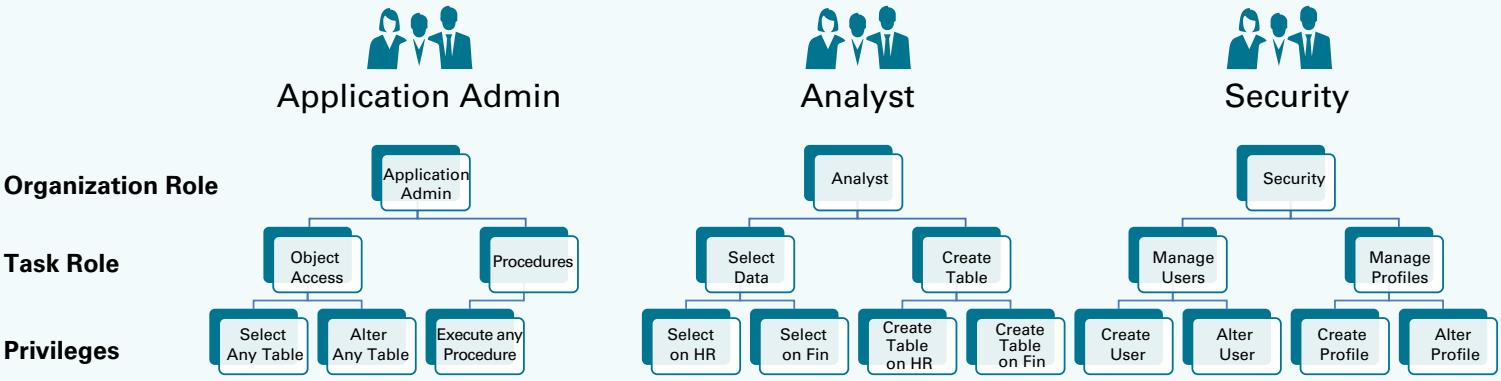


Figure 2.1: Sample Role and Privilege Hierarchy

Multiple users can be granted the same role to simplify privilege management for users. Using privileges and roles in this fashion makes it easy to add a new person to an organization. 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 simply moved instead of managing individual privileges or having to redefine all the organizational level roles. Roles simplify the management of user 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, which is a set of tables and views that provides information about the database, including definitions (metadata) about all objects and users in the database. The dictionary views listed below show the roles and privileges granted to users or roles.

If one were to analyze the information in these tables, they would see that the out-of-the-box Oracle DBA role is extremely powerful, with more than two hundred system privileges (depending on which database options were installed) including ALTER SESSION; CREATE, ALTER and DROP USER; CREATE and ALTER ANY TABLE; SELECT, INSERT, UPDATE, and DELETE ANY TABLE; EXPORT and IMPORT FULL DATABASE, and over eighty roles.

Dictionary Views	Contents
DBA_TAB_PRIVS	Object privilege grants to roles or users
DBA_SYS_PRIVS	System privilege grants to roles or users
DBA_ROLE_PRIVS	Role grants to users or other roles
DBA_ROLES	All defined roles

The Oracle DBA role shouldn't be modified and in most cases should not be used by the customer. Instead of using the default DBA role, each organization should create one or more custom DBA roles (i.e. ops_dba, backup_dba) that have the roles and privileges required for that job function.

MANAGING USERS CENTRALLY

In an enterprise with many users accessing a number of databases, users have difficulty in keeping every password compliant with each database's policy, and remembering different passwords for their various accounts. Further, it is difficult for administrators to manage unique accounts for each user in every database as each user needs to be provisioned separately along with their passwords. More importantly, each account needs to be de-provisioned when the user changes role and / or leaves the organization. Hackers often exploit such dead / orphan accounts to gain unauthorized access to data.

Oracle Enterprise User Security (EUS) centrally manages users and roles across multiple databases in Oracle Directory Services, which optionally can integrate with other corporate directories. EUS allows users and administrators to be authenticated by Oracle Directory Services using passwords, Kerberos and certificates. Upon connecting, 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.

Introduced in Oracle Database 18c, 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 (exclusive mapping) or they can share a schema through Active Directory groups. Active Directory groups can also map directly to database roles.

PROTECTING YOUR USERS FROM GETTING HACKED

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

- Impose a strong password profile for all of the users controlling the password strength, the inactivity period, number of passwords retries, etc.
- Implement strong authentication mechanisms with PKI or Kerberos.
- Use Oracle Wallets instead of writing down the password somewhere.
- Manage users centrally using directories to reduce the probability of orphan accounts when the employees leave the organization or change their roles.
- Create users with shared schemas if they do not need their own schema for their objects.
- Regularly examine if users have been granted additional fine-grained privileges to objects outside of their schema. Implement privilege analysis to ensure that users only have the privileges that they really need for the job. More on this in later chapters.

Database Administrator (DBA) accounts typically have wide access within the database – usually much more access than is actually needed just 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 and/or strong authentication mechanisms such as PKI and Kerberos.
- 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.
- Block access to user schemas using Database Vault. More on this in later chapters.
- Enforce a trouble ticket mechanism using Database Vault.
- Fully audit all DBA activities for accountability and tracking
- Use a Privileged Account Management (PAM) system when operating system accounts for root and database owner are needed for SYSDBA access during upgrade.

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 well protected using the following best practices:

- Use strong authentication mechanisms (PKI, Kerberos) to authenticate to the database.
- Use Oracle Wallets to store credential information.

- Use proxy authentication instead of sharing passwords across multiple administrators.
- To minimize direct login access for the application schema, create a schema-only account for the application objects and procedures, and 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 with no password or other ability to login. Starting with Oracle Database 19c, almost all Oracle accounts that are installed with the database are installed as schema-only accounts to prevent login to these privileged accounts by malicious users. 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.
- 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 not only take over the application, but also change the stored procedures and delete tables.
- Grant Application DBAs access to only the application schema objects, and not database-wide privileges even though this may be convenient.

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:

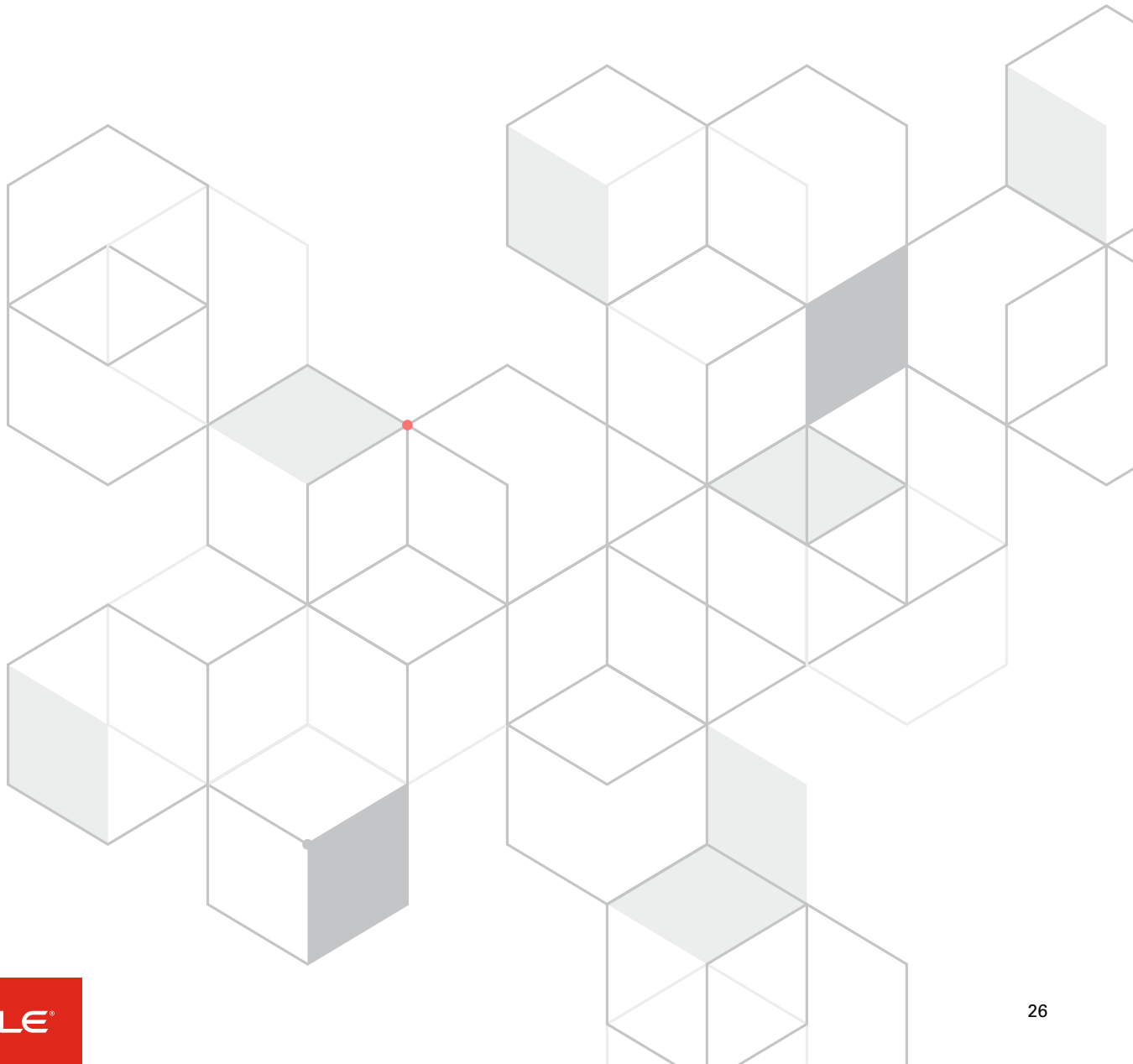
- Use strong authentication mechanisms (PKI, Kerberos) to authenticate to the database.
- Enforce proper separation of duty between database administrators/users and security administrators to prevent DBA/user accounts from being able to alter security records, create fake users and change security controls.
- Limit roles granted to the Security administrators such they don't have wide access to the sensitive data in the database in addition to managing the security controls.
- All of their activities should be fully audited for accountability and tracking.

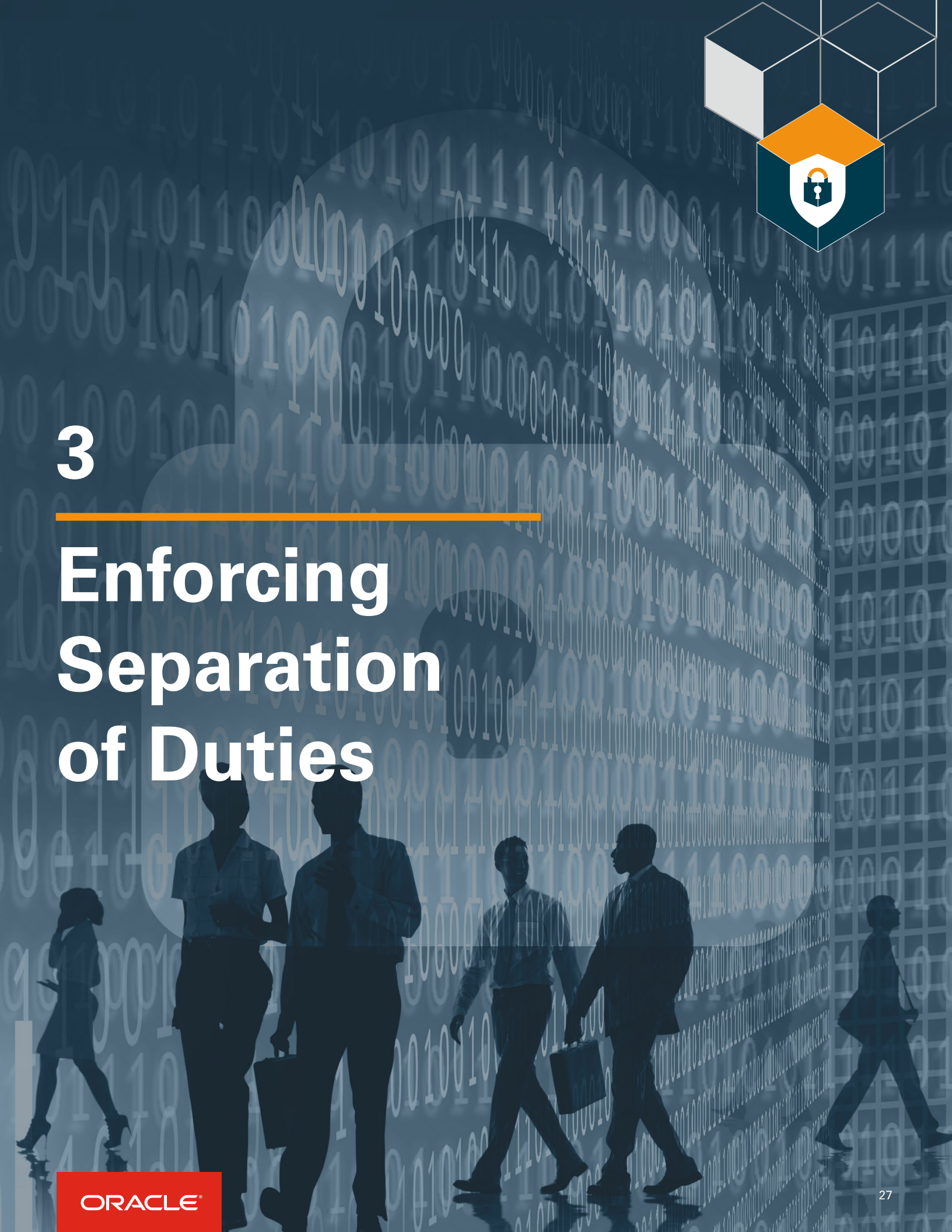
SUMMARY

As most attacks target the authorized users of the database, properly managing users and their authorization is the first step in protecting the database. Oracle provides many options for user authentication to meet customer requirements with central or local management of users. Fine grained access controls using privileges and task oriented roles give a great level of control that is easy to manage.

Upcoming chapters will show how tools can help identify areas of concern more quickly. For example, Database Security Assessment Tool (DBSAT) scans databases and provides information about user account configuration and privilege grants. Finding the right number 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.





3

Enforcing Separation of Duties

CONTROLLING PRIVILEGED USERS

Cybersecurity and regulatory concerns are driving strong security controls for insiders and privileged (administrative) users. Privileged users usually have powerful system privileges which give them unrestricted access to the database so they can easily manage the database 24 x 7 whether it is for performance management, diagnostics and tuning, upgrade and patching, database startup and shutdown, or database backup. However, this also gives them full access to all the sensitive data in the database such as salary, social security numbers, company financial forecasts and intellectual property. While most organizations trust their DBAs and power users, someone attempting to break into the system can – and frequently does – target over-privileged user accounts so that they can get full access to all the data.

This chapter introduces best practices such as the least privilege model and separation of duties for controlling access by privileged users' accounts. It discusses how Oracle Database Vault prevents privileged users from abusing their system privileges to access sensitive data and also to prevent accidental or malicious changes to the database. It continues with how Privilege Analysis helps implement the least privilege model.

CONTROLLING POWERFUL ADMINISTRATORS

DBAs essentially hold the keys to your data kingdom, and hence are typically fully trusted within their organizations. Unfortunately, this makes them prime targets of the hackers. Organizations also have a responsibility to protect their sensitive data from unauthorized access. A balance must be created between the thinking that DBAs need full access and the need to protect sensitive data.

An example of a powerful system privilege is the `SELECT ANY TABLE` privilege granted to DBAs. With this, the DBA can view almost any data on the database including salary, social security, phone numbers, corporate financial forecasts, intellectual property and other sensitive data. If a cyber-criminal is successful in compromising the credentials of a DBA, they get access to the sensitive data. They can now easily export this data into a file and exfiltrate it.

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

- **Separation of Duties (SoD) –** Here the administration 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, operations and security duties make 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.
- **Trusted Path** – In the likelihood that administrator credentials get compromised, access to the database should be allowed only if it follows a trusted path which can include IP address, time of day, authentication type, etc. Adding additional checks reduces the probability that an attack using compromised credentials succeeds.

- **Least Privilege** – The users should be granted only the minimum set of privileges needed to accomplish their intended tasks or functions, and no more. When granting privileges to a user or role, it is preferable to grant specific object privileges that are needed rather than broad system privileges that allow access to all objects in the database. Similarly, it is better to create roles that contain the least number of 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 close match to the tasks that the user needs to perform without granting extra privileges. The least privilege model helps reduce the attack surface for the database by limiting what the attacker could do even if the credentials for the privileged account were somehow compromised.
- **Named Users** – 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. Every DBA should have an individual named account with appropriately tailored privileges and roles.
- **Controlling Database Owner Account** – The `SYSDBA` administrative privilege provides full access to the `SYS` (Database owner) account. Use of 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.
- **Activity Audit** – Audit records provide an irrefutable record of actions on 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 Chapter 7.)

These security principles are well known – but difficult to enforce without tools such as Oracle Database Vault and Privilege Analysis.

Oracle Database Vault enforces separation of duties, prevents access to sensitive data regardless of system privileges through realms, and provides SQL command rules to prevent accidental or malicious execution of SQL commands. Trusted path rule sets enforce the context for authorized users to access realms and to execute SQL commands.

Privilege Analysis – a powerful analysis mechanism — dynamically determines privileges and roles that were used, and were not used for a given user. This helps in implementing least privileged named users and applications to reduce damage in case of a breach.

Oracle Database Vault and Privilege Analysis work together to make it easier for administrators to follow and enforce good security practices for their database.

LIMITING RISK EXPOSURE WITH SEPARATION OF DUTIES

By force of habit, administrators find it easy to use the super powerful `SYSDBA` privilege or `DBA` role to perform critical database operations such as startup/shutdown, backup, managing Data Guard, etc. Not only are you increasing your dependence on this all-powerful privilege or role, you are also increasing the probability of a mistake or a breach.

Many of these DBA operations can be done using 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 conferred by the SYSDBA privilege. Oracle Database 12c adds additional administrative privileges called SYSBACKUP, SYSDG, SYSKM and SYSRAC to enable database backups, Data Guard administration, key management, and RAC management, respectively. Each of these administrative privileges is associated with an operating system group (e.g.: DBA). Depending on one's organizational structure, one may assign all these administrative privileges to the same group (default) or assign each privilege to its own unique OS group.

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. This still does not prevent the administrators from accessing user specific data. Read on to see how to further enforce SoD.

CONTROLLING 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 the 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 all across the database.

What one needs is a set of operational controls within the database to enforce limits on privileged users from accessing sensitive schemas, tables and views.

Oracle Database Vault, first introduced in Oracle Database 9i, restricts privileged users or DBAs with system privileges such as SELECT ANY TABLE from accessing sensitive data. It introduces the concept of Realm, which is essentially a list of schemas or specific sensitive objects that only the object owner and those with direct object grants can access. Thus 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 do their DBA tasks.

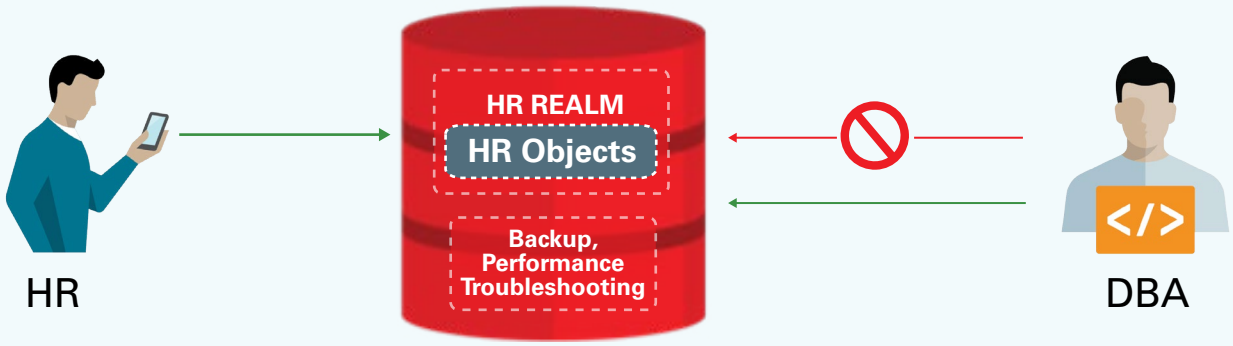


Figure 3.1: Oracle Database Vault uses realms protect the sensitive data

There are however other situations where you don't even want the owner or even somebody with the direct object grant to have access to the data. Thus, by default, nobody is allowed access until they are specifically granted access and only IF they are granted access by the security team for SoD purposes. They are also useful during patching and upgrades when an administrator needs to make changes during an application upgrade. 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 the sensitive tables and views and still allow access to update the procedures.

To implement the above use cases, Oracle Database Vault supports "mandatory realms" that block even the object owner or other users with direct object privileges from accessing the protected data. The only way to access such protected data is to be authorized to the mandatory realm. In this example, the application administrator is granted access to the regular Oracle Database Vault realm protecting the entire schema for the upgrade, but a mandatory realm is put around the sensitive data tables to prevent the administrator from accessing the sensitive data. This is safer than temporarily dropping all protection to the sensitive data objects or authorizing access to all the sensitive objects.

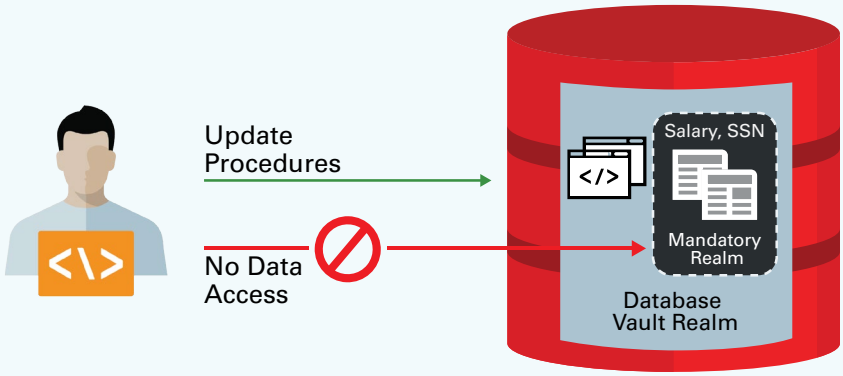


Figure 3.2: Mandatory Realms in Database Vault

The only user that can grant access to the data is the security officer, not the owner of the data. This simplifies identifying which users have access since the auditors don't need to traverse a potentially complex chain of direct object grants to different users and roles. All object authorizations are maintained as realm authorizations, managed by a security officer.

ENFORCING SEPARATION OF DUTIES WITH ORACLE DATABASE VAULT

When Oracle Database Vault is configured, additional roles are created for separation of duties including account management and Oracle Database Vault management.

- **Oracle Database Vault Account Manager Role** provides out of the box enforced SoD for a security role to create and manage users in the database along with their password profiles.
- **Oracle Database Vault Owner Role** is the only role that allows a security DBA to create, modify and manage Oracle Database Vault security controls including the ability to manage realms and add realm participants.

The DBA role under Oracle Database Vault does not have the ability to manage users or manage Oracle Database Vault controls out of the box. This prevents a common attack vector 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 leaves the original compromised DBA account available for future malicious acts.

A separate role to manage the Oracle Database Vault security controls ensures separation between operational DBAs and the security staff. This prevents either the DBA or security DBA from disabling security controls and stealing sensitive data using a single account. Since these account management and Oracle Database Vault security control tasks are controlled by roles, organizations can enable Oracle Database Vault and implement separation of duties over time by initially granting these roles to existing DBAs.

Common DBA tasks such as performance tuning, memory management, backup, and others can still be done by the DBA team (infrastructure or application DBAs). Certain DBA tasks which may expose the DBA to sensitive data like export and jobs scheduling will require an additional authorization step by the security team.

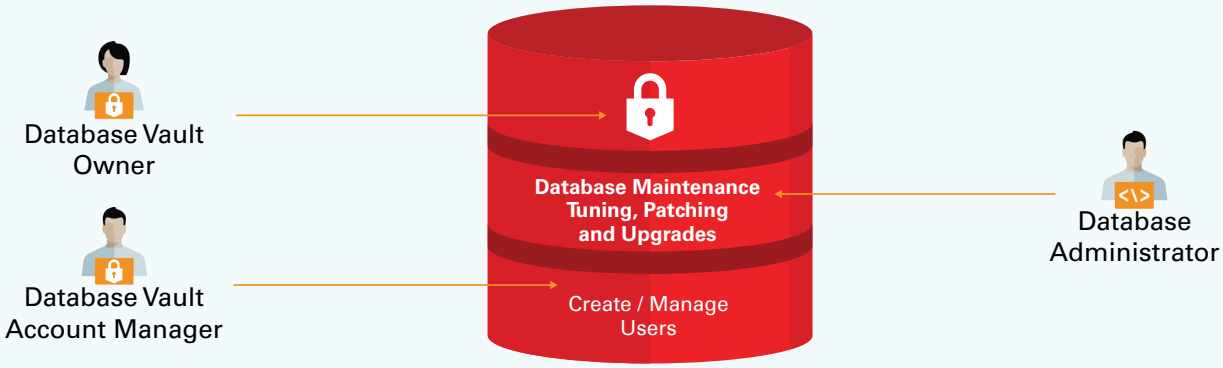


Figure 3.3: Database Vault Separation of Duties

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

CONTROLLING DATABASE COMMANDS WITH ORACLE DATABASE VAULT

Privileged users on production systems should not be dropping tables, changing system parameters, or modifying the 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.

Oracle 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 they can dictate the time of day when commands can be run along with the permissible

client IP address and other context data. For example, organizations can limit DBAs to run certain `ALTER SYSTEM` commands only from their desktops at work and only during working hours to prevent unauthorized remote access off-hours. Another example is to limit SQL commands like `CREATE, DROP` or `ALTER TABLE` to maintenance windows, require that any change has to be associated with a trouble ticket number, or ensure that two DBAs have logged in at the same time before the command can succeed.

LIMITING BREACH IMPACT WITH PRIVILEGE ANALYSIS

So far we have discussed how to limit access to sensitive data by privileged users using Database Vault. Let's consider the scenario where one of the application or user accounts does get compromised due to some reason. If that compromised account had access to many schemas or had many privileges, the hacker could exploit all of them. One way to reduce the loss is by restricting the privileges and the access this particular user has to the bare minimal they need to do their assigned set of responsibility. This set of rules and privileges is called as the "least privileged" set.

While there are dictionary tables and views to show which privileges and roles have been granted, it is much harder to determine which ones are actually needed. This is especially true in systems that have been in use for some time as privilege and role grants tend to accumulate and it is difficult to know when it is safe to revoke them. When a new administrator joins the organization, or is assigned a new task, it may be important to limit this user from doing anything else but the new task. Database accounts for applications also tend to accumulate more roles and privileges over time with patches requiring more privileges.

Oracle Database 12c introduced Privilege Analysis that captures privileges and roles actually used at run-time. After capturing the usage for some period of time, or running through all the test cases, the `DBA_USED_PRIVS` and `DBA_UNUSED_PRIVS` views show which privileges and roles have been used or not used, respectively.

The security administrator may decide to revoke the unused privileges from the user, or at minimum, audit the usage of unused roles and privileges to detect misuse. Privilege analysis helps you reduce the impact due to a compromised account.

OPERATIONALIZING ORACLE DATABASE VAULT

There are three key elements to consider when operationalizing Database Vault in your environment:

- **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 new enforced separation of duties**

Prior to creating the Database Vault realms, you have to know which data is sensitive. One could argue that all user data is sensitive, and that a realm should be put automatically on every user schema. Starting with Oracle Database 19c, Database Vault Operations Control blocks common users (infrastructure DBAs, for example) by default from accessing local data in pluggable databases (PDBs) in autonomous, regular Cloud, or on-premises environments. No realms have to be specifically created.

To determine which SQL commands to allow, one can look at their audit records to discover what critical SQL commands (i.e. ALTER SYSTEM, ALTER DIRECTORY,…) were run and the context they were run (IP address, program name, user name, etc.). Command Rules with trusted path can be created to constrain authorized commands to the time of day, incoming IP address and other rules to ensure changes are done in accordance with policy.

Most thought, however, has to be given to the Separation of Duties with Database Vault as it enforces the separation of duties. Just as the initiator of Purchase Orders cannot be the same person who approves Purchase Orders, Separation of Duties with Database Vault enforcement improves security, and reduces chances of a breach. Some organizations take multiple steps before they implement full SoD. For example, since the SoD is divided by the new Database Vault roles, the new roles can be granted to the same DBA(s) initially and then steps can be taken towards separating SoD. This allows some controls in Database Vault to be implemented quicker and for SoD changes to be spread out over a longer time frame.

DBA staff can be hesitant to turn on preventive controls in case applications stop functioning or any administrative scripts stop working. Oracle Database Vault introduced Simulation Mode with Database 12c Release 2 so one can verify Realms and SQL command rules without having any impact on application run-time and operations. Simulation Mode only logs violations instead of blocking them, so a full end-to-end regression test can be done without stopping. Oracle Database Vault controls can then be updated based on the simulation log data.

Finally, audit records need to be captured and alerted on when a violation occurs. These audit records are high value since any alert either indicates an initial probe by a malicious user or additional training for existing users on accessing sensitive data.

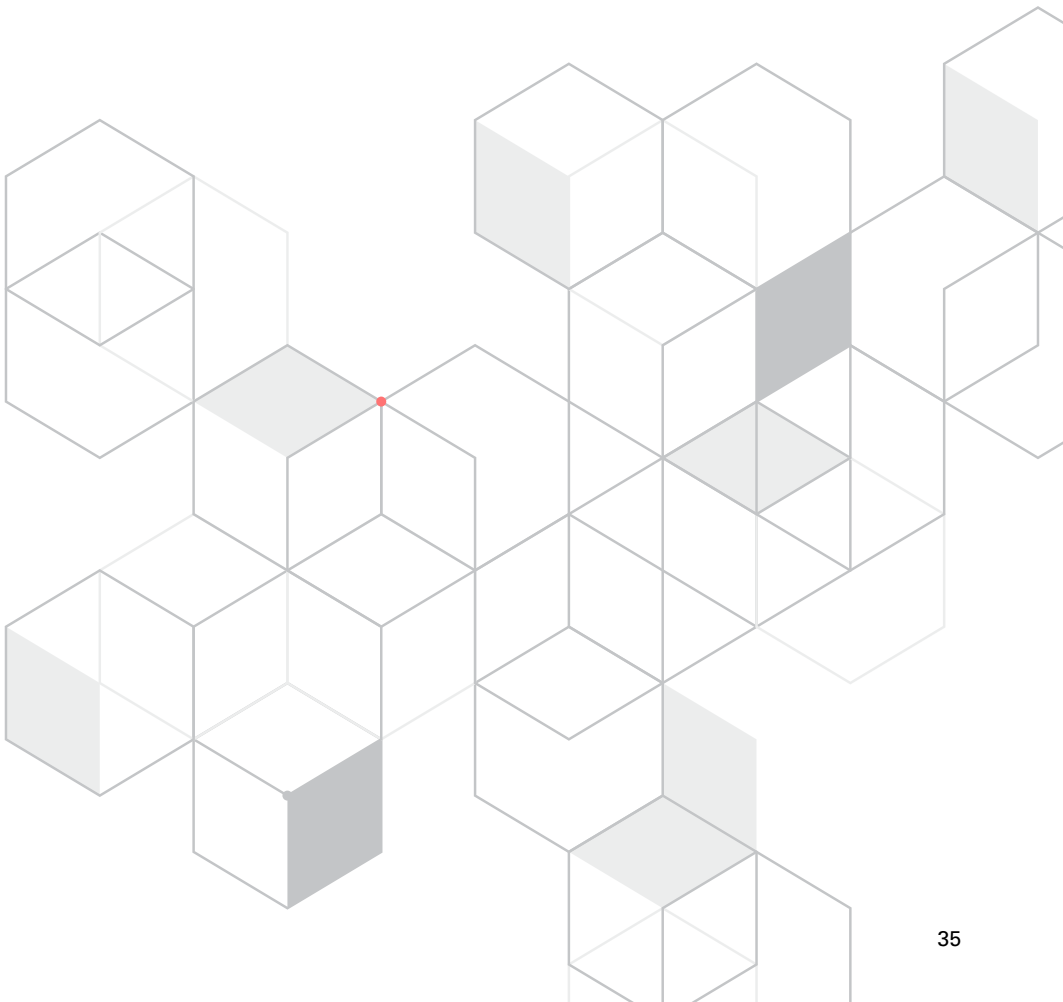
ORACLE CLOUD AND APPLICATIONS

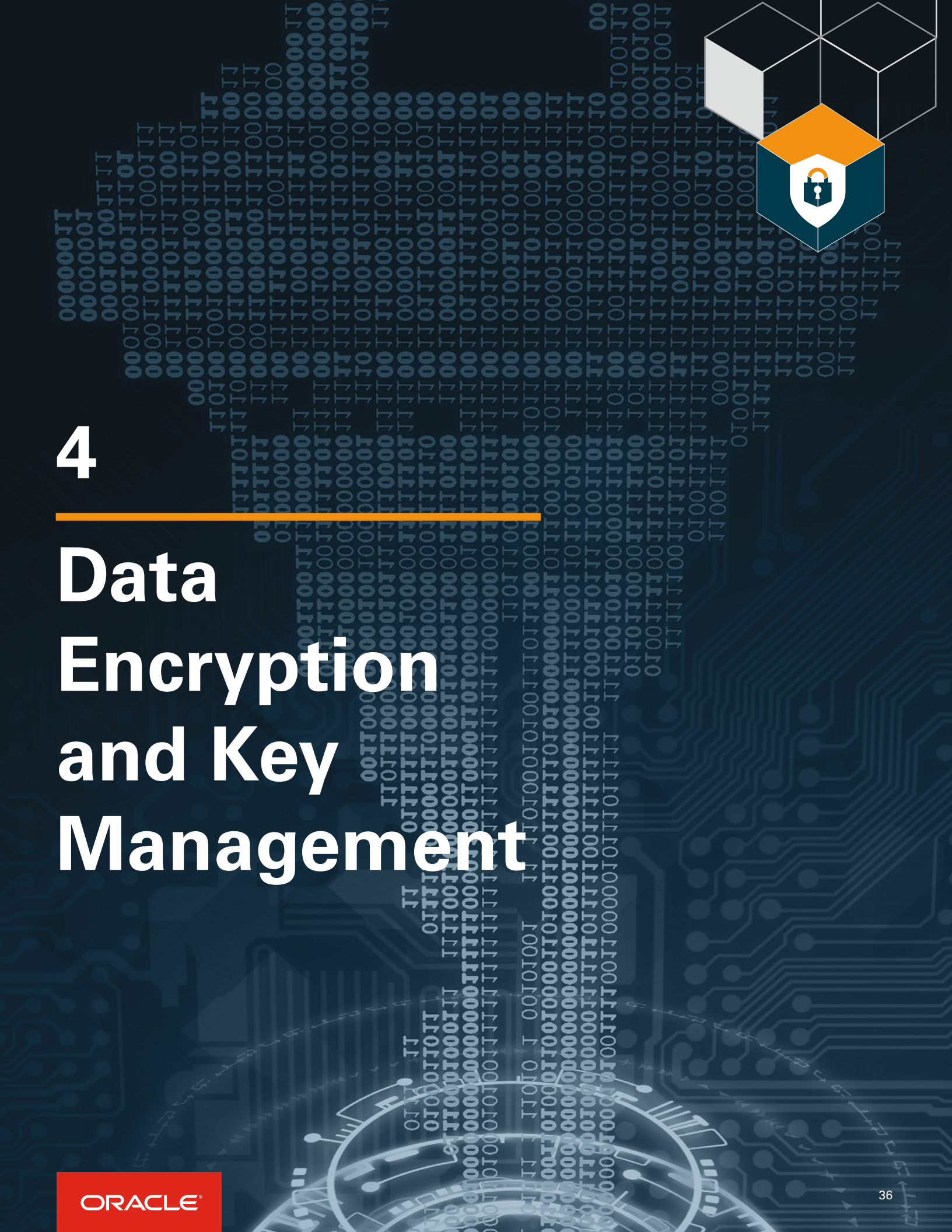
A number of Oracle applications – both on the Oracle Cloud and on-premises – have been certified to work with Oracle Database Vault, including Oracle Fusion Human Capital Management, Enterprise Resource Planning Cloud Services, Oracle E-Business Suite, and Peoplesoft. Please review the certification matrix on Oracle Support for more information about on-premises certifications and the Oracle Cloud website for Cloud support.

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 SQL command rules protect sensitive data and sensitive database operations from privileged or compromised insiders.

By implementing Separation of Duties and Least Privilege, we can minimize the losses from compromised accounts. While a single administrator may want to perform multiple functions for convenience, the ability to divide these duties among multiple users using Oracle Database Vault greatly improves security.





4

Data Encryption and Key Management

WHY ENCRYPT DATA?

The strongest defense imaginable is useless if an attacker can just go around it. Attackers are not limited to attacking where defenses may be the strongest; they will bypass the front door protected with a deadbolt and look for a less secure back door or even an open window. Database authentication and authorization ensures that data is only available to those who are authorized, and not accessible to anyone else. If the attacker is unable to gain access via the database (i.e. the front door), then the attacker will try to circumvent database access controls and attack the underlying data files, backups, or even just copy the data as it travels across the network. Attackers can gain access to the system as a privileged operating system user and directly read the database files (.dbf) – bypassing database controls. Attackers can also go after the database backup files which are often shipped to a remote location.

One easy way for attackers to gain access to data is to intercept the data as it travels over the network, such as between a client and a database server or between database servers. On many networks it is relatively easy for an attacker to capture network traffic and then extract whatever information was transmitted between the two systems since many internal network connections are left unencrypted.

Encryption is the best technique for protecting data in these situations as it renders the data unintelligible to those who access it directly. With encryption, the problem of protecting a large amount of data is reduced to a simpler problem of protecting an encryption key. As long as the attackers do not have the key, any encrypted data they manage to intercept provides nothing useful. Encryption is also frequently required in order to comply with regulations or security standards regarding sensitive or personally identifiable information (more in Chapter 11).

This chapter explains how encryption protects data in motion and at rest. It also explains some of the considerations and options available for securely storing the encryption keys that ultimately protect the encrypted data.

ENCRYPTING DATA IN MOTION FOR DATABASES

The ability to encrypt data on the network, using either Transport Layer Security (TLS) or Oracle Native Encryption, is a standard feature of the Oracle Database. This feature can be configured to provide both confidentiality (to prevent others from reading data sent over the network) and/or integrity protection (to prevent others from modifying or replaying the data). Additionally, TLS can provide client and server authentication. The network data protection settings for each Oracle client and server are configured in their respective `sqlnet.ora` file. The applicable settings for each system include the list of cryptographic algorithms it supports as well as options for how connections are negotiated. When the encryption and checksum settings are turned on within a database server, all clients automatically follow the configuration settings in the server.

ENCRYPTING DATA AT REST IN DATABASES

When information is written to the database, it is stored in files, either on a local file system or in some other form of storage (ASM for example). Encryption of this data at rest ensures that an attacker cannot read the information directly from those files.

There are multiple ways to encrypt the data: at the application layer, at the file or volume layer, or at the database layer. Irrespective of the layer where encryption is done, the key considerations include security, performance, simplicity of installation and use, transparency to existing applications, migrating data from plaintext to encrypted form, patching, and key management.

Application-level-encryption is driven from the application code where it encrypts the data before storing it in the database. This not only requires each application to know how to encrypt and decrypt within different parts of the process flow, it also has to manage encryption keys and securely store them somewhere. If multiple applications share the information in the same database, they would need to collaborate to encrypt and decrypt the same data in the same way. Encryption in the application tier also adversely limits the types of queries that can be run outside of equivalency searches on encrypted columns. For example, common analytic queries that match against data ranges or computed values will not work. Further, application tier encryption does not benefit from Oracle Database In-Memory and Exadata high performance architecture. 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. Another problem that needs to be addressed is that the database might no longer be able to use indexes, which can cause severe performance problems.

File or volume encryption, while convenient, has a number of disadvantages when used to protect tablespace data in Oracle databases. First and foremost, the volume encryption software cannot distinguish between a legitimate and an illegitimate user. This means the Oracle Database running as user 'oracle' and a human (malicious) user who also logs in as 'oracle' would both have access to the decrypted data. The other big disadvantage is the overall inefficiency of the system. For example, in order to transfer redo logs from a primary to standby database, first the data has to be decrypted at the primary, encrypted while being transported across the network, decrypted at the destination, and then re-encrypted for storage on the standby system. Data cannot be just copied to the receiving system because both source and target are encrypted with their own encryption key. Finally, in the case of third-party encryption solutions, there is a potential for introducing system instabilities and upgrade issues through invasive operating system and/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.

These two techniques are not optimized for the database encryption use case. What we need is a mechanism that improves security and reliability, while minimizing the management burden

TRANSPARENT DATA ENCRYPTION IN ORACLE DATABASES

Transparent Data Encryption (TDE) for Oracle databases encrypts data within the database tier. The encryption is transparent to authorized applications and users because the database automatically encrypts data before it is written to storage and automatically decrypts it when reading from storage. Authorized applications that store and retrieve data in the database only see the decrypted (or "plaintext") data. Because the encryption and decryption takes place automatically, this is not an access control mechanism for database users, but rather a way to prevent privileged operating system users,

network and storage administrators, or someone masquerading as them, from bypassing the database controls to access the data directly. Authorized database users and applications do not need to present the decryption key when they process encrypted data. Instead, the database enforces the access control rules described in the previous chapters and denies access if the user is not authorized to see the data. TDE can encrypt both application table columns, and tablespaces with all application data. With tablespace encryption, you do not need to keep track of which columns to encrypt and column characteristics such as indexes and constraints.

Creating an encrypted tablespace is easy, as shown here:

```
CREATE TABLESPACE investigations
DATAFILE '$ORACLE_HOME/dbs/investigations.dbf'
ENCRYPTION USING 'AES256' DEFAULT STORAGE (ENCRYPT);
```

TDE automatically leverages special instructions in Intel (AES-NI) and SPARC CPUs to accelerate cryptographic operations. TDE tablespace encryption seamlessly integrates with the performance optimizations built into Oracle's engineered systems. All Oracle engineered systems use CPUs with hardware crypto acceleration, and TDE tablespace encryption does not interfere with Exadata Hybrid Columnar Compression (EHCC) or Smart Scan technology.

Transparent Data Encryption leverages a "FIPS Inside" approach to provide secure and validated cryptographic services used by the database. This approach is based on the TDE use of FIPS 140-2 validated cryptographic module.

With TDE, sensitive data remains encrypted throughout the database, whether it is in tablespace storage files, temporary or undo tablespaces, or other files such as redo logs. Also, TDE can encrypt entire database backups and Data Pump exports. Oracle Recovery Manager (RMAN) and Data Pump Export / Import both integrate with TDE encrypted data.

MIGRATING CLEARTEXT DATA TO ENCRYPTED TABLESPACES

TDE has now become one of the most well-established ways to ensure that data-at-rest remains encrypted. However, most customers have terabytes of clear text data in their existing databases, and they need some way to encrypt them quickly without any downtime.

Online tablespace encryption, introduced with Oracle Database 12c Release 2, enables zero downtime during migration from plaintext to encrypted data, or while converting from one encryption algorithm to another. However, it temporarily requires extra storage of the same size as the processed tablespace. An existing tablespace can easily be encrypted online with a command similar to:

```
ALTER TABLESPACE online_accounts ENCRYPTION ONLINE using 'AES256' ENCRYPT;
```

There are two offline encryption methods that do not require additional storage. Both Offline encryption methods encrypt with AES128. These require the tablespace to be offline, or the database in mount mode. In Oracle Database 12.2.0.1 and later, a single tablespace can be encrypted with a command similar to:

```
ALTER TABLESPACE court_documents ENCRYPTION OFFLINE ENCRYPT;
```

Alternatively, multiple datafiles that belong to any number of application tablespaces can be offline-encrypted in parallel. The datafile(s) of an existing tablespace in Oracle Database 11.2.0.4, 12.1.0.2 and later can easily be converted offline with a command similar to:

```
ALTER DATABASE DATAFILE '$ORACLE_BASE/oradata/investigations.dbf' ENCRYPT;
```

In Data Guard environments, where the standby databases are already in mount mode, offline encryption can be used to minimize the down time. The procedure is to turn off the managed recovery process, then encrypt the standby database's data files, perform a switch-over and encrypt the former primary database. In this scenario, the downtime for encrypting any size database is as short as the duration of one or two switch-overs.

In order to ensure all new tablespaces are encrypted by default, set the system parameter `encrypt_new_tablespaces to ALWAYS`. Note that such tablespaces are encrypted with AES128, but they can be later rekeyed to use AES256 or any other supported encryption algorithm by using Online Tablespace Encryption. The system parameter `encrypt_new_tablespaces` is available from Oracle Database 11.2.0.4, 12.1.0.2 and later.

ORACLE TDE TWO-TIER KEY ARCHITECTURE

The security of the encrypted data depends on maintaining the secrecy of the keys used for encryption. Proper key management is essential to keep an attacker from discovering the encryption key and gaining access to data. It is also vital to ensure that active keys as well as all previously used keys are safely stored, that keys are rotated periodically, and that old keys are securely archived to provide continued access to backup data sets. Many regulations, such as those developed by the Payment Card Industry (PCI), require physical separation between encrypted data and encryption keys, as well as periodic rotation of encryption keys to limit the exposure period if a single key somehow gets exposed.

Transparent Data Encryption (TDE) uses a two-tier key architecture to create and manage the keys used for encryption. TDE uses the master encryption key (MEK) to encrypt the internally generated data encryption keys (DEK) that are used to encrypt columns and tablespaces. DEKs are managed by the database behind the scenes. The two-tier key architecture simplifies key rotation and thus makes it practical to rotate master keys more frequently. When the MEK is replaced with a new MEK, there is no need to re-encrypt all data, only the much smaller set of DEKs. If DEKs need to be rotated in addition to the MEK, customers can use Online tablespace encryption to rotate DEKs.

Administrators perform key management operations (create, open, rotate, backup, etc.) either by using a series of SQL commands, or through Oracle Enterprise Manager. Beginning with Oracle Database 12c, users can connect to the database as SYSKM instead of SYSDBA to perform key management operations. The SYSKM administrative privilege supports separation of duties by providing the ability to perform all key management tasks but without the level of broad access granted to SYSDBA.

The MEK is separated from encrypted data, stored outside of the database, and directly managed by the database security administrator in a keystore. The keystore can be a local keystore like the Oracle wallet or in a centralized keystore such as Oracle Key Vault.

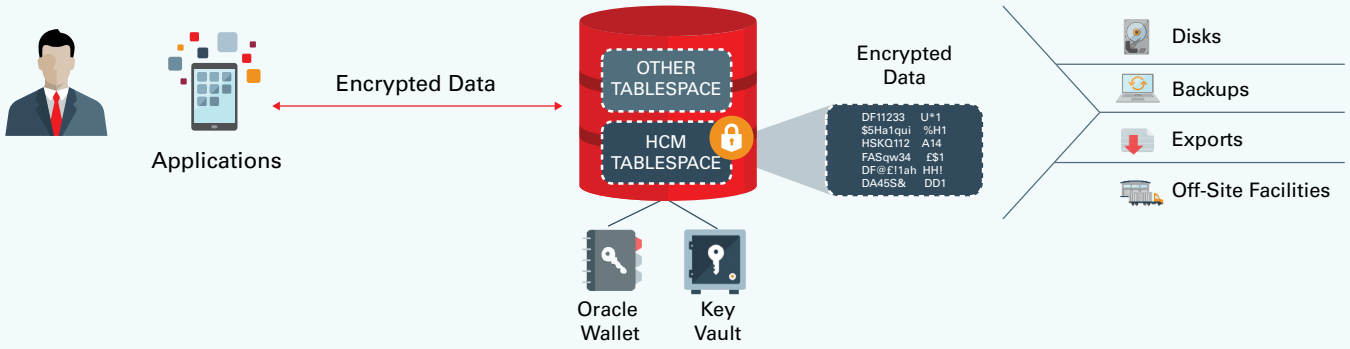


Figure 4.1: Oracle Transparent Data Encryption

LOCAL KEY MANAGEMENT WITH ORACLE WALLETS

The most important aspect of key management is storing the keys in a safe location and restricting access to authorized entities. By default, TDE stores MEKs in an Oracle wallet, a PKCS#12 standard-based key storage file. The content of the wallet is encrypted using a key derived from a password that must be provided by a user whenever the wallet is opened to access the keys inside. It is important that the password has a minimum of 12 characters and includes both alphanumeric and special characters. Note that if the wallet is somehow lost or corrupted or the password is forgotten, there is no way to access the data that is encrypted using the MEK in the wallet. Hence, it is important to back up the password protected wallet elsewhere. While wallets provide a key management solution for a few encrypted databases, customers are responsible for safeguarding and archiving wallets, as well as performing other key management operations.

CENTRALIZED KEY MANAGEMENT WITH ORACLE KEY VAULT

With the number of encrypted databases quickly rising, the management of local wallets becomes a management burden. Data can be at risk of compromise if wallets are lost or stolen, and if passwords are forgotten the data could become inaccessible. Furthermore, many regulations do not allow the encryption keys to be stored with the encrypted data, and many organizations require a clear separation of duties between DBAs and key management administrators. Centralized key management can also introduce a single point of failure, if the key management solution is not highly available and does not provide continuous key availability.

Oracle Key Vault (OKV) is the only enterprise-grade key management solution for your entire Oracle database real estate that addresses continuous key availability, security, and scalability requirements. Oracle Key Vault is a security-hardened software appliance that uses the industry standard OASIS Key Management Interoperability Protocol (KMIP) for communications. All network connections between OKV and database servers are encrypted and mutually authenticated using TLS.

Oracle Key Vault 18 provides continuous availability and high scalability for both read and write operations without any data loss. Customers can group up to 16 nodes to form a multi-master cluster that can be deployed across geographically distributed data centers. All nodes run in active mode, significantly lowering the total cost of ownership.

Databases can connect to any node in the Oracle Key Vault cluster to get encryption keys. Any updates to keys or changes to authorization rules are quickly replicated to all other Oracle Key Vault nodes so they are available on at least one other node providing zero data loss. If the Oracle Key Vault connection fails or an Oracle Key Vault node goes down for any reason, the database servers transparently failover to the nearby active Oracle Key Vault nodes for read / write operations without any down time, hiccups, or user intervention required.

OKV supports TDE keys across enterprise database deployments using Oracle Multitenant, Oracle Real Application Cluster (RAC), Oracle Data Guard, Oracle GoldenGate, and other Oracle products and technologies. In addition to TDE keys, Oracle Key Vault also manages Java Keystores, MySQL TDE master encryption keys, Solaris Crypto keys, and ASM Cluster File System (ACFS) volume encryption keys. Customers can periodically back up their local Oracle wallets and Java Keystores in the centralized Oracle Key Vault, or they can remove the wallet files from their environment entirely by using always-on OKV connections.

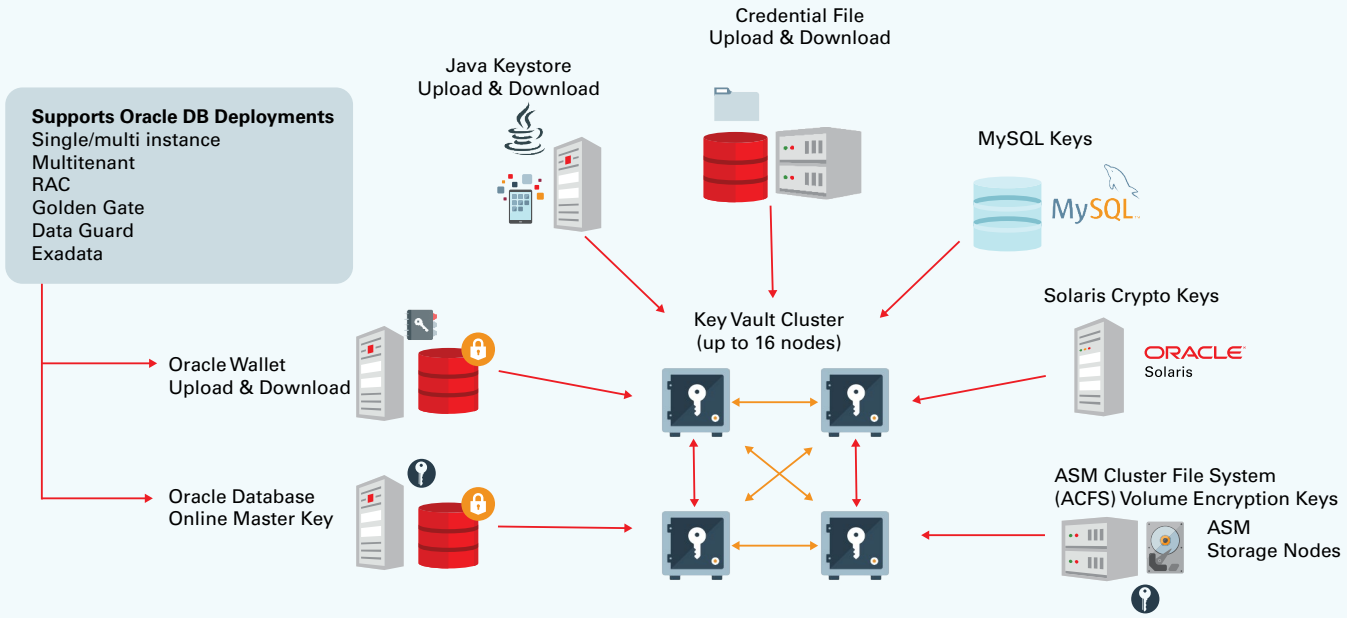


Figure 4.2: Oracle Key Vault use cases

OKV's RESTful API allows easy and secure on-boarding of large numbers of Oracle and MySQL databases in a self-service fashion. The OKV administrators prepare a simple set of scripts for the DBAs to run. Seconds later, the DBAs can begin encrypting their databases while the security team securely manages the MEKs within OKV. Oracle Database versions from 12.1.0.2 can register external keys uploaded into OKV as their MEK, enabling "bring your own key" for OKV-based TDE deployments.

Oracle Key Vault provides separation of duties for its administrators. The OKV System Administrator, Key Administrator, and Auditor role can be granted to individual administrators or the roles can be collapsed and be granted to fewer administrators. All administrative actions, access to key material, as well as automated onboarding of databases using the RESTful API, are audited. Optionally, Oracle Audit Vault can collect OKVs audit records.

In addition to providing key management for on-premises database deployments, Oracle Key Vault can also provide key management for database instances deployed in Oracle Cloud Infrastructure (OCI). Hybrid cloud key management lets users maintain control and visibility of master encryption keys regardless of where those databases are deployed.

Oracle Key Vault administrators can suspend an endpoint, denying access to encrypted data for all users and applications with a single click of a button.

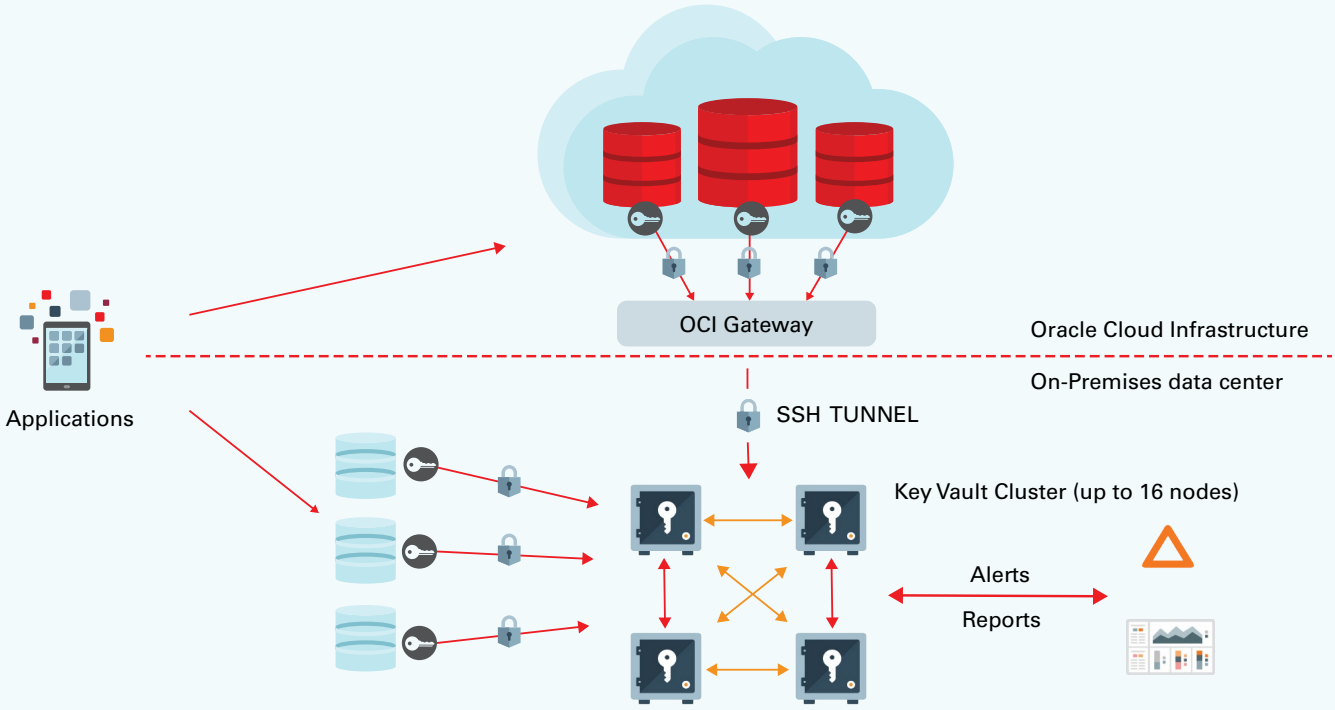


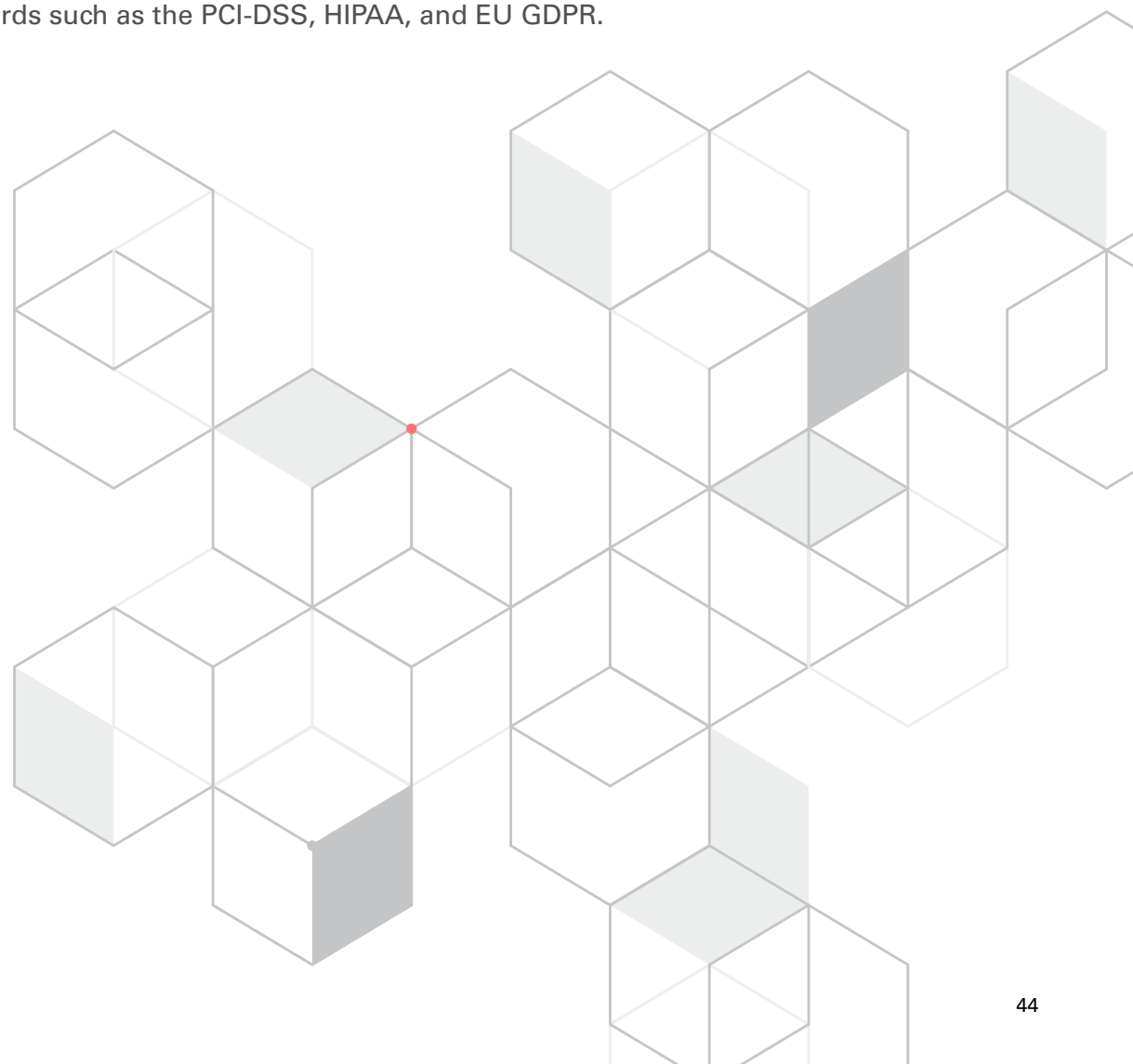
Figure 4.3: Oracle Key Vault supports your journey to the cloud

SUMMARY

Rising security threats, expanding compliance requirements, and cloud computing are just a few of the reasons why encryption has become critical for enterprises. Encrypting data-in-motion helps maintain 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, and simplicity of installation and use, transparency for existing applications, data migration from plaintext to encrypted form, patching, and key management.

TLS and Oracle native encryption are standard features of the Oracle Database for protecting data in motion. Oracle Advanced Security Transparent Data Encryption safeguards sensitive data against unauthorized access from outside of the database by encrypting data at rest.

TDE prevents privileged operating system users from directly accessing sensitive information by 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. Encryption and centralized key management are critical for complying with privacy regulations and standards such as the PCI-DSS, HIPAA, and EU GDPR.



5

Discovering Sensitive Data

The amount of data that organizations collect and manage is growing every day. From supporting operations, driving analytics and automation, to identifying market trends and deciding advertising budgets, enterprises need data to make smarter business decisions, improve user experience, and develop better products and services. Even government policies and programs are driven by data today. It makes data the most valuable resource and a necessity for every organization.

A significant percentage of the collected data is usually sensitive and personal. This data, if in the wrong hands, can be monetized by committing identity theft and financial frauds, selling military and trade secrets, or using it for future attacks. It makes data the prime target of cyber criminals. On the other hand, loss of sensitive data can be catastrophic for business, impacting companies' finances, reputations, customer trust, and competitiveness. The importance of data and growing security threats make it necessary to protect sensitive data. At the same time, the data privacy laws and standards such as EU GDPR, PCI-DSS, and HIPAA mandate protection of personal data. EU GDPR fines to both large and small organizations across industries attest to that.

But one can't protect sensitive data if one doesn't know it exists. Knowing what kinds of sensitive data reside in an application is the first step to implementing appropriate security controls to protect them. Identifying and locating sensitive data is hard to do and often does not happen reliably. Many organizations have difficulty discovering all of their sensitive data due to the complexity of large applications, which have often been developed over long periods of time. Frequently, the sensitive data outlast their original owners and exist beyond the knowledge of the person who currently owns and manages that data.

Sensitive data discovery tools automate the process of discovering and classifying sensitive data. This chapter introduces the basic elements of sensitive data discovery and gives an overview of the Oracle technologies that can be used to discover sensitive data: Database Security Assessment Tool (DBSAT), Enterprise Manager Application Data Modeling, and Data Safe.

SENSITIVE DATA

Sensitive data is, quite simply, any information that should not be made available to unauthorized people, whether they operate inside or outside the organization. Figure 5.1 shows some examples of sensitive data which is usually found in database applications.

Category of Sensitive Information	Examples
Identification Information	Name, Email Address, Phone Number, National ID, Driver License Number, Passport Number, Visa Number, Tax ID, Voter ID
Biographic Information	Date of Birth, Place of Birth, Address, Postal Code, Marital Status, Gender, Religion, Race, Citizenship, Nationality
IT Information	IP Address, MAC Address, Hostname, User ID, Password, Cookie, URL, Twitter Handle, GPS Location, IMEI Number
Financial Information	Credit Card Number, Card Type, Card Security PIN, Card Expiration Date, Bank Name, Bank Account Number, IBAN, SWIFT Code
Healthcare Information	Health Insurance Number, Insurance Provider, Healthcare Provider, Height, Weight, Blood Type, Disability, Test Result
Employment Information	Employee ID, Income, Stock, Job Title, Job Code, Organization Name, Department Name, Hire Date, Termination Date
Academic Information	Student ID, Grade, School, Academic Degree, Field of Study, Academic Level, Course, Attendance Record, Disciplinary Record

Figure 5.1: Examples of Sensitive Data

DISCOVERING SENSITIVE DATA

The most common way to discover sensitive data is to search for column names using keywords or search patterns defined as regular expressions. For example, the following search pattern can be used to discover columns containing Social Security numbers. It matches column names such as SSN#, 'SOCIAL_SECURITY_NO', and 'SSN_NUMBER'.

```
COLUMN NAME PATTERN = (SOCIAL.?SECURITY|SSN).?(NO|NUM|#)
```

Sometimes, column names are obscure and not easy to catch using search patterns. If a database has column comments, they can be searched to discover sensitive columns. For example, the search pattern shown below can be used to find keywords such as 'SSN No' and 'Social Security Numbers' in column comments.

```
COLUMN COMMENT PATTERN = (SOCIAL SECURITY|SSN) (NO|NUM)
```

Going one step further, the actual data in database columns can be checked to improve accuracy and discover additional columns. For example, the following search pattern can be used to search for Social Security numbers such as 333-93-2585 and 383368610.

```
COLUMN DATA PATTERN = [0-9]{3}[- ]?[0-9]{2}[- ]?[0-9]{4}
```

These data patterns can be used to check the first value in columns. As there can be data quality issues, a better approach is to check multiple values, say 10, and report a column as sensitive if a high fraction of these values match, say 60% or 80%. In addition, data types of columns can be checked to speed up the discovery process and improve result accuracy. For example, birth dates are more likely to be in date type columns and person names can be searched in varchar type columns only.

Data discovery tools automate the discovery process by combining some or all the above discussed factors. This multipronged approach helps minimize false positive and false negative rates. Oracle provides multiple sensitive data discovery tools which use sensitive types to drive data discovery. Sensitive types define patterns (regular expressions) that are used to search for sensitive columns. For example, a sensitive type consisting of the following patterns can be used to search for columns containing Social Security numbers.

```
[Social Security Number]
COLUMN NAME PATTERN = (SOCIAL.?SECURITY|SSN).?(NO|NUM|#)
COLUMN COMMENT PATTERN = (SOCIAL SECURITY|SSN) (NO|NUm)
COLUMN DATA PATTERN = [0-9]{3}[- ]?[0-9]{2}[- ]?[0-9]{4}
```

DISCOVERING SENSITIVE DATA USING DBSAT

Oracle Database Security Assessment Tool (DBSAT) is a stand-alone command line tool that helps discover sensitive data and analyzes database configurations, users, and security policies. It helps uncover security risks and provides recommendations to improve the security posture of Oracle Databases. DBSAT is available free of cost to all Oracle customers. This section covers the data discovery component of DBSAT. See Chapter 10 to learn about the other DBSAT capabilities.

One of the components of DBSAT is Discoverer, which helps identify sensitive columns in Oracle Databases using sensitive types. DBSAT Discoverer collects the metadata, i.e. column names and comments, and matches them against the patterns defined for sensitive types. It generates detailed data assessment reports in HTML and CSV formats.

Sensitive Types

DBSAT provides 125+ predefined sensitive types to help discover common sensitive and personal data. These sensitive types are arranged under sensitive categories such as identification, biographic, healthcare, financial, employment and academic data. Users can modify the predefined sensitive types and create new ones to meet specific requirements. DBSAT helps discover sensitive columns in English and seven major European languages: Dutch, French, German, Greek, Italian, Portuguese, and Spanish.

The following example shows a DBSAT sensitive type that can be used to discover columns containing person names. It has column name and comment patterns and belongs to "Identification Info" category and "Public IDs" subcategory.

```
[FULL NAME]
COL_NAME_PATTERN = (CUSTOMER|CUST|CLIENT|FULL|PATIENT|PERSON).?(NAME|NM)
COL_COMMENT_PATTERN = (CUSTOMER|CUST|CLIENT|FULL|PATIENT|PERSON).?NAME
SENSITIVE_CATEGORY = Identification Info - Public IDs
```

Sensitive Data Assessment Report

DBSAT Sensitive Data Assessment report helps understand what kind and how much sensitive data a database has, where it is located, and the associated risk. Figure 5.2 shows the summary section which provides information about the number of tables, columns, and rows identified as sensitive, grouped by sensitive category and subcategory. In the figure, * means number of unique tables with sensitive data, and ** means number of unique rows with sensitive data.

Sensitive Category	# Sensitive Tables	# Sensitive Columns	# Sensitive Rows
BIOGRAPHIC INFO – ADDRESS	7	18	244
FINANCIAL INFO – CARD DATA	2	2	256
HEALTH INFO – PROVIDER DATA	1	1	149
IDENTIFICATION INFO – PERSONAL IDS	3	3	356
IDENTIFICATION INFO – PUBLIC IDS	3	12	321
IT INFO – USER DATA	1	1	149
JOB INFO – COMPENSATION DATA	7	10	527
JOB INFO – EMPLOYEE DATA	12	25	569
JOB INFO – ORG DATA	7	8	412
TOTAL	21*	80	989**

Figure 5.2: Sensitive Category Level Summary

Each sensitive category has an associated risk level (high, medium, or low), which helps classify the discovered sensitive tables and columns. The report also provides recommendations on how to protect the sensitive data associated with a risk level. Figure 5.3 shows some recommendations for protecting data with high risk.

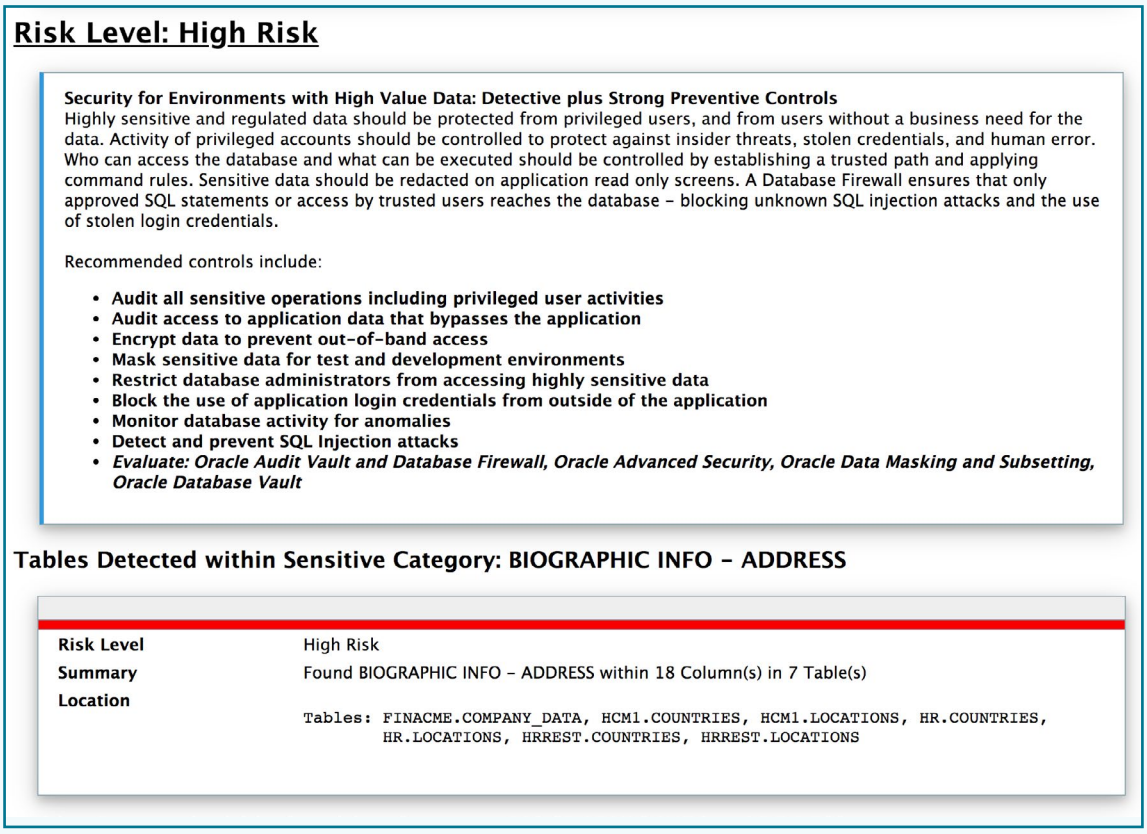


Figure 5.3: Recommendations for Protecting Data with High Risk ve Category Level Summary

As shown in Figures 5.4 and 5.5, the report also provides schema, table, and column level details along with some statistics to help understand the sensitive data present in a database.

Schema	Table Name	Columns	Sensitive Columns	Rows	Sensitive Category
FINACME	COMPANY_DATA	9	4	100	BIOGRAPHIC INFO – ADDRESS, IDENTIFICATION INFO – PERSONAL IDS
HCM1	COUNTRIES	3	1	25	BIOGRAPHIC INFO – ADDRESS
HCM1	DEPARTMENTS	4	1	27	JOB INFO – ORG DATA
HCM1	EMPLOYEES	11	8	107	IDENTIFICATION INFO – PUBLIC IDS, JOB INFO – COMPENSATION DATA, JOB INFO – EMPLOYEE DATA, JOB INFO – ORG DATA
HCM1	EMP_EXTENDED	3	3	107	FINANCIAL INFO – CARD DATA, IDENTIFICATION INFO – PERSONAL IDS, JOB INFO – EMPLOYEE DATA
HCM1	JOBS	4	4	19	JOB INFO – COMPENSATION DATA, JOB INFO – EMPLOYEE DATA

Figure 5.4: Table Level Summary

Schema Name	Table Name	Column Name	Column Comment	Sensitive Category	Sensitive Type	Risk Level
FINACME	COMPANY_DATA	CITY	--	BIOGRAPHIC INFO – ADDRESS	CITY	High Risk
FINACME	COMPANY_DATA	STATE	--	BIOGRAPHIC INFO – ADDRESS	STATE	High Risk
FINACME	COMPANY_DATA	TAX_PAYER_ID	--	IDENTIFICATION INFO – PERSONAL IDS	TAX ID NUMBER (TIN)	High Risk
FINACME	COMPANY_DATA	ZIP	--	BIOGRAPHIC INFO – ADDRESS	POSTAL CODE	High Risk
HCM1	COUNTRIES	COUNTRY_NAME	--	BIOGRAPHIC INFO – ADDRESS	COUNTRY	High Risk
HCM1	DEPARTMENTS	DEPARTMENT_NAME	--	JOB INFO – ORG DATA	DEPARTMENT NAME	Low Risk
HCM1	EMPLOYEES	EMAIL	This is the e...	IDENTIFICATION INFO – PUBLIC IDS	EMAIL ADDRESS	High Risk

Figure 5.5: Sensitive Column Details

The data discovery results from DBSAT can be used to implement appropriate security controls to protect sensitive data. The DBSAT report can also be loaded into Oracle Audit Vault and Database Firewall to add sensitive data context to the Data Privacy reports and can be used to define policies to audit access to sensitive data. See Chapter 7 to learn about Oracle Audit Vault and Database Firewall.

DISCOVERING SENSITIVE DATA USING ORACLE ENTERPRISE MANAGER

The application data modeling capability in Oracle Enterprise Manager enables users to quickly analyze databases to identify sensitive data present within an application and where it resides within the database schema. Similar to the DBSAT Discoverer, application data modeling examines the metadata, i.e. column names and comments, to discover sensitive columns. In addition, it also examines the actual data present in columns, which helps drive down false negative and false positive rates associated with the data discovery process. For example, application data modeling can help locate credit card and national identification numbers based on the column name, column comment, and the data itself. It also discovers referential relationships between application objects so that application integrity can be preserved during data masking and subsetting. The next chapter discusses data masking and subsetting.

Sensitive Column Types

A sensitive column type in Enterprise Manager consists of patterns for column name, comment and data. Figure 5.6 shows a sensitive column type that can be used for locating credit card numbers within an application schema. The Search Type is used to specify whether any one or all the three patterns must

match for identifying a column as sensitive. The "Or" option helps discover more columns while the "And" option helps minimize false positives.

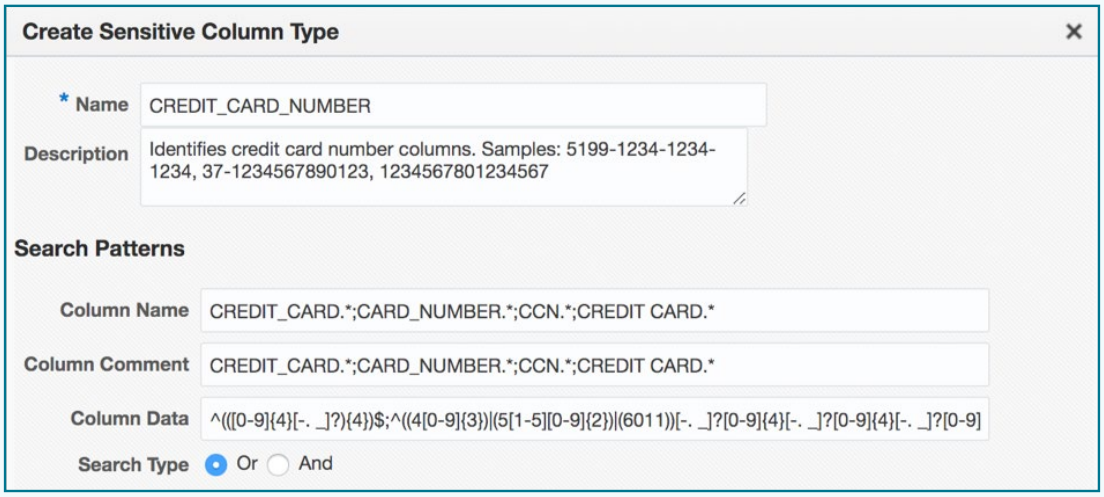


Figure 5.6: Sensitive Column Type Definition

Oracle provides a set of predefined sensitive column types, and users can create new sensitive column types to search domain specific data. Figure 5.7 shows some predefined sensitive column types available in Enterprise Manager.

Name	Description	Author
CREDIT_CARD_NUMBER	Identifies credit card number columns. Samples: 5199-1234-1234-1234, 37-1234...	Oracle
EMAIL_ID	Identifies email address columns. Samples: jsmith@comgmt.com, JackieSmith-4...	Oracle
IP_ADDRESS	Identifies IP address columns. Samples: 7.7.7.1, 78.78.78.12, 789.789.789.123	Oracle
ISBN_10	Identifies 10 digit International Standard Book Number columns. Samples: ISBN-...	Oracle
ISBN_13	Identifies 13 digit International Standard Book Number columns. Samples: ISBN-...	Oracle
NATIONAL_INSURANCE_NUMBER	Identifies National Insurance number (UK) columns. Samples: ZR 50 16 33 A, ZR...	Oracle
PHONE_NUMBER	Identifies phone number columns. Samples: 555-1212, (123)555-1212, 12355512...	Oracle
SOCIAL_INSURANCE_NUMBER	Identifies Social Insurance Number (Canada) columns. Samples: 884-099-029, 2...	Oracle
SOCIAL_SECURITY_NUMBER	Identifies Social Security number columns. Samples: 123-45-6789, 123456789	Oracle
UNDEFINED	Sensitive column type not defined.	Oracle
UNIVERSAL_PRODUCT_CODE	Identifies Universal Product Code columns. Samples: 1-23456-78901-2, 1 23456 ...	Oracle

Figure 5.7: Predefined Sensitive Column Types

Discovering Sensitive Columns and Referential Relationships

Application data modeling uses sensitive column types to perform pattern matching and identify sensitive columns. Users can review the discovered sensitive columns and, if required, add additional columns to the list manually. Figure 5.8 shows a list of discovered as well as user-defined sensitive columns.

Application	Object	Object Type	Column	Type	Source	Comment
HR	EMPLOYEES	Table	EMAIL	EMAIL_ID	Sensitive Column Discovery	Email id of the employee
HR	EMPLOYEES	Table	FIRST_NAME	UNDEFINED	User Defined	First name of the employee.
HR	EMPLOYEES	Table	LAST_NAME	UNDEFINED	User Defined	Last name of the employee.
HR	EMPLOYEES	Table	PHONE_NUMBER	PHONE_NUMBER	Sensitive Column Discovery	Phone number of the emplo
HR	EMPLOYEES	Table	SALARY	UNDEFINED	User Defined	Monthly salary of the emplo

Figure 5.8: Discovered Sensitive Columns

Application data modeling analyzes the referential relationships between application objects using the foreign key constraints defined inside the database. It also provides the option to automatically discover application-level referential relationships, which are not defined at database level inside data dictionary. 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 5.9 shows a list of parent-child relationships found inside the data dictionary.

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

Figure 5.9: Discovered Referential Relationships

Figure 5.10 presents the overall picture putting all this together. Application data modeling automatically discovers sensitive columns, database-defined referential relationships, and application-level referential relationships. Information about the discovered sensitive columns and relationships is contained in an application data model stored in the Enterprise Manager repository. This application data model can be used to implement security controls such as data masking and subsetting.

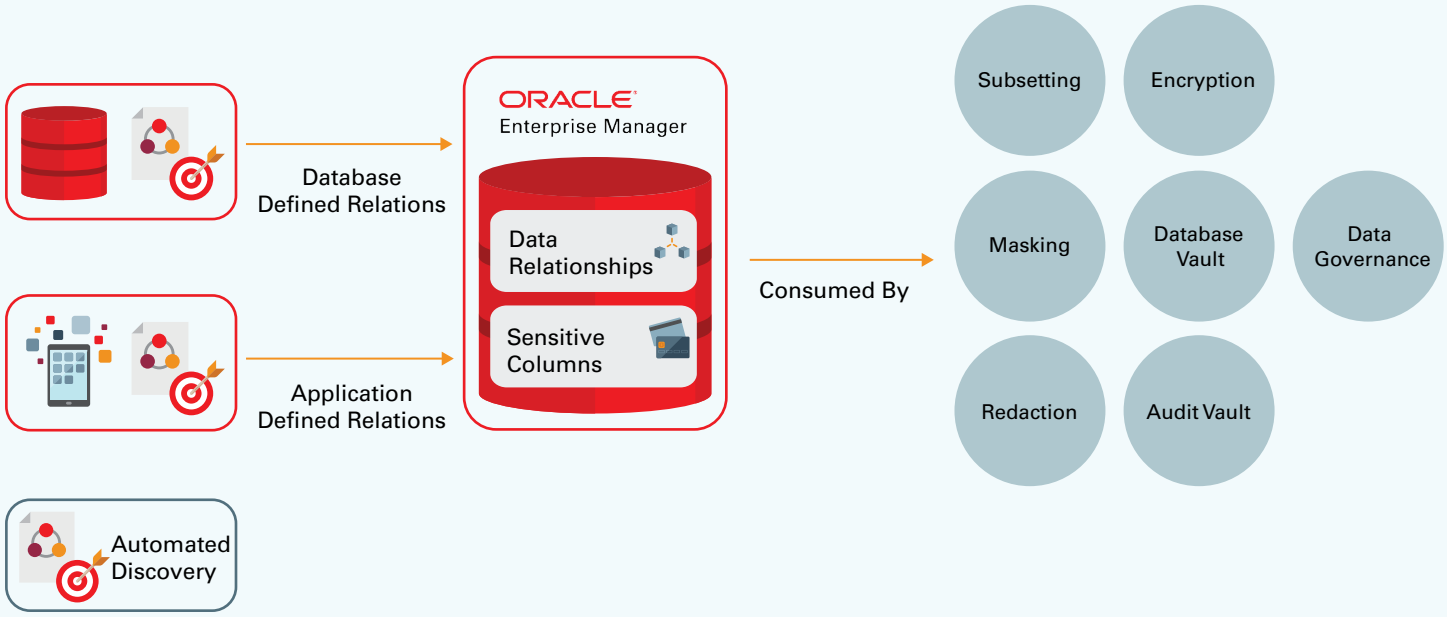


Figure 5.10: Overview of Application Data Modeling

DISCOVERING SENSITIVE DATA IN ORACLE CLOUD

Many organizations have either already moved to the cloud or are planning to in the near future. Cloud vendors already have a great share of responsibility by taking care of many security aspects, but consumers still need to protect and manage their data, users, and some security configurations. This makes sensitive data discovery equally important in cloud environments.

Oracle Data Safe, a data security cloud service, provides a comprehensive suite of security capabilities, including sensitive data discovery. Using an extensible set of 125+ sensitive types, it helps discover and classify sensitive data in Oracle Cloud databases. Figure 5.11 shows a sample data discovery report provided in Data Safe. See Chapter 13 for more information on how Oracle Data Safe can help meet data security requirements, including sensitive data discovery, in Oracle Cloud.

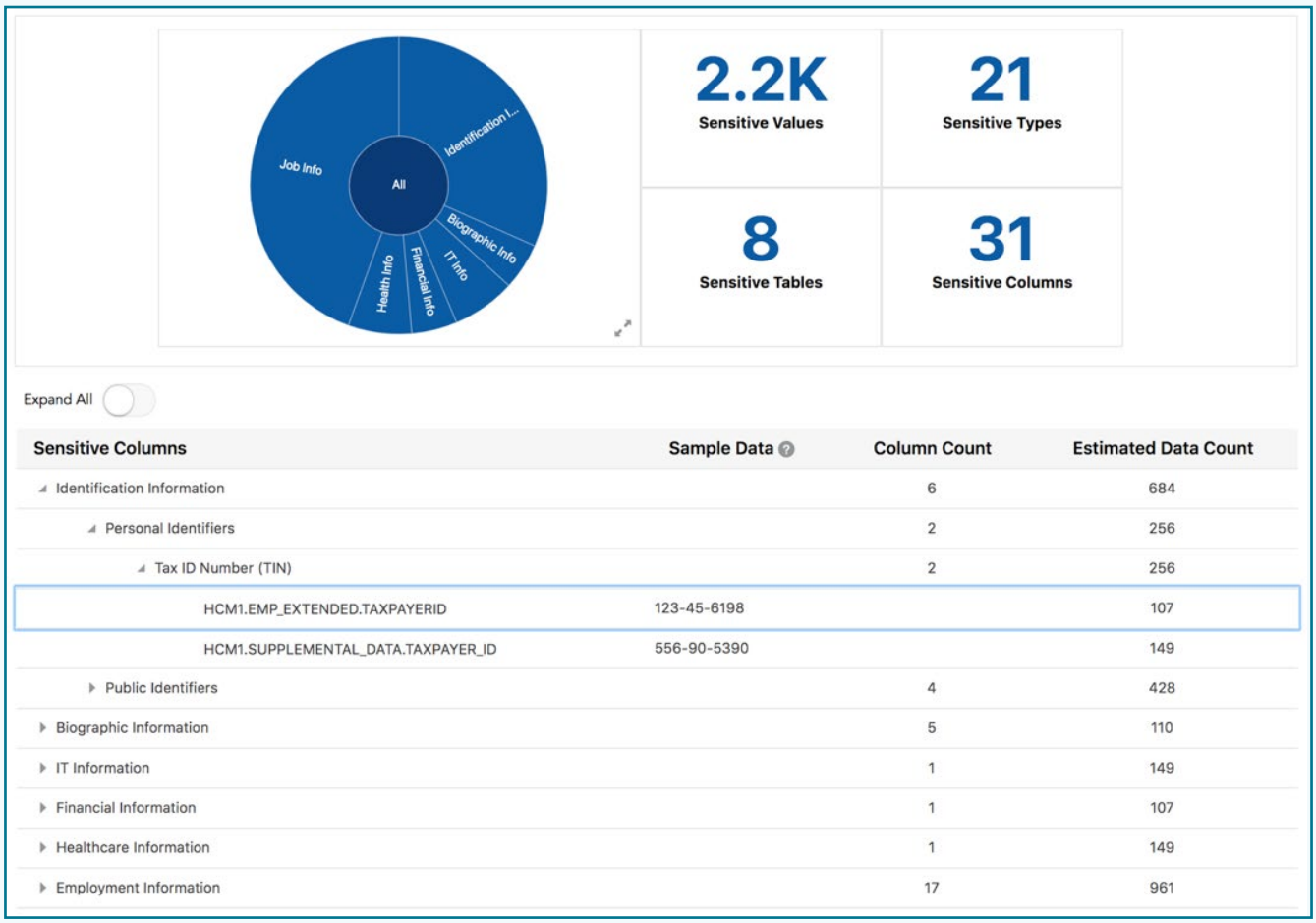
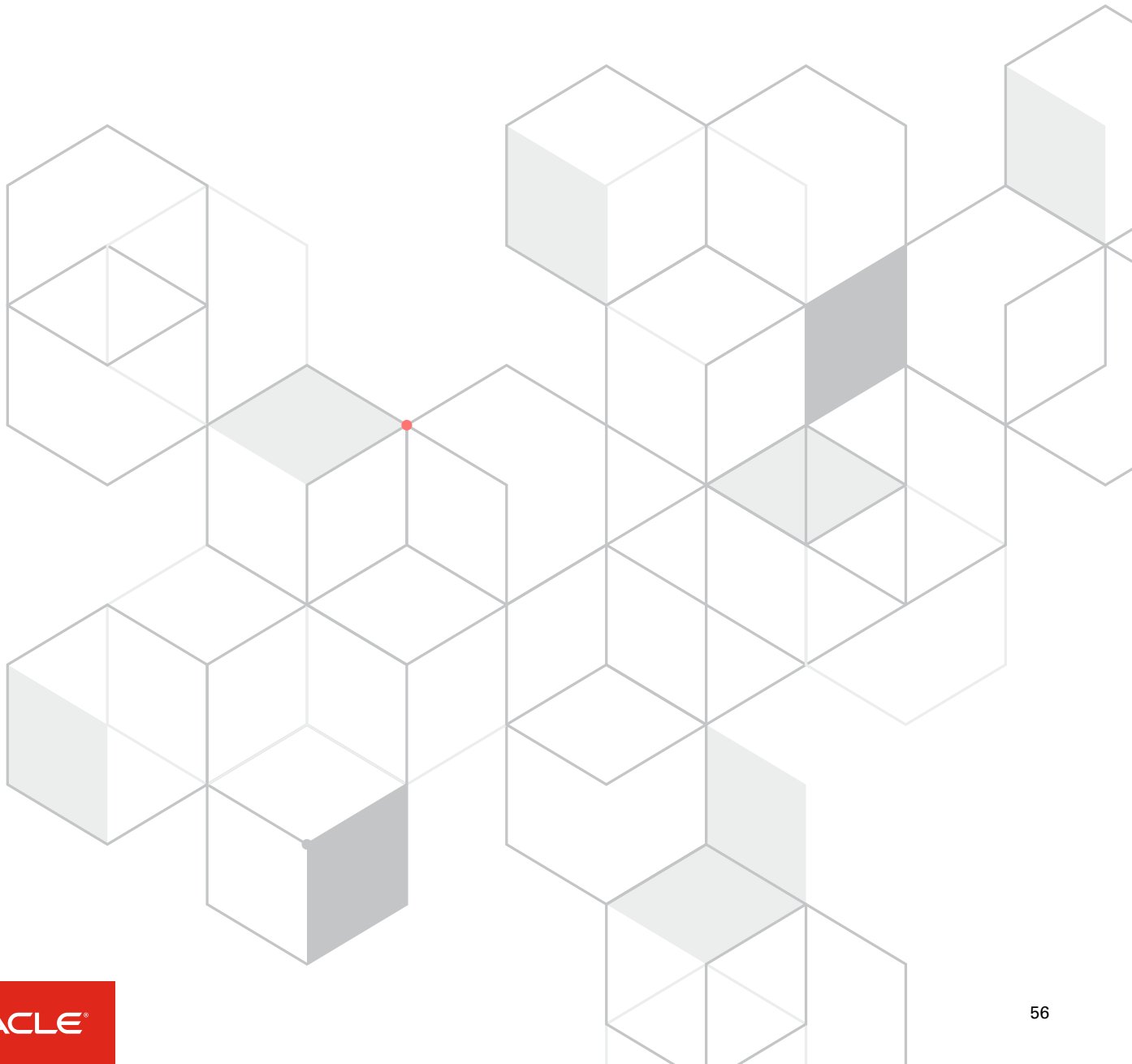


Figure 5.11: Data Discovery Report in Data Safe

SUMMARY

Ever-growing security threats and data privacy regulations have made data security mandatory. Understanding sensitive data is the first step to implementing appropriate security controls to protect it. Oracle provides multiple sensitive data discovery tools: Database Security Assessment Tool (DBSAT), Enterprise Manager Application Data Modeling, and Data Safe. 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. Application data modeling is a full-blown option that scans both metadata and data to discover sensitive columns and referential relationships. It is the default approach and a prerequisite for data masking and subsetting as it helps understand the dependencies and preserve application integrity. Data Safe provides a comprehensive data discovery capability and is the recommended option for discovering sensitive data in Oracle Cloud databases.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 EU GDPR.



6

Masking Sensitive Data

Reducing exposure of sensitive data is a challenge faced by many organizations. For example, snapshots of live production data are typically shared with development and test environments so that these teams have access to realistic data to support their activities. This data can include credit card numbers, Social Security numbers, sales figures, and other personal or proprietary information. Attackers have learned that this kind of information often ends up in test and development systems where it is generally less protected than in production systems – making those systems attractive targets for attacks.

Even in production systems, organizations should limit the exposure of sensitive data so that the information displayed to various users is limited to the minimum required by them to perform their jobs. For example, in the case of a customer service application, a call center representative may only require access to the last four digits of a customer's Social Security number in order to verify their identity. While the system might require the Social Security number for other purposes such as credit reporting, displaying the Social Security number to the telephone representative through the application's user interface violates the principle of least privilege and creates an unnecessary avenue for compromise of this information.

In both of these cases, organizations need to mask sensitive data to minimize exposure and the associated risk. Data Masking, also called "static masking", involves permanently altering data so that it is no longer sensitive. Masking sensitive data before it is handed over to test and development teams eliminates the risk of data breaches in non-production environments by irreversibly replacing the original sensitive data with realistic, but fictitious, data. In contrast, Data Redaction, also referred to as "dynamic masking", obfuscates data on the fly as it is retrieved by a database session but does not actually change the stored data value. Data Redaction limits the exposure of sensitive data within application user interfaces and helps reduce disclosure risks.

Another related requirement is to extract and share a portion or subset of data instead of sharing the entire production dataset. For example, a development team might need only 20% of the production data, data collected only over the last year, or data specific to a region. Data Subsetting reduces security risk and minimizes storage costs by removing unnecessary data from a database.

Together, Data Masking, Data Subsetting, and Data Redaction limit the avenues available to hackers to steal sensitive data and helps facilitate compliance with various regulatory requirements such as PCI-DSS and the EU GDPR. See Chapter 11 for more information on how data masking helps support regulatory compliance.

Knowing what kinds of sensitive data reside in an application is the first step to masking sensitive data. The previous chapter describes various ways to discover sensitive data. This chapter introduces the basic elements of data masking and data redaction, and describes how these techniques can be applied to reduce the attack surface for sensitive data within applications.

MASKING SENSITIVE DATA

Security-conscious organizations usually implement multiple security controls in their production environments to ensure that access to sensitive data is tightly controlled. But it is equally important to protect sensitive data in non-production environments. Data privacy standards such as PCI-DSS and EU GDPR also emphasize protecting sensitive information in non-production environments because these systems are typically not as protected or monitored as production systems.

Oracle Data Masking and Subsetting, part of Oracle Enterprise Manager Database Plug-in, provides the capability to mask and subset data for non-production use. After creating an application data model consisting of sensitive columns and referential relationships, Oracle Data Masking is used to create a masking definition that defines the rules to mask sensitive columns. For example, credit card numbers and Social Security numbers can be replaced with fictitious values; names and birth dates can be shuffled; and salaries can be randomized within the given range. This masking definition is then used to obfuscate sensitive data in application schemas while preserving application integrity. The resulting data set may then be used for non-production purposes such as development, testing, and analytics. Data Masking helps minimize proliferation of sensitive data and reduce the security and compliance boundary, while maintaining usability of the data.

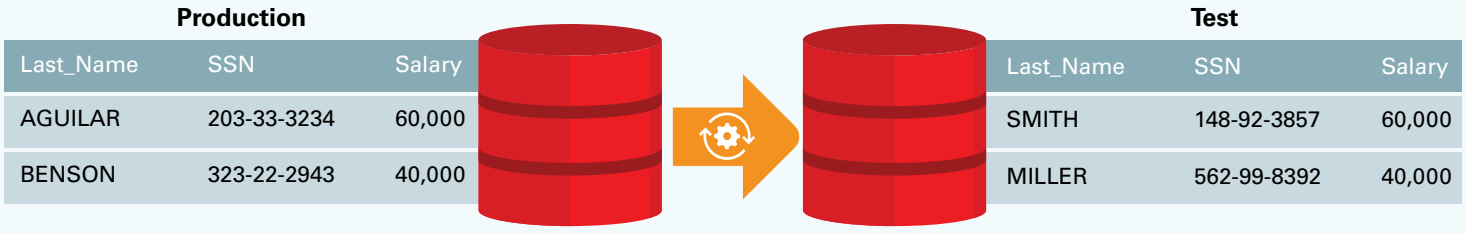


Figure 6.1: Oracle Data Masking Overview

Masking Formats

Masking formats define how original data is transformed during the masking process to create an anonymized and sanitized data set. Oracle provides a comprehensive set of predefined masking formats for common types of sensitive data such as credit card numbers, phone numbers, Social Security numbers, and national identifiers. Figure 6.2 shows some of the masking formats provided for different types of credit card numbers. The ability to create fictitious credit card numbers for multiple varieties of credit cards preserves diversity in the test data set and helps ensure generation of a realistic test data set.

By leveraging the masking format library, organizations can apply data masking and anonymization rules to sensitive data across enterprise-wide databases and ensure consistent compliance with regulations. New masking formats can be created using various masking techniques to meet specific requirements. For example, custom email addresses could be generated in a test data set, consistent with a fictitious set of first and last names contained in other columns within the database. These user-defined formats can also be stored in the masking format library for future use.

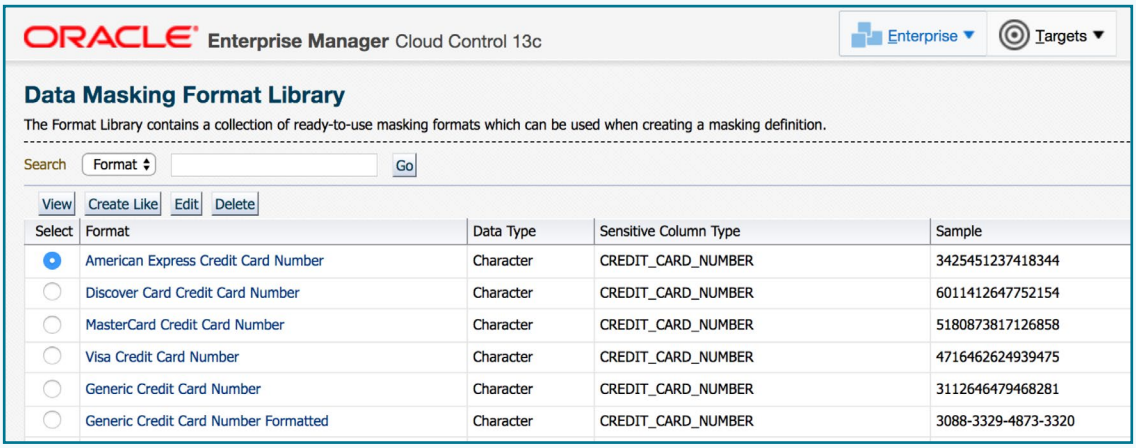


Figure 6.2: Credit Card Masking Formats

Powerful Masking Techniques

Oracle supports different masking techniques, enabling complex applications to function with masked data in a variety of test and development environments. Figure 6.3 describes some common masking techniques, and Figure 6.4 shows some masking techniques in action.

Masking Technique	Description
Random numbers strings/dates	Replaces values in a column with random numbers, strings or dates within a user specified range.
Format preserving randomization	Randomizes data while preserving its format. Replaces letter with letter and digit with digit, but preserves the data length, special characters, and the case (upper or lower) of characters. This can be applied to a variety of data such as national identifiers, zip codes and license plate numbers.
Shuffle masking	Randomly reorders the values within a column so that statistical relevance is maintained. This can be applied to ages of employees in a company, for example.
Array list	Replaces original sensitive values in a column with random values from a user provided list. This can be applied to partner names, age, gender or religion, for example.
SQL expression masking	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 names and last names. This may be used to mask data contained in large objects (LOBs) including BLOBs, CLOBs and NCLOBs.
Conditional masking	Applies different masking formats to different rows based on a condition. For example, data about citizens from multiple countries can have unique masked national identifiers based on their country. Similarly, credit card numbers can be masked while preserving their original type and format.
Compound masking	Masks related data stored in multiple columns as a group. For example, city, state, and zip code may need to be masked together so that the masked values correlate with each other.
Deterministic masking	Masks data to the same consistent value across multiple databases or applications. This is used to ensure that certain values such as customer number gets masked to the same value across all databases. This can be used to maintain referential integrity in a multi-application test environment.
Reversible masking	Encrypts and decrypts sensitive data using a cryptographic key. It is helpful in scenarios where you want to retrieve the original data from the masked data. For example, you might have to share masked data with a third party for some business processing and want to recover the original data after receiving the processed data.

Figure 6.3: Representative Masking Techniques

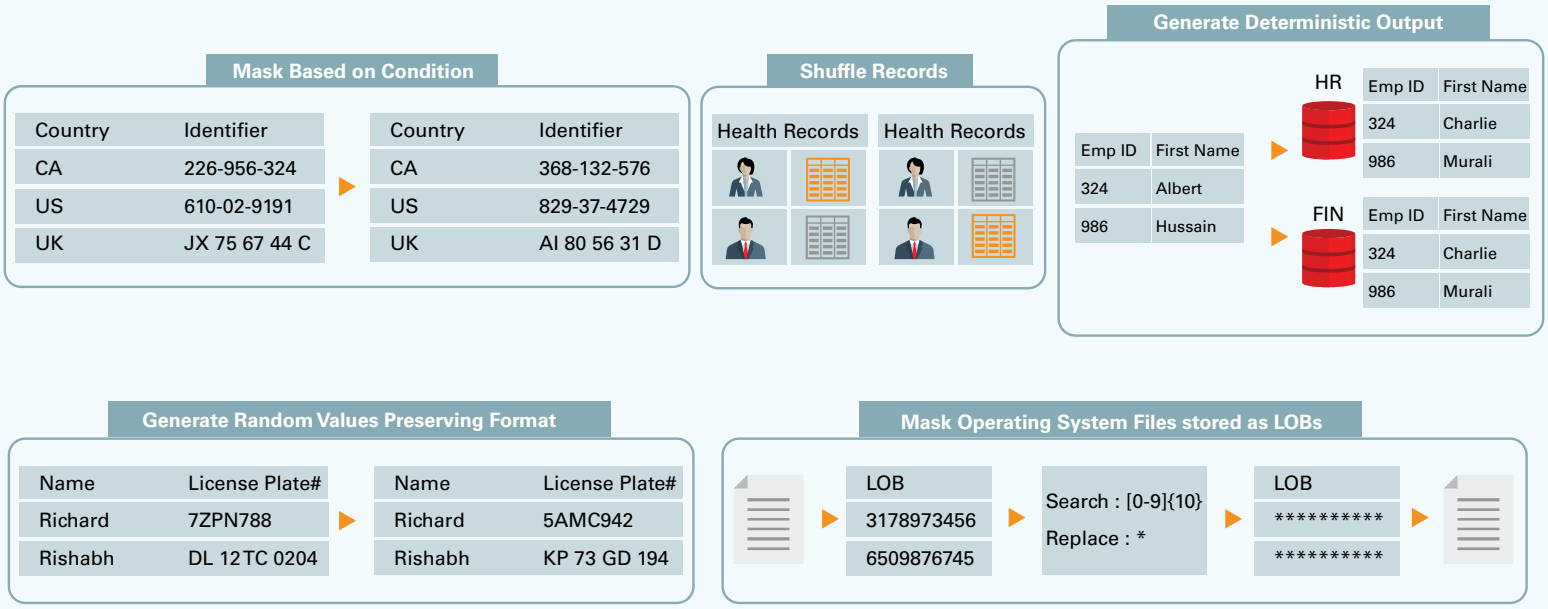


Figure 6.4: Data Masking Examples

SUBSETTING DATA

The Data Subsetting component of Oracle Data Masking and Subsetting can also leverage the application data model to generate a reduced data set of the original data while preserving referential integrity. For example, a large (>2TB) data set might be subsetted to generate a smaller (100 GB) data set for test and development, thus enabling these activities to be performed efficiently while consuming fewer resources. As shown in Figure 6.5, data subsetting supports different use cases such as extracting a fraction of a large table for application development, fetching data belonging to a particular region for analytics, or extracting rows belonging to a particular partition or sub-partition of a table.

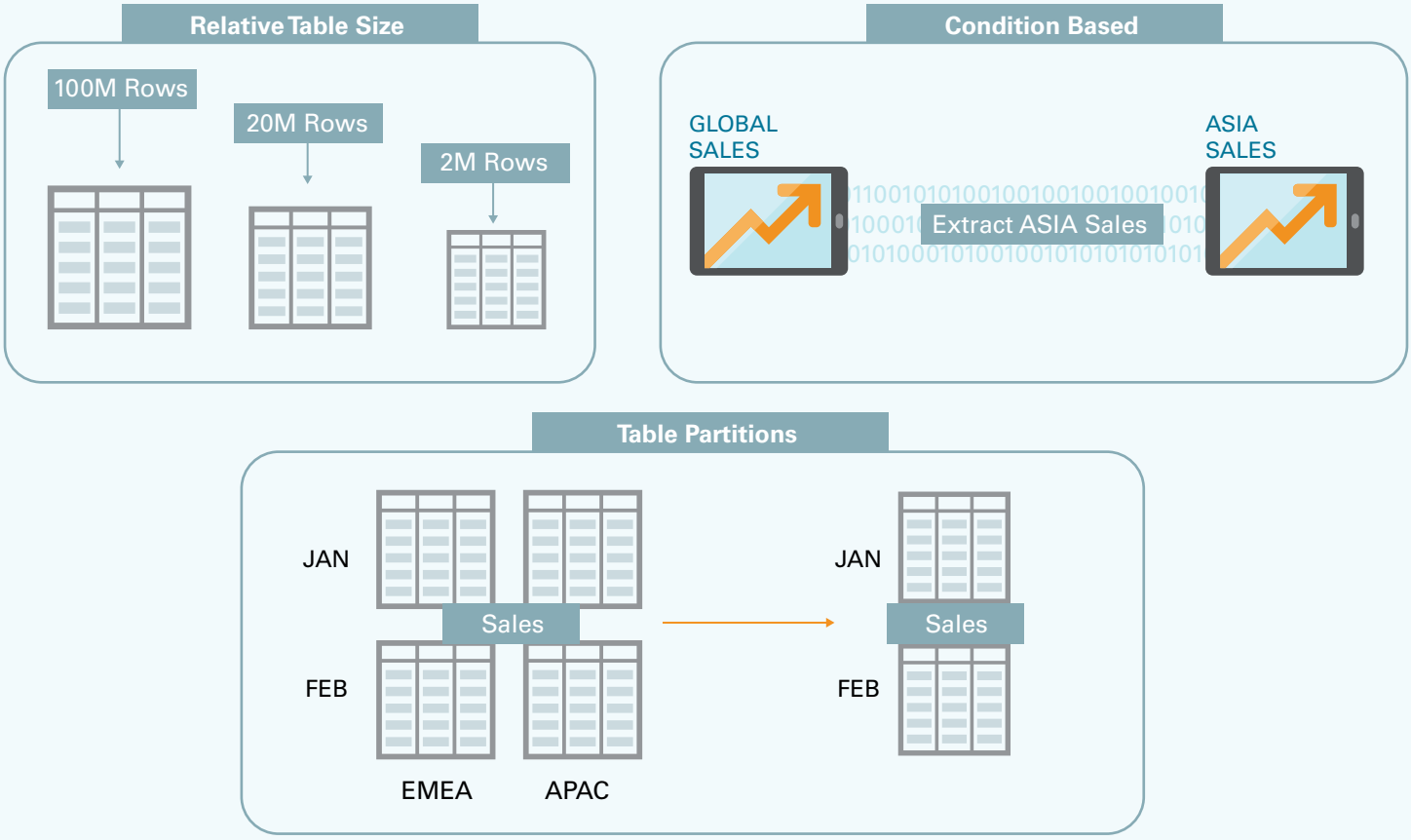


Figure 6.5: Data Subsetting Use Cases

Users can also perform integrated data masking and subsetting to extract a subset of production data and then mask sensitive data in this subset so that it can be shared for non-production use. It maximizes the business value of production data without compromising sensitive information or wasting resources.

FLEXIBLE MASKING AND SUBSETTING MODES

Usually, organizations mask sensitive data outside of the production environment. A clone of the production database is staged in a restricted area outside production. It is isolated from all users except for the administrators that run masking on it. After masking data and validating that no sensitive data is at risk, this masked copy is then further cloned and shared with developers, QA, analysts, and third-party consumers. But some organizations may have stricter security policy mandating that sensitive data never leave the production environment. In this case, they need to mask sensitive data before moving it out of the production environment.

As shown in Figure 6.6, Oracle Data Masking and Subsetting supports two different modes, providing options to choose where masking and subsetting should happen.

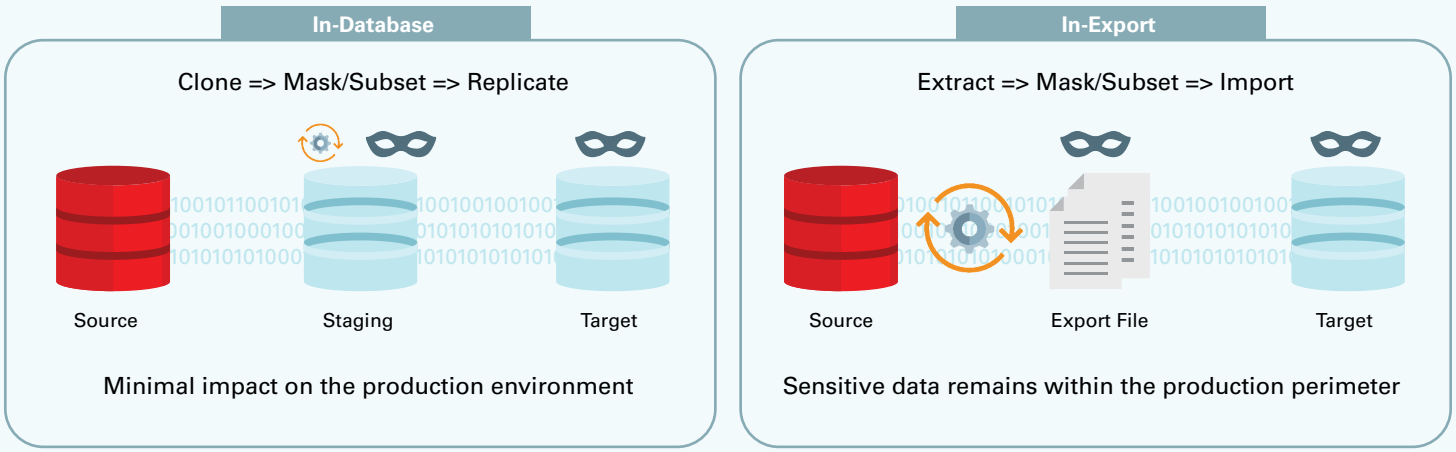


Figure 6.6: Data Masking and Subsetting Modes

In-database mode allows masking and subsetting of data within a non-production database, with minimal or zero impact on production environment. As this mode permanently changes the data stored in a database, it is recommended for non-production environments such as staging and development.

Alternatively, in-export mode can be used to mask and subset data while data is being extracted from the production database, leaving the original data unaltered on production servers. The extracted and masked data is written to Oracle Data Pump dump files, which can be further imported into non-production databases. This mode eliminates the need for staging servers and helps ensure that the sensitive data never leaves the production perimeter.

MASKING SENSITIVE DATA IN THE CLOUD

Organizations are rapidly adopting cloud, with many preferring it for application development as it enables them to cost-effectively build, deploy, and manage applications. Oracle Data Safe provides tightly integrated sensitive data discovery and data masking capabilities that help identify and mask sensitive data efficiently. See Chapter 13 for more information on how Oracle Data Safe can help meet security requirements, including data masking, in Oracle Cloud.

REDACTING SENSITIVE DATA

There is an increasing need to control the display of sensitive data contained within applications. For example, a call center application may have a page that exposes customer credit card and other personally identifiable information to call center operators. Exposing that information, even to legitimate application users, may violate privacy regulations and put the data at unnecessary risk exposing it to those who don't have the need to know.

One common technique for restricting sensitive data displayed in an application is to redact it. With redaction, displayed data might be hidden completely, replaced with random data, or special characters may be applied to part of a retrieved data element such as the first twelve digits of a credit card number or the username portion of an e-mail address.

Organizations redact sensitive data shown on application screens to better protect personal information and adhere to regulations. However, implementing redaction inside application code can be difficult to maintain. Strict controls must be placed on new application development to make sure that custom application code and new objects are properly accessed. In addition, redaction approaches implemented with custom application logic can result in disparate solutions that are inconsistent across the enterprise, especially in consolidated environments where multiple applications access the same data.

The best approach for controlling the exposure of sensitive data in applications is by enforcing redaction policies inside the production database as the data is being served live to the application.

Oracle Data Redaction performs selective, on-the-fly redaction of sensitive data in query results priorto display by applications. This enables consistent redaction of database columns across application modules accessing the same data. Oracle Data Redaction minimizes changes to applications because it does not alter actual data in internal database buffers, caches, or storage, and it preserves the original data type and formatting when transformed data is returned to the application. Figure 6.7 shows how the first five digits of Social Security numbers are redacted prior to display through a call center application.

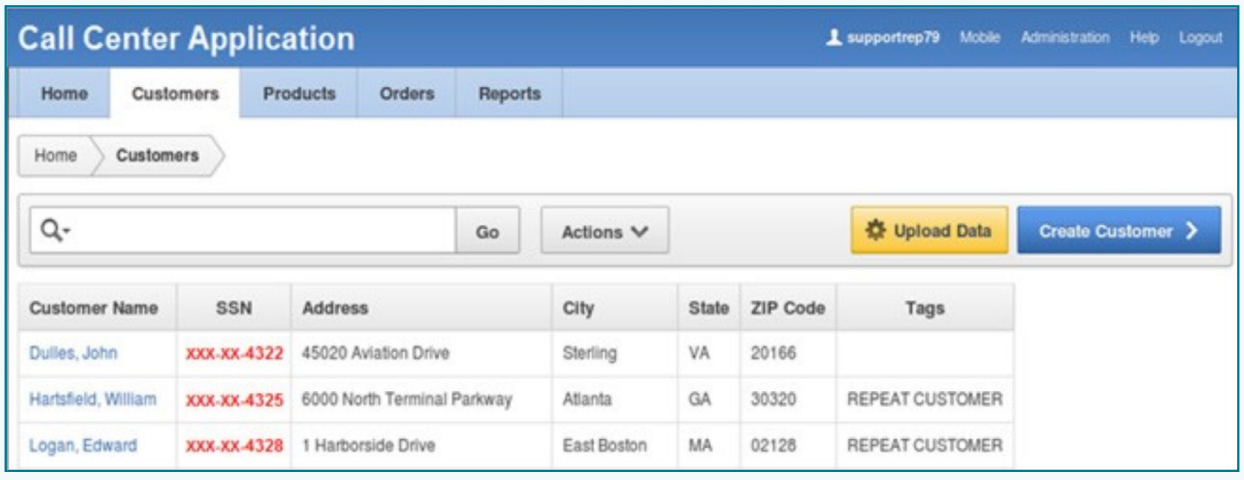


Figure 6.7: Partial Data Redaction

Oracle Data Redaction supports a number of different transformations as shown in Figure 6.8.

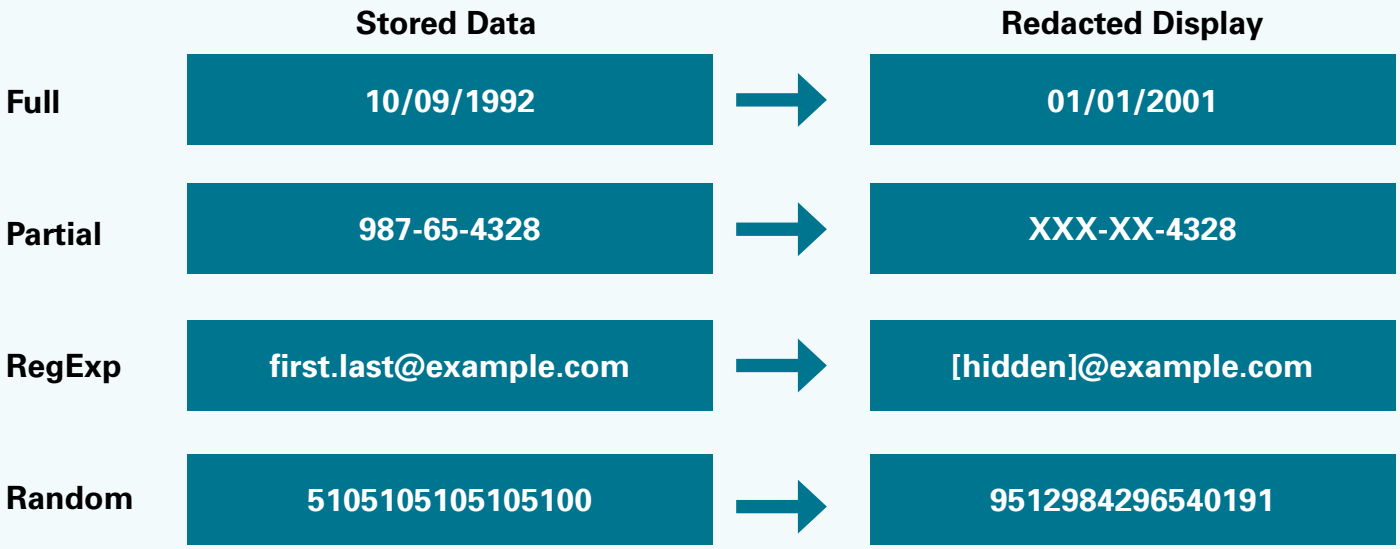


Figure 6.8: Examples of Data Redaction

Oracle Data Redaction supports a number of different transformations as shown in Figure 6.8. Predefined redaction formats are available for redacting common sensitive data such as credit card numbers, Social Security numbers, zip codes, email addresses, and phone numbers. Users can also create new redaction formats to meet specific requirements. Figure 6.9 shows some predefined redaction formats provided by Oracle.

Format Name	Sensitive Column Type	Function Type
American Express Credit Card Numbers - Formatted	CREDIT_CARD_NUMBER	PARTIAL
American Express Credit Card Numbers - NUMBER	CREDIT_CARD_NUMBER	PARTIAL
American Express Credit Card Numbers - Partially Redacted	CREDIT_CARD_NUMBER	REGEX
American Express Credit Card Numbers - Random	CREDIT_CARD_NUMBER	RANDOM
Canadian Social Insurance Numbers - Formatted	SOCIAL_INSURANCE_NUMBER	PARTIAL
Canadian Social Insurance Numbers - NUMBER	SOCIAL_INSURANCE_NUMBER	PARTIAL
Canadian Social Insurance Numbers - Random	SOCIAL_INSURANCE_NUMBER	RANDOM
Canadian Social Insurance Numbers - VARCHAR	SOCIAL_INSURANCE_NUMBER	PARTIAL
Credit Card Numbers - Formatted	CREDIT_CARD_NUMBER	PARTIAL
Credit Card Numbers - NUMBER	CREDIT_CARD_NUMBER	PARTIAL
Credit Card Numbers - Partially Redacted	CREDIT_CARD_NUMBER	REGEX
Credit Card Numbers - Random	CREDIT_CARD_NUMBER	RANDOM
Email Addresses	EMAIL_ID	REGEX
IP Addresses	IP_ADDRESS	REGEX
North American Phone Numbers - Formatted	PHONE_NUMBER	REGEX
North American Phone Numbers - NUMBER	PHONE_NUMBER	PARTIAL
North American Phone Numbers - Random	PHONE_NUMBER	RANDOM
North American Phone Numbers - VARCHAR	PHONE_NUMBER	PARTIAL

Figure 6.9: Predefined Redaction Formats

Oracle Data Redaction can be applied selectively based on policies that use the runtime contexts available from the database and from the running application. These include user names, client identifiers, database roles, and session information including client IP addresses and program modules. Context information available from Oracle Application Express (APEX), Oracle Real Application Security (RAS), and Oracle Label Security (OLS) can also be utilized. Multiple runtime conditions can be joined together within a data redaction policy for fine-grained control over when redaction occurs.

The redaction policy expression builder within Oracle Enterprise Manager enables administrators to define and apply redaction policies on existing applications. It guides the user through creating policy conditions that use context obtained from applications, the database, the APEX framework, and other database security solutions.

DEPLOYING DATA REDACTION

The power of Oracle Data Redaction resides in its efficient transformation and enforcement inside the database kernel, declarative policy conditions, and transparency.

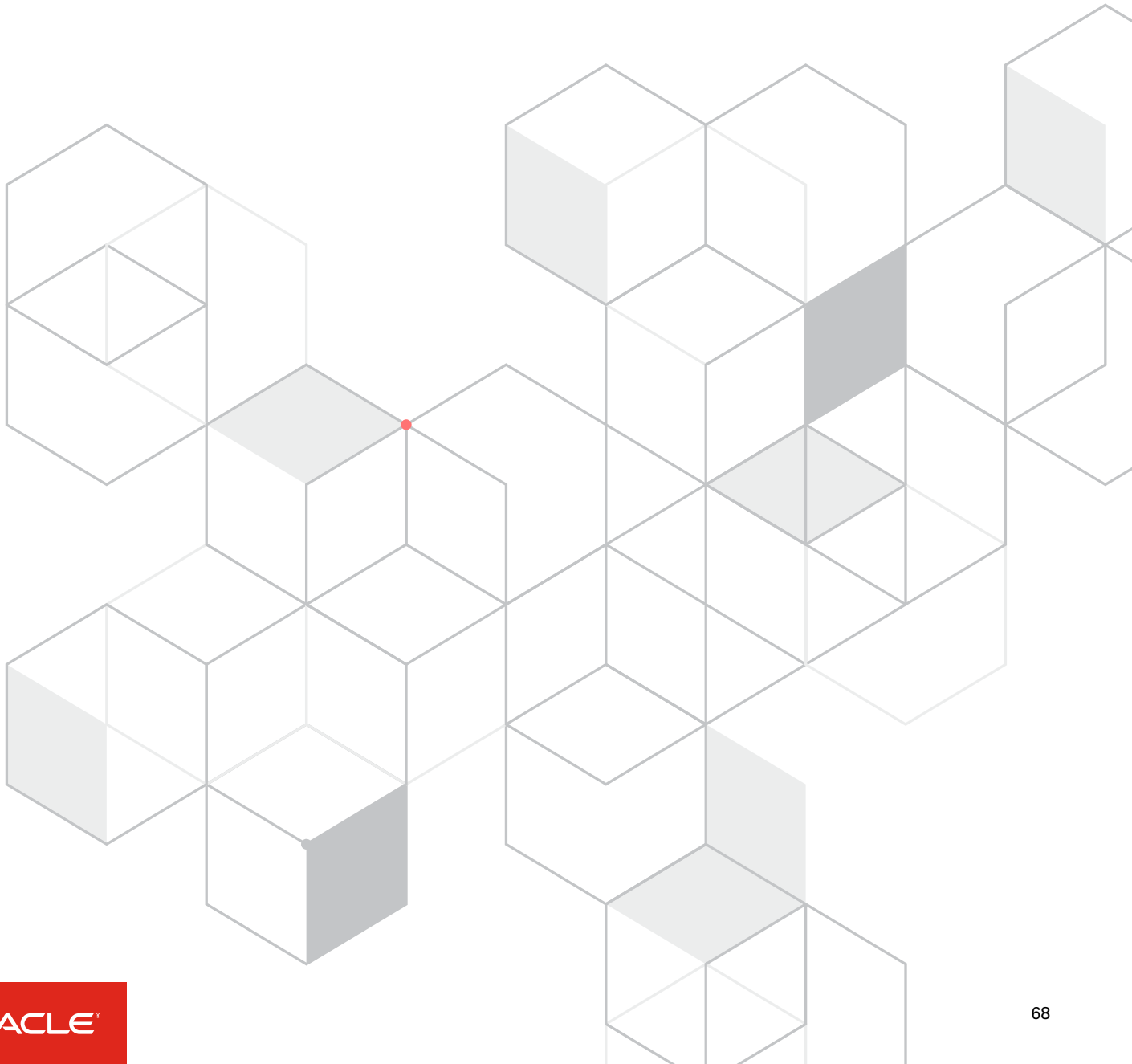
Unlike traditional approaches that rely on application coding, data redaction policies are enforced directly in the Oracle Database kernel, ensuring consistency across application modules and providing stronger security. Redaction policies are stored and managed inside the database, and they go into effect immediately upon being enabled. Redaction policies can be applied to all database users, or selectively applied based on a user's environment or other factors.

Oracle Enterprise Manager provides an easy-to-use interface for creating and applying redaction policies, allowing users to specify the protected columns, transformation types, and conditions. In addition to the Oracle Enterprise Manager administrative interface, PL / SQL APIs can also be used for scripting and applying redaction policies.

Oracle Data Redaction is transparent to applications and the database. It supports the column data types that are frequently used by applications and various database objects including tables, views, and materialized views. Redacted values retain key characteristics of the original data such as the data type and optional formatting characters. For transparency to the database, Oracle Data Redaction does not affect administrative tasks such as data movement using Oracle Data Pump or database backup and restore using Oracle Recovery Manager. It does not interfere with database cluster configurations such as Oracle Real Application Clusters, Oracle Active Data Guard, and Oracle GoldenGate.

SUMMARY

Data masking, data subsetting, and data redaction are powerful techniques for limiting exposure of sensitive data contained within applications. Permanently masking a database before it is exported to a test and development system helps ensure the sensitive information cannot be compromised during these activities. Redacting sensitive data so that it can be displayed on a production system, without altering the stored data, limits exposure of sensitive information to application users. These technologies play a key role in addressing anonymization and pseudonymization requirements associated with regulations such as EU GDPR.



7

Auditing Database Activity

NEED FOR AUDITING

Assuming that access controls are properly configured and privilege grants have been minimized, two important risks still remain. The first risk is that users who need significant privileges to perform their jobs may misuse these privileges. The second risk is that a hacker may take over those user accounts and misuse them. It is therefore very important to audit the activities of these users to know who did what and when, enabling organizations to meet stringent security controls and compliance requirements. Auditing is the primary tool for tracking these user activities.

Regulations also drive the need for auditing. In addition to countering cyber threats, organizations audit databases to comply with EU GDPR, PCI DSS, SOX, HIPAA, GLBA, FISMA, and other security and compliance regulations. Internal governance, company specific security policies, and forensic reporting also contribute to the need to audit. Most enterprises want to be able to attribute any access or change to data, or modification to the database, to specific events that are authorized in terms of who or what was issuing the command, and the business justification behind such an interaction. Compliance auditors expect that the custodians of the database, the database administrator, and their management can account for all access and changes to the database.

In order for audit data to be used for security and compliance purposes, it must be a reliable record of events which cannot be deleted or manipulated to hide suspicious activities. As many database breaches take months to get discovered, it is important that audit data be kept for as long as possible to analyze past events. Finally, we need easy ways to search the collected audit data to find specific information for forensics or anomaly detection.

AUDITING IN ORACLE DATABASE

Oracle Database provides the industry's most comprehensive auditing that includes information about the database, the client from where the request is being made, the details of the operation and the SQL statement itself. One can enable auditing on individual actions, such as the type of SQL statement executed, or the use of powerful system privileges or specific SQL statements on a particular schema object. It is typical to turn on auditing for all activities by the privileged user such as the DBA. Auditing can be configured to log both successful and unsuccessful events and include or exclude certain users from being audited as well.

Auditing is the most accurate record of any database activity, not just the ones happening over the wire but also those triggered within, through direct login, which bypasses any network monitoring. Auditing works even if the network is encrypted, which normal network monitoring systems are not able to do.

Oracle Database provides predefined audit policies that cover commonly used security-relevant audit settings such as logon failures, database configuration parameter changes, user account and privilege management, etc. Additionally, one can create custom audit policies. One can build selective and effective audit policies by adding various conditions including SYS_CONTEXT and Application Context values. All audit records with detailed information are written to the audit trail. Sample audit record detail is shown in Figure 7.1

Database details	Client details	Operation details	Statement details
Database name	Host name	Current / Proxy User	SQL statement
Instance number	IP address	Application user	Bind values
Database Identifier	Client Program	Action	Schema and Object
Database User Name	End User Identifier	Success/Failure	Application Context
DB Link Info	OS User	Object owner	
	OS Terminal	Time	

Figure 7.1: Audit Record Details

Starting with Oracle Database 12c, there is one audit trail thereby providing a simplified way for collection, analysis, and management of all of the following audit sources:

- Audit system related: Audit records (including SYS audit records), Audit configuration or policy changes
- Security controls related: Oracle Database Vault, Oracle Label Security, Oracle Real Application Security
- Database operations related: Oracle Recovery Manager, Oracle Data Pump, Oracle SQL*Loader Direct Load

The audit trail, which resides in the `AUDSYS` schema in the `SYSAUX` tablespace, makes the audit data available in a uniform format through the `UNIFIED_AUDIT_TRAIL` data dictionary view.

When the database is writeable, audit records are written to the audit trail. When the database is not writable, then audit records are written to operating system files in the `$ORACLE_BASE/audit/$ORACLE_SID` directory.

To provide separation of duty, Oracle provides two separate audit related roles: `AUDIT_ADMIN` and AUDIT_VIEWER. AUDIT_ADMIN role enables users to create audit policies, use the `AUDIT` and `NOAUDIT` `SQL` statements, view audit data, and manage the audit trail. `AUDIT_VIEWER` role is limited to viewing and analyzing audit data.

Audit data can only be managed using the built-in audit data management package within the database and cannot be directly updated or removed using `SQL UPDATE` or `DELETE` commands.

Audit admins can enable predefined audit policies, or create custom audit policies, as explained below.

PREDEFINED AUDIT POLICIES IN ORACLE DATABASE

The Oracle Database supports multiple pre-defined, out-of-the-box audit policies to address common audit use cases. These audit policies are essentially groups of audit options with different configurations.

- **Logon Failures:** Tracks failed login attempts.
- **Oracle Database Parameter Changes:** Captures changes in database parameters and configurations.

- **User Account and Privilege Management:** Audits changes related to commonly accessed user account, role, and privilege settings.
- **Oracle Database Vault Security:** Audits all the changes made to the Oracle Database Vault and Oracle Label Security policies.
- **Center for Internet Security Recommendations:** Audits events that the Center for Internet Security (CIS) recommends.

CUSTOM AUDIT POLICIES IN ORACLE DATABASE

When default policies do not capture the desired audit data, or when fewer events need auditing, a custom audit policy can be created. For example, custom audit policies can be created on tables which contain sensitive columns. In this example, the audit policy will audit any INSERT, UPDATE, and DELETE statements on the SALES.CUSTOMERS table when the client IP address is not on the trusted list.

```
CREATE AUDIT POLICY customer_audpol ACTIONS

INSERT ON SALES.CUSTOMERS, UPDATE ON SALES.CUSTOMERS,

DELETE on SALES.CUSTOMERS

WHEN 'SYS_CONTEXT(''USERNEV'',''IP_ADDRESS'') NOT IN ( ''192.0.2.1'',''192.0.2.2''
,''192.0.2.3'') ' EVALUATE PER STATEMENT;

AUDIT POLICY customer_audpol;
```

Using conditions, one can define precise, context-dependent auditing, for example, to capture just the required events during certain non-working hours and weekends.

PERFORMANCE CONSIDERATIONS FOR CREATING AUDIT POLICIES

Auditing in Oracle Database is performant and incurs minimal overhead. In Oracle's internal TPC-C test environment, it was observed that audit record generation had approximately 2% throughput overhead.

In order to ensure that auditing is generating the needed amount of information with minimal system impact, it is important to limit the number of enabled audit policies for a user session to what is needed for fulfilling security and compliance needs. This allows the internal audit check to be more efficient, as fewer audit policies will need to be evaluated before an event is generated, and limits the amount of audit data generated as well. It is important to fine-tune audit policies to be selective. For example, users can restrict the audit policies to specified sensitive tables so they can monitor access to their most critical data resources and not monitor a broader set of tables.

MANAGING AND MONITORING AUDIT DATA WITH ORACLE AUDIT VAULT

There are many instances in which customers may have a large deployment of Oracle and non-Oracle databases that can generate a large amount of audit data. Good practice dictates that audit data should be forwarded to a remote centralized location where it is secure from tampering by the individuals whose activities are being audited. Additionally, it is important to have a way to efficiently monitor the ongoing stream of audit data to find the particular events with security implications and identify problems that need immediate attention.

Oracle Audit Vault and Database Firewall (AVDF) simplifies the collection of audit records in a secure repository, creates separation of duty from the databases it is auditing, and produces consolidated reports and alerts on audit event data. AVDF collects audit data not just from Oracle databases, but also from Oracle MySQL, Microsoft SQL Server, SAP Sybase, and IBM DB2 for LUW databases, along with the commonly used operating system platforms such as Linux, Solaris, Windows, and AIX. AVDF also has plans to expand the audit collection support to Hadoop and NoSQL databases.

AVDF is designed for enterprises that wish to collate audit information from hundreds of database instances and provide unified policy management and reporting from a centralized server. Figure 7.2 shows the deployment of AVDF in a heterogeneous environment.

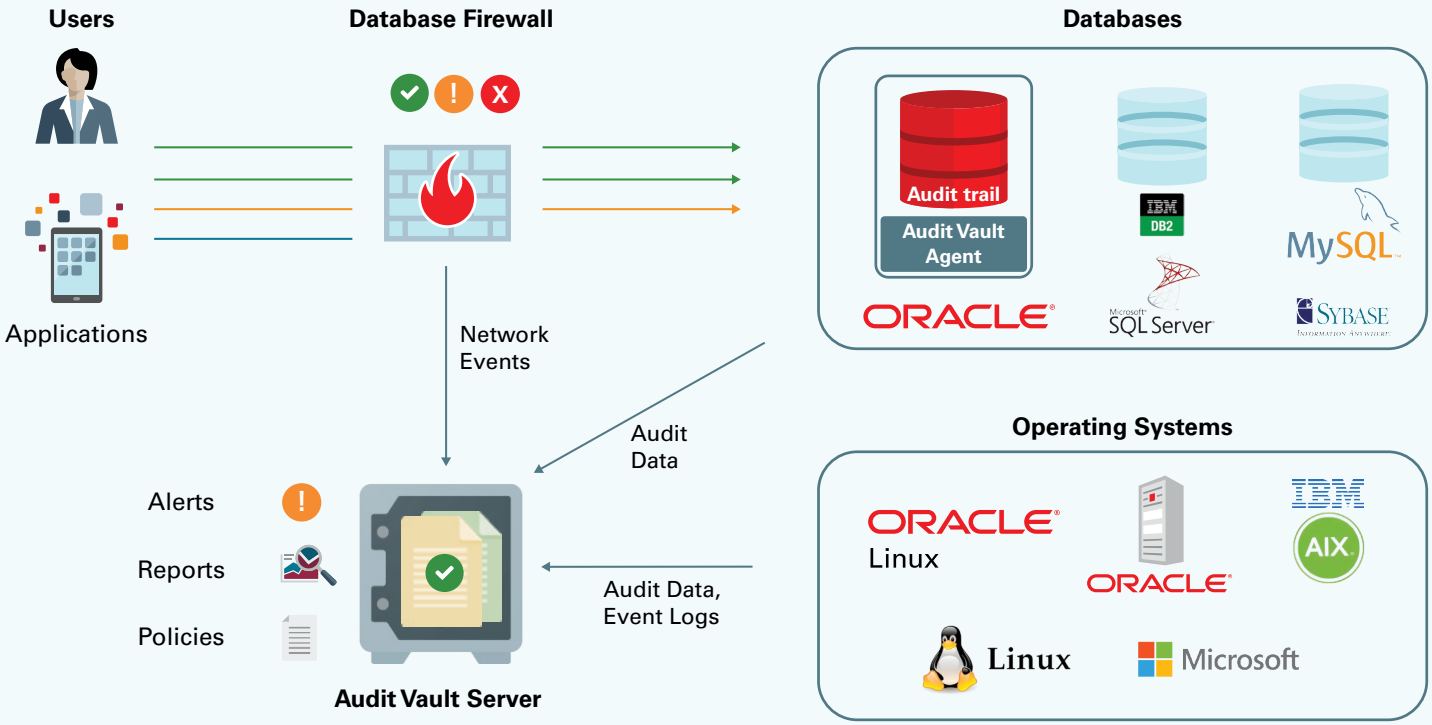


Figure 7.2: Oracle Audit Vault Functional Architecture

COMPONENTS OF ORACLE AUDIT VAULT

AVDF has four main components, namely, Audit Vault Server, Audit Vault Agents, Audit Trails and Database Firewall:

- **Audit Vault Server** is the central repository of audit records. It is highly scalable, performant and secure through the use of database compression, partitioning, encryption, in-memory, and Database Vault. Audit Vault Server performs three primary functions:
  - Consolidation of audit data and event logs from Oracle and non-Oracle databases, operating systems, directories, Database Firewall, and other custom sources.
  - Reporting and distribution of out-of-the-box and custom reports.
  - Creating alerts for potentially suspicious activities.

Audit Vault Server supports a high-availability system architecture to ensure that audit records are always collected despite any outages. One can setup a pair of Audit Vault servers, designating one of them as the primary and the other as secondary. All data and configuration in the primary server is automatically copied to the secondary and kept in sync. In the event of a failover, the secondary server becomes the primary.

- **Audit Vault Agents** manage the collection of data from systems where they are installed, and communicate with the Audit Vault Server. Audit Vault Agent retrieves audit trails from various types of secured targets and sends the audit data to the Audit Vault Server. Audit Vault Agent is deployed on a host, usually the same host running the audit data source (secured target), such as a database or operating system. However, one can also deploy the agent on a remote host.
- **Audit Trails** are the sources of audit data:
  - Database audit trails including Oracle Database, Microsoft SQL Server, SAP Sybase, IBM DB2 for LUW, and MySQL. The audit data can come from audit tables or files.
  - Before/After values from REDO records of Oracle databases.
  - OS audit trails from Oracle Linux, Red Hat Linux, Oracle Solaris, Microsoft Windows, and IBM AIX.
  - Microsoft Active Directory
  - File systems such as Oracle ACFS.
  - Custom audit data in either database tables or XML files so that audit data for custom applications can also be collected and reported in the same way without having to write additional code.
- **Database Firewall** monitors SQL transactions over the network and determines whether a SQL statement should be permitted, modified, blocked, or substituted before it reaches the database server. The next chapter describes the operation of Database Firewall. For now, it is sufficient to state that it forwards relevant events to the Audit Vault Server.

REPORTING

Oracle Audit Vault reports cover a wide range of activities including privileged user activity, changes to database structures, and SQL statements being executed. In addition, reports include information on changes in database account management, roles and privileges, object management, and stored procedure changes. Reports consolidate audit information from databases, operating systems, and directories, providing an aggregate view of activities across the enterprise.

Auditors access reports interactively through a web interface, or through PDF or XLS report files. The web console provides the ability to create color-coded charts and graphs. Report columns can be sorted, filtered, re-ordered, added, or removed. Rules can automatically highlight specific rows so that users can quickly spot suspicious or unauthorized activity. PDF and XLS reports can be generated and delivered via e-mail. Reports can also be defined to require signoff by multiple auditors. Users can use Oracle BI Publisher to create new or customize PDF and XLS report templates to meet specific compliance and security requirements. Furthermore, the Audit Vault Server repository schema is documented, enabling integration with third-party reporting solutions.

Various categories of built-in reports are provided, such as:

1. Activity Reports including User Entitlement Reports and Stored Procedure Audit Reports.

2. Summary Reports including Trend Charts and Anomaly Reports.

3. Compliance Reports.

ACTIVITY REPORTS

Activity reports track general database access activities such as audited SQL statements, application access and user logins. Specialized activity reports cover failed logins, user entitlements, before-after data modifications, changes to application tables, and database schema changes. For example, if we need to audit each time a user performs DDL SQL statements such as DROP or ALTER, the pre-built Database Schema Changes report highlights rows associated with that particular user and drills down to individual event details. Activity Reports, as shown in Figure 7.3, can be used to monitor a wide range of activity from databases, operating systems, and directories, providing an aggregate view of activities across the enterprise.

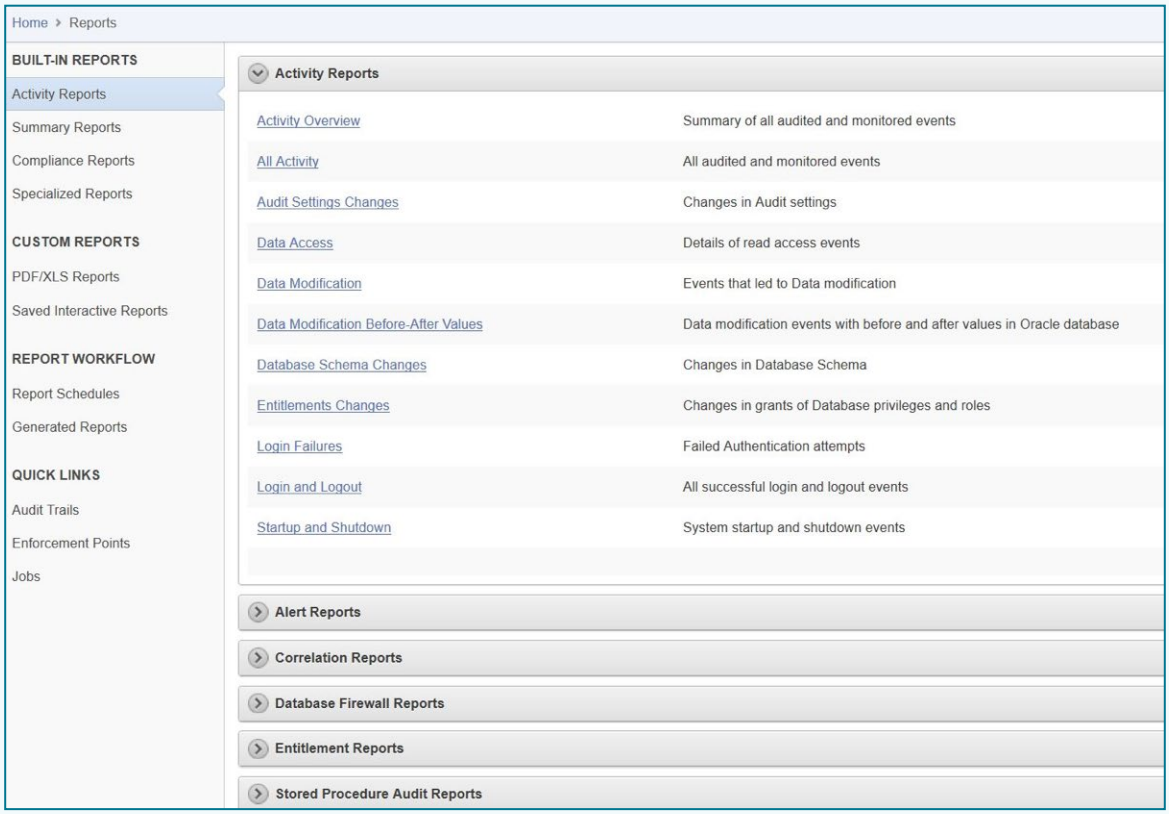


Figure 7.3: Oracle Audit Vault Built-in Activity Reports

User Entitlement reports, as shown in Figure 7.4, describe the types of access that users have to an Oracle database, providing information about the users, roles, profiles, and privileges used. These reports are useful for tracking unnecessary access to data, finding duplicate privileges, and simplifying privilege grants. After an entitlement snapshot is generated, one can compare different snapshots to find how the entitlement information has changed over time. This is particularly useful for identifying any drift from an approved database entitlement baseline and can also pinpoint privilege escalations due to possible malicious activities.

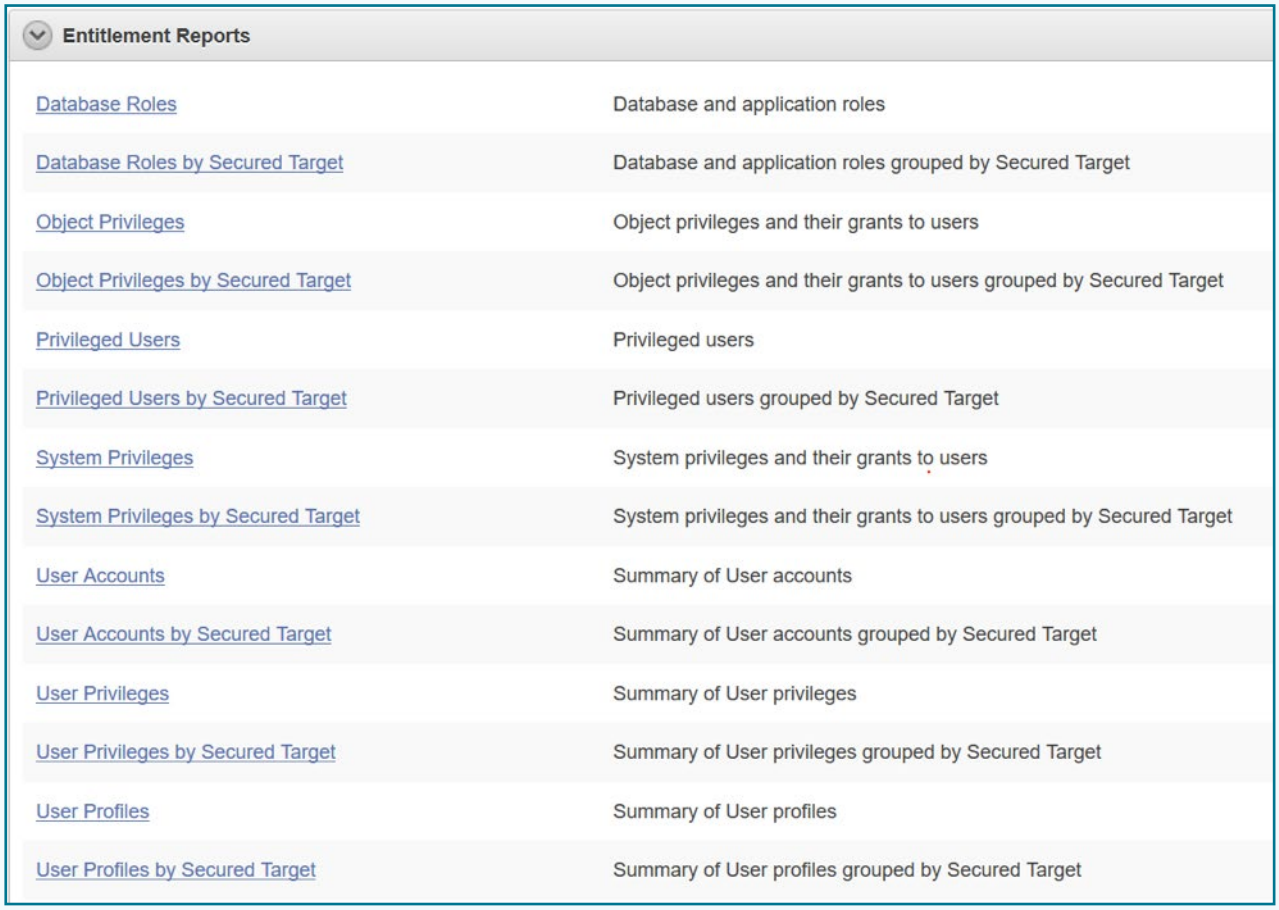


Figure 7.4: Entitlement Reports

Stored Procedure Audit Reports can help keep track the changes made to the stored procedures. Correlation Reports identifies events on the database with the original Linux OS user for secured Oracle Database targets running on Linux. This is useful in cases where this user runs a shell or executes a command on the database as another user by using su or sudo.

SUMMARY REPORTS

The report group contains Summary Reports, Trend Charts and Anomaly Reports as shown below. These reports can be used to quickly review characteristics of user activity on specific targets or across the enterprise.

Summary Reports shown in Figure 7.5 focus on statistical occurrence of various types of events generated by individual users or initiated from specific client IP addresses. Trend charts graphically present general event trends and also trends based on specific users, client IPs and secured targets.

Reports could be used to identify anomalies such as new and dormant user and client IP anomalies over time. Activities by new users, or previously dormant users, can be an indication of account hijacking.

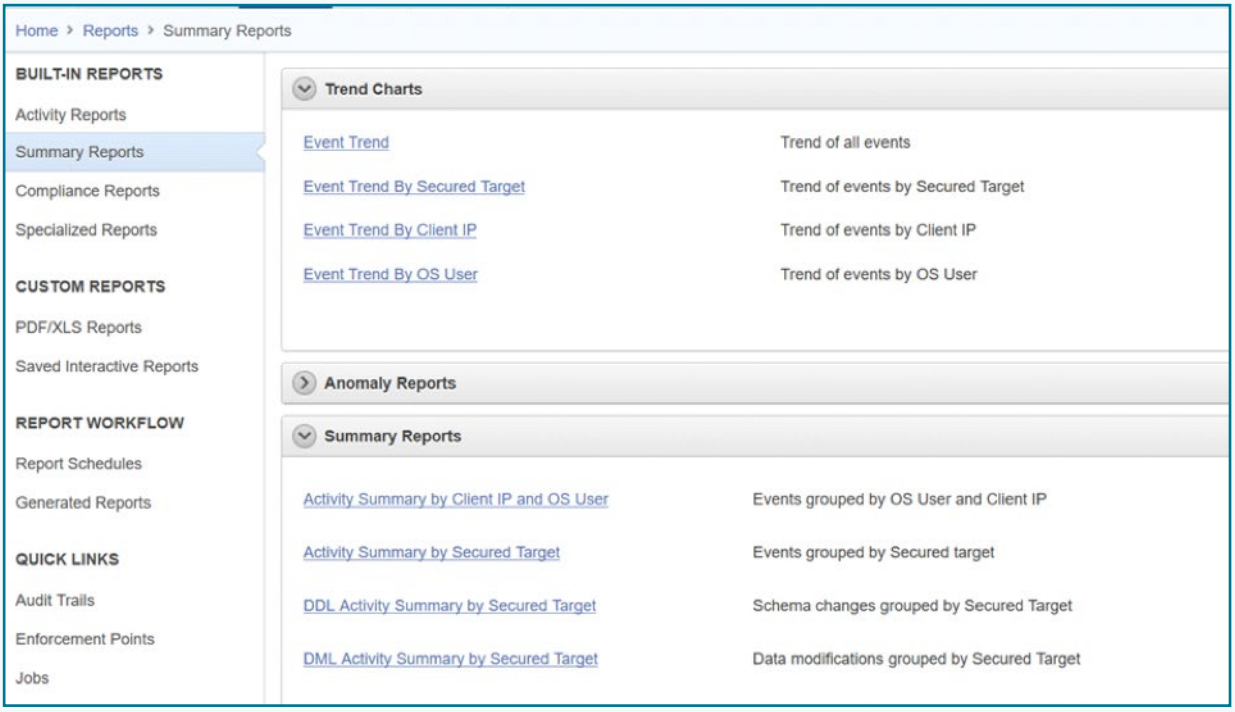


Figure 7.5: Audit Vault Summary Reports

COMPLIANCE REPORTS

Standard out-of-the-box audit assessment reports are categorized to help meet regulations such as:

- General Data Protection Regulation (GDPR)
- Payment Card Industry Data Security Standard (PCI-DSS)
- Sarbanes-Oxley Act (SOX)
- Health Insurance Portability and Accountability Act (HIPAA)
- Gramm-Leach-Bliley Act (GLBA)
- Data Protection Act (DPA)
- IRS Publication 1075

As shown in Figure 7.6, Oracle Audit Vault complies with data protection directives and regulations by offering capabilities like centralized auditing, monitoring, reporting, and alerting of anomalous activity on the database. It also reports on the sensitive data in the database targets as well as any access to sensitive data by all users including privileged users.

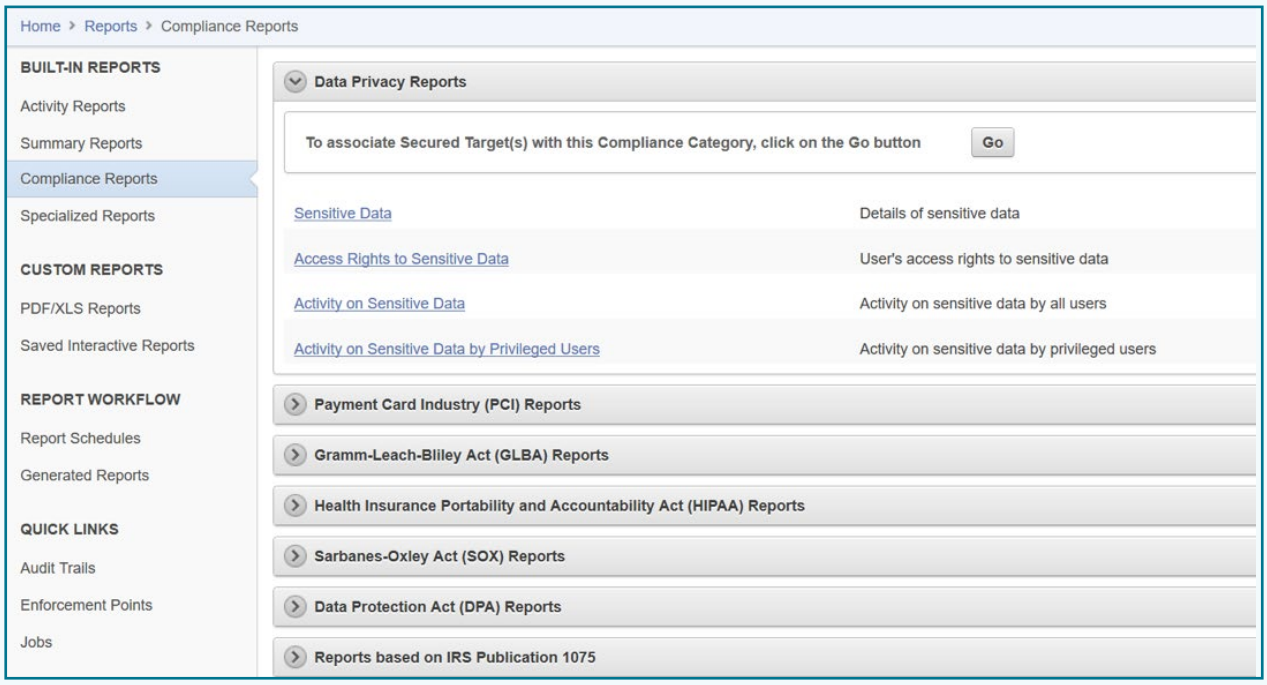
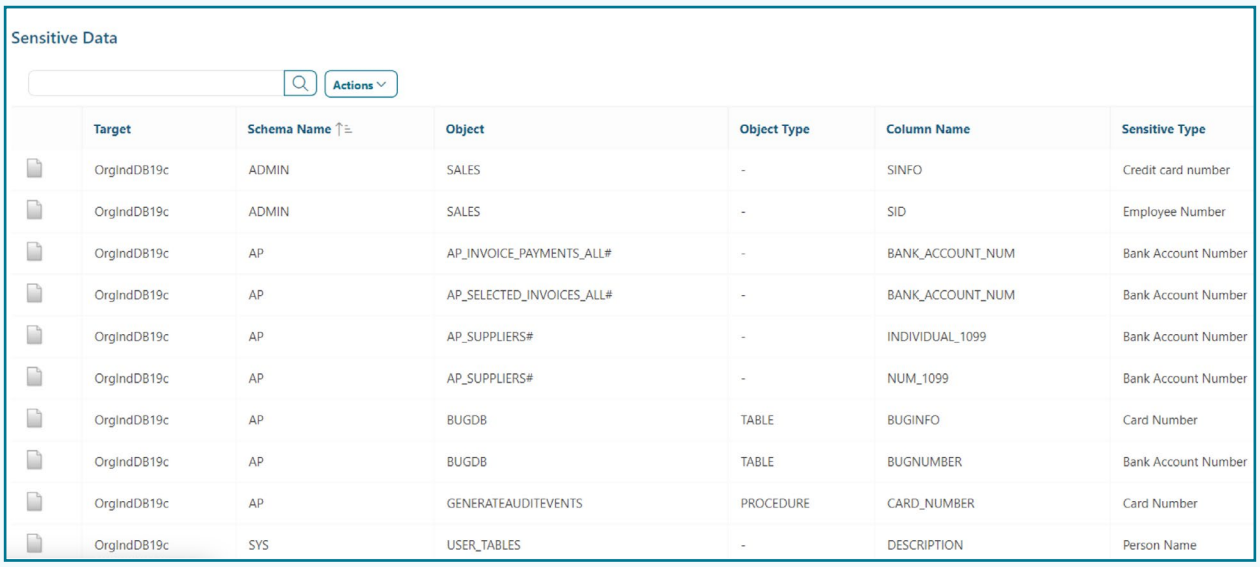


Figure 7.6: Compliance Reports

The Sensitive Data Report shown in Figure 7.7 contains information regarding activity on sensitive data by all users including privileged users.



Target	Schema Name	Object	Object Type	Column Name	Sensitive Type
OrgIndDB19c	ADMIN	SALES	-	SINFO	Credit card number
OrgIndDB19c	ADMIN	SALES	-	SID	Employee Number
OrgIndDB19c	AP	AP_INVOICE_PAYMENTS_ALL#	-	BANK_ACCOUNT_NUM	Bank Account Number
OrgIndDB19c	AP	AP_SELECTED_INVOICES_ALL#	-	BANK_ACCOUNT_NUM	Bank Account Number
OrgIndDB19c	AP	AP_SUPPLIERS#	-	INDIVIDUAL_1099	Bank Account Number
OrgIndDB19c	AP	AP_SUPPLIERS#	-	NUM_1099	Bank Account Number
OrgIndDB19c	AP	BUGDB	TABLE	BUGINFO	Card Number
OrgIndDB19c	AP	BUGDB	TABLE	BUGNUMBER	Bank Account Number
OrgIndDB19c	AP	GENERATEAUDITEVENTS	PROCEDURE	CARD_NUMBER	Card Number
OrgIndDB19c	SYS	USER_TABLES	-	DESCRIPTION	Person Name

Figure 7.7: Sensitive Data Report

ALERTING

In many instances, customers want to be notified when certain events happen. The Alert Management capability in AVDF helps create alert definitions that raise alerts on the auditor's dashboard and sends notifications to multiple users for investigations and signoff. Alerts can be defined using SQL comparison operators (=, <, LIKE, IN, NULL, and so on) and logical operations (NOT, AND, OR). As shown in Figure 7.8, alerts can be threshold and time based. For example, if five login failures occur within a one-minute window, possibly indicating a brute force attack, then an alert can be raised. For easy management, alerts can be forwarded via email or sent to Syslog.

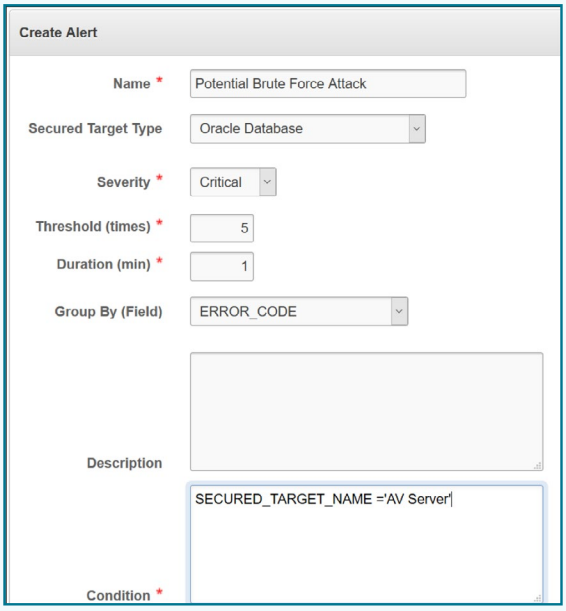


Figure 7.8: Alert Definition

USING ORACLE AUDIT VAULT FOR CLOUD AND HYBRID DEPLOYMENTS

With the rapid adoption of the cloud, companies increasingly face the situation where some of their databases are deployed on-premises while others are deployed in the cloud. The challenge is to audit and monitor them all, ideally with a unified tool. Utilizing an on-premises audit tool for both on-premises and cloud database targets has many advantages including consistent policies, unified reporting, and common alert management. Existing alert configurations and data retention polices can be applied for cloud databases. Thus, the same resources can be utilized for configuration and maintenance tasks across on-premises and cloud, and as such, Oracle Audit Vault Server can consolidate auditing across hybrid cloud environments.

In hybrid cloud deployments, as shown in Figure 7.9, the on-premises Audit Vault Server collects audit data from both on-premises and Oracle Database Cloud Service (DBCS) instances and autonomous databases on Oracle Cloud Infrastructure. On-premises agents retrieve audit data from the DBCS instances over encrypted channels, and then transfer it to the on-premises Audit Vault Server. Outside of opening the appropriate network ports on the cloud infrastructure for communication with Audit Vault agents, no other on-premises networking changes are necessary.

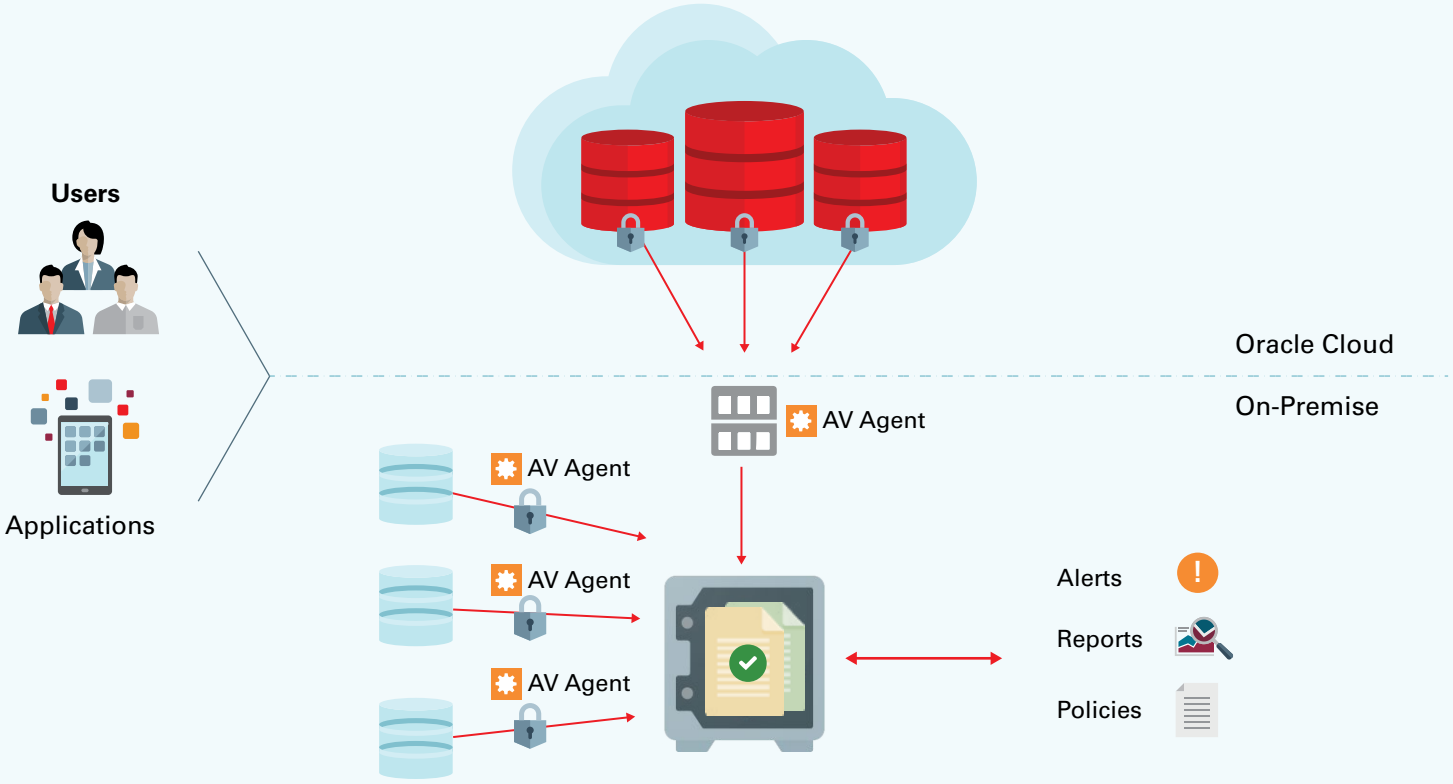


Figure 7.9: Hybrid Cloud Deployment for Oracle Audit Vault

AUDITING DATABASE ACTIVITY IN THE ORACLE CLOUD

With Data Safe, customers can monitor activities on Oracle Cloud databases, collect and retain audit records to meet industry and regulatory compliance requirements, and trigger alerts as needed for unusual behavior. They can audit sensitive database changes, administrator user activities, activities recommended by the Center for Internet Security (CIS), and activities defined by their own organization.

Alerts can be triggered when a database parameter or audit policy changes, a failed login by an admin occurs, a user entitlement changes, and when a user is created or deleted.

Data Safe provides a wide range of interactive audit reports, including the All Activity report, which is a comprehensive report that contains every audited activity. Other reports focus on specific areas, such as user and entitlement changes, administrative activity, data access, database schema changes, and

login sessions. Customers can also download a report as a spreadsheet or PDF file, which is useful for compliance reporting. See Chapter 13 for more information on how Oracle Data Safe can help audit database activity in Oracle cloud.

SUMMARY

Database auditing is an integral component of an organization's data security architecture, enabling organizations to monitor database activity and identify anomalous behavior to detect and block threats and support compliance reporting. Effective auditing requires that audit policies capture the important details about significant events, with minimal impact on the environment. The collected audit data must be a reliable record of events, and not something that could be deleted or manipulated to hide suspicious activities.

Oracle Database provides the industry's most comprehensive auditing capabilities, collecting detailed information on critical events. With conditional auditing, users have the ability to configure precise and context-dependent auditing. Additionally, the predefined audit policies simplify the configuration and management process.

Large deployments of Oracle and non-Oracle databases can produce a huge amount of audit data which needs to be consolidated and secured for alerting and reporting. Oracle Audit Vault consolidates and secures audit data from databases, operating systems, and directories. Utilizing its extensive reporting and alerting capabilities, Oracle Audit Vault provides auditors and security personnel with access to detailed information and early warning alerts on potentially malicious activities. The wide variety of reports including Activity Reports, Correlation Reports, Summary Reports, Anomaly Reports, Trend Charts, Entitlement Reports, and Stored Procedure Audit Reports allow a quick and efficient way to detect and investigate data breaches. For auditing database activity in the Oracle cloud, customers can also use Data Safe to ensure security and regulatory compliance.

8

Database Activity Monitoring with Database Firewall

NEED FOR FIREWALL MONITORING

The widespread use of applications and the relative ease of exploitability have made them number one entry point for hackers. Various studies have concluded that a significant percentage of data breaches are caused by misusing credentials of insiders, compromised clients, or through SQL injection. SQL injection has been the number one application security risk on the OWASP Top 10 for several years.

The three-tier application architecture, combined with the increasing number of attacks on databases via SQL injection or insiders with access to privileged accounts, has made network-based database activity monitoring an important component of the overall security architecture. With near-zero overhead on the monitored databases, network-based database activity monitoring can monitor independent of platform, requires no modification to the databases, and can monitor multiple heterogeneous database systems at once. Network-based database activity monitoring is to be used in conjunction with database auditing that can capture not only local activity on the database, but also any database activity that doesn't cross the network as SQL, such as logging local or remote console connections, to make database and user changes. An effective database activity monitoring solution needs to combine local database auditing with network-based database activity monitoring for detecting intrusions.

This chapter describes some common network attack vectors used to compromise the application data and the security techniques that can be applied to thwart them. Oracle Database Firewall not only monitors and blocks attacks such as SQL injection and misuse of credentials, but also provides the ability to define policies and raise alerts based upon multiple factors such as SQL category, application, user, and IP address.

SQL INJECTION OVERVIEW

SQL injection, perhaps the most common approach to attack databases, is a code injection technique that involves insertion or injection of a malicious SQL statement via the input data from the browser end user to the application. If the input data is not validated properly by the application and the input data is used to dynamically construct a SQL query, the malicious SQL gets executed with the privileges of the application runtime. Let's take an example such as the user query below:

```
SELECT * from stock where catalog-no= '<unvalidated-input>';
```

Here, a malicious user can try to exploit a SQL injection vulnerability by entering one of the following payloads as the input.

```
101' OR '1'= '1
1001' OR full_name LIKE '%bob%
' UNION SELECT cardNo from Orders--'
1001'; DROP TABLE Orders; --'
```

Instead of the simple select query that the application was going to execute, the malicious input data can change the query parameters, resulting in the database returning all the data, or executing some other SQL statement, or even deleting the data. The possibilities are quite endless. The risk here is huge because the malicious SQL statement is executed with all the privileges of the original user, and this can be done using a browser anywhere on the Internet.

COMMON MEASURES TO BLOCK NETWORK ATTACKS

Traditional perimeter-focused network firewalls used to be one of the ways to stop unwanted traffic to your databases. However, network firewalls are not adequate as they are just filtering systems that allow or deny network communication to and from a certain port and IP address. They are not capable of evaluating the actual content of the packets and providing full SQL control. Moreover, the blurring network boundaries and the increasing insider threats have rendered traditional network firewalls inadequate.

Another approach is to add protection at the application level by improving user authentication, enforcing least privilege, using prepared statements, avoiding dynamic queries, performing input validation, and so on. Although these methods provide stronger level of protection and should be practiced, even minor gaps in development or configuration controls can easily introduce vulnerabilities. Application developers need to be trained to follow secure coding practices, but developers may not follow these practices with the rigor that is needed, or they may never get around to fixing such vulnerabilities in legacy applications. Further, most applications today operate using a single powerful application user account for communicating with the database, and thus any weakness may expose the entire application data.

Database firewalls monitor database network activity to identify and protect against SQL-based attacks that access or modify sensitive information stored in the databases. Most network-based database monitoring solutions on the market today rely on regular expressions to determine which SQL statements should be blocked from reaching the database. The challenge with these first-generation solutions is that regular expressions do not match the expressive power of the SQL language. Because there are many ways to write a malicious SQL statement, it is nearly impossible to write a regular expression rule that will detect all such statements.

ORACLE DATABASE FIREWALL

Oracle Database Firewall provides a comprehensive, scalable and flexible solution for monitoring and protecting database systems. Oracle Database Firewall, a component of Oracle Audit Vault and Database Firewall, acts as the database's first line of defense on the network, monitoring SQL traffic and enforcing expected application behavior while helping prevent SQL injection, application bypass, and other malicious activities from reaching the database. The Database Firewall can block specific types of SQL statements and replace them with harmless statements that you specify. Based on the policy, the Database Firewall can also terminate an application's connection to the database. After evaluating the SQL statements and acting upon it.

Database Firewall forwards the database activity logs to Oracle Audit Vault Server, which then provides comprehensive Database Firewall reports along with other reports.

Oracle Database Firewall collects SQL statements using highly optimized traffic capture techniques. Each SQL query is associated with full session information including context attributes listed in Figure 8.1.

Attribute	Example
Target name	db12gr2.internal.example.com
Database service name	accounts.service1
Database user name	dba_001
OS user name	Fred
Client IP address	192.0.2.12
Client program name	sqlplus.exe

Figure 8.1: SQL Context Attributes

This information is then combined with further analysis of the SQL statement based on its type and structure as shown in the following Figure 8.2.

Structure Component	Example
SQL Category	SELECT, DML, DDL, DCL, TCL
Table name	tbl_customers

Figure 8.2: SQL Structure Components

Oracle Database Firewall uses SQL grammar-based engine to parse the SQL syntax to understand the SQL with high accuracy and recognize the different ways a statement can be expressed. It groups these SQL statements with the same grammatical structure into "clusters." For example, a SQL query that searches for a specific order number 234324 is essentially the same as the one that searches for another order number 333221. Understanding the similarity between different statements filters millions of SQL statements down to just a few hundred, which in turn can be further subcategorized by IP addresses, client programs, SQL types, and user names. Using these SQL clusters, firewall policies can be defined that include both allowed SQL (a whitelist) and disallowed SQL (a blacklist). The SQL statements that do not belong to the whitelist or the ones that belong to the blacklist could possibly be an attack and need to be dealt adequately.

A blacklist policy specifies a set of harmful statements that are disallowed. It also provides information about the various parameters and properties attached to the statements such as the incoming IP address, user name, and so on. A blacklist policy can be defined to disallow certain SQL statements based on the type of statement, the database object it acts upon, or session information such as IP addresses, user names, or client application names.

Since the set of harmful statements do not remain constant, instead of blocking a fixed set of “bad" statements, it is much more effective to allow only “good" statements based on the normal activity of the applications and users that connect to the database. This set of good statements is a whitelist that helps implement a positive security enforcement model. Whitelists of “normal” behavior for applications and users who generally perform the same type of SQL interaction can be defined.

Once the firewall policy is defined and deployed, Oracle Database Firewall captures incoming SQL statements, associates session information, analyzes the SQL, and decides based on the policy whether the requests should be permitted, logged, blocked, substituted or alerted on. All this work is completed in real time, without affecting the database resources.

DATABASE FIREWALL POLICIES

Successful deployment of Database Firewall depends on an effective firewall policy. A policy defines rules for Database Firewall to decide whether to permit, block, or substitute a SQL statement. These rules also help decide whether a particular event should be logged and an alert should be raised. Policy rules can depend on any combination of the SQL statement type, database user name, IP address of the database client, operating system user name, client program name, or any specified exceptions

Database Firewall policies focus on the positive enforcement model of whitelists, but can also support blacklists to disallow specific SQL statements. The firewall policy editor can be used to design custom policies quickly and efficiently. Using the firewall policy editor, you can create sets of IP addresses, database users, database clients and OS users, as shown in the Figure 8.3. These sets help arrange the expected session attributes into groups that would be allowed or denied access.

Sets of Session Attributes	Description
IP Address Sets	Specified set of IP addresses of database clients
Database User Sets	Specified set of database user login names
Database Client Sets	Specified set of client programs
OS User Sets	Specified set of operating system user names

Figure 8.3: Sets of Sessions Attributes

Database Firewall policy is essentially a multi-stage rules engine formed by a combination of rules such as Exception Rules, Analyzed SQL Rules and Object Access Policy Rules.

- **Exception Rules:** Exception rules provide powerful means of allowing or denying a SQL request based on sets of sessions attributes, without looking at the specific SQL statement. Exceptions override all other policy rules and determine the action, logging level and threat severity (as shown in the Figure 8.4) to use when certain session data is encountered.

Setting	Options
Action	Pass, Warn, Block
Logging Level	Don't Log, Sample, Once, Unique, Always
Threat Severity	Insignificant, Minor, Moderate, Major, Catastrophic

Figure 8.4: Rule Settings

The Figure 8.5 shows an exception rule that logs and blocks all the SQL statements that do not originate from IP addresses in the Application IP set. One can also provide substitution SQL statements for the SQL traffic received from any unauthorized sets.

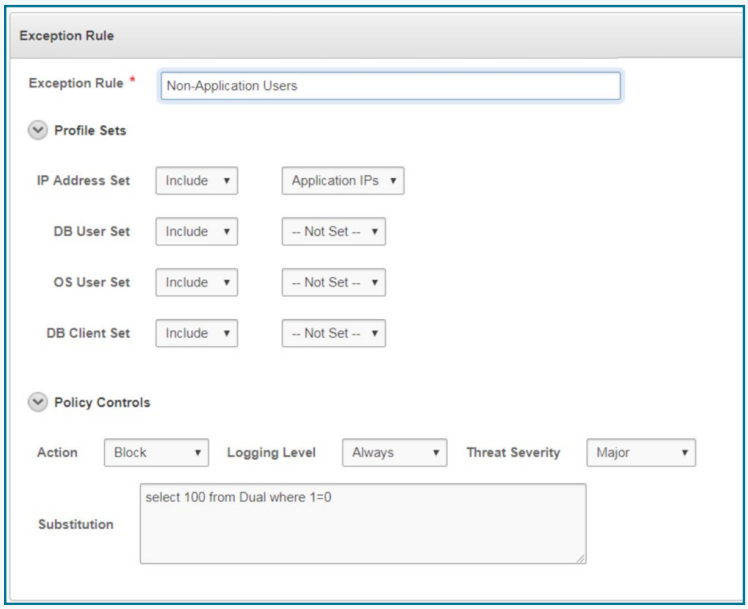


Figure 8.5: Exception Rule Example

Similarly, an exception rule could be used to enable a specific remote administrator, coming from a predetermined IP address, to diagnose an application performance issue without being bound by the rules for "normal" SQL set in the firewall policy. As these examples show, exceptions can be seen either in terms of a whitelist or a blacklist depending on how it is defined.

- **Policy Rules for Analyzed SQL:** These rules are executed after Exception Rule. Database Firewall automatically captures, analyzes actual SQL traffic and puts similar SQL statements into groups known as clusters. You can use the policy editor to quickly set up whitelists of "normal" behavior and/or blacklists of disallowed SQL statements by specifying action, logging level and threat severity for each cluster in the analyzed SQL.

Whitelists of "normal" behavior could also be automatically created by running the firewall and capturing a controlled set of expected SQLs, such as those generated in a test or QA system. Session profiles are created by grouping one or more sets of session attributes. When a session profile is assigned to the policy rules, these rules are applied only when the session data of an incoming SQL statement matches the associated session profile.
For example, when a profile is created, a completely different set of rules for SQL originating from a certain set of database users can be defined. When a user in this database user set, accesses the database, this profile's policy rules are used instead of normal policy rules.

- **Object Access Policy Rules:** Object Access policy can be used to prevent or allow specific types of SQL statements (DML, DCL, etc.) acting on specific database objects such as tables and views. These rules are often used for controlling behavior of DBAs over the network where it might be necessary to stop them from accessing specific database objects. By associating SQL categories with specific objects, Object Access Policy Rule provides easy ways to disable logging for entire classes of objects, such as the underlying database dictionary tables that are not related to the application itself. As with other parts of a firewall policy, Object Access Policy can be seen either in terms of a whitelist or blacklist, depending on how the policy is defined.

ORDER OF EXECUTION OF RULES

The order of Firewall policy rules evaluation on any incoming SQL traffic starts with Exception Rules to see if there is a match and if not, then the analyzed SQL rules are evaluated, followed by the Object Access Policy Rules and finally the Default Rule as shown in Figure 8.6. Once the SQL matches a rule in the firewall policy, the Database Firewall takes the action as defined in the policy.

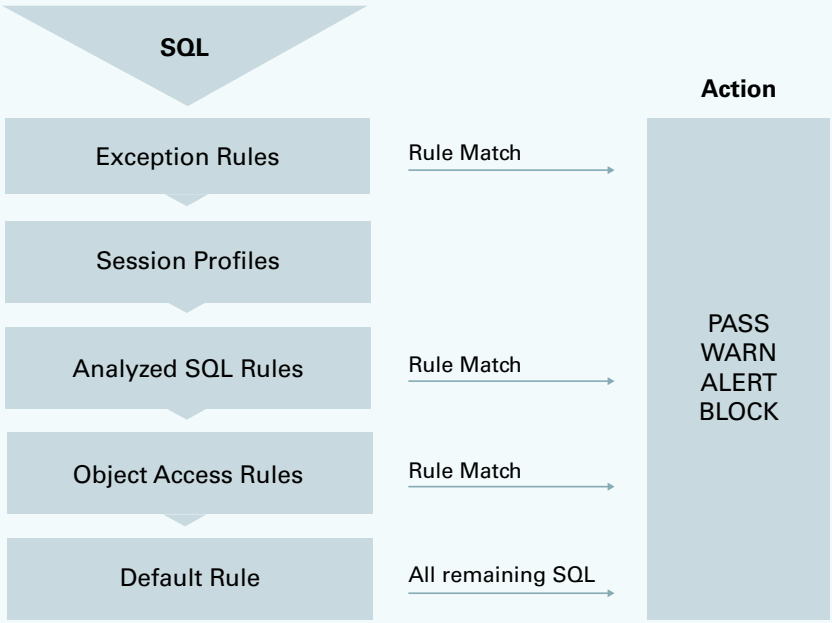


Figure 8.6: Firewall policy rules evaluation order

HANDLING UNAUTHORIZED SQL

When Database Firewall finds an unauthorized SQL statement, it can log and raise alert on the out-of-policy SQL statement, and/or block the SQL statement and take one of the following actions:

- Do nothing after blocking the statement. The actual end-user experience would depend upon how the application handles this case where the database server does not respond.
- Substitute the out-of-policy statement with a new harmless statement that returns success but with no data or returns an error. It minimizes the impact as most robust applications can deal with no data coming back or an error being returned.
- Drop the connection to the client. This blocks all traffic from that specific connection to the database. This is the most aggressive action, and if the application is using connection pooling, this will impact all users using the connection pool.

DATABASE FIREWALL REPORTS

Database Firewall sends alerts and event logs to Audit Vault Server. In addition, rule-based alerts can be defined on the event data received from the Database Firewall. Alert conditions are flexible and can include more than one event, and the events can come from different databases.

The Database Firewall reports give detailed event information on the SQL traffic to the monitored databases. For example, details of statements that had warnings or were blocked can be seen. General information about SQL traffic to the databases can also be seen, including statement type (DDL, DML, etc.), database username, OS username, client application name, client IP address, and Database Firewall action and threat level.

These Database Firewall reports as shown in Figure 8.7, along with the other built-in audit reports provide a comprehensive view of the security and compliance status of the database environment.

Report Name	Description
Database Traffic Analysis by Client IP	All events grouped by client IP and database
Database Traffic Analysis by OS User	All events grouped by OS user and database
Database Traffic Analysis by Blocked Statements	SQL statements blocked by the DB Firewall
Database Traffic Analysis by Warned Statements	SQL statements marked as WARN by the Database Firewall
Database Traffic Analysis by Invalid Statements	SQL statements marked as INVALID by the Database Firewall

Figure 8.7: Database Firewall Reports

MONITORING AND BLOCKING MODES

Depending on the required protection level, Oracle Database Firewall can operate in one of the following modes:

- **Database Activity Monitoring (DAM):** In this mode, Database Firewall monitors the SQL traffic to the database and raises alerts based on the firewall policy. It sends event logs to Audit Vault Server, enabling reports to span information observed on the network alongside audit information from the databases, operating systems, and directories. Organizations can use this passive monitoring mode to alert security operations team of unexpected activity and utilize this audit trail to address compliance requirements.
- **Database Policy Enforcement (DPE):** In this active monitoring mode, Database Firewall acts as an application layer firewall, transparently intercepting the incoming SQL traffic and analyzing the SQL payload in the TCP packets. In addition to monitoring, logging, and alerting, a variety of attacks can be prevented by blocking the SQL statements violating the firewall policy and optionally substituting them with harmless SQL. Support for policies based on whitelists and blacklists provides a high degree of deployment flexibility.

DATABASE FIREWALL NETWORK DEPLOYMENT

Database Firewall can be deployed in one of the following ways shown in the Figure 8.8:

- **Proxy:** Database Firewall can be configured as a proxy such that all traffic to the database server is routed through the Database firewall. This requires the database server IP address/port on the database client or application to be changed to the IP address/port of the Database Firewall proxy, along with changes to the database listener to reject direct connections. Most enterprise network switches and traditional firewalls can also be used to redirect database traffic to a Database Firewall proxy port, allowing SQL traffic to be protected without any changes to database clients or applications. The proxy mode of deployment supports both active and passive monitoring of database activity.
- **Out-of-Band**: In this passive mode of monitoring, Database Firewall receives a copy of the network traffic, including client requests to the database and the database's response to those requests. Spanning ports, network taps, and packet replicators can be used to copy database traffic to the firewall. In this mode, the Database Firewall can monitor and alert on SQL traffic, but cannot block or substitute SQL statements.
- **Host Monitor:** Database Firewall supports a local server-side agent. Host Monitor, part of Audit Vault Agent, captures SQL traffic reaching the database server and securely forwards it to Database Firewall.
- **In-line (bridge):** In this mode, the Database Firewall is inserted into the network in a segment that lies between database client application servers and the databases being protected. The network must be configured to route the traffic to the Database Firewall which intercepts traffic, analyzes the same, makes an explicit outbound connection to Database Server, and forwards all the received data. In this mode, the Database Firewall can both monitor and block SQL, as well as optionally substitute SQL statements. While this is possible, we don't recommend this mode as this is deprecated and instead the proxy mode is recommended.

Deployment can be customized with monitoring and blocking on some databases and monitoring only on other databases. Database Firewall can also be configured in a High Availability mode for fault tolerance.

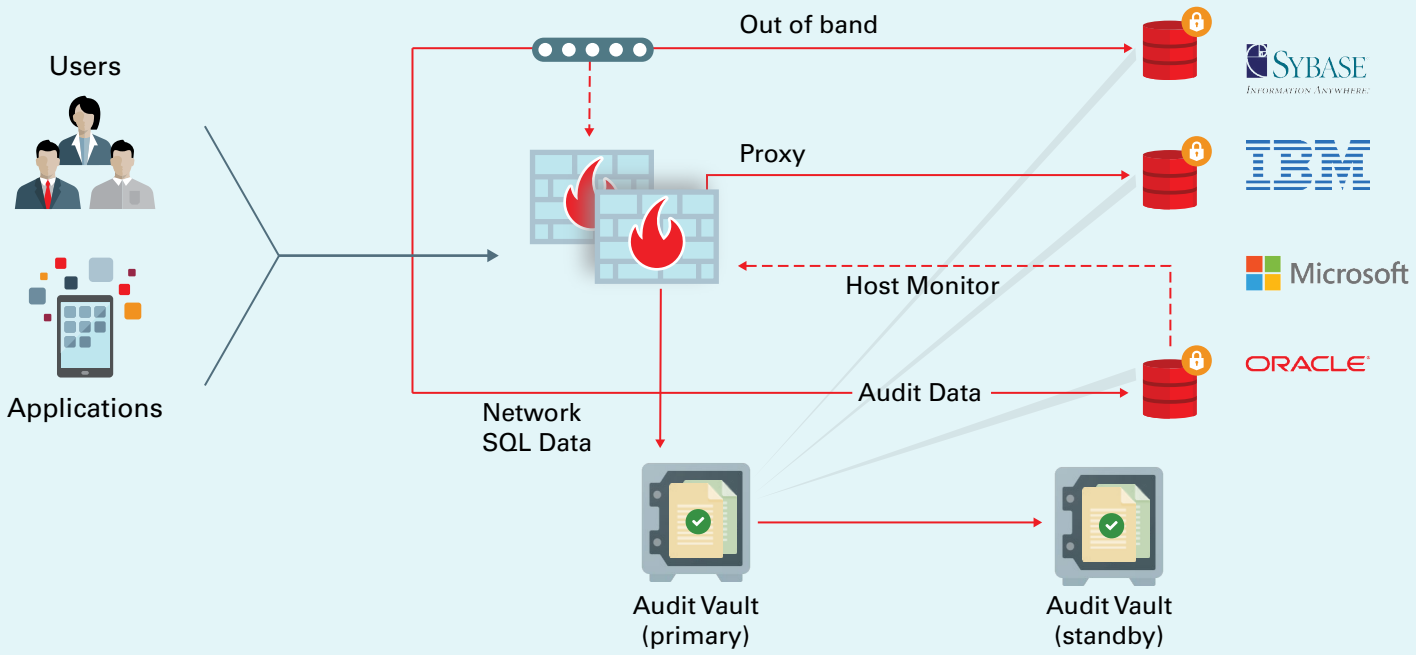


Figure 8.8: Oracle Audit Vault and Database Firewall Deployment

SUMMARY

The increasing number of attacks on databases, via SQL injection or misused credentials of insiders, has made database activity monitoring an important component of the overall security architecture. Oracle Database Firewall provides a sophisticated next-generation SQL grammar analysis engine that inspects SQL statements going to the database and determines with high accuracy whether to allow, log, alert, substitute, or block the SQL.

Oracle Database Firewall supports whitelist, blacklist, and exception list based polices. Policies can be enforced based upon attributes, including SQL category, application, user, and IP address. This flexibility, combined with highly accurate SQL grammar analysis, enables organizations to monitor database activity, minimize false alerts, and only collect data that is important. Database Firewall events are logged to the Oracle Audit Vault Server enabling reports to span information observed on the network alongside audit data, providing a complete picture of the security and compliance status of the environment.

9

Data-Driven Application Authorization

CONTROLLING ACCESS TO APPLICATION DATA

The easiest way to attack a database is through the browser accessing applications that connect with the database. The attacker could take advantage of the SQL injection vulnerabilities within the application, or any inconsistencies in the way the application authorizes its users. Today's applications have become highly complex with different authorization policies based upon the session attributes, user attributes, organization attributes, their role, and so on. Implementing appropriate access control checks at all the right places is quite cumbersome, and difficult to maintain over the lifecycle of the application.

Application users should have access to only the tables containing data they need to perform their tasks. However, when an object privilege such as `SELECT` or `INSERT` is granted to a database user for a specific table, the privilege provides access to everything within that table. Database tables for most applications, however, 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 other customer 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 image shows a table with both sensitive and non-sensitive HR data. Nancy is able to see her sensitive data and also the salaries of her direct reports. But all other sensitive data is hidden.

Name	Manager	SSN	Salary	Phone Number
Adam Fripp	Steven Stiles			650-123-4234
Neena Kochhar	Steven Stiles			650-124-8234
Nancy Greenberg	**Neena Kochhar**	**000-51-4569**	**120300**	**515-123-4567**
Luis Popp	Nancy Greenberg		69000	515-123-4234
John Chen	Nancy Greenberg		82000	515-123-8181
Daniel Faviet	Nancy Greenberg		90000	515-123-7777

Figure 9.1: Example: Fine Grained Access Control Applied to a Table

Applications typically manage and enforce their own user authorizations, but these authorizations do not apply to other applications accessing the same database tables, or a reporting tool with direct access to the database.

This chapter discusses how applications have typically handled this fine-grained authorization problem. The challenge is implementing these policies in an easy to understand way, without coding these rules within the application itself.

CHALLENGES WITH APPLICATION AUTHORIZATION

Consider the `EMPLOYEE` table in the `HR` sample table earlier where it might be appropriate to allow all users to view the basic information about employee names and office phone numbers, but not information about `SALARY` and `SSN` columns. Managers should be able to access the `SALARY` column for their direct reports. It is common for applications to write complex application authorization code to determine which rows and columns each user has access to, and under what conditions. An extra database schema or set of tables is typically dedicated to the user and role authorization information, which is then used to build the SQL that is generated for a particular application user request.

Some common problems with this approach:

- An application developer has to write this complex code, and ensure that the application logic is applied to all scenarios where the application user needs to access a given table.
- Every application accessing the same data must re-implement their version of the authorization policy as the database itself knows nothing about this security policy.
- Tools that are given direct access to 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.

Oracle Database addresses this challenge with several technologies known collectively as Data-Driven Authorization. Centralizing these application authorization controls in the database simplifies and accelerates application development, and provides a single set of policies to define and maintain.

Oracle introduced the well-known feature called Virtual Private Database (VPD), an automated predicate-based row-filtering security technology, twenty years ago – at that time, it was the only database with this innovative capability. Following that, Oracle Label Security (OLS) was introduced to automatically filter rows based upon data and user labels. The most recent technology in this line-up is Real Application Security (RAS) introduced with Oracle Database 12c which 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 different rows depending on the operation the end user is performing. This is useful, for example, to allow users to view information about all employees but only update their own rows. A VPD policy can include sensitive columns so that only certain end users under certain conditions can get access to its values 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, the `SALARY` data is hidden. When users execute such a generic query without specifying any condition or `WHERE` clause, they still only get the rows and

columns that they are authorized to get. This significantly simplifies application development as the developer does not have to think about how they need to change the query depending upon 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 9.2: 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 data item that describes its sensitivity or importance. In many government and corporate environments, a document might be labeled as TOP SECRET or INTERNAL USE ONLY, and 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 that he or she is permitted to access. This capability when attached to the rows of a table, allows the database to inherently know which data is sensitive and restrict access to it accordingly.

Oracle Label Security (OLS) simplifies the process of assigning labels to data and users and enforces access control based on those labels. Associated with every row in a table protected by OLS is a label that indicates the sensitivity of the data. The label for each row can be set explicitly based on business logic, but more often the system sets the label automatically based on the label of the application or user session that inserted the row. The label of the user session, in turn, is calculated from a variety of factors, including the label assigned to the user, the session, the type of connection to the database, and so on. A label can be thought of as 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 can be used to segregate information based on attributes such as project or department. Compartments are used to segregate and compartmentalize data such as special project information. Groups provide a convenient way to represent hierarchical authorization based on 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 are allowed to access the quarterly financial data and reports which have not yet been released. A Finance compartment has been defined to only allow access to those with the need to see financial data. Upcoming pricing changes are also sensitive so a Pricing compartment protects pricing data. Groups are used to represent the retail stores' geographic hierarchical structure.

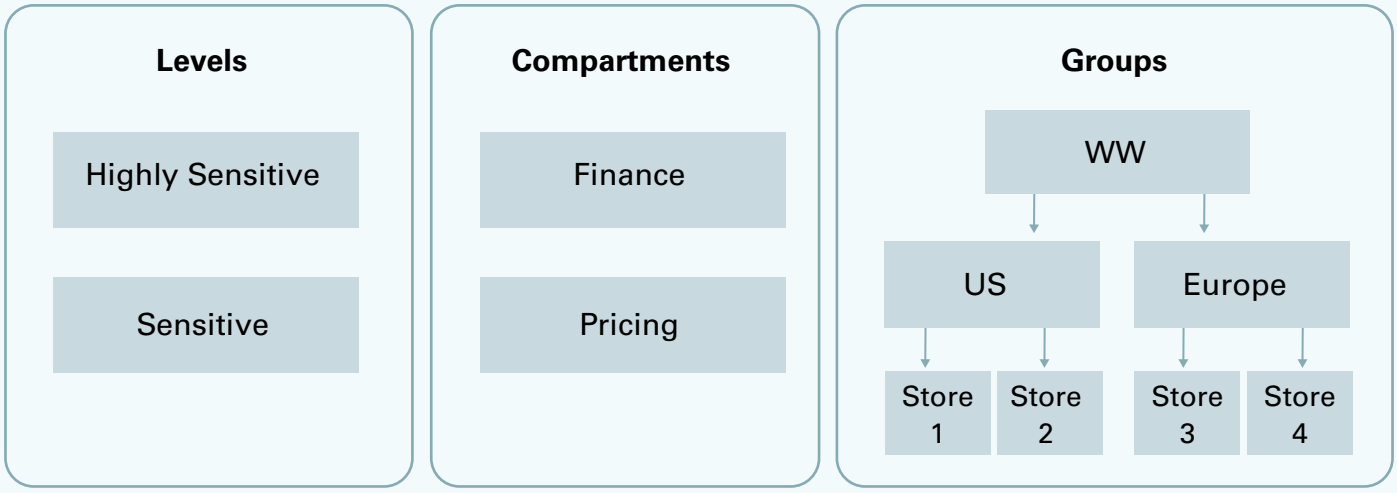


Figure 9.3: Oracle Label Security Components – Levels, Compartments, Groups

Labels are represented by the three components separated by a colon (:). A data label that is 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, he/she will have the Pricing compartment as part of his/her user label. We will review how labels work by starting the example with the group component, then adding in compartment and level.

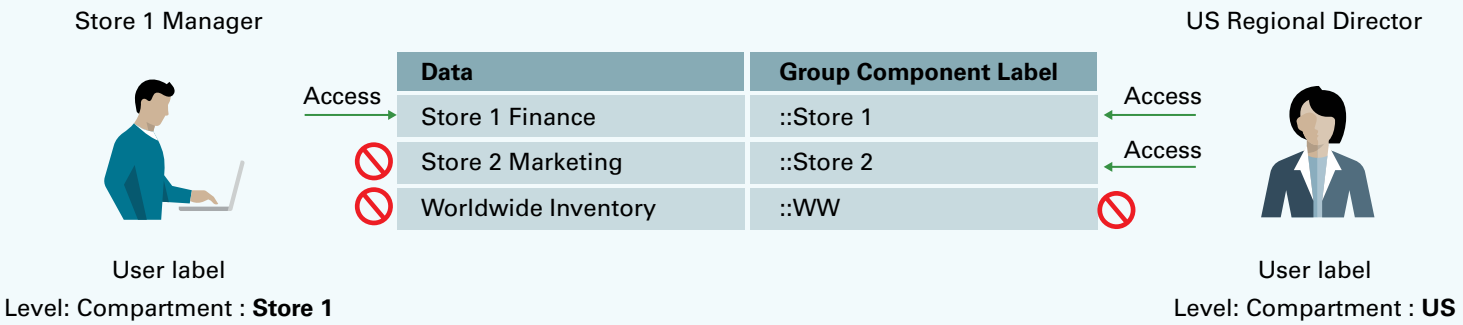


Figure 9.4: Using the Group Component

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 manager would have 'US' as their group label and this allows 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 needs 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.

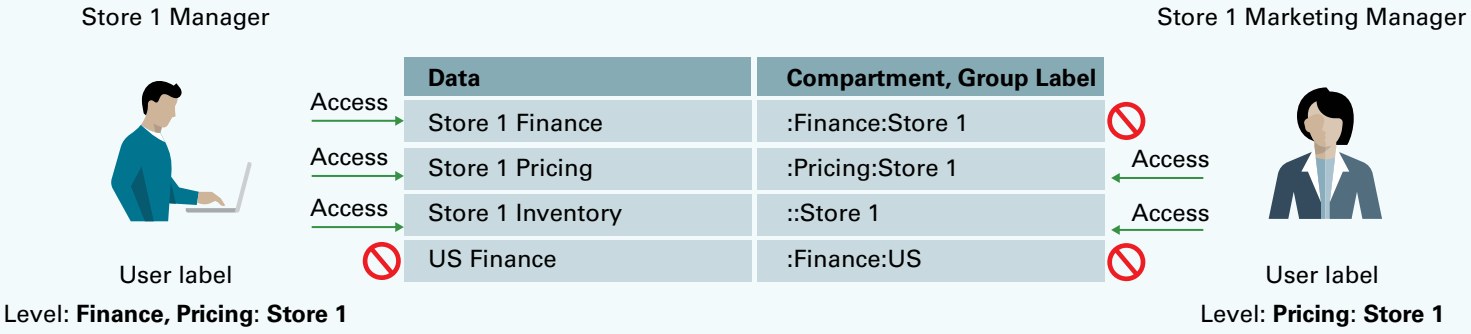


Figure 9.5: Using the Compartment and Group Components

Since this is a public company, all non-released quarterly financial data and reports are considered highly sensitive. All data was 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 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 Sensitive level label can access data that is 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.

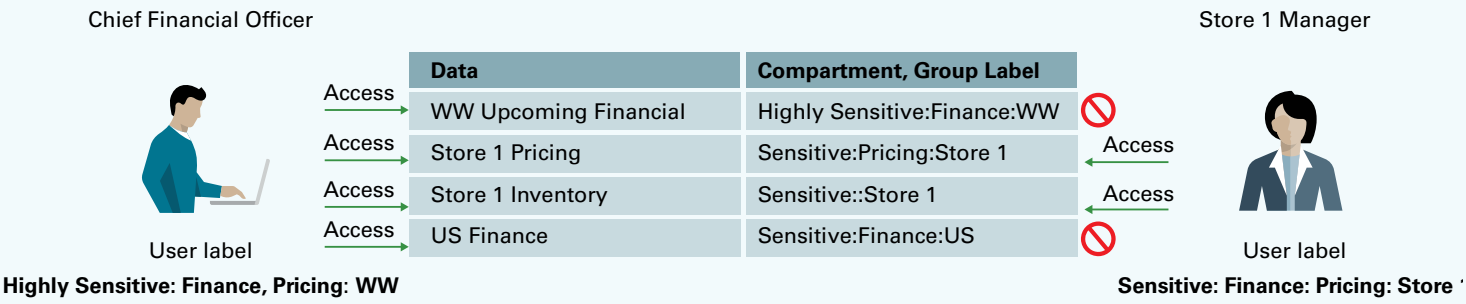


Figure 9.6: Using the Level, Compartment, and Group Components

While Label Security has all the functionality to meet government, military and intelligence agency requirements for labeling, commercial organizations have also found Label Security to simplify 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 to do along with a high price tag for maintaining many duplicate systems. Label Security was used to consolidate the data into a single database while maintaining the strong controls on data access.

The key advantage of using Oracle Label Security is a full authorization infrastructure with mediation and simplified administration. OLS provides the same type of row-level access control as VPD but implements the concept and management of user and data levels, compartments, and groups which map closely to many real-world use cases and does 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 should be able to access the data in each individual 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 by users. In modern three-tier applications, the application end users do not interact directly with the database. Database queries and updates are typically performed by a single database user that represents the entire application rather than the individual end user.

Because the end user's identity is unknown to the database, per-user access control policies must be enforced by the application instead. Besides requiring extra software development in the application, this approach can lead to inconsistent enforcement, especially when there are elements of the system 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 framework within the database, enabling three-tier and two-tier applications to declaratively define, provision, and enforce their 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 on both static and dynamic collections of records representing business objects. RAS overcomes the maintainability, scalability and security issues faced by complex VPD policies, and also provide support for 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 so that access control policies can then be enforced within the database itself. 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, however, does not have its own schema to store data objects or dedicated connection to the database as a regular database user would. RAS controls can also enforce access control policies on database users.

RAS enforces fine-grained restrictions on access to columns and rows within a database table, just like VPD. However, RAS uses a more general, declarative syntax to specify these restrictions. First, the administrator creates one or more "data-realms" for each table to be protected. 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, each shown through a different color. The '`public`' data-realm includes all employee records. The '`self`' data-realm is dynamic and only includes the user's record. The '`manager`' data-realm is also dynamic and includes all the employees that report to the user.

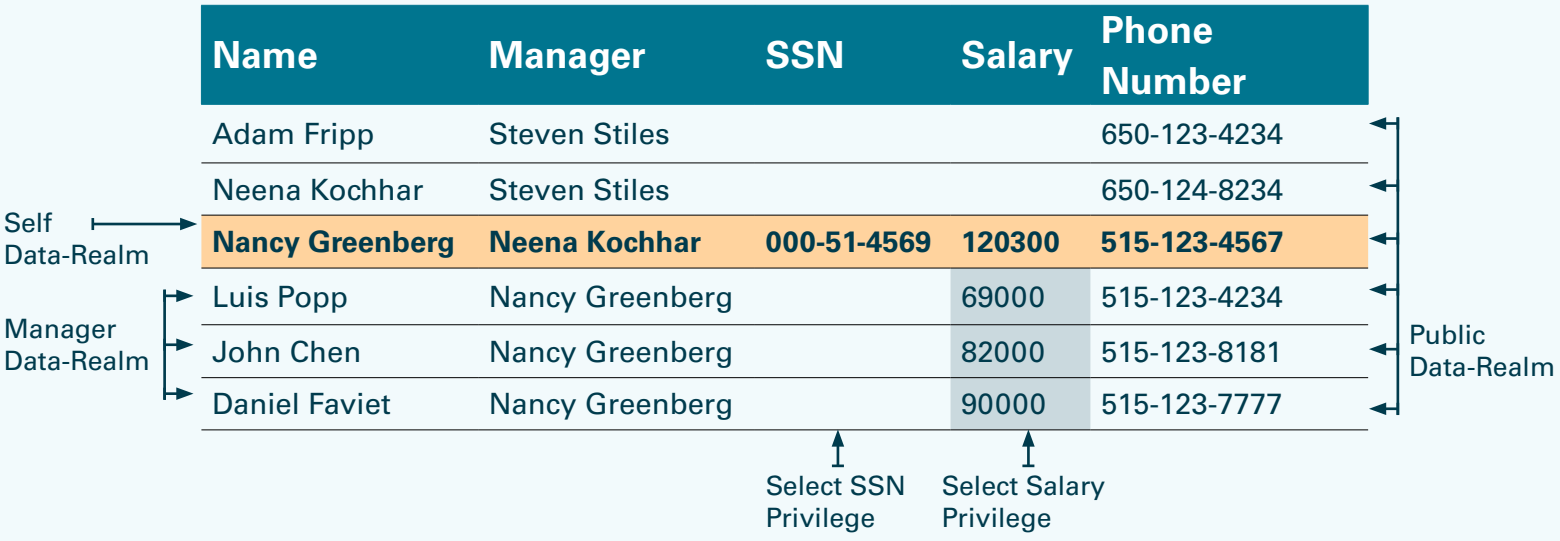


Figure 9.7: Controlling Data Access with RAS

For each of these data-realms, an Access Control List (ACL) is attached that specifies who can access that data-realm and under what condition. For the `'public'` data-realm, all employees have the '`SELECT`' privilege only. All non-protected data would be visible to any employee. However, since `SSN` and `Salary` data is protected, it doesn't show up since it's not included in the ACL for the 'public' data-realm. "The `'self'` data-realm, seen below in Figure 9.8, is associated with an ACL to view SSN (Authorize SSN) and Salary (Authorize Salary) data so an employee (role) can view their own sensitive data.

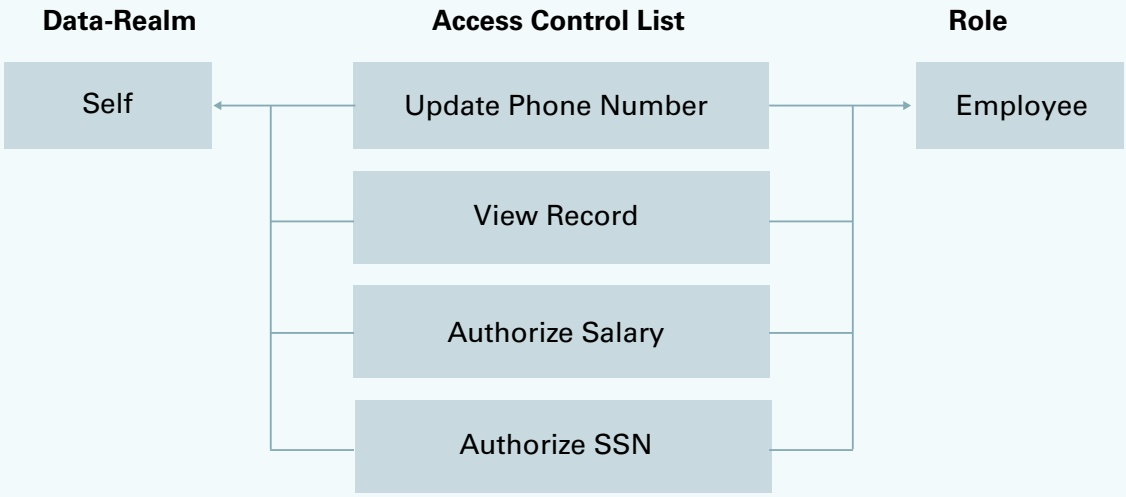


Figure 9.8: Access Control List (ACL)

A manager can view the salary of their reports using the '`manager`' data-realm which has an ACL that allows the `Salary` data to be shown. The '`self`' and `'manager'` data-realms are dynamic -- that is, RAS returns data from appropriate rows and columns based upon the application user identity for that particular session. Furthermore, RAS allows the database to enforce additional security policies that are unique to each application. The application can define its own 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`).

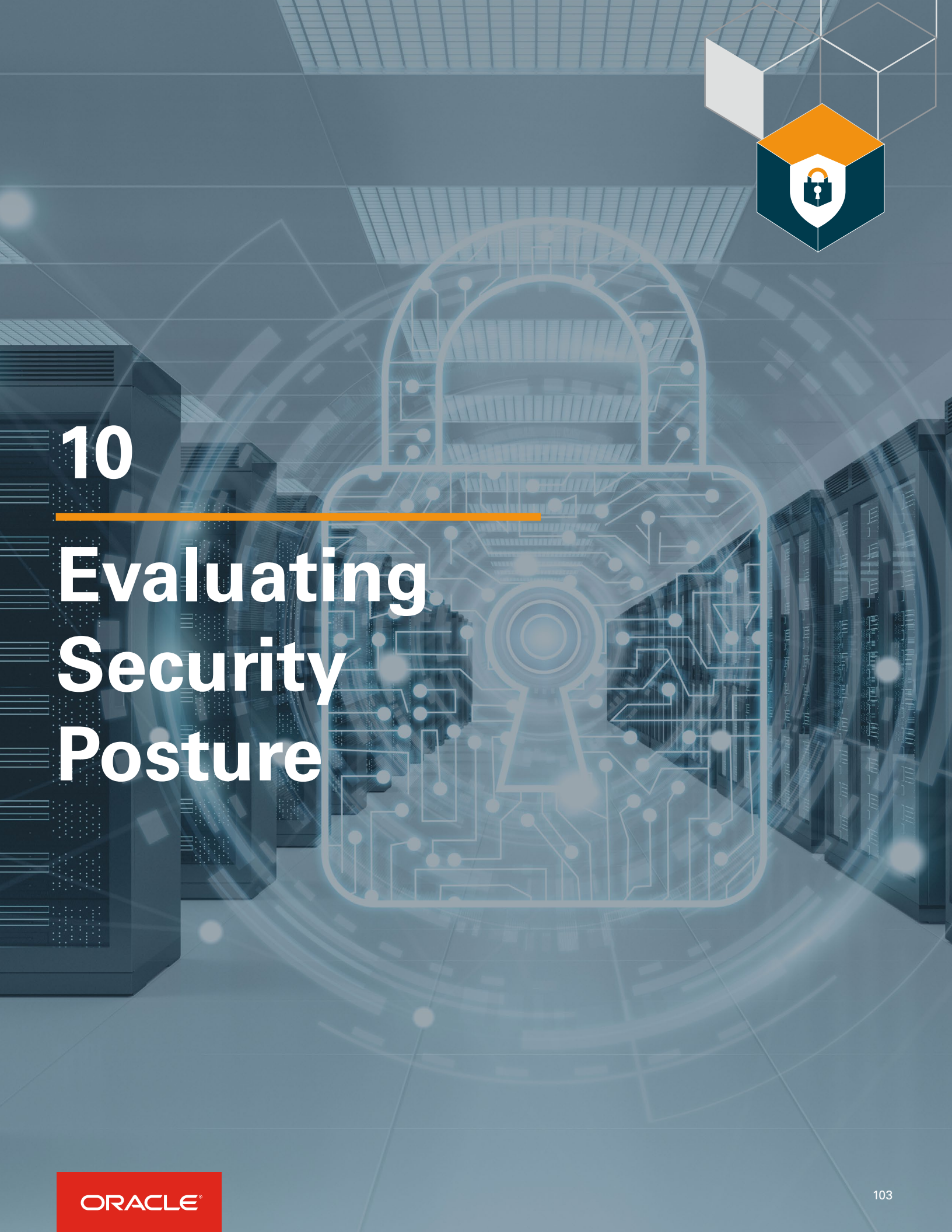
Connecting applications through the RAS Java interface to the database is more secure than using traditional connections to the database. A traditional application connection to the database needs to include every privilege and role that every application user needs to run the application and typically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

These highly privileged accounts are a top target for cyber criminals looking to get into a database. The RAS application user connection to the database uses a minimally privileged account so even if the account is compromised, attackers could not use it to access sensitive data. RAS data-realms, ACLs, and policy management can be done completely 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 with VPD.

SUMMARY

Hackers and malicious insiders can take advantage of weak application authorization policies if they are implemented inconsistently and improperly. Oracle authorization technologies can centralize and secure all database clients including applications, analysis tools and command-line tools. Centralized application authorization can also help accelerate application development and reduce the complexity of maintaining and upgrading multiple authorization policies in different applications.



10

Evaluating Security Posture

NEED FOR SECURITY ASSESSMENT

Today's systems have become very sophisticated with many configuration settings. As recent data breaches have demonstrated, it is critical to have properly configured and secure systems. A human error could potentially leave your database open to everyone, or an attacker could maliciously exploit vulnerabilities in configuration to gain unauthorized access to sensitive data. By overlooking basic security controls, organizations might end up losing their customer names, addresses, date of birth, account information, and other personal data impacting their reputation and bottom line. It is important to scan the database to ensure that it is securely configured, and to remediate the situation if there is a deviation.

Security configuration assessment has also become an important part of many regulations such as the EU GDPR, PCI DSS, Sarbanes-Oxley, and numerous breach notification laws. Various organizations such as the Center for Internet Security (CIS) and U.S. Department of Defense also have recommendations for security configuration best practices. The importance of configuration cannot be understated as new regulations are being released and existing regulations are evolving. As a result, having a solution that can adapt to new regulations is critical.

This chapter describes how to quickly evaluate your database security posture and come up with a strategy to keep your databases secure.

EVALUATE AND ASSESS DATABASE CONFIGURATION

Attackers take their time to prepare the attack and spend considerable time doing reconnaissance. They use several 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 identify the weakest links and then determine the next steps. In essence, the attackers are evaluating your current security status to find out the easiest way to get to the data.

- 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?
- Is auditing on? For whom? Which conditions?
- Which packaged applications are running on this database? Are those running with all powerful privileges?
- Is the data encrypted? If not, can we get access to the underlying storage or a backup?

All these questions are inside the hacker's mind, and the answers help them come up with a plan to break into the database and the data. Organizations, as the owners of the data, need to do similar thinking, but with the aim to improve the security posture.

Properly hardening and securing a database is not an easy task. It needs an understanding of the users and their roles, the data and its sensitivity, the security configuration parameters, the enabled features, knowledge about the database attack vectors, and finally the available security controls to protect it. As the Oracle Database is highly customizable, it also needs an understanding of the impact of configuration choices on the overall security.

Here are the key considerations on protecting your databases:

- Almost all databases hold sensitive data, but the level of importance may differ. For example, date of birth may be more sensitive than your email address. It is important to find out which databases contain what type of sensitive data so that controls can be put in place accordingly.
- Database vulnerabilities include vulnerabilities from unpatched systems, application design, weak credentials, excessive privileges, lack of a trusted path, no separation of duties, no encryption, and no auditing.
- Security configuration parameters are tightly related to how the database behaves and require an understanding of what the parameters are, what they do, the impact of changing them, and their dependencies.
- Not all database users are equal. Apart from the DBAs there are several other actors / processes that need to interact with your data through an account – the application itself, application administrators, security administrators, and others like service accounts, batch programs, etc. Clearly identifying the different types of users and the different types of activities they need to execute on the database will help to properly manage privileges and roles and implement the principle of least privilege.
- Not all databases are created equal and rarely take into consideration the controls needed to protect sensitive data. The database might enforce a certain password complexity factor but may not protect customer table data out-of-the-box. Attackers exploit this "vulnerability" that comes with installing a system by default, or databases that are badly configured.

QUICKLY EVALUATING THE SECURITY STATE OF ORACLE DATABASES

The Oracle Database Security Assessment Tool (DBSAT) identifies potential sensitive data and areas where your database configuration, operation, or implementation introduces risk. DBSAT collects and analyzes different types of data from the database. DBSAT further 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 quickly assess the current security status of a database and recommends best practices. Updated best practices rules are delivered periodically with new versions of the tool.

One challenge that comes with doing database security assessments and vulnerability assessments is that the data about the vulnerabilities becomes unmanageable. A lot of findings are presented and it is hard to find a way to prioritize and act upon critical findings first.

DBSAT not only scans the database for weaknesses and vulnerabilities but also indicates the priority level to help prioritize work on the most critical weaknesses first. DBSAT also provide high-level details, along with specific recommendations for each of the issues, making it simpler and quicker to act.

DBSAT is a free tool available to all Oracle customers so that they can quickly find sensitive data, evaluate their Database Security posture, identify gaps, and implement the recommended security best practices for their organization. DBSAT is also used by Oracle consultants and partners while executing Database security assessments.

DBSAT COMPONENTS

DBSAT has three components: Collector, Reporter, and Discoverer. The Collector and Reporter are used for generating Database security risk assessments, and the Discoverer to discover the different types of sensitive data in the database.

The Collector first gathers security configuration information from the database and underlying OS. The Reporter then analyzes the collected data and generates detailed findings and recommendations. The output reports are in HTML, spreadsheet, text and JSON formats. The Discoverer (described earlier in Chapter 5) helps to identify sensitive data by looking into table metadata (comments and column names), classifies, and summarizes the findings in HTML and spreadsheet reports.

The HTML reports provide detailed results of the assessment 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 usage. The JSON output is convenient for data aggregation and integration purposes.

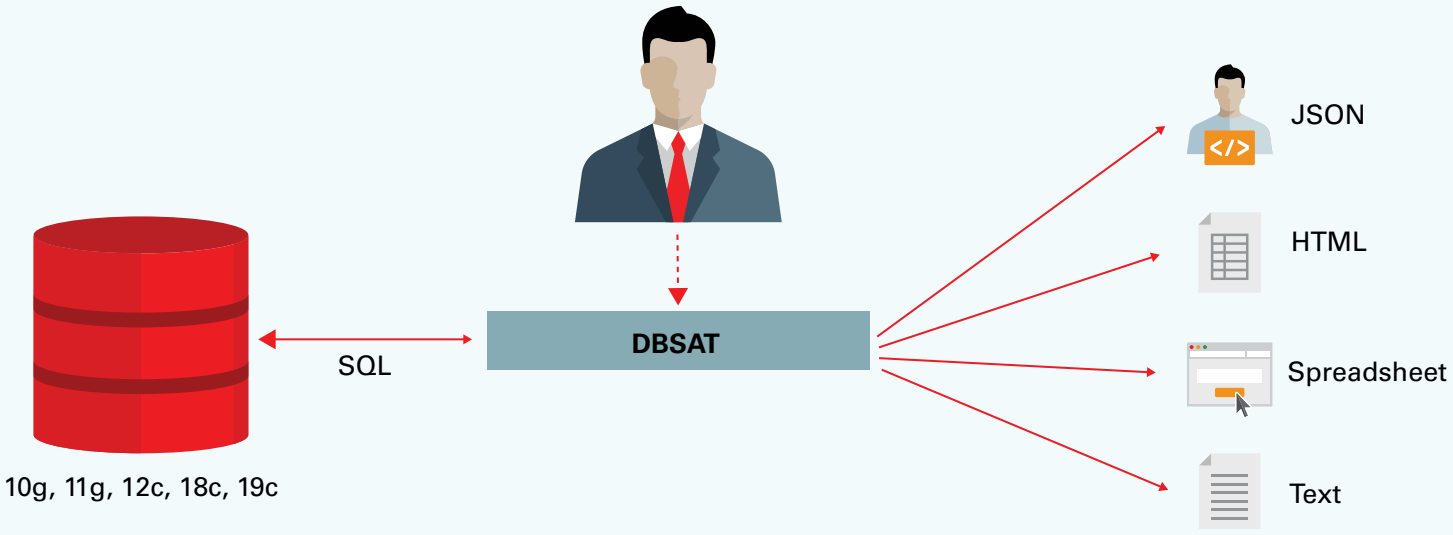
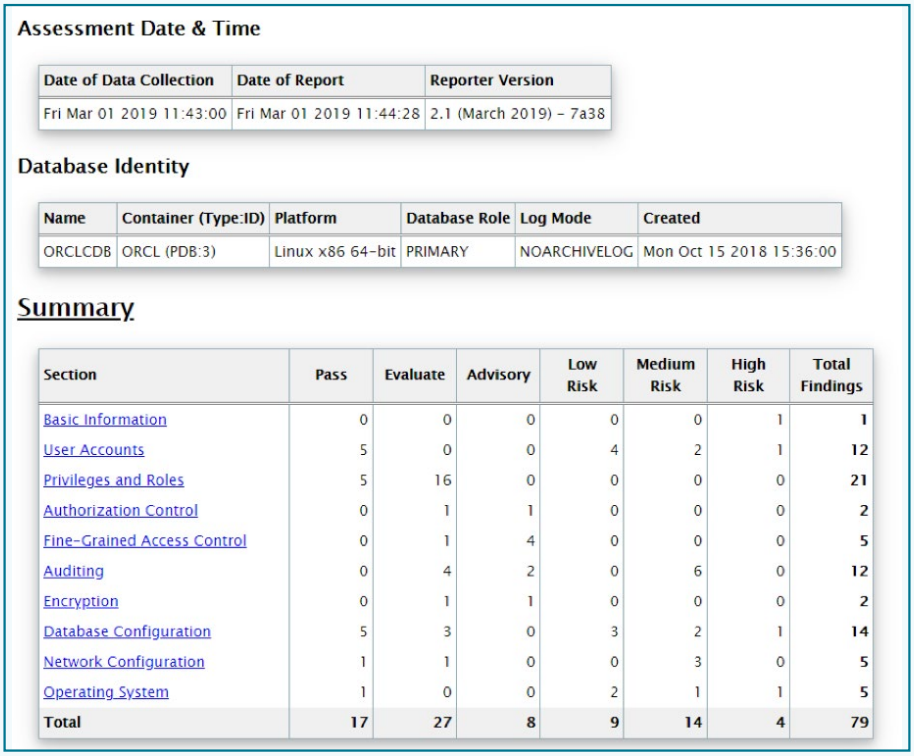


Figure 10.1: Database Security Assessment Tool

The following figure displays the Database Security Assessment report.



Assessment Date & Time

Date of Data Collection	Date of Report	Reporter Version
Fri Mar 01 2019 11:43:00	Fri Mar 01 2019 11:44:28	2.1 (March 2019) - 7a38

Database Identity

Name	Container (Type:ID)	Platform	Database Role	Log Mode	Created
ORCLCDB	ORCL (PDB:3)	Linux x86 64-bit	PRIMARY	NOARCHIVELOG	Mon Oct 15 2018 15:36:00

Summary

Section	Pass	Evaluate	Advisory	Low Risk	Medium Risk	High Risk	Total Findings
Basic Information	0	0	0	0	0	1	1
User Accounts	5	0	0	4	2	1	12
Privileges and Roles	5	16	0	0	0	0	21
Authorization Control	0	1	1	0	0	0	2
Fine-Grained Access Control	0	1	4	0	0	0	5
Auditing	0	4	2	0	6	0	12
Encryption	0	1	1	0	0	0	2
Database Configuration	5	3	0	3	2	1	14
Network Configuration	1	1	0	0	3	0	5
Operating System	1	0	0	2	1	1	5
Total	**17**	**27**	**8**	**9**	**14**	**4**	**79**

Figure 10.2: Database Security Assessment Summary

The resulting analysis is reported in units called Findings and consists of the following:

- 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 that is not currently in use. In cases where further analysis is needed, the status is shown as "Evaluate."
- Summary – Presents a brief summary of the finding.
- Details – Presents the details of the results, followed by recommendations.
- Remarks – Explains the reason for the rule and recommended actions for remediation.
- References –When applicable it will reference the corresponding CIS Oracle Database 12c benchmark recommendation, Oracle Database STIG Rule and the EU GDPR article/recital.

In the Finding below, DBSAT identified that there are five 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 CIS Oracle Database 12c Benchmark recommendation 4.4.4

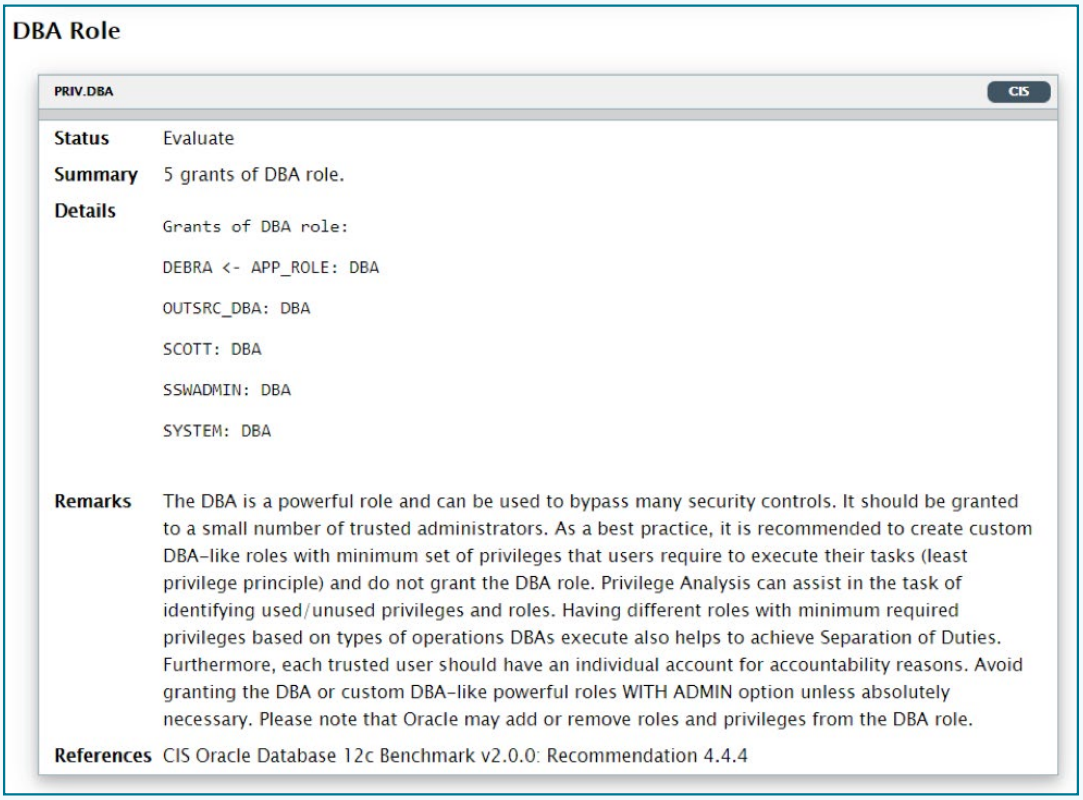


Figure 10.3: Sample Finding – Users with DBA Role

DISCOVERING SENSITIVE DATA WITH DBSAT

Part of DBSAT, Discoverer helps to identify which tables and columns contain sensitive data and how many rows with sensitive data there are. To learn more about discovering sensitive data with DBSAT please refer to "Discovering Sensitive Data", Chapter 5.

DATABASE SECURITY ASSESSMENT IN THE CLOUD

Customer with database on Oracle Cloud can use Data Safe, a data security cloud service that provides a comprehensive suite of security capabilities, including user and security assessment. Tightly integrated assessment capabilities help identify and fix vulnerabilities effectively. See chapter 13 for more information on how Oracle Data Safe can help meet data security requirements in Oracle Cloud including user and security assessments.

ENTERPRISE LEVEL MONITORING AND ASSESSMENT

While DBSAT collects configuration information and identifies gaps along with recommendations, Oracle also provides the Enterprise Manager Database Lifecycle Management (DBLM) to address your enterprise needs for assessing security configuration. DBLM helps database, system and application administrators automate the processes required to manage the Oracle Database lifecycle processes. DBLM provides numerous reports for security configuration checks as well as a comprehensive compliance framework. Reports include information on initialization parameters, operating system directory permissions, user account profiles, and sensitive object reports.

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

Figure 10.4: Oracle Enterprise Manager General Security Reports

Oracle Enterprise Manager 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, the trend analysis allows fine-grained tracking of compliance scores over time.

Figure 10.5: STIG Compliance Standard detail

COMPLIANCE FRAMEWORKS, STANDARDS, AND RULES

DBLM provides an extensible framework that supports the compliance frameworks, standards, and rules. This extensibility enables organizations to adapt to the ever-changing world of compliance and also create custom configuration scans to test systems against internal best practices (see figure 10.6 for details).

Compliance Framework	A compliance framework is an industry-specified best- practices guideline that deals with the underlying IT infrastructure, applications, and processes. e.g. PCI
Compliance Standard	A compliance standard is a collection of checks or rules for a specific target type. They represent best practices and help maintain consistency across systems
Compliance Standard Rule	A compliance standard rule is a test to determine a specific configuration parameter or real-time status

Figure 10.6: Descriptions of compliance frameworks, standards, and rules

ASSET DISCOVERY AND GROUPING

The Enterprise Manager DBLM Pack eliminates the need to manually track IT assets including databases. 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, easily perform administrative operations against the targets, and consolidate and monitor your distributed targets as one logical entity. For example, you can define a group called TEST that contains all applications, databases, and hosts targets within your test environment.

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

- Quickly determine the overall 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 turned on auditing.

REAL-TIME MONITORING

Oracle DBLM Pack enables real-time monitoring to monitor any action that can happen against a file (log, configuration, binary), a database object, or a Microsoft Windows Registry key. The real-time aspect of this monitoring means that it captures the action as it happens.

Results from real-time monitoring can be reconciled with a Change Management system to determine if an action was authorized or not.

Enterprise Manager can be configured to send an email and notify in the event of a configuration change. Notifications can perform actions such as executing operating system commands (including scripts) and PL/SQL procedures allowing you to automate IT practices. Enterprise Manager can also send SNMP messages for events published in Oracle Enterprise Manager, such as when a certain metric has exceeded a threshold.

SECURITY PATCH MANAGEMENT

The timely application of patches is necessary for organizations to maintain a proper security posture. In order to give customers choice and flexibility, Oracle provides different patch strategies: Patch Set Updates (PSU), Bundle Patches (BP) and more recently Release Updates (RU) and Release Update Revisions (RUR).

- PSUs contain a cumulative collection of fixes for proven high impact bugs encountered in the field plus the security fixes that are released as part of the Critical Patch Update program.
- BPs are a superset of the PSU and include a cumulative collection of fixes to address bugs in a given feature, product, or configuration. They are also delivered on a pre-defined schedule and may be more frequent than PSUs. Customers should choose the best strategy that adapts to their needs and organization and follow it (either PSU or BP).

- Starting July 2017 for the Oracle Database and Grid Infrastructure release 12.2 and beyond, Oracle provides quarterly Release Updates (RU) and Release Update Revisions (RUR). RUs are proactive, highly tested bundles of critical fixes which enable customers to avoid known issues. RURs contain security and regression fixes to a RU that extend the RU's lifetime up to two quarters. They are specific to a particular RU. Both RUs and RURs contain all security fixes, thus eliminating any 'security' versus 'stability' tradeoff.

Oracle strongly recommends to keep your database up to date to fix known security vulnerabilities and minimize the risk of a successful attack.

PATCHING THROUGH DBLM

Patching requires planning as the process is complex, and might lead to downtime. The EM DBLM supports the entire Patch Management Lifecycle including patch advisories, pre-deployment analysis, rollout, and reporting. It is integrated with My Oracle Support to provide a synchronized view of available and recommended patches.

SUMMARY

Knowing where sensitive data is, and how the database is configured is the foundation for a defense-in-depth strategy. Further, the configuration drifts need to be monitored, the database needs to be patched, and finally, appropriate controls need to be put into place. No system is 100% secure but overlooking the basics will only make life easier for attackers.

Oracle Database Security Assessment Tool and Oracle Database Lifecycle Management pack provide the tools to help you identify sensitive data and areas where your database configuration, operation, or implementation introduces risk.

11

EU GDPR and Database Security

EU GDPR AND DATABASE SECURITY

Executives tend to be concerned about database security for two main reasons: First, they need to minimize the risk of experiencing a data breach, and second, they need to comply with the requirements of national/state laws, industry regulations, and contractual agreements. Organizations must consider a host of laws that may apply to their activities such as the European Union's General Data Protection Regulation (EU GDPR), the California Consumer Privacy Act (CCPA), industry regulations such as the Payment Card Industry Council's Data Security Standard (PCI-DSS), services contracts stipulating ISO 27001 compliance, or Defense Information Security Agency (DISA) Secure Technical Implementation Guide (STIG) standards.

Organizations typically must additionally consider dozens of different regulations for data privacy and data security. Privacy laws regulate the collection, storage, sharing, control, and use of personal information about individuals, also called personal data or personally identifiable information (PII). There are hundreds of data privacy regulations in the United States alone at both the state and federal level dealing with PII. Globally, more than 100 different governments have enacted privacy legislation and many more have pending bills or initiatives. Since the last edition of this book, we've seen the EU GDPR widely enforced, with significant fines of hundreds of millions of Euros levied. The global march of privacy regulations continued with new laws in Brazil (LGPD), strengthening of privacy protection in Japan, significant revisions to Canada's data privacy structure, and much more.

Because EU GDPR is one of the most comprehensive in its coverage, it is of interest to many organizations. Several other countries model their data privacy laws on the framework provided by EU GDPR and the concepts being discussed in the context of EU GDPR are applicable for most other data privacy regulations. For example, the California Consumer Privacy Act (CCPA) contains many of the same provisions and protections of EU GDPR. Brazil's Lei Geral de Proteção de Dados (LGPD) and Thailand's Personal Data Protection Act (PDPA) also closely follow EU GDPR and are similar in terms of scope and financial penalties for non-compliance. This commonality between various country-specific privacy laws and EU GDPR helps multi-national organizations better manage their compliance requirements. For that reason, it is valuable to have a good working knowledge of the EU GDPR.

The details amongst regulations may vary, and the article numbers and titles will change, but at heart all data privacy regulations attempt to safeguard a class of people's personal information by regulating security practices protecting that data.

There are numerous sources available for these regulations including books, white papers, or other material such as this book that can help you understand them, but none of them should take the place of competent legal advice. You should always consult with your corporate legal counsel to understand the applicability of any law or regulation, and the relevance of security controls to your specific environment. Also, keep in mind that the security controls and practices discussed in this book are important even in the absence of regulations – there are good reasons to reduce security risk and protect data that go way beyond just complying with a given regulation. Most regulations are an attempt by the regulator to encourage conformance with their idea of best practices.

INTRODUCTION TO EU GDPR

Before EU GDPR, each of the European Union countries had their own internal national Data Protection laws, guided by the EU Data Protection Directive. EU GDPR provides a single common privacy framework for safeguarding EU residents from misuse of their data. This common privacy framework reduced the cost of compliance and simplified the process of dealing with personal information for businesses serving residents of the European Union.

EU GDPR has 99 articles and 173 recitals. The Article 29 Working Party, the EU's main privacy watchdog, has also provided numerous documents to advise and clarify EU GDPR requirements. EU GDPR was passed in April 2016 and became effective on 25 May 2018. On that date, the Article 29 Working Party was replaced by the European Data Protection Board (EDPB). The EDPB is an independent body, which aims to contribute to the consistent application of data protection rules throughout the European Union, and promotes cooperation between the EU's data protection authorities.

EU-GDPR is applicable to all companies and organizations, in all industries, whether the company is in Europe or outside of Europe, so long as the company collects, processes, or maintains data about an individual in the EU, including through the use of online tracking tools or when offering goods or services to individuals in the EU.

Under EU GDPR, there are several roles involved:

- **Data Subject** – the individual whose data is being collected
- **Data Controller** – the legal entity, public authority, agency, company or any other body that determines the purposes and means of processing personal data
- **Data Processor** – the legal entity, public authority, agency, company or any other body that may collect, store, maintain or otherwise process data about the Data Subject on behalf of the Data Controller. They are responsible for safeguarding a subject's data and for ensuring that data is used only with the consent of the subject. In many cases, the Data Controller and Data Processor will be the same entity.
- **Data Protection Officer (DPO**) – monitors compliance with EU GDPR. DPO keeps management informed about their responsibility under the law. The DPO is also responsible for monitoring and overseeing notification and communication about personal data breaches with Supervisory Authority.
- **Third Party** – a person or organization who, under the direct authority of the Data Controller or processor, is authorized to access or process personal data
- **Supervisory Authority**, sometimes referred to as the Data Protection Authority (DPA) – responsible for enforcing EU GDPR. They also provide guidance on the interpretation of that law. Each country will have its own supervisory authority.

Under EU GDPR, the protection of personal data is a basic human right. Put another way, under EU GDPR, the Data Subjects own their personal data, and if you are storing their data you are responsible for protecting it.

To understand the various actors and their roles, as well as how they relate to each other, consider a hypothetical company based in France. Customers of this company place orders online through the company's web store.

As part of its multi-national business model, the company stores and processes personal information about individuals (Data Subjects). This EU-based company determines the purposes and the means of processing personal data, and is therefore considered to be a Data Controller. The development, testing, customer care, and billing are outsourced to external subcontractors in Brazil and India – these are Data Processors. The company is responsible to the French Supervisory Authority for ensuring that data is appropriately protected. Fraud analytics is contracted to a company in the United States - this a a Third Party. The company's Data Protection Officer is the liaison between the Supervisory Authority and the company.

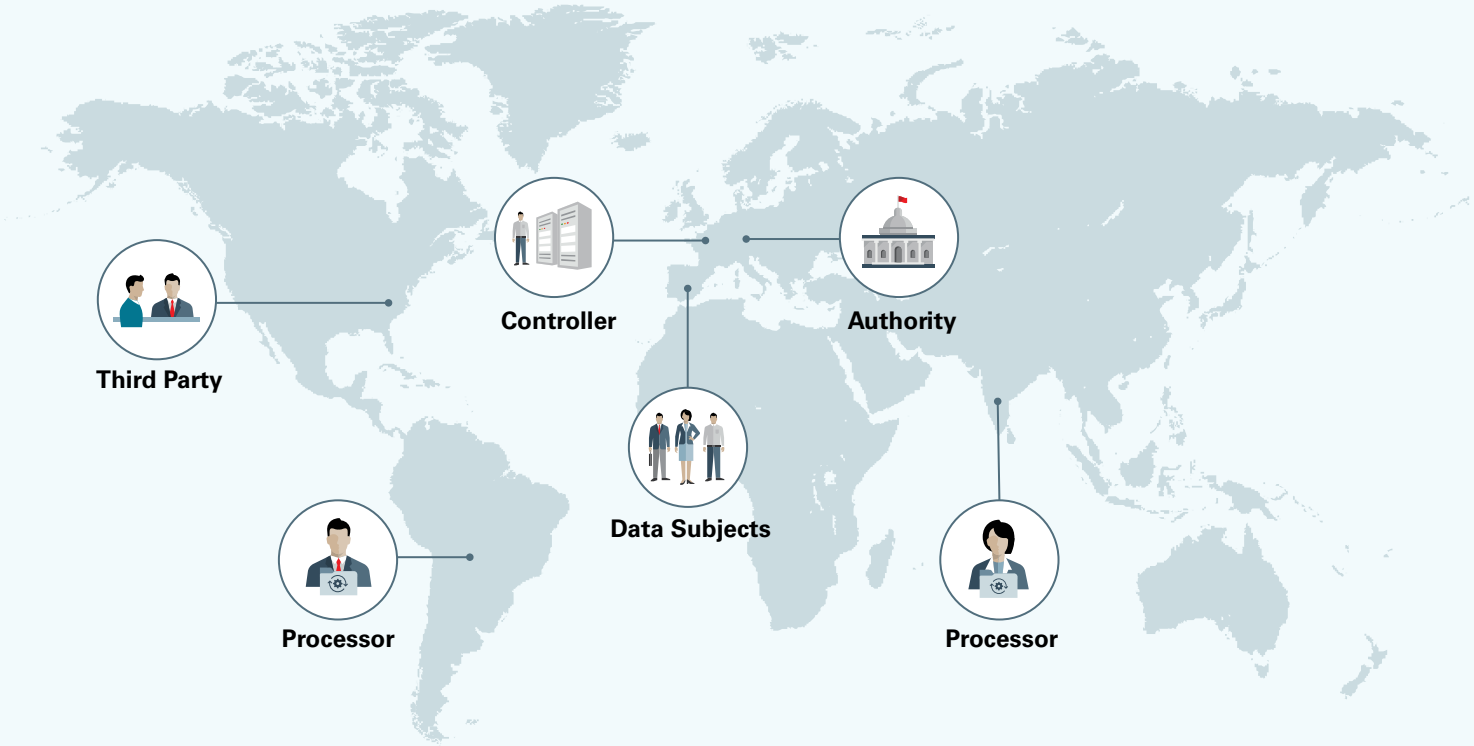


Figure 11.1: EU GDPR Roles

Let's consider another example. In this case the hypothetical company is based in the United States and sells goods and services globally – including to residents within the European Union. As part of its business, the US-based company stores profile information about its customers that includes name, email address, phone number, physical address, and product preference data like preferred clothing sizes. In this example, the US-based company is the Data Controller and, even though it is based outside of the European Union, it is still subject to EU GDPR.

FUNDAMENTAL PRINCIPLES OF EU GDPR

Article 5 of EU GDPR outlines fundamental principles for data protection. These principals require that personal data be processed in a manner that ensures appropriate security of the personal data, including protection against unauthorized or unlawful processing and against accidental loss, destruction, or damage, using appropriate technical or organizational measures is what almost universally concerns us where databases are involved. This principle is reinforced in EU GDPR article 25 – "Data Protection by design and default" — which adds "implement appropriate technical and organizational measures" and to "integrate the necessary safeguards into the processing in order to meet the requirements of this Regulation." As custodians of personal data, organizations have a legal obligation to protect the personal data they store.

EU GDPR doesn't go into specifics on exactly what those technical measures and safeguards are – they are left for each organization to determine how to guarantee the confidentiality, integrity, availability, and resiliency of data and systems. Following a well-known and documented security architecture is an effective way to start when working with any regulation. For applying that protection to an Oracle Database, an organization will usually follow the blueprint already seen in the preceding chapters:

- Assess the risks involved with collecting, processing, and storing personal data including discovering and documenting where sensitive data is stored and how it is processed (Chapters 5 and 10)
- Protect against access to data from outside of the database with encryption (Chapter 4)
- Control access to data to prevent unauthorized use, and inappropriate disclosure including enforcing separation of duties to limit access to data (Chapters 2, 3, and 9)
- Monitor access to data to detect potential data breaches (Chapters 7 and 8)
- Minimize the personal data you must protect, especially in non-production systems (chapter 6).

ASSESS YOUR RISK

EU GDPR article 35 along with recitals 84, and 90 – 95 require the Data Controller under certain circumstances to conduct a Data Protection Impact Assessment (DPIA) to evaluate the origin, nature, particularity, and severity of risk. The assessment is then used to determine appropriate measures to safeguard personal data and to mitigate the risks. The assessment should include:

- Discover sensitive data – the organization needs to understand where sensitive data is located and how it is being processed. Article 30 requires a data inventory as part of an organization's duty to maintain records of processing activities. Both DBSAT Discovery and Enterprise Manager Application Data Modeling can help with this – remember, Sensitive Data Discovery is included with ALL database security products and options. Data Safe cloud service data discovery capabilities can help identify sensitive data hosted in Oracle's databases on Oracle cloud.
- Validate database configuration – systems should be configured in accordance with an organization's own standards, usually those will be based on a widely documented and accepted standard. The Database Security Assessment Tool (DBSAT) can help with this. If your database is running in Oracle's

cloud, you can validate the database configuration with Oracle Data Safe – also included with your cloud subscription. Refer to Chapter 10 for more information on assessing database configuration.

- Document risks and the measures used to mitigate those risks – the remediation recommendations within the DBSAT or Data Safe output can help make a start at this documentation.

PROTECT FROM DATABASE BYPASS

For Oracle Databases, encryption is used to render out-of-band access to data as useless – in other words, data is encrypted so that someone can't go around access control mechanisms to view data directly. Examples of "going around" might include somehow getting access to the tablespace files, and then using a `strings` command to view data in a data file or redo log. It might also include using `tcpdump` to print out network traffic to/from the database. Encryption renders this type of attack useless and forces someone who wishes to view data to use database commands – at which point controls like authentication, authorization, and audit come into play.

While EU GDPR is technology neutral and does not mandate specific security controls, encryption is called out as an example of an appropriate measure to mitigate risk (Recital 83), an appropriate safeguard (Article 6.4(e)), a method of helping to ensure the security of processing (Article 31.1(a)), and a mitigating factor that can relieve the Data Controller of the requirement to notify a Data Subject in the event of a personal data breach (Article 34.3(a)). This is one of the core reasons why Transparent Database Encryption (TDE) is now so prevalent for Oracle databases. TDE ensures that there is no data loss if there is any unauthorized access to the data through the storage layer, including copies (e.g.: backups) of the database. Database encryption is now such a common requirement that all databases in Oracle Cloud are encrypted by default.

Encryption always comes with the additional responsibility to store and manage encryption keys. A common best practice is to separate the keys and the encrypted data, otherwise access to both of them would allow the hackers to decrypt this data. As discussed in Chapter 4, the encryption keys should be protected with Oracle Key Vault.

CONTROL ACCESS TO DATA

The organization must decide how to control access to data and how those controls should be configured to meet Article 25's obligation to protect data by default and by design. A few good rules of thumb for this are:

- Separate database administration from data administration – use Oracle Database Vault realms as discussed in Chapter 3.
- Lock down application accounts so they can only be used by the application – use Oracle Database Vault command rules to prevent misuse of the application credentials.
- Strengthen database authentication to reduce the likelihood of compromised credentials. This might be through the use of Kerberos, PKI, or Centrally Managed Users/Enterprise Users. At an absolute minimum, use strong password policies along with failed login lockouts and unsuccessful login auditing/alerting.
- Periodically review role and privilege grants to identify over-privileged users. As privileges tend to accumulate over time, you should periodically review privilege grants and remove those that are no longer needed. Use Oracle Database's Privilege Analysis to identify unused grants.

The main body of EU GDPR makes several references to pseudonymization. Article 4.5 of EU GDPR defines pseudonymization as "the processing of personal data in such a manner that the personal data can no longer be attributed to a specific Data Subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organizational measures to ensure that the personal data are not attributed to an identified or identifiable natural person." In Articles 6.4(e) and 32.1(a), pseudonymization is identified along with encryption as an appropriate safeguard for personal data. In Article 25.1, pseudonymization is identified as an appropriate technical measure that can help demonstrate data protection by design.

Pseudonymization in production systems is usually a function of access control for the identifying attributes (those attributes that can be uniquely used to identify a Data Subject – examples include taxpayer ID, account number, email address). Data Redaction, Real Application Security, or Virtual Private Database could be used for controlling access to the identifying attributes.

MONITOR ACCESS TO DATA

Article 33 of EU GDPR requires that the Supervisory Authority be notified within 72 hours of a personal data breach. Article 34 requires that Data Subjects be notified about a personal data breach if the breach "is likely to result in a high risk to the rights and freedoms of natural persons."

These notification requirements mean that auditing and monitoring database activity to detect inappropriate access is critical – the organizations should be able to tell who accessed personal data, when the access occurred, and where the access came from. As discussed in Chapters 7 and 8, this will typically require both auditing of high-risk operations and monitoring of database activity to identify anomalies using Oracle Audit Vault and Database Firewall. We will discuss another option for monitoring database activity in Chapter 13, Keeping Data Safe.

MINIMIZE PERSONAL DATA

One of the best ways to reduce the EU GDPR compliance burden is to not collect and store more personal data than is absolutely required. As with encryption and pseudonymization, minimization is identified several times (Article 25.1, Article 47.1(d), and Article 89.1) as a component of data protection by design and default. Data Minimization means first, collect as little data about the Data Subjects as needed to run the business, and second, remove that data once it is no longer needed.

For non-production systems, organizations can consider permanently masking identifying attributes using Oracle Data Masking as discussed in Chapter 6.

Data Subsetting also helps minimize the volume of personal data by creating smaller test environments with a lower volume of personal data. Data Subsetting is also a very useful way to limit how much data the organization keeps on production servers and even in archives.

DATABASE DATA PROCESSING RESTRICTIONS

In certain cases, data subjects may have the right to request an organization to stop processing their

data. In such cases, the organization needs to block individual records from continued processing. Depending on the application and schema design, the use of application customization, or use of Oracle Label Security may be appropriate.

As discussed in Chapter 9, Label Security can be used to attach security related metadata to individual rows of data. That metadata can take the form of levels, security groups, or compartments. Using Label Security access control mechanism, one can allow or deny access to a record based upon the row and user labels. Data rows that are available for processing might be assigned a low security level. When processing of certain rows needs to be restricted, assigning them a high level means that a process can't interact with them unless that process is operating under an even higher permission level.

CONSENT MANAGEMENT

For Data Controllers that collect personal data with consent, they also need to manage, collect, store, and enforce such consent. Oracle Label Security may be able to help here also by recording membership in security groups. Sales and Marketing might be one group, while Product Support might be a different group. When a Data Subject consents to different processes, it could be recorded as different groups in the Data Subject's row label. A process would not be able to interact with the Data Subject's row if the process were not a member of one or more of the groups attached to the row.

SUMMARY

Vipin Samar, Senior Vice President of Database Security for Oracle, says "Data is today's capital, but in the wrong hands, data becomes the new liability." We must minimize the risk of retaining that data in order to maximize opportunity provided by the data. Throughout this chapter we have discussed how the comprehensive set of tools and technologies from Oracle can significantly enhance your readiness to meet compliance requirements for EU GDPR and other regulations.

Remember, databases are core assets that may store significant amounts of personal data. Protecting this data is central to all privacy legislation, and using a comprehensive set of security controls is one of your best aids in meeting regulatory requirements.

.

12

Securing Databases in the Cloud

ORACLE

From a security threat perspective, many risks are the same for both on-premises and cloud deployments, but some risks are quite different in the cloud. Securing a database in the cloud requires a different mindset, and recognition that while in many ways the controls used to secure the environment are the same as on-premises, the focus and control points are necessarily different.

On-premises systems benefit from known, trusted administrators and an inherently restricted threat environment. They are inside the organization's firewall, and benefit from protected and limited access.

Cloud services benefit from dedicated security administrators, binding service level agreements, and an increased focus on process. However, resources in the cloud may be exposed to different threats vectors. Attackers can reach the database through an internal network connected to a cloud infrastructure over VPN, through direct attack on the cloud provider's network, and even through the networks of other customers of the cloud provider who may be connected to the provider's network.

While there are advantages and disadvantages to both models, the broad industry consensus is that the cloud's advantages outweigh the disadvantages – a properly configured database service in the cloud is usually MORE secure than an on-premises database.

SOME THREATS IN THE CLOUD REMAIN THE SAME

Some threats faced in the cloud mirror the threats faced on-premises and can be protected using similar controls. The importance or value (in terms of risk mitigation) of a particular security control may vary, but the application of the control is essentially the same.

- Encrypt data at rest and in motion to force attackers to come at the system using database APIs.
- Control what people who connect to the database can do with privileges, roles, Oracle Database Vault, Data Redaction, Label Security, and Real Application Security.
- Monitor database activity to detect anomalies and block malicious activity.
- Minimize the amount of sensitive data needing protection with Oracle Data Masking and Subsetting or Oracle Data Safe.

UNIQUE THREATS IN THE CLOUD

Some threats faced in the cloud are different – and that difference can be a difference of scope, of emphasis, or even a completely new threat that is different in kind. Let's take a look at a few of these differences.

Service Provider Access. One thing that makes the cloud threat environment different is the differing trust level for the operating team. In on-premises environments, the organization knows their employees and contractors. There is some ability to check on-premises employee backgrounds, even it it's just a credit check. There are also on-premises physical controls to supplement IT access

controls, and an external attacker must first penetrate the internal network before beginning to attack the database.

There is no knowledge of who the cloud service providers' employees are, and no ability to check their backgrounds. However, cloud being the business of cloud vendors, all reputable cloud vendors ensure the strongest levels of background checks. Service providers invest in controls to detect and prevent unwanted intrusions – and the major providers all have fully-staffed security operations center that monitor operations around the clock. Cloud vendors block inappropriate access to service consumer data with strong operational controls designed to minimize administrator access to data.

The risk of data access from outside of the service consumer's organization is mitigated through access controls – and in the cloud, access controls, particularly separation of duties, are even more important than on-premises.

Shared Infrastructure. Another difference in the cloud is the use of shared resources. On-premises, all servers, networks, compute, and storage support the company's goals. This dedicated infrastructure limits perceived risk because all the applications belong to the organization and operate within the boundaries and policies of the organization's network – a trusted zone. In a worst-case scenario, a server can be physically disconnected from the network or powered down while the situation is evaluated, and ways are devised to stop data from being extracted and exfiltrated.

In the cloud, you can remove logical access to a service, you may be able to deprovision the service – but you have no physical access to power down a server or unplug the network cable. Examine the cloud provider's architecture to identify how systems are isolated from one another, and what the process is to isolate a system during an emergency.

In the cloud, resources are provisioned and de-provisioned as needed. A server and storage area that was used by one company yesterday may be used by another company today. There are justifiable concerns about remnants of data from yesterday's company being visible to today's consumer of the same storage, and still another consumer tomorrow. Depending on the level of access they have, the new user of hardware once dedicated to your organization may be able to forensically recover data fragments that contain sensitive data. This is one reason why encryption is even MORE important in the cloud than on-premises. At Oracle, we believe that encryption is so important that we force encryption for databases in our cloud – we don't want to assume the risk of storing our customer's data in unencrypted form even if the customers themselves might be willing to assume that risk.

And just as with on-premises, encryption in the cloud is only as good as the security of the keys used to encrypt / decrypt the data. The Cloud Security Alliance (CSA) recommends that encryption keys for cloud-based systems be retained outside of the cloud service provider's environment, either in a different provider's system or on-premises. Oracle Key Vault satisfies this need – in case of a loss of confidence in the security of a database in the cloud, simply suspend access to that database's master encryption key stored in Oracle Key Vault. Once concerns about the cloud service are resolved, restore access to the key and the database is now ready for use again. The combination of on-premises control (Key Vault) with cloud-based service (database) is known as a hybrid cloud security model.

Note that no one is saying there are no on-premises risks – malware, botnets, and compromised servers can exist within your firewall – but a growing body of experience shows that a reputable cloud service provider's internal network and infrastructure is likely to be MORE secure than your network.

Anomaly Detection and Activity Monitoring. In the cloud, auditing and monitoring is even MORE important than on-premises. With unknown administrators managing your infrastructure, it is only common sense that you will monitor their activity. Here is another case where a hybrid cloud control could use an on-premises monitoring system (for example, Oracle Audit Vault and Database Firewall)

to monitor databases in the cloud. Much like when Oracle Key Vault is used in a hybrid cloud scenario, Oracle Audit Vault and Database Firewall in this mode provides local control (with attendant separation from the cloud provider) of cloud resources. This hybrid model for monitoring cloud database activity is decreasing in popularity as the availability of mature, cloud-based activity monitoring services grow.

Configuration Management. Most organizations have configuration standards for hardware, operating systems, and databases. When a new server is installed, database software is installed, or a new database is created for on-premises, those standards are followed, and the system can move to production knowing that it is secured according to established risk acceptance. Organizations have their own rules for patching and upgrading systems, and while they may not always apply patches immediately, at least they know the state of their systems and what risk they are choosing to accept.

With cloud, the service provider's configuration standards are followed, and they may or may not match your on-premises standard. Further, because there is seldom visibility of the complete application stack – the cloud provider is trusted to configure systems appropriately, and the service consumer usually does not have visibility into their standards or the ability to change them.

Although organizations almost always have standards for configuration, patching, and operations, the actual situation is usually different than those standards dictate. Most security assessments of on-premises databases find default passwords in use, systems that have not been patched in years, and configuration settings exposing the system to very high levels of risk. The reality is that most IT organizations simply don't have the time to do everything well, and the limited DBA staff is heavily focused on availability and performance, with very little time left over for security. The average security staff has no background in database administration, and just hopes that the DBAs are doing everything right.

Ensure you understand how your database is configured in the cloud. Consider using the Database Security Assessment Tool (DBSAT) to scan the database configuration – it's possible to run DBSAT remotely, so even if you don't have access to the operating system you can still review the configuration and scan for sensitive data.

SECURITY IN THE CLOUD IS USUALLY BETTER

In the cloud, consumers gain from economies of scale when it comes to security. The major cloud providers have dedicated security staff that DOES have database expertise. They have SLAs around configuration, patching, and management. Most cloud providers operate on a shared responsibility model, where the cloud provider's duties are clearly outlined.

Knowing where the cloud provider's responsibility stops and the consumer's responsibility begins, is one of the first things to establish before creating the security action plan.

There are several things that cloud providers can do to make the job easier – for example, Oracle encrypts database in its cloud by default, and includes security options like Oracle Database Vault, Label Security, and Data Masking and Subsetting for the Oracle Database Cloud Service High and Extreme Performance editions. This means consumers of those services get to take advantage of the capabilities discussed in earlier chapters.

SHARED RESPONSIBILITY

Security in the cloud imposes a shared responsivity model, with some tasks being the responsibility of the service provider, and others the responsibility of the service consumer. Most of the time, the goal is to shift as much responsibility as possible onto the service provider. How much can actually be shifted to the provider depends upon the type of service being provided and the level of access granted to the service consumer. The more access the service consumer has, the more security responsibility they need to assume. For on-premises services, all the responsibility falls upon the service consumer (who is usually part of the same organization as the service provider).

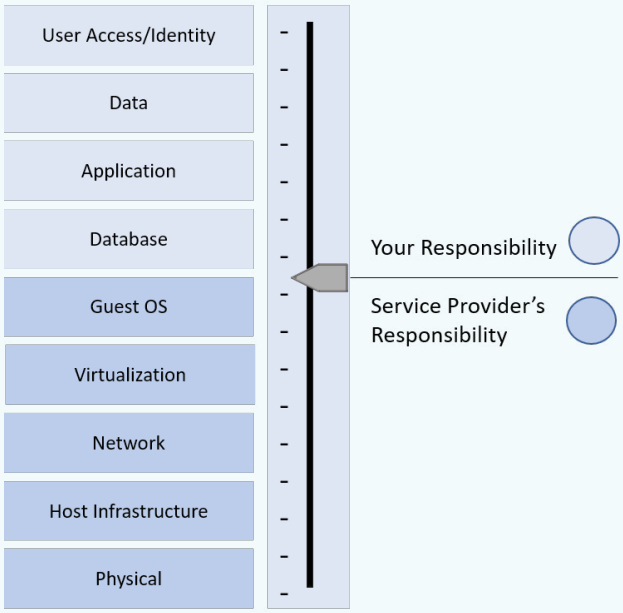


Figure 12.1: Cloud Shared Responsibility Model for IaaS

In the cloud, Infrastructure as a Service (IaaS) offers consumers access to the system at the operating system level, meaning the consumer assumes responsibility for everything at or above the OS. Conversely, Software as a Service (SaaS) usually only allows the consumer to access the application's user interface – everything below that interface is the responsibility of the service provider and the service consumer is only responsible for the data they place into the service and the users they allow access to that data. Platform as a Service (PaaS) falls in-between IaaS and SaaS, with the service provider having responsibility further up the stack than with IaaS, but not as far as with SaaS.

Database as a Service is an example of Platform as a Service – PaaS. With DBaaS, the consumer has access at the database level, so assumes responsibility for the database and higher in the stack. However, even within Database as a Service offerings, the responsibility dividing line may vary. For example, Oracle's Autonomous Database shifts more of the security responsibility onto the cloud service provider, bringing the Autonomous Database close to the SaaS model as opposed to the standard PaaS model followed by other database services. Responsibility for the database is split, with the majority of the responsibility (including application of security patches) lying with the provider, and a small portion of the responsibility belonging to the consumer.

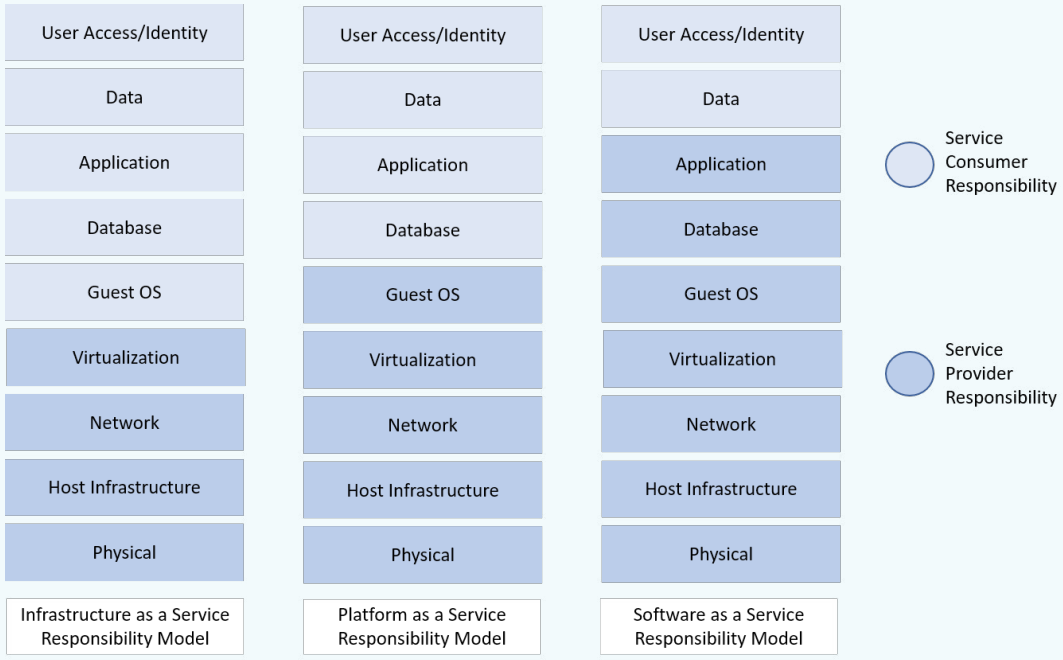


Figure 12.2: Cloud Service Responsibility Model

With Autonomous Database, the service consumer is responsible for a limited amount of database configuration, but fully responsible for what data is added to the system and the applications or user(s) who access that data.

The common DBaaS shared responsibility model is also modified if the service platform moves into the consumer's physical data center. For example, Oracle offers Exadata Cloud-at-Customer, where physical possession of the cloud infrastructure is retained by the service consumer. In this case the consumer assumes responsibility for physical security of the system, most of the network and a small part of the host infrastructure responsibility (e.g.: power, HVAC) while the service provider assumes responsibility for the rest of the host infrastructure, a small portion of the network (internal to the cloud environment), and all of the virtualization and guest OS layers.

As of the time of this writing there is no Autonomous Cloud at Customer available, but one can probably see how that would further blend the responsibility model.

Why is all this talk of shared responsibility important? Because before adding security controls into your cloud environment, you should first understand what you are responsible for, and ensure your cloud provider clearly understands what you expect them to be doing.

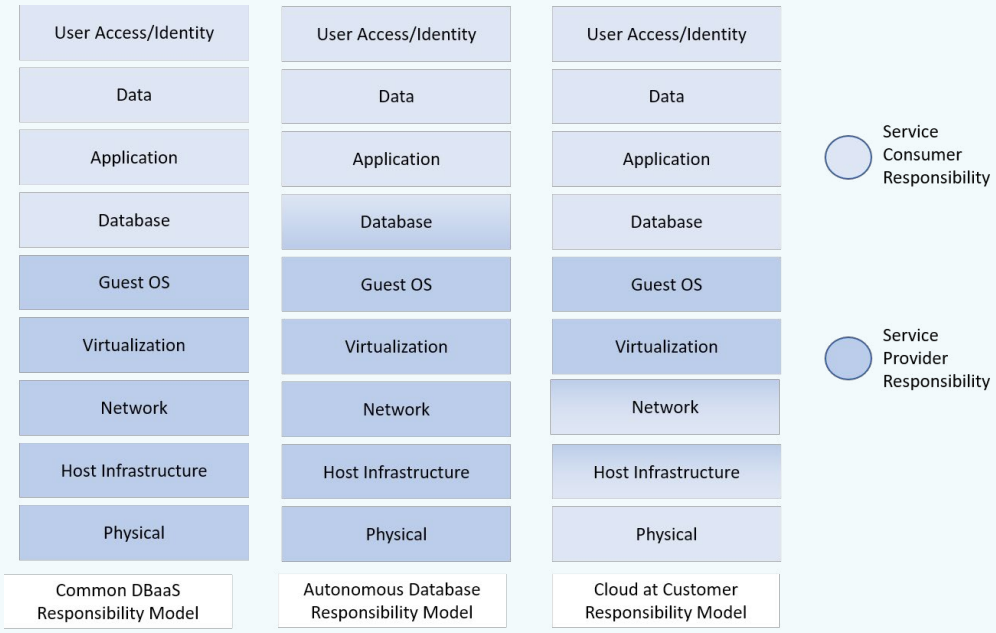


Figure 12.3: DBaaS Shared Responsibility Model

SO, WHAT SHOULD WE DO?

Which cloud services are used determines the configuration work and periodic security assessment that should be performed as part of the organization's security responsibilities. Once you understand which areas you are responsible, you should assess the system's current state. That should include some validation that the cloud service provider is also doing their part. Assessments may be as procedural as requesting third-party validation reports (e.g.: SOC2, FedRAMP, EU GDPR, or other security reviews) or as tactical as periodically reviewing exposed ports, patch levels, etc.

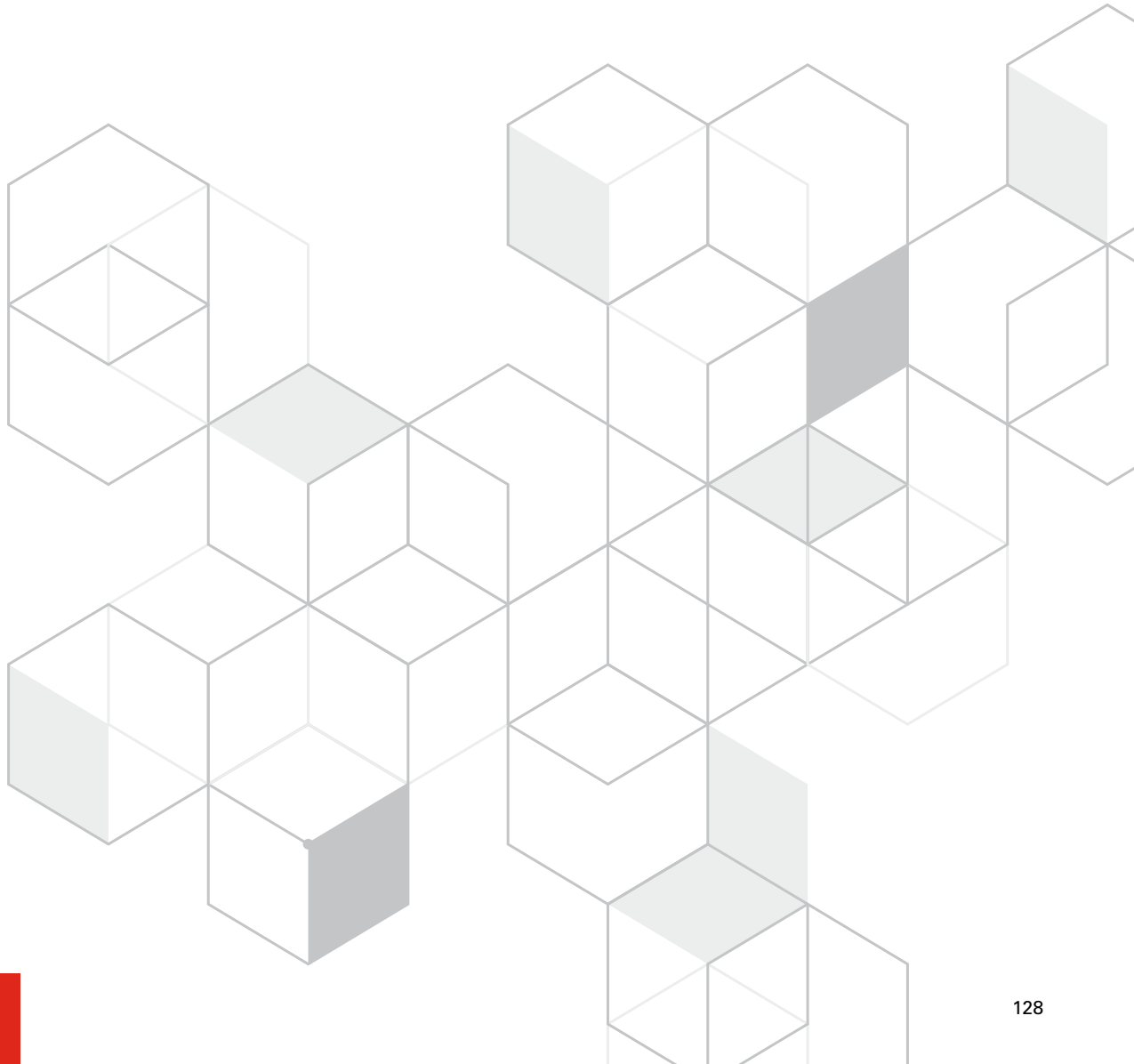
Now look at the remaining tasks – the ones not owned by the cloud provider. The capabilities and methodologies outlined earlier in this book continue to protect the database in the cloud. Here are a few general-purpose guidelines to start with:

- **Encryption** – DO NOT place unencrypted data in the cloud. This is an area of risk an organization should be unwilling to accept.
- **Data Minimization** – DO NOT leave sensitive data in test and development systems. DO reduce the size of test systems using subsetting where possible. Where possible, anonymize that data using masking and scrambling techniques that remove sensitive data, or use completely artificial data sets that present no security risk.
- **Access Control** – DO implement separation of duties – plan as if the cloud vendor will be breached at some time, and the attacker will gain access to the environment hosting your data. If there are no security realms to protect sensitive data, there is nothing to stop the attacker from stealing it.

- **Activity Monitoring** – DO monitor the audit trail. Ensure the organization or cloud provider's audit facility forwards alerts to the organization security monitoring service – either on-premises or in the cloud.

SUMMARY

Cloud database deployments can reduce costs, free up staff for more important work, and support a more agile and responsive IT organization. But those benefits come with additional risk, including an extended network perimeter, expanded threat surface with an unknowable administrative group, and shared infrastructure. Despite that, with the right procedures in place, and with just a little attention to the things outlined in this short chapter, the cloud can provide better security than most organizations have on-premises and do it at a fraction of the cost in time and manpower.



13

Keeping Data Safe

The previous chapter, "Securing Databases in the Cloud," introduced the shared responsibility model, which describes how the job of protecting data and applications must be shared between the cloud service provider and the cloud service consumer. We learned how Oracle Autonomous Cloud provides considerable security as an integral part of the service. These capabilities include network security and monitoring, OS, VM and container security and patches, database security patches and upgrades, web application firewalls, as well as regulatory compliance processes. We also learned how various layers of technology including data encryption, activity monitoring, and administrative separation of duties are used to protect customer data from accidental or deliberate exposure to cloud operators.

Regardless of how secure the platform is out-of-the-box, however, there are certain security activities that can only be performed by the cloud customer. For example, customers are responsible for adding their database users, managing those users' privileges, and invoking appropriate security controls to protect their data based on how mission-critical and/or sensitive that data is. To accomplish this task, cloud database administrators need to be able to answer some fundamental questions about their users and data. These questions include:

- Is my database properly configured?
- Who are my risky users?
- What are my users doing?
- What sensitive data do I have in my system?
- How do I protect my sensitive data from exposure when used outside of the production environment?

We've seen in previous chapters how Oracle offers a variety of solutions for answering these important questions. However, wouldn't it be nice if these capabilities came pre-integrated in the cloud with an easy-to-use-interface?

ENTER ORACLE DATA SAFE

Data Safe cloud service provides an integrated set of security features for databases enabling users to understand the sensitivity of their data, evaluate risks to data, mask sensitive data, implement and monitor security controls, assess user security, monitor user activity, and address data security compliance requirements. Data Safe is a unified control center for managing database security in the Oracle Cloud.

Data Safe employs a simple "click-and-secure" model and is designed to be accessible to cloud users with no special security expertise required. 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 visible. Implemented as a cloud service, Data Safe helps Oracle Cloud customers manage their portion of the shared responsibility model.

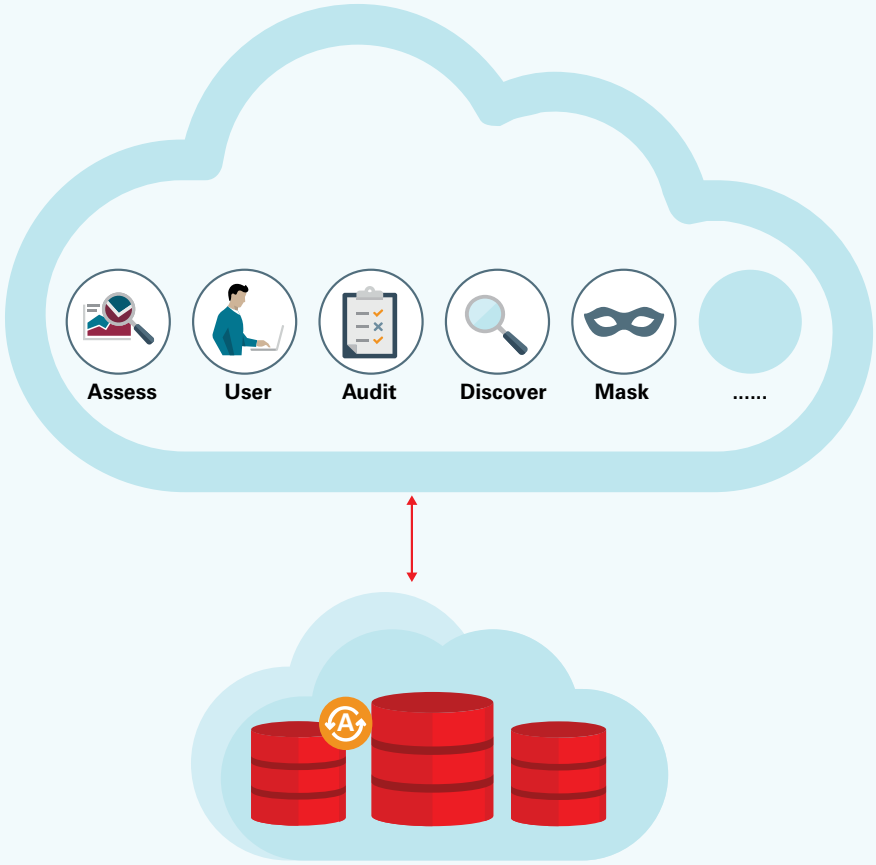
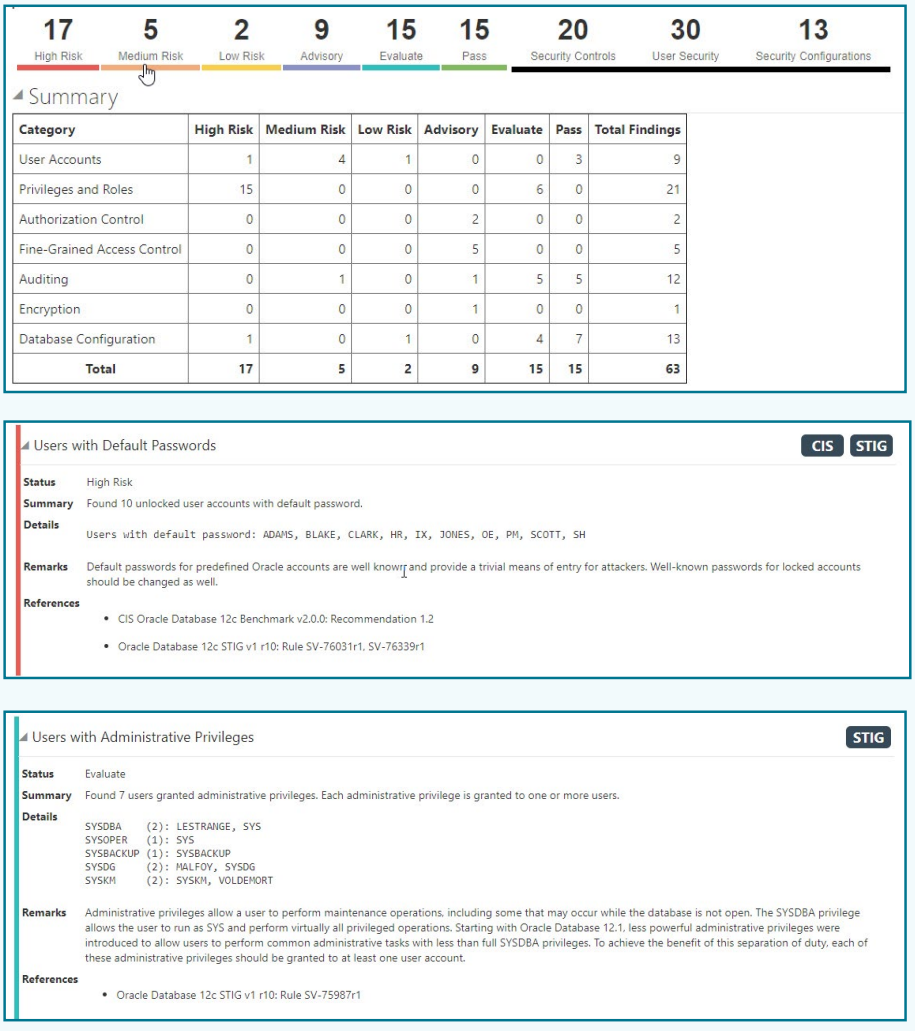


Figure 13.1: Oracle Data Safe services for protecting databases in the cloud.

ASSESSING SECURITY

Even with a managed database service like Oracle Autonomous Database, users have considerable latitude in how they configure their databases. Data Safe's database security assessment helps to identify configuration gaps that could represent a vulnerability. It performs a comprehensive check of database configuration. It examines areas like user accounts, privilege and role grants, authorization controls, fine-grained controls, auditing, encryption, and configuration parameters. It identifies gaps versus organizational best practices and delivers actionable reports, with prioritized recommendations as well as mappings, to common compliance mandates like EU GDPR, DISA STIGs, and CIS benchmarks.



Category	High Risk	Medium Risk	Low Risk	Advisory	Evaluate	Pass	Total Findings
User Accounts	1	4	1	0	0	3	9
Privileges and Roles	15	0	0	0	6	0	21
Authorization Control	0	0	0	2	0	0	2
Fine-Grained Access Control	0	0	0	5	0	0	5
Auditing	0	1	0	1	5	5	12
Encryption	0	0	0	1	0	0	1
Database Configuration	1	0	1	0	4	7	13
Total	**17**	**5**	**2**	**9**	**15**	**15**	**63**

Figure 13.2: Oracle Data Safe – Security Assessment

USER RISK

Data Safe includes a unique capability that allows security administrators to evaluate the risk represented by various database users. The user risk assessment feature evaluates database users, looking at both static and dynamic characteristics of the user's profile, in order to identify the highest risk users.

User risk is presented graphically, allowing administrators to very quickly determine which users may be over privileged or require compensating control such as auditing. It helps you understand how many users have not logged in last 3 months, or have not changed their passwords. If there is some suspicion about any user, it can help you understand all details about the user including when they were created, to their roles and privileges and related audit records.

We've already seen how appropriating the credentials of a privileged user is the most common method used by hackers to access sensitive data. By providing the ability to assess and visualize user risk, database administrators can take the necessary steps to make their applications more secure.

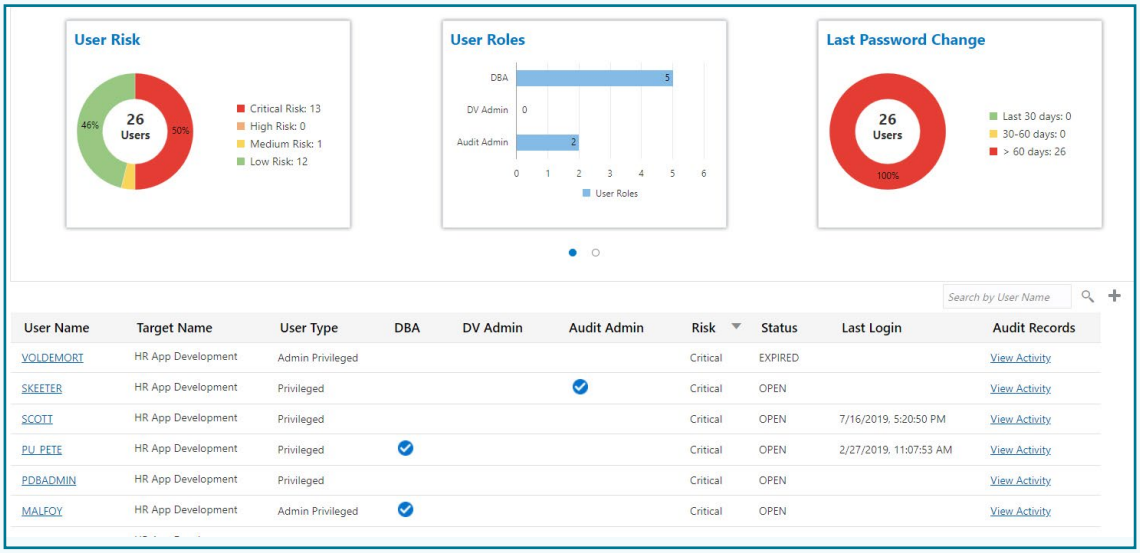


Figure 13.3: Oracle Data Safe – User Risk Assessment

AUDITING USER ACTIVITIES

We've seen in previous chapters how 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. Data Safe's user activity auditing feature allows administrators to select from a variety of predefined audit policies and provision them to the database. Data Safe can then start collecting and storing audit records obtained from the cloud databases.

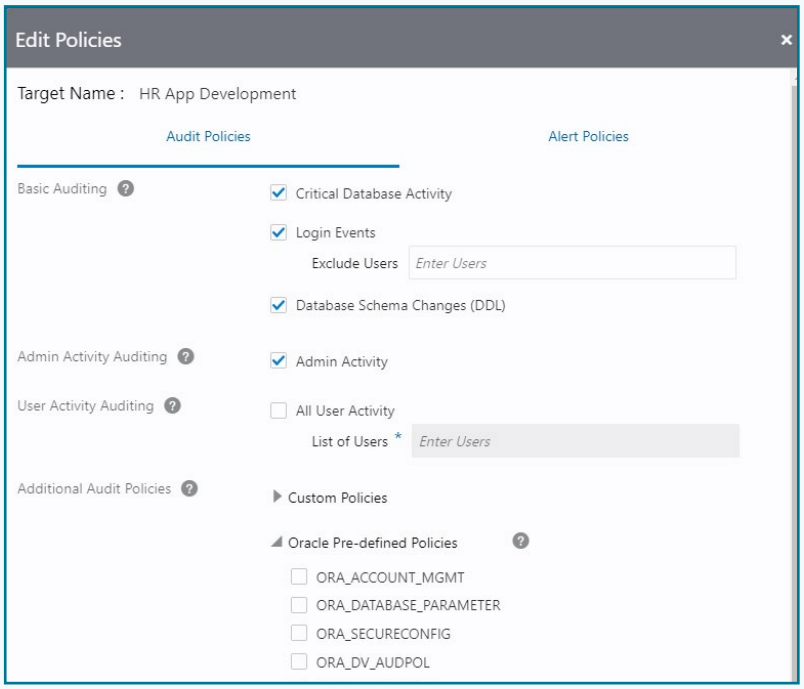


Figure 13.4: Oracle Data Safe – Audit Policy Provisioning

Data Safe users can access interactive reports for user activity tracking or forensics, as well as summary reports for routine collection and reporting. Finally, these reports can be downloaded as PDFs to help with organizations' compliance programs. Administrators can also select from a number of predefined alert policies so they are immediately notified of unusual activities that may indicate compromise.

DISCOVERING SENSITIVE DATA

The types of data contained within the database, and their sensitivity, helps determine what controls should be used to protect that data. Data Safe includes a sensitive data discovery feature that allows security administrators to quickly answer the critical questions of "What types of sensitive data do I have?" and "How much of it do I have?"

Data Safe's sensitive data discovery feature provides automated discovery of over 125 sensitive data types across categories including personally identifiable information, financial information, health information, job-related information, and education information. Data Safe examines column names, comments, and data values to identify columns containing sensitive data. Data Safe users can also extend these sensitive data types to include custom data types. Sensitive data discovery helps users to understand the value of the data and enables them to prioritize their defenses.

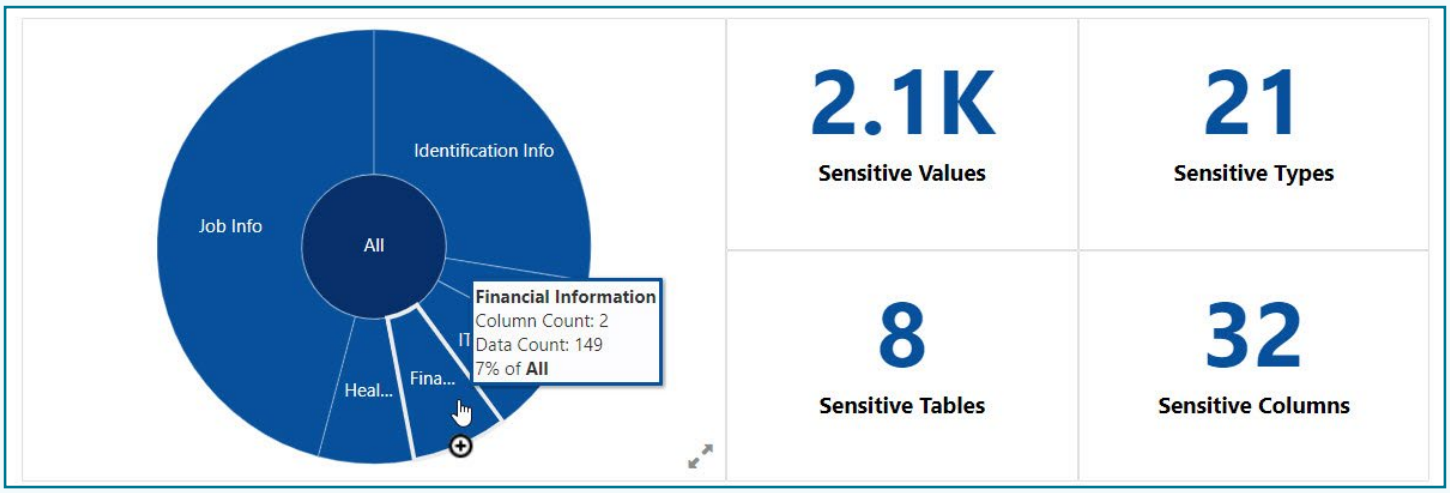


Figure 13.5: Oracle Data Safe – Sensitive Data Discovery

MASKING SENSITIVE DATA

In the earlier chapter on data masking, we learned how masking removes security risk from test and development systems, and help minimize the amount of sensitive data stored. Data Safe's data masking feature provides the ability to quickly mask sensitive application data with a library of over 50 predefined masking formats. Default masking formats are automatically suggested based on the type of sensitive data discovered using the sensitive data discovery feature. Data masking can be used to transform columns of sensitive information such as birth dates and credit card numbers, and can also support more complex data masking use cases such as conditional and compound masking. Each masking run is summarized with an interactive masking report.

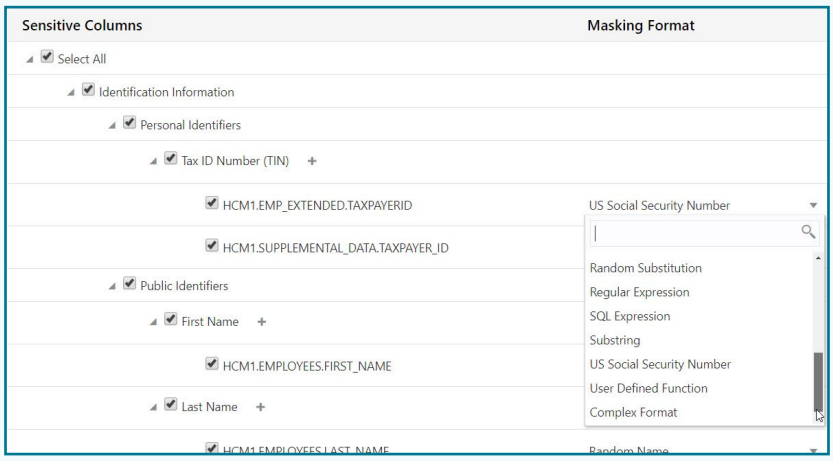


Figure 13.6: Oracle Data Safe – Sensitive Data Discovery

VISUALIZING RISK WITH THE DASHBOARD

Data Safe includes a built-in dashboard that summarizes risk elements derived from user and security assessments, sensitive data discovery, audit trails, and alerts. This dashboard provides a 'single pane of glass' snapshot that allows Data Safe users to visualize their configuration, user, and data risks, identify open issues requiring immediate attention, and spot opportunities to better secure their application data.

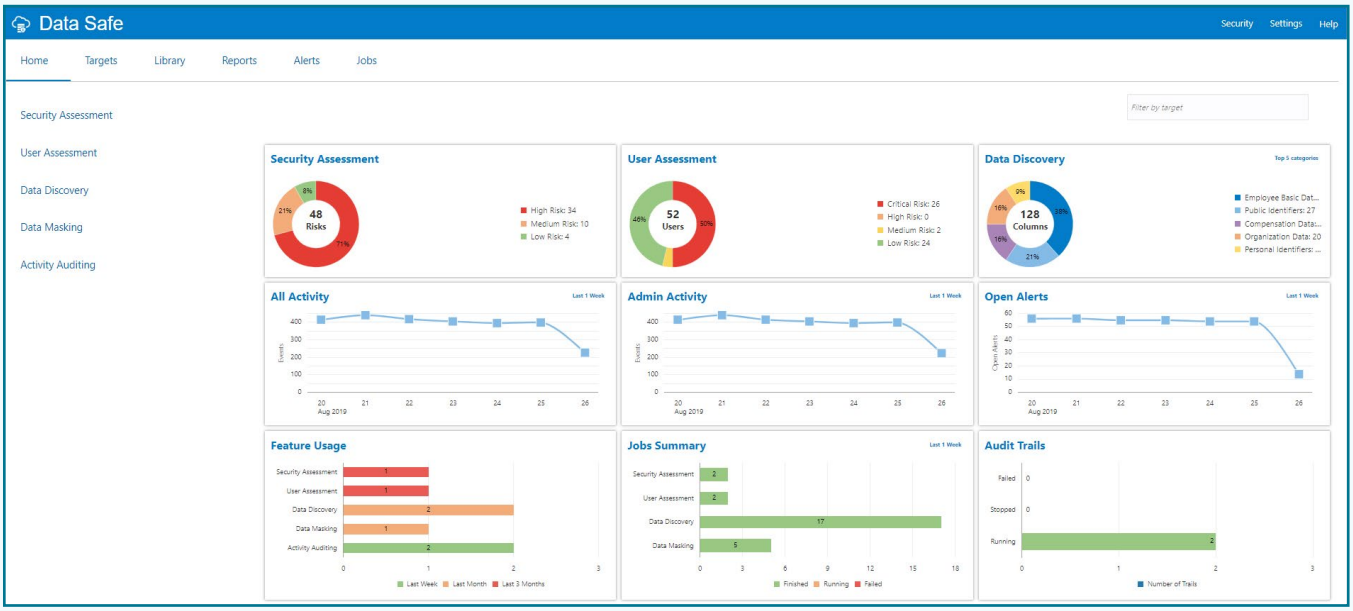
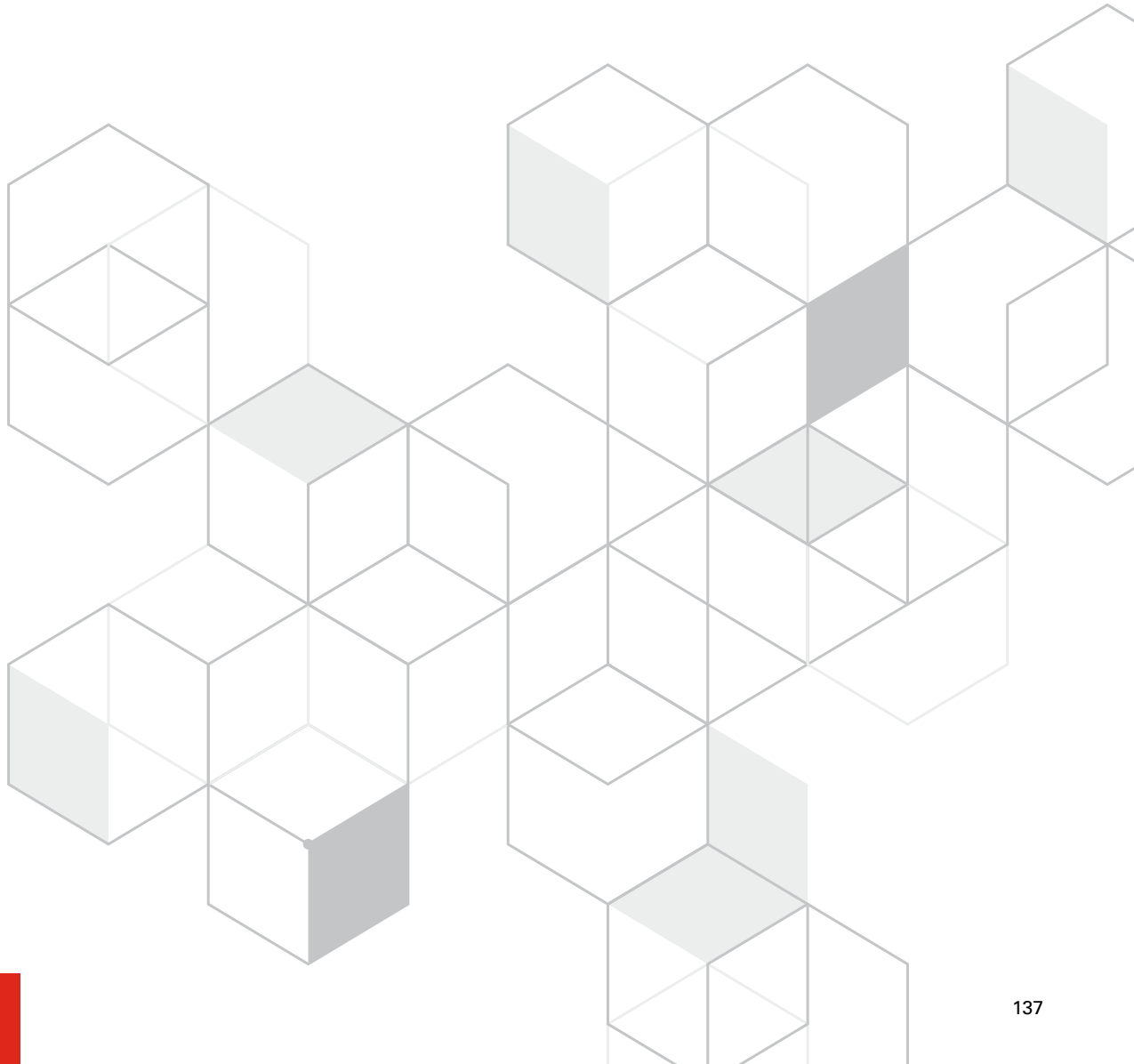


Figure 13.7: Oracle Data Safe – Security Console

SUMMARY

In the previous chapters, we've learned how the security technologies and capabilities built into the Oracle Database enable Oracle Database customers to deploy and maintain a highly secure database environment. With Oracle Data Safe, critical functionalities for securing databases in the cloud are now under a simple click-and-secure interface. The most common security tasks can be completed without requiring any deep security expertise. Data Safe helps all customers, big or small, keep their data safe. With Data Safe, security is now truly the reason to move to the cloud.



Putting it All Together

Putting it All Together

> 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**

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 controls available for keeping your data secure. Indeed, with so many technologies available for securing data with evaluative, preventive, detective, and data- driven security rings of controls, you may be wondering how you would actually go about implementing these controls. What should be the order? What should I do first? How much time will it take? How much risk am I reducing?

The best advice we can give is that you should 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. Let's say that you are able to assign different sensitivity levels to your data and systems ranging from bronze to silver, gold, and platinum. Bronze systems might include internal portals, employee directories, and wikis. Silver systems could include business transaction systems, supplier information, and parts catalogs. Gold systems might include data subject to regulatory compliance, whether it be 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.

One possible approach to protecting these systems is illustrated in the figure below. Bronze and above databases should at minimum be securely configured and current with security patches. It is very easy for hackers to break into an unpatched system and exploit it as a command and control base for further attacks or a staging area for all the data they discovered. In addition, we certainly want to monitor and audit all the activities done by the privileged users on this machine so that if any significant changes are being made, they can be tracked.

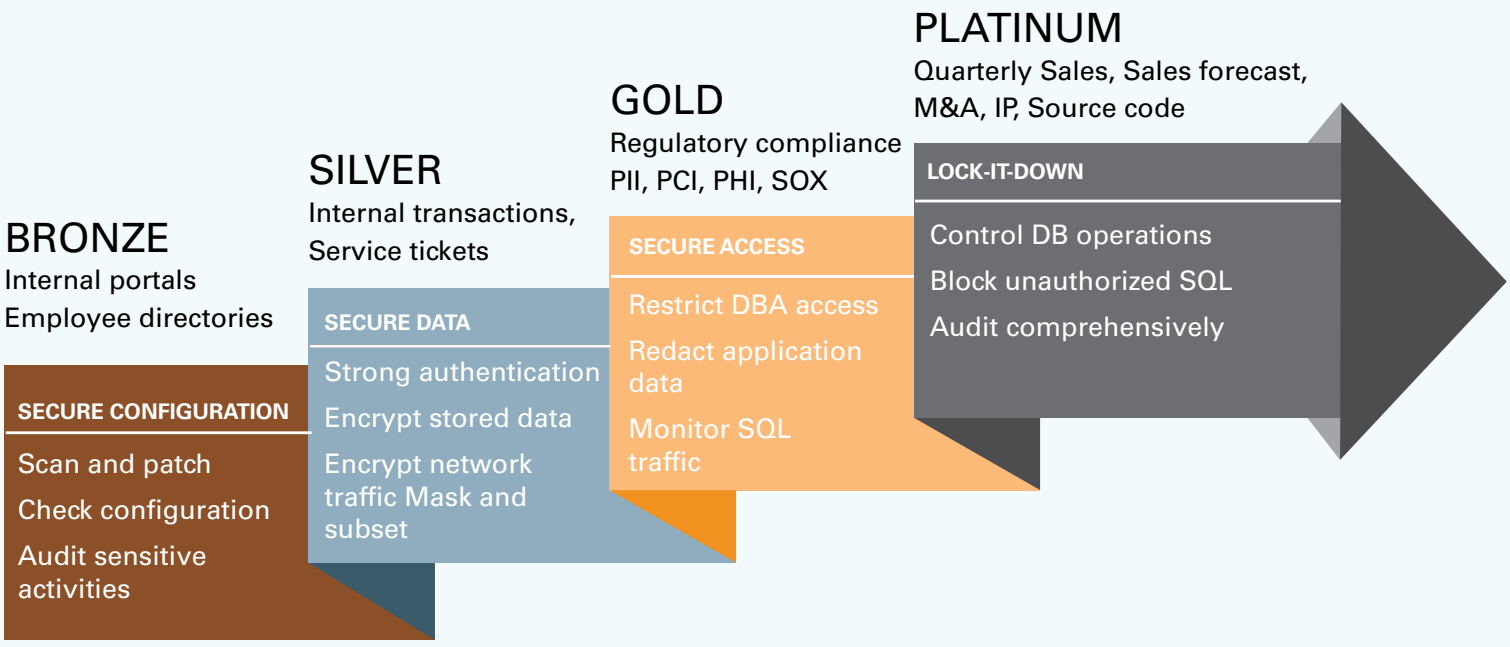


Figure 14.1: Approach for Mapping Security Controls

For silver and above databases, data at minimum should be protected from unauthorized users. This means 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. Make sure that privileged users are authenticated with strong passwords or with PKI or Kerberos based authentication. The basic security controls used for silver are encryption, masking, and strong authentication.

For gold and above databases, data should be protected from privileged users and from those users who do not have a business need to access the data. We need to restrict privileged users while still enabling them to perform their duties and monitor SQL activities over the network to quickly identify malicious attempts to exploit application vulnerabilities. To address basic compliance requirements, we need to protect all PII, PCI, PHI data.

Platinum databases should have all the security controls used for bronze, silver, and gold. In addition, platinum databases should be locked down as they contain the largest number of sensitive critical assets. One needs to control who can log on to the database machine, monitor every single operation in real time, ensure that SQL injection attacks cannot succeed, and audit everything that happens on this machine so that in case of a breach, we would be able to figure out what actually was lost, and how.

Depending upon the priorities and the security strategy of the company, deployment of these controls could start from either edge of the spectrum. You might take a controls-based approach and start securing the configuration of all your databases and then move on to the next function and encrypt all your databases, and so on – implementing one control at a time across your environment. Or, you might take a systems based approach and do a complete lock-down of individual systems one at a time, according to their level of risk. Both approaches are valid, and in real life we usually see a combination of the two approaches.

Either way, you need to have a proper strategy in place which takes into consideration the overall business objectives along with the 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 one's security effort.

ADDITIONAL READING

For Further Reading

The topic of Database Security is broad, and there is a good deal of material available if you'd like to learn more. Here are links to a few resources you may want to review:

Oracle Database Security Homepage – this is where we post updated product data sheets, white papers, demo viewlets, and more.

https://www.oracle.com/database/technologies/security.html

Database Documentation – this link takes you to the latest database product manuals. Just select your database version from the drop-down list and click Security to see the security related documentation.

https://docs.oracle.com/en/database/oracle/oracle-database/index.html

Oracle Critical Patch Updates, Security Alerts, and Bulletins – this page lists announcements of security fixes made during critical patch updates. You'll also find security alerts and bulletins.

https://www.oracle.com/technetwork/topics/security/alerts-086861.html

Oracle Database Security Blog – articles on topics of interest to the database security development team and field implementation resources.

https://blogs.oracle.com/cloudsecurity/db-sec

Oracle Software Assurance Blog – information on new critical patches and alerts from our software assurance team

https://blogs.oracle.com/security/

AskTom monthly community calls – we hold a community call the second Thursday of most months, where we'll update you on the latest product announcements and cover some technical topic. You can register to be notified of new sessions and the topics. We record these sessions, and you'll find the recordings listed on this site:

https://bit.ly/asktomdbsec

Oracle Corporation

WORLDWIDE HEADQUARTERS

500 Oracle Parkway
Redwood Shores
CA 94065
USA

WORLDWIDE INQUIRIES

Phone: +1.650.506.7000
+1.800.ORACLE1

Fax: +1.650.506.7200

oracle.com

Integrated Cloud Applications & Platform Services

Oracle is committed to developing practices and products that help protect the environment

CONNECT WITH US

ORACLE