ORACLE

Fourth edition

Securing the Oracle Database

A technical primer

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 while everybody else trusted the administrators, users, and computing environment to keep their data secure. It was only when browsers opened up new vistas for commerce over the net in the 90s, companies began to understand the strong need for security. This new perspective led to SSL, network firewalls, and strong cryptography.

Fast forward to the present and, just like before, we find ourselves living in a dramatically different world where every piece of data is online and available 24/7. To address this new reality, we see many different security technologies protecting various layers of the IT stack all the way from the applications down to the chipsets. While the global security spend is expected to exceed $130 billion in 2020, hacks are becoming bigger and bolder, and impacting everything from customer and citizen databases to vaccine data and Wi-Fi routers.

Hackers have built sophisticated tools along with a thriving underground market to go after everything we have, whether on mobile devices, laptops, file servers, or databases. For most hackers, the target of choice is not a laptop or a spreadsheet–the target is most often a database with its hundreds of millions of records. The hackers may try to break-in through attacks on the network, operating system, database, and applications. 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 they are vulnerable, and what the fixes might be. They might also fear that the fixes may break their applications, or that the insiders may exploit the trust placed in them. Too many stop at securing the perimeter, not recognizing how easily hackers can bypass the network perimeter, get to the databases, and quietly walk away with their data. It is not surprising that on average, it takes the victims six months to even know that they have been breached, and it also isn't surprising that they typically learn about the breach from customers or law enforcement.

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

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

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

Vipin Samar

Senior Vice President, Oracle Database Security Development

January 2021

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 Polytechnic Institute.

Angeline Janet Dhanarani is a Product Manager for Oracle Database Security, focusing on auditing and activity monitoring. With close to 16 years of experience in Oracle spanning multiple products, she now helps Oracle customers adopt comprehensive database security strategies and closely works with the engineering team to define the product roadmap for auditing and activity monitoring.

Ashok Swaminathan is a Senior Director of Product Management with overall responsibilities for the auditing and activity monitoring 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 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.

Bettina Schäumer is a Senior Principal Product Manager for Oracle Database Security, responsible for Oracle Data Safe. She has over 20 years of experience in product and solution management, go-to-market strategies, sales operations, sales enablement, program management and consulting for major software companies. Prior to joining Oracle, Bettina worked at SAP with global responsibilities for end-to-end scenarios and key capabilities within the SAP HANA platform. Throughout her career, she covered a variety of solutions in enterprise software, business networks, business analytics, internet of things, technology and database systems. Bettina has a German degree in Computer Science.

Manish Choudhary is a Product Manager for Oracle Database Security focused on sensitive data discovery and masking technologies. Prior to joining the Oracle Database Security team, he worked with the Oracle Solaris group as a product manager and a security engineer. While with McAfee Labs, he worked on Host Intrusion Prevention System and Database Activity Monitoring. His interest areas are data security and privacy, cryptography, cloud security, and machine learning in security. Manish has an MS in Security from Georgia Institute of Technology.

Michael Mesaros is Director of Product Management for Oracle Database Security 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.

Pedro Lopes is a Product Manager in the Oracle Database Security group. He covers Europe, Middle East, and Africa (EMEA), and Latin America regions for all Database Security features and products and manages the Security Assessment technologies (DBSAT, Data Safe). He played numerous roles from Consulting to Presales during the last 20 years at Oracle. Pedro is helping customers to adopt Oracle Data Safe and to understand how Oracle Database Security solutions may help address EU GDPR and other regulatory requirements. Pedro's certifications include CISSP, ITIL v3 Foundation, and International Project Management Association Level D (IPMA).

Peter Wahl is the Product Manager for Oracle Transparent Data Encryption and Oracle Key Vault. Between 2004 and 2015 he helped introduce Transparent Data Encryption and wrote security articles/tools to migrate 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 and holds a Master's Degree in Electrical Engineering from the University of Applied Sciences in Ravensburg.

Russ Lowenthal is a Senior Director of Outbound Product Management for Database Security helping Global 1000 customers understand how to mitigate the risks with their databases using Maximum Security Architecture. Leveraging more than thirty years of experience in IT including database, UNIX systems, and network administration, he advises Oracle's customers on database security strategy and implementations. Russ' certifications include CISSP, Certified Information Systems Auditor (CISA), Certified Information Systems Manager (CISM), Oracle Certified Master (OCM), Microsoft Certified Systems Engineer (MCSE) and Certified Technical Trainer (CTT).

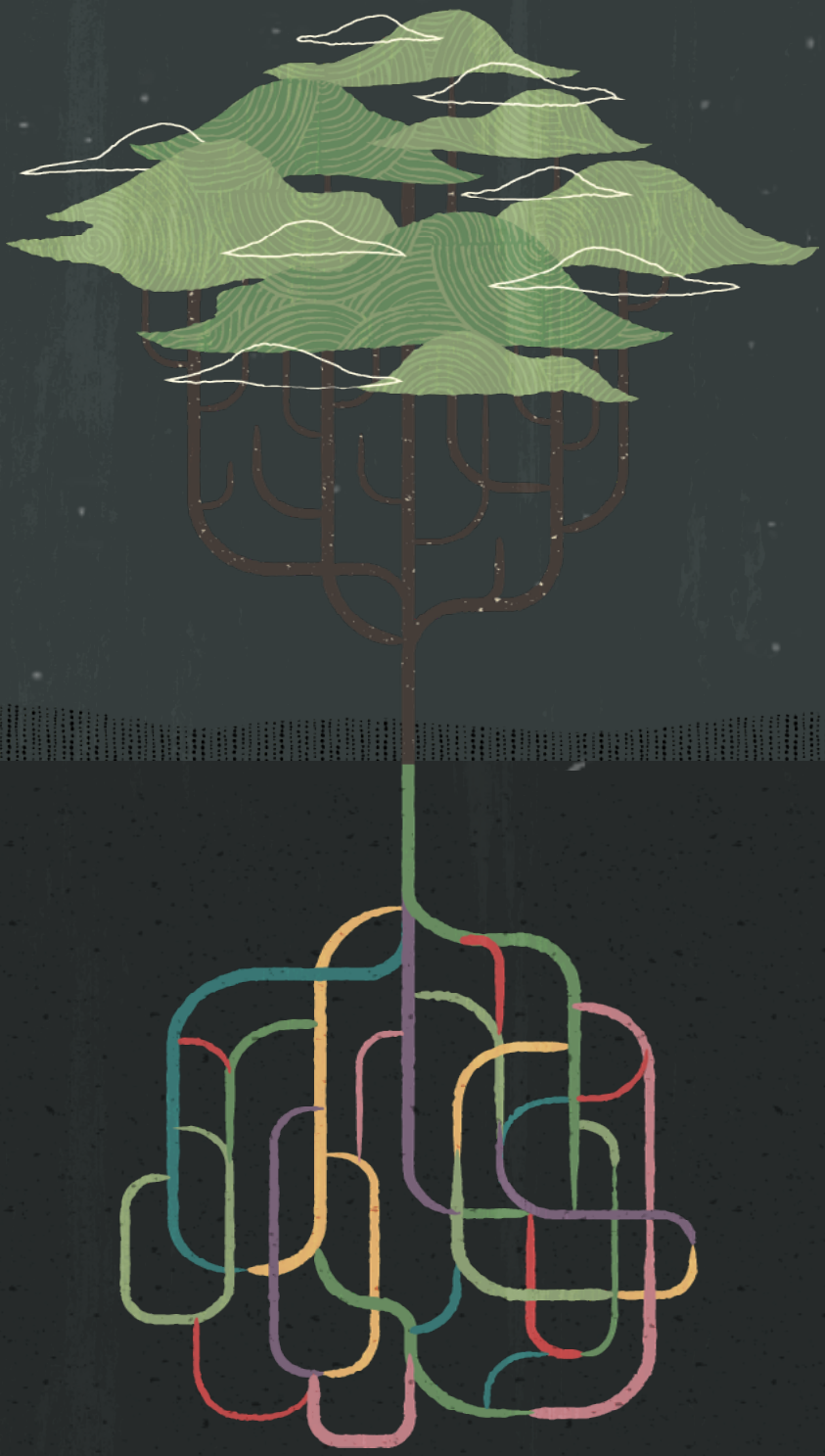
Acknowledgements

The authors wish to gratefully acknowledge Michelle Malcher, Paul Needham, Scott Rotondo, George Csaba, and Saikat Saha who contributed towards the first three editions of this Oracle Database Security primer.

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

01 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 such as databases from data theft.

However, the threat environment for organizations has changed considerably in recent years. Tools vary widely depending upon the attackers, from exploiting unpatched systems to very advanced methods where hackers penetrate a network, search for vulnerabilities, and then covertly exfiltrate data from servers. These attacks can go undetected for weeks, months, or even years.

The need to protect data has never been greater. 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 more than 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 including comprehensive data privacy laws in Japan, Australia, New Zealand, India, South Korea, Chile, and 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: “outsiders” and “insiders.” Outsiders vary widely in their level of skill and resources. They include everyone from lone “hacktivists” and cyber criminals seeking business disruption or financial gain, to criminal groups and nation state-sponsored organizations seeking to perpetrate fraud and create disruption at a national scale. Insiders include current or former employees, curiosity seekers, and customers or partners who take advantage of their position of trust to steal data. The target for both of these groups includes personal data, financial data, trade secrets and regulated data.

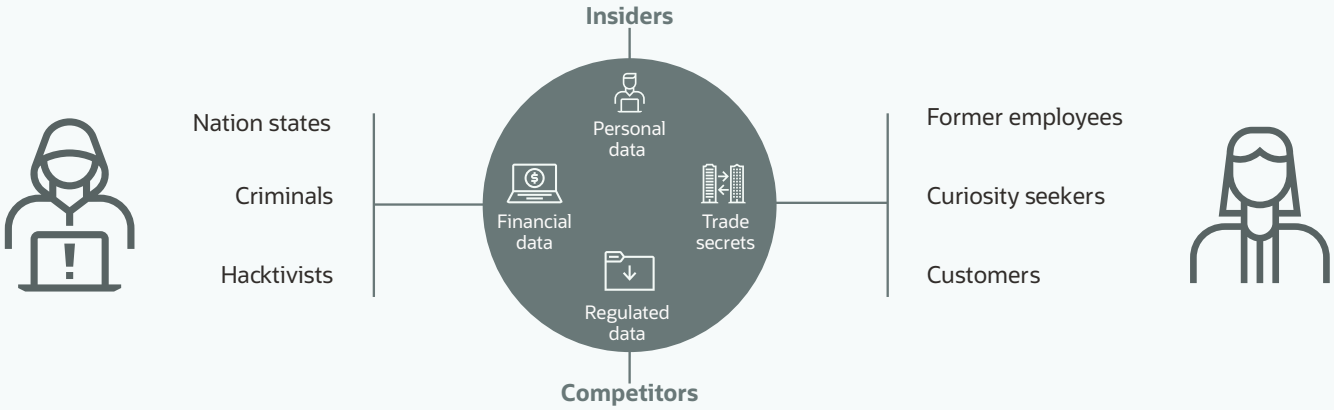


Figure 1.2: Threat actors

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, 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 system 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.
- 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.
- Stealing the keys and demanding ransom.

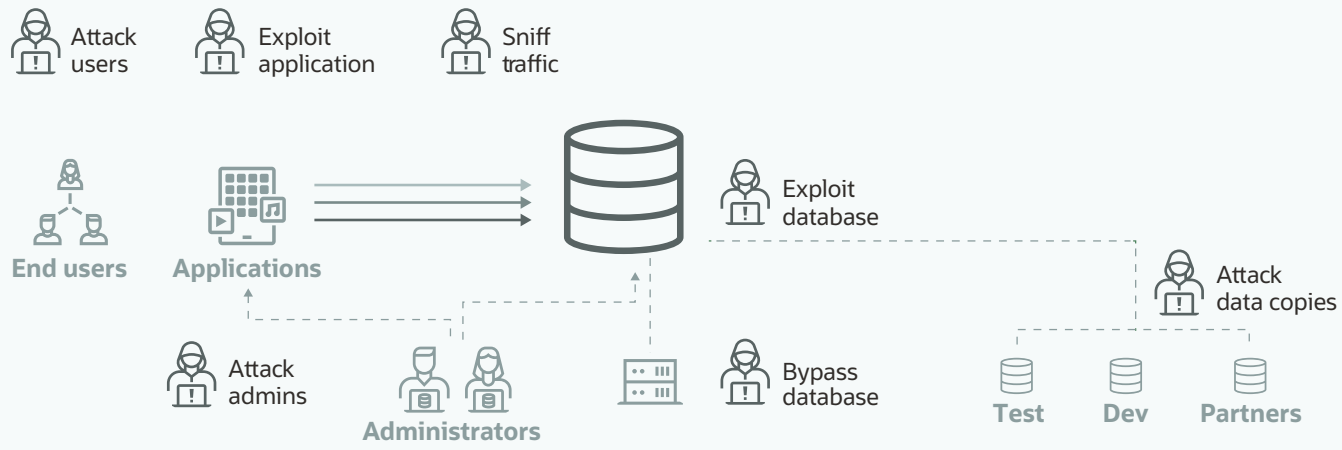


Figure 1.3: How threat actors exploit databases

Rings of security control

A well-structured data security solution must provide multiple 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 it resides.
- Preventive controls block access to data by unauthorized users with technologies such as encryption, redaction, masking, or database system-level controls.
- 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.

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.

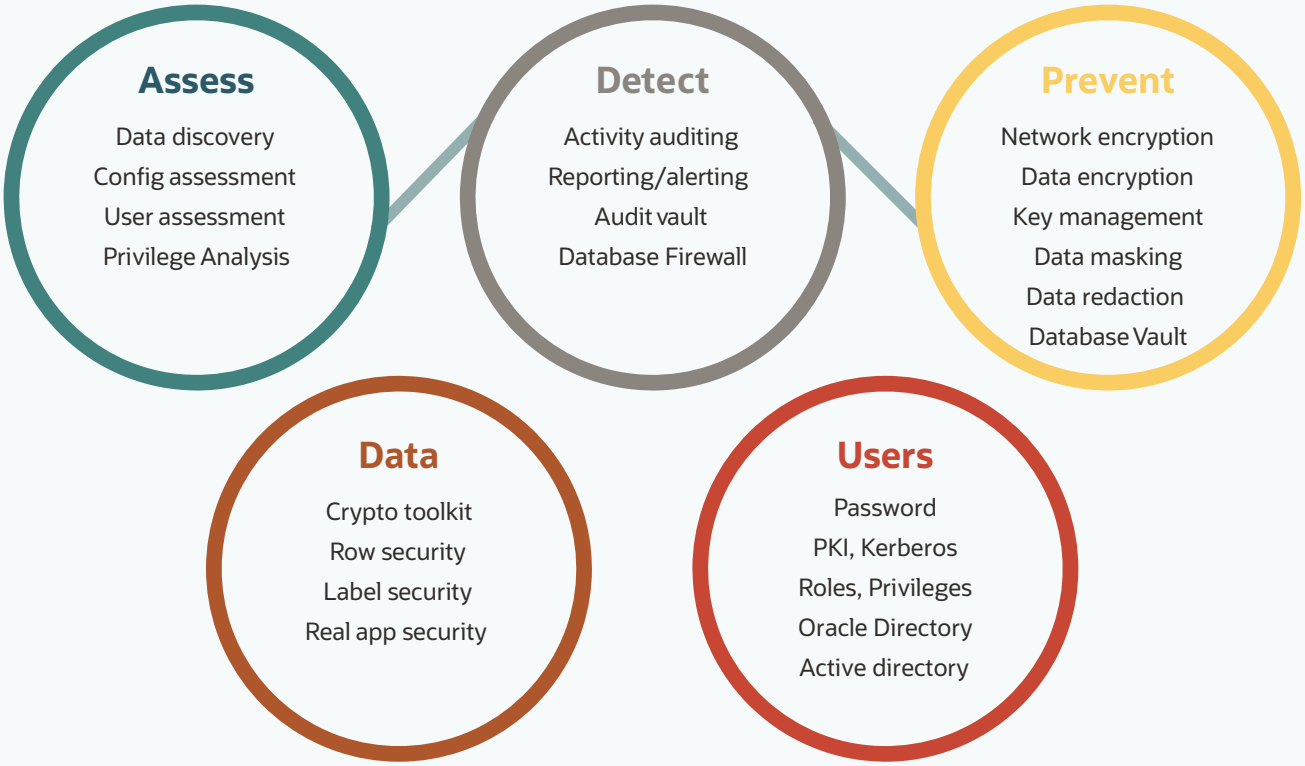


Figure 1.4: Rings of security controls for Oracle Database

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 sensitive data, and its location, are critical to protecting your data. It briefly covers various Oracle technologies that enable you to automatically discover, classify and analyze sensitive data.

Chapter 6: Masking sensitive data discusses how to limit exposure to sensitive data in production systems as well as non-production environments such as development and analytics. It explains how data masking and subsetting technologies can help you minimize risk and maximize value of your data.

Chapter 7: Database auditing and activity monitoring 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: Network-based SQL monitoring 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: Need for security assessment 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 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.

02 Database authentication and authorization

A fundamental step in securing a database system is validating the identity of the users 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 centralized 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 typically 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 24/7, 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.
- **Data 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.

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

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 can 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, 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 and privileges. 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 provide mechanisms (described below) that puts constraints on 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 but insecure 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 24/7 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 primary 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, Label Security, Redaction and Real Application Security *(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 can be used to create a common policy of password and resource authorization parameters for user accounts. Each user account can be associated with a selected user profile to simplify 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; -- STIG password complexity rules
```

The ora12c_stig_verify_function imposes several password requirements that correspond with the United States Department of Defense Security Technical Implementation Guide (STIG) including a minimum length, minimum number of alphanumeric characters and 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: Downtime when changing application account passwords

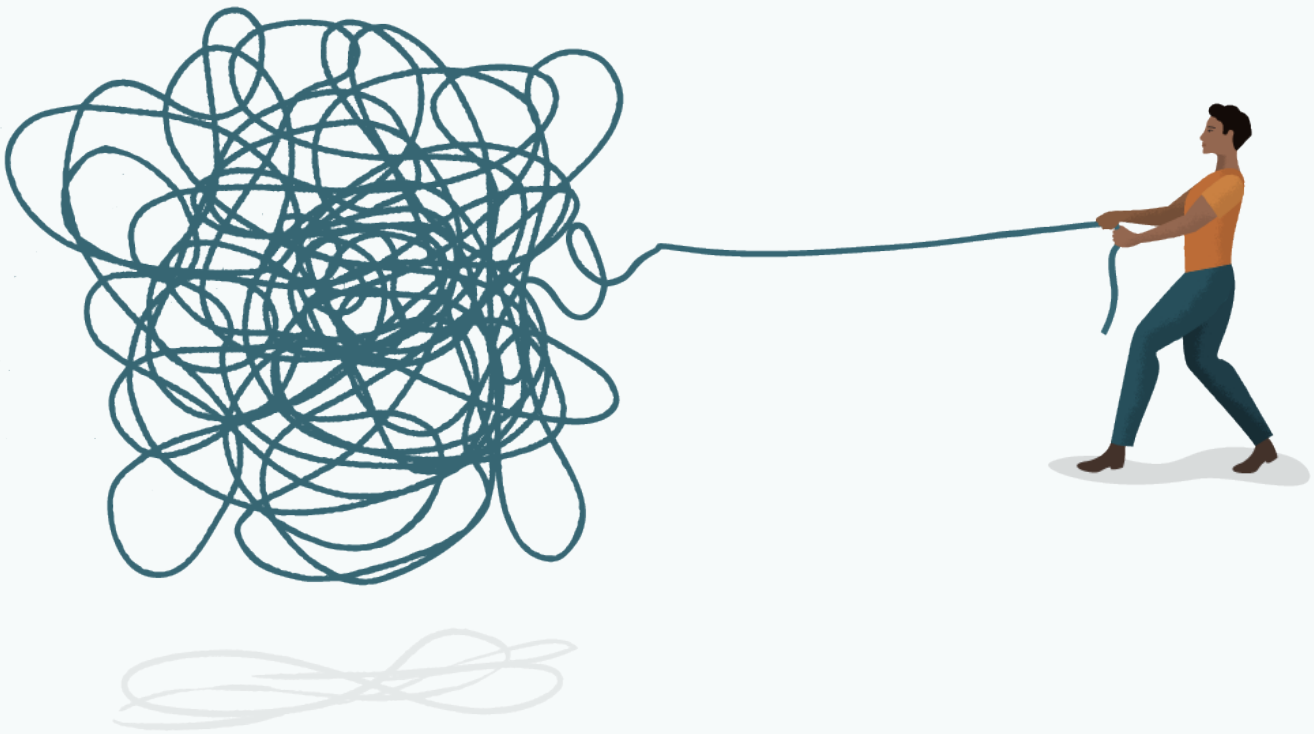
The database accounts for applications need to update their passwords periodically to comply with corporate policies or follow security best practices guidelines. Changing an application account password requires the password to be changed in the database and also with the application (hopefully stored in a client wallet). To eliminate the chance that the application would fail when it tries to login during a password change, most organizations have to schedule downtime for their applications. Some organizations even forego password updates to minimize application downtime.

Starting with Oracle Database 21c, users can create a new password for an application while still allowing other application clients to use the old password for a limited period. With gradual password rollover, the new password can be updated with all application clients or mid-tiers without having to schedule downtime or risk an adverse application event.

Problem: Managing multiple accounts and passwords

If users have accounts on multiple databases, they tend to keep the same password for convenience. However, they then have to update the password on every database when any single database requires a password update. Further, when an 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 them a very attractive target for hackers.

Using 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 needs to decide what this user can do. In general, the user is permitted to connect to their own schema and access, modify, or delete all the objects in their own schema. Through privileges and roles, user can get permission to perform a specific operation, such as updating an 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 database 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` allow the user to read data from almost any table in the database including sensitive data stored in application schemas. `CREATE USER` is another example of a system privilege allowing new users to be created in the database.
- 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 be done by the SYS account (Database owner). The administrative privilege SYSDBA allows a user to become SYS for these tasks. The SYS account and the related SYSDBA administrative privilege should only be used when absolutely necessary and be safeguarded. Any such use should also be fully audited.

Roles

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

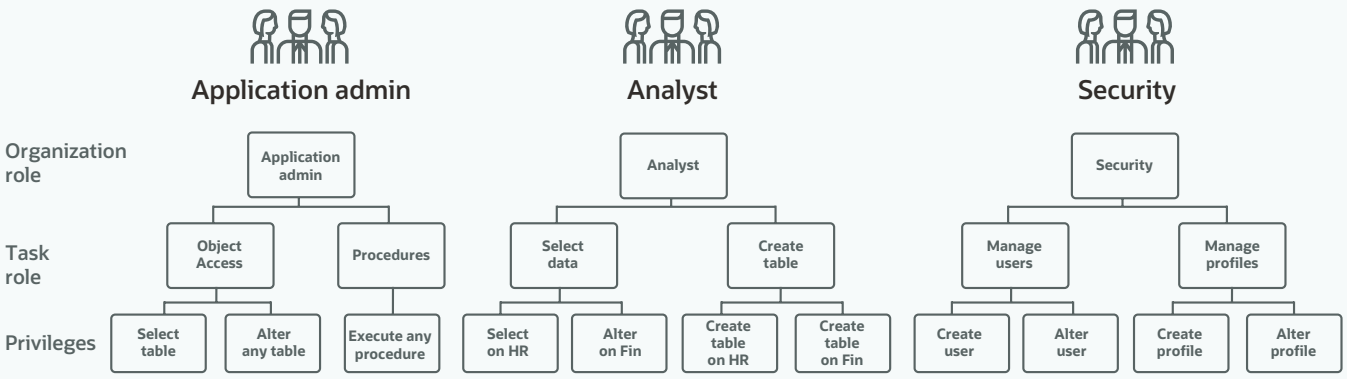


Figure 2.1: Sample role and privilege hierarchy

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

Table 2.2: Data dictionary views for roles and privileges

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 one of the Oracle directory services, which optionally can integrate with other corporate directories. After successfully authenticating the user, the database refers to the directory for authorization (roles) information. Database users managed through the directory service are called enterprise users as they span multiple databases across the enterprise. Such enterprise users can be assigned enterprise roles or groups that determine access privileges across multiple databases. An enterprise role maps to one or more roles on individual databases.

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.

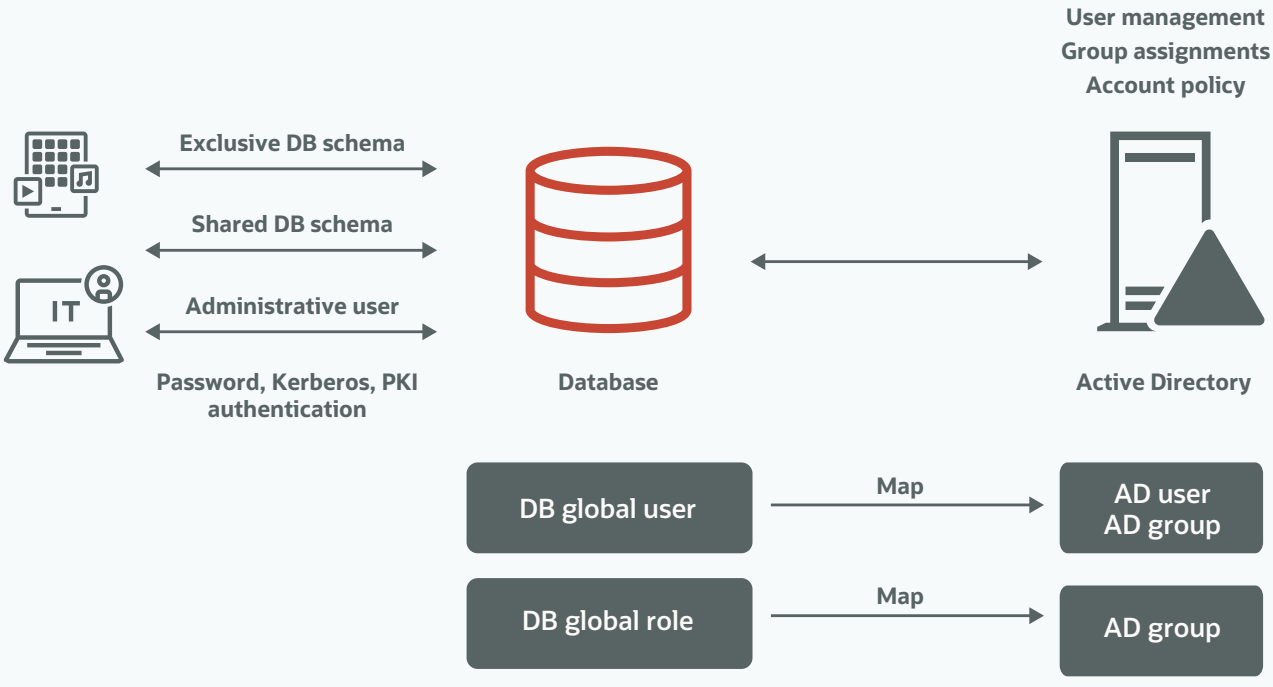


Figure 2.2: Centrally Managed Users (CMU)

Both EUS and CMU allow users and administrators to be authenticated using passwords, Kerberos and certificates. Kerberos is becoming more commonly used when using centralized directory services for human users while passwords and certificates are used for application accounts.

Protecting your users from getting hacked

Protecting regular users

One cannot expect regular database users to remember and voluntarily follow all the security guidelines regarding their password strength, password management hygiene, and the roles/privileges granted to them. These security guidelines need to be enforced 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, and the number of passwords retries.
- 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.

Protecting DBA accounts

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 or SQL commands using Database Vault. More on this in later chapters.
- 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.

Protecting application accounts

Application DBA and application service accounts in the database not only have all the roles and privileges required to run the application on behalf of every application user, but for the sake of convenience, they may also have roles and privileges to perform application installation, upgrades, patching, and other maintenance activities. Such accounts need to be 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 gradual password rollover to periodically rotate the application password in the database.
- 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 installed with the database are schema-only accounts to prevent login to these privileged accounts. In the past, these accounts would have passwords that would need to be periodically rotated. Other users can still be granted read/write access to these schemas.
- 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.
- Fully audit all of their activities for accountability and tracking.

Protecting security administrator accounts

Security administrators manage security controls, encryption keys, database users and audit records. They ensure the 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.

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

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 set of privileges and roles can be done easily using Privilege Analysis.

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

03 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 24/7 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 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.

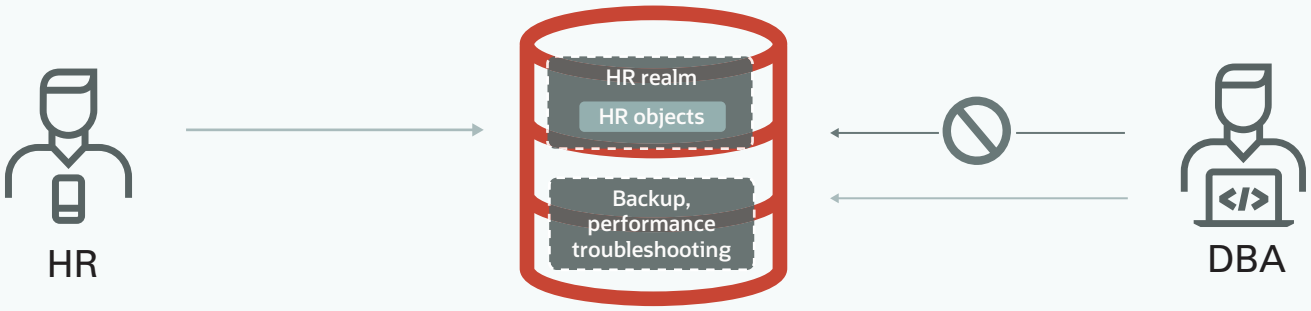


Figure 3.1: Oracle Database Vault uses realms to protect the sensitive data

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.

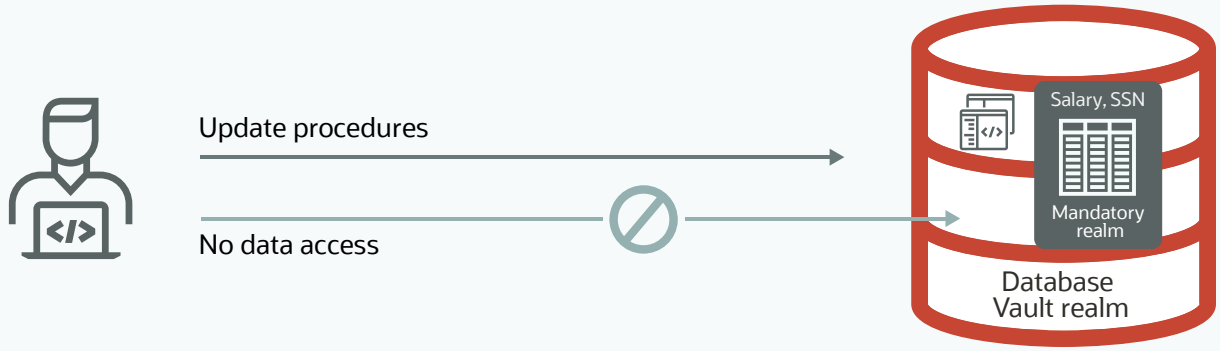


Figure 3.2: Mandatory realms in Database Vault

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.

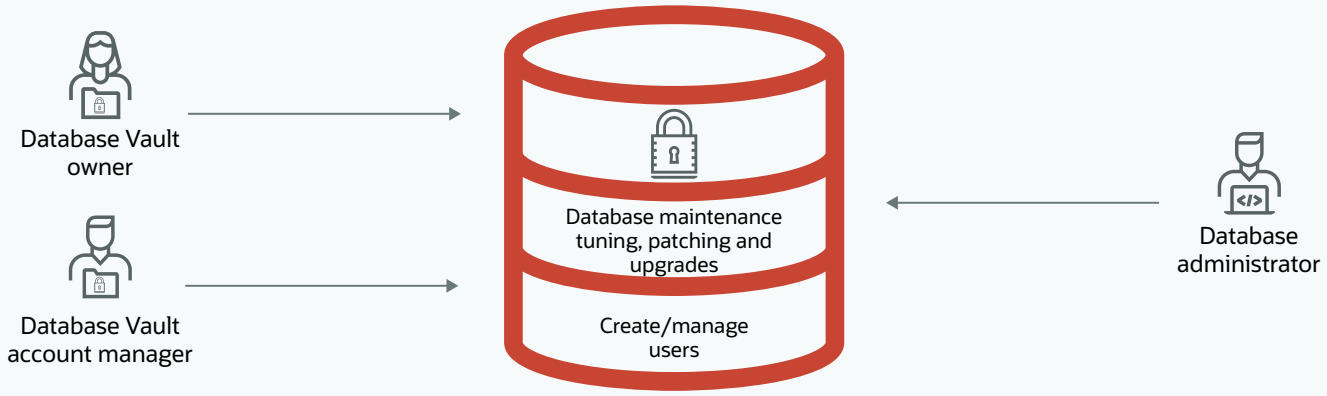


Figure 3.3: Database Vault separation of duties

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 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 enforced 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 quickly 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 the need for 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.

04 Data encryption and key management

Why encrypt data?

Even the strongest defense is useless if an attacker can just go around it. We can appreciate this with an example from the physical world. When attempting to rob a building, thieves will bypass the front door protected with a deadbolt and look for a less secure back door or an unlocked window. Similarly, database authentication and authorization secure the "front door," ensuring that data is only available to those who are authorized and not accessible to anyone else. However, if attackers are unable to gain access via the normal means, they might try to circumvent database access controls and go after the data in another way.

One way attackers can access data is by intercepting the data as it travels over the network, for example between a database client and a database server. On many networks, it is relatively easy for an attacker to capture network traffic and then extract whatever information was transmitted between the two systems as many internal network connections are not encrypted.

Another way for attackers to access data is by gaining access to the system as a privileged operating system user and directly read the database files, bypassing database controls. Attackers can also go after database backup files which might be stored on physical media and shipped to a remote location.

Encryption is the best technique for protecting data in these situations as it renders the data unintelligible to those who attempt to access it directly. With encryption, the problem of protecting a large amount of data is reduced to a much simpler problem of protecting an encryption key. As long as the attackers do not have the key, any encrypted data they manage to access 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 for the database. It also discusses the considerations for securely storing and managing 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 Network Encryption, is a standard feature of the Oracle Database. This feature can 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. All Oracle client software (thin and thick JDBC, instant client, SQL*Plus) support native network encryption and integrity protection as well as TLS.

Because of the very simple setup, and the fact that no additional external infrastructure or certificates are required, Oracle's native network encryption is used by a large majority of customers.

Oracle Database also supports industry-standard Transport Layer Security (TLS) which can support both authentication and encryption. TLS offers considerable flexibility. For example, TLS may be configured in server-authenticated mode (where the server presents an identifying certificate) to provide encryption of data in motion between the database and client. TLS may also be configured in mutually-authenticated mode (where both the server and the client possess an identifying certificate) to provide both encryption of data in motion and authentication of the database user. TLS certificates are stored in an Oracle wallet and may be self-signed or issued by a recognized certificate authority.

For TLS encryption, in most cases you will allow the server and client to negotiate the cipher suite used during communication (they will select the strongest mutually supported algorithm). However, if desired you may explicitly require certain algorithms. If desired, TLS encryption may be configured to operate in FIPS-140 mode, which enforces stricter control over which encryption libraries are used.

Security and performance often have an inverse relationship, and the same is generally true with TLS cipher suites. For example, `ECDHE`, the strongest default cipher suite, is also the one with the highest overhead. Although stronger from a security viewpoint, these algorithms will be less performant than algorithms using static keys (`RSA`, `ECDH` or `DHE`), and within that family, `ECDHE_RSA` is less performant than `ECDHE_ECDSA` and so on. If throughput degradation is a concern, encryption overhead should be considered when choosing your cipher suite. Note that either the database client OR server can override the default list, so if throughput concerns are only applicable to a limited number of clients, users have the flexibility to specify the stronger algorithms only for certain clients. For more information on configuring TLS encryption, refer to the Oracle Database Security Guide.

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 ensures that an attacker cannot read the information directly from those files.

There are multiple ways you could 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, time taken in converting data from plaintext to encrypted form, patching, and key management. One really needs to think through all these topics before selecting any given encryption strategy as it can have long-term effects.

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, but it also must manage encryption keys and securely store them somewhere. 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 on the server will not work. If multiple applications share the information in the same database, encrypting in the application requires they collaborate to encrypt and decrypt the same data in the same way. Further, application-level 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. It can also cause severe performance problems because the database might no longer be able to use indexes. Having said that, there are some narrow use cases where application-level encryption may make sense. For example, if you want to store some secret string or a blob where you do not want it to be visible to any other database user including the administrators, and where you are okay with not doing any database operations on that value, then application-level encryption could be used.

File or volume encryption is about encrypting the files with the database data. For example, you could use an encrypting file system (such as NFS). This seems simple but has a number of disadvantages when used to protect tablespace data in Oracle databases. First, file/volume encryption software lacks database user context. This means the Oracle Database running as OS user 'oracle' on the operating system and a human (malicious) user who also logs in as 'oracle' would both have access to the decrypted data. Another disadvantage is the overall inefficiency of a distributed database system when using volume encryption. For example, in order to transfer redo logs from a primary to standby database, first the data must be decrypted at the primary database, encrypted for transport across the network, decrypted at the destination, and then re-encrypted for storage on the standby database. Encrypted data cannot be simply copied to the receiving system because the source and target are each 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 encrypting the data in Oracle databases. What we need is a mechanism that improves security and reliability, while minimizing the management burden.

Transparent Data Encryption in Oracle databases

Oracle Transparent Data Encryption (TDE) 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.

TDE prevents 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 do anything special or different from the way they normally access the database. Instead, the database enforces the access control rules described in the previous chapters as before and denies access if the user is not authorized to see the data.

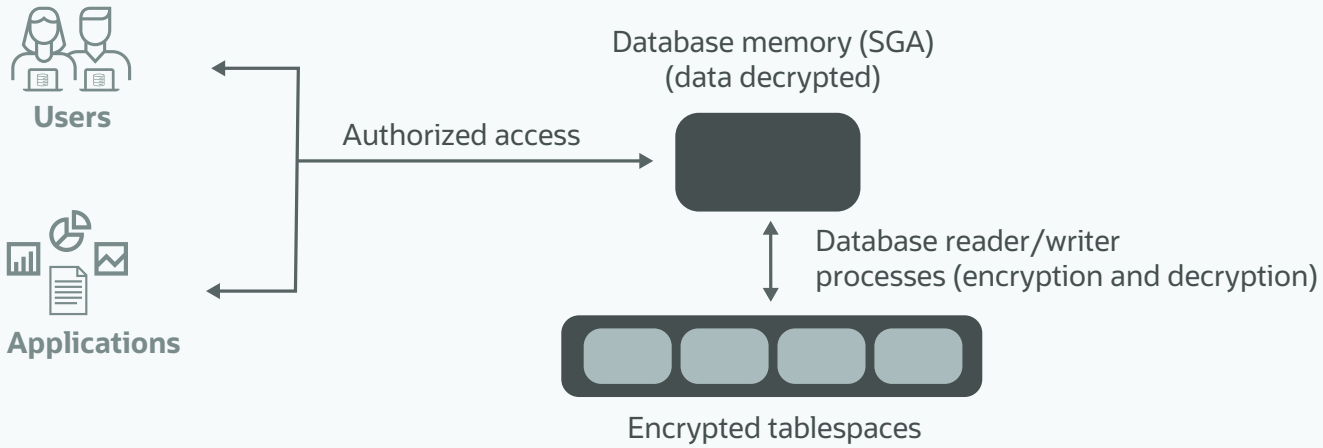


Figure 4.1: Users and applications work against cleartext data in database memory; for them, Transparent Data Encryption is 'transparent'.

TDE can encrypt both select application table columns, as well as entire tablespaces containing all application data. With tablespace encryption, you do not need to track which columns to encrypt or worry about 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';
```

Here 256 bit AES algorithm is used as the encryption algorithm. Even if your applications do not create encrypted tablespaces as part of the installation process, the database can be configured to automatically encrypt new tablespaces regardless.

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. In addition, TDE can encrypt entire database backups and Data Pump exports, and Oracle Recovery Manager (RMAN) and Data Pump both integrate with TDE encrypted data.

Oracle TDE is engineered to be highly performant. It automatically leverages special instructions in Intel (AES-NI) and SPARC CPUs to accelerate cryptographic operations. In addition, TDE tablespace encryption integrates with the performance optimizations built into Oracle's Engineered Systems. For example, TDE tablespace encryption works seamlessly with Exadata Hybrid Columnar Compression (EHCC) and Smart Scan technology.

Migrating cleartext data to encrypted tablespaces

Oracle TDE has become one of the most well-established ways to encrypt data-at-rest. However, many 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 migration from plaintext to encrypted data, or while converting from one encryption algorithm to another. An existing tablespace can easily be encrypted online with a command similar to:

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

Online encryption temporarily uses additional storage of the same size as the processed tablespace. Oracle also provides two offline encryption methods that do not require additional storage, but these require the tablespace to be offline, or the database to be in mount mode. In Oracle Database 12.2 and later, an existing tablespace can be encrypted with a command similar to:

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

In Oracle Database 11.2.0.4 and later, multiple datafiles can be encrypted offline in parallel with a command similar to:

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

In you are using Oracle Data Guard where the standby databases are already in mount mode, offline encryption can be used to minimize the downtime without the need for additional temporary storage space. The procedure is to turn off the managed recovery process, encrypt the standby database's data files, perform a switch-over, and then encrypt the former primary database. In this scenario, the downtime for encrypting any size database is as short as the duration of switch-overs.

Oracle TDE two-tier key architecture

The security of encrypted data depends on maintaining the secrecy of the keys used for encryption. Proper key management is essential for preventing an attacker from discovering the encryption key and gaining access to data. Proper key management also ensures that active keys are not lost, that keys are rotated periodically, and that old keys are securely archived to provide continued access to encrypted 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 key somehow gets exposed.

Oracle 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 in turn used to encrypt columns and tablespaces. DEKs are managed by the database behind the scenes. The two-tier key architecture simplifies key rotation. When the MEK is replaced, there is no need to re-encrypt all data, but only the much smaller set of DEKs. If DEKs need to be rotated or if they need to encrypt data with another algorithm, customers can use online tablespace encryption as described above.

Administrators perform key management operations (create, open, rotate, backup, etc.) either through a series of SQL commands, or by using Oracle Enterprise Manager. Oracle Database 12c introduced the SYSKM privilege to allow key management operations without requiring the powerful SYSDBA privilege.

The MEK is separated from encrypted data, stored outside of the database, and directly managed by the database security administrator in a keystore. This can be a local keystore like the Oracle wallet or a centralized keystore such as Oracle Key Vault. Starting with Oracle Database 18c, customers can import externally generated MEKs into the wallet, enabling “bring your own key” for wallet-based TDE deployments.

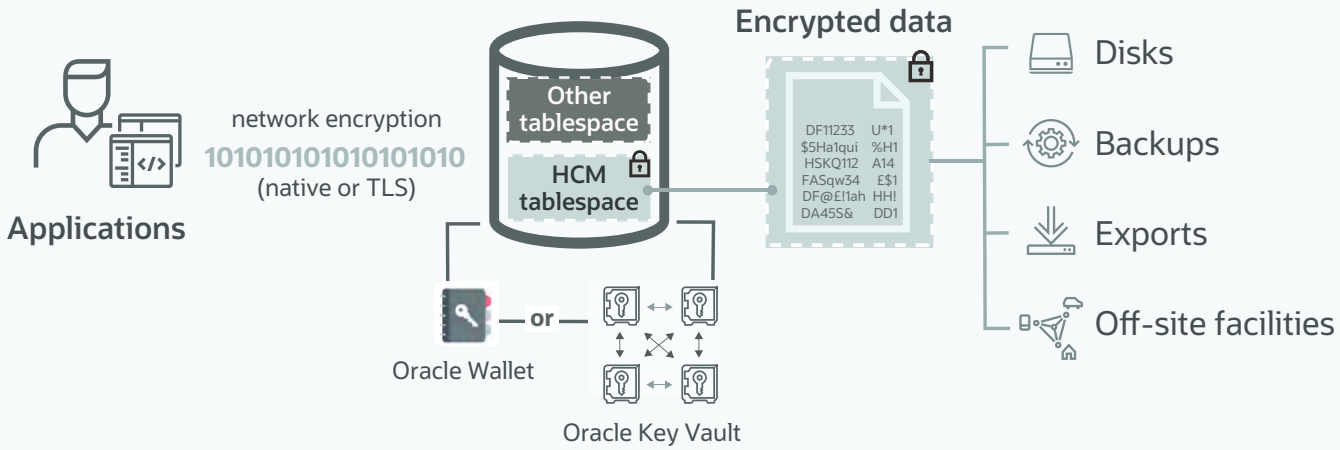


Figure 4.2: Oracle TDE encryption keys can be managed locally on the server in Oracle wallets, or in a centralized keystore such as Oracle Key Vault.

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 the master encryption key in a PKCS#12 standard-based file called Oracle Wallet. The content of the wallet is encrypted using a key derived from the wallet password. Note that if the wallet is somehow lost or corrupted or the password is forgotten, there is no way to recover the data that is encrypted using the MEK in the wallet. Hence, it is important to securely back up the password protected wallet. Oracle Key Vault, described in the next section, provides an automated solution for backing up and managing these wallets.

Centralized key management with Oracle Key Vault

With the increasing number of encrypted databases, managing wallets can become a burden. Data can be at risk if wallets are lost or stolen, or if passwords are forgotten. Furthermore, many regulations do not allow the encryption keys to be stored on the same server as the encrypted data, and many organizations require a clear separation of duties between DBAs and key management administrators. To ease the key management burden, and reduce risk of losing access to wallets, customers need external key management solutions.

Oracle Key Vault (OKV) is an enterprise-grade centralized key manager for your Oracle databases and provides continuous availability, security, and scalability for master keys, wallets, and secrets. Oracle Key Vault 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 are active, significantly lowering the total cost of ownership.

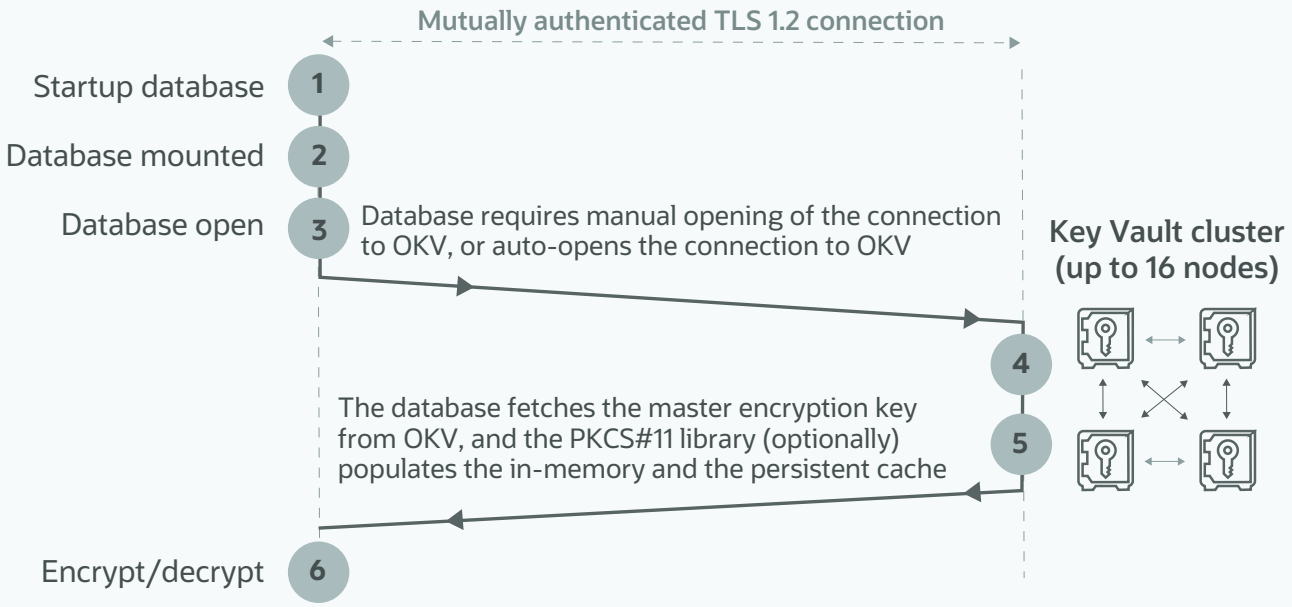


Figure 4.3: Oracle Key Vault securely delivers MEKs to Oracle databases on demand.

Databases can connect to any node in the Oracle Key Vault cluster to retrieve master encryption keys. Any updates to keys or changes to authorization rules are synchronously replicated to a partner Oracle Key Vault node, ensuring zero data loss in the event of a node failure. If the current Oracle Key Vault connection fails or if a node goes down for any reason, the database servers transparently failover to the nearby active nodes for read/write operations without down time or user intervention.

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

Oracle Key Vault also supports **Secrets Management**. This capability allows you to upload database account passwords for maintenance scripts (nightly RMAN backups, batch loading into a data warehouse, refreshing materialized views) into Oracle Key Vault and securely retrieve them from there. This reduces the management burden of maintaining individual Secure External Password Stores (SEPS) for automated database operations and simplifies management and sharing of those passwords between databases. Additionally, private SSH keys can be uploaded into Oracle Key Vault to protect from loss and abuse of stolen keys.

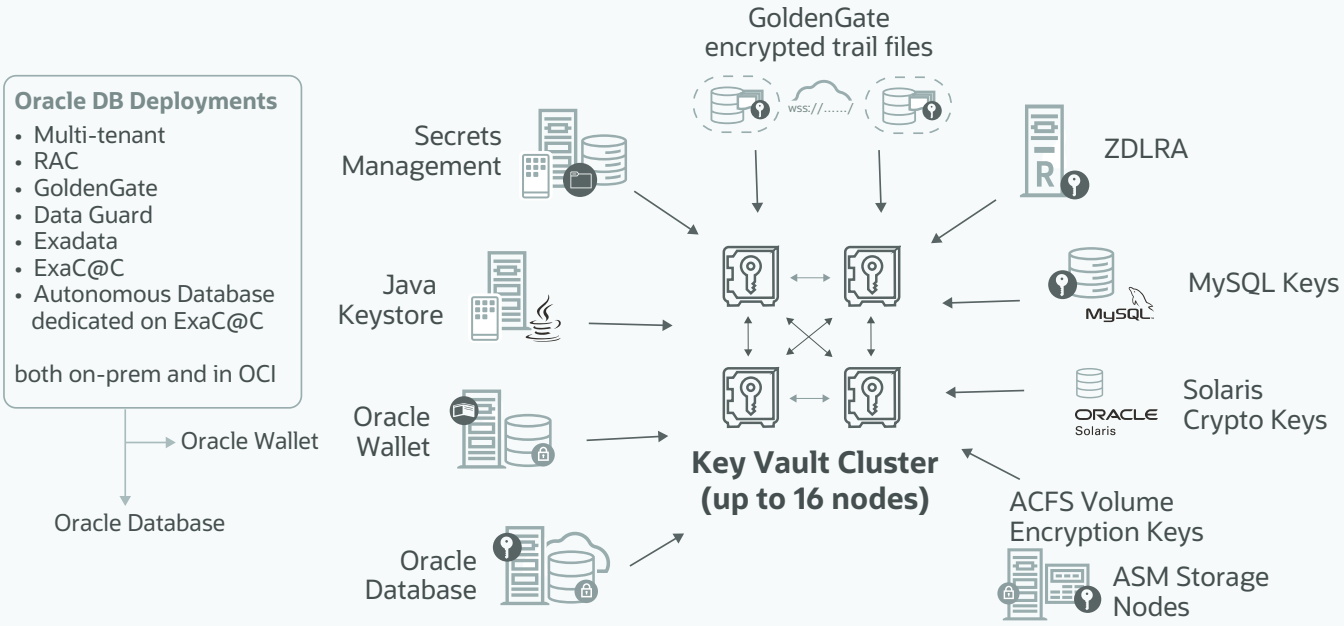


Figure 4.4: Oracle Key Vault manages Oracle TDE MEKs, as well as other keys and security objects.

To on-board a large number of databases, Oracle Key Vault administrators can use the supplied RESTful APIs to prepare a set of scripts for the DBAs to run. Seconds later, the DBAs can begin encrypting their databases while the security team securely manages the master keys within Oracle Key Vault. Oracle Key Vault also supports "bring your own key" for Oracle Key Vault-based TDE deployments.

Oracle Key Vault provides separation of duties for its administrators. The Oracle Key Vault System Administrator, Key Administrator, and Auditor roles 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 be used to automate the collection of audit records from Oracle Key Vault.

Oracle Key Vault deploys as a security-hardened software appliance that uses the industry standard OASIS Key Management Interoperability Protocol (KMIP) for communications. It is also available from the Oracle Cloud Marketplace (https://cloudmarketplace.oracle.com/marketplace/app/OracleKeyVault), from where it can be deployed in your Oracle Cloud Infrastructure (OCI) tenancy in minutes. Oracle Key Vault servers can span your data centers and OCI regions, providing fault-tolerant hybrid key management for your hybrid-cloud database deployments.

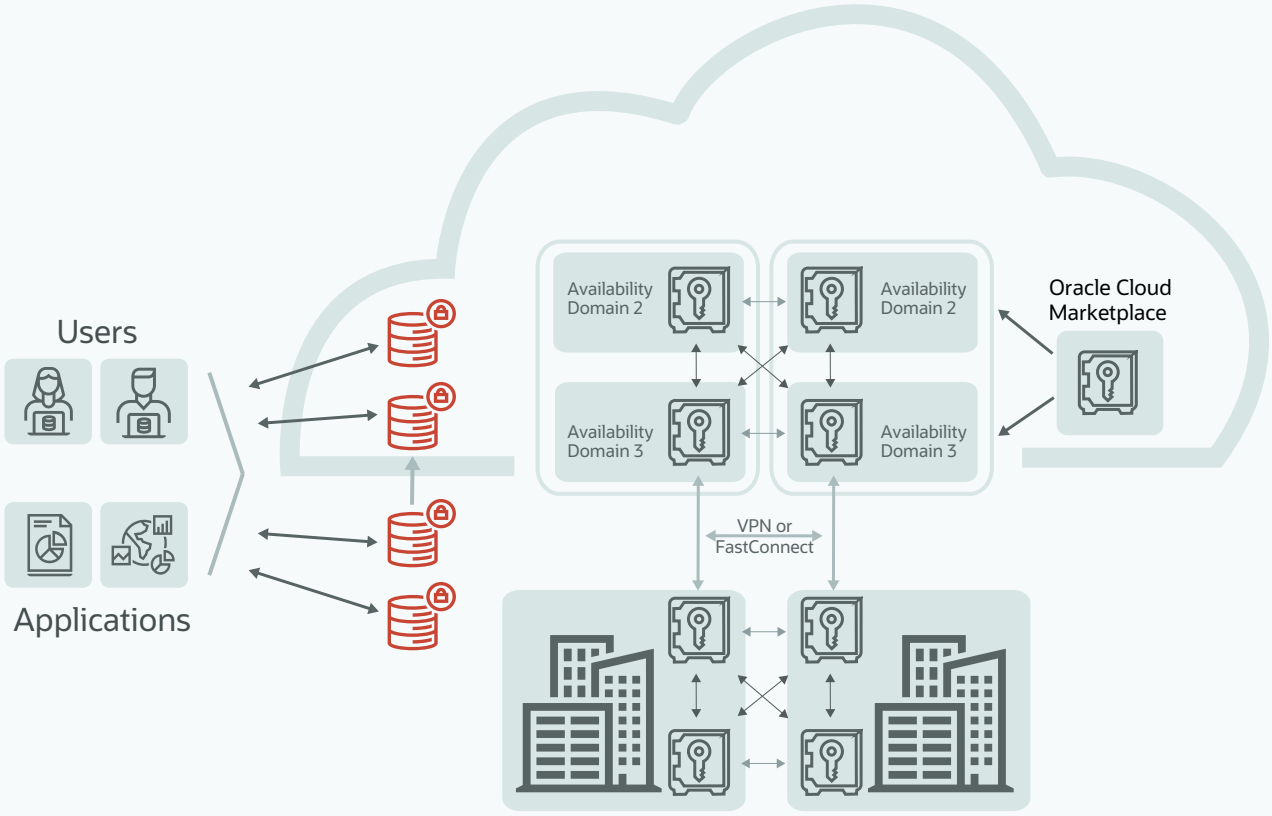


Figure 4.5: Oracle Key Vault supports your journey to the cloud.

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, simplicity of installation and use, transparency for existing applications, data migration from plaintext to encrypted form, patching, and key management.

TLS and Oracle native network encryption are standard features of the Oracle Database for protecting data in motion. TDE safeguards sensitive data against unauthorized access from outside of the database by encrypting data at rest. It prevents privileged operating system users from bypassing controls and directly inspecting the contents of database files. It also protects against theft, loss, or improper decommissioning of database storage media and backups. Oracle Key Vault eases the management overhead of mass deployment of TDE by providing a centralized key management platform. Encryption and centralized key management work together to help address privacy regulations and standards such as the PCI-DSS, HIPAA, and EU GDPR.

05 Discovering sensitive data

The amount of data that organizations collect and manage is growing every day. Data has become the fuel for developing better products and services, making smarter business decisions, gaining strategic advantage over competitors, and growing business. Even government policies and programs are driven by data today. In today's world, data has become the most valuable resource and a necessity for every organization.

A significant percentage of data is usually sensitive and personal. This data, if in the wrong hands, can be monetized by committing identity theft and financial frauds, selling government and trade secrets, or using it for future attacks. Data, being the most valuable resource, has become the prime target of cybercriminals. 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.

To protect sensitive data, the first step is to understand what kind and how much sensitive data a database has and where it is located. This knowledge can be used to implement appropriate security controls to protect data.

Identifying and locating sensitive data is hard and often does not happen reliably. Many organizations have difficulty finding 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, quite simply, is 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 the categories and types of sensitive data usually found.

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

Figure 5.1: Examples of sensitive data categories and types

Discovering sensitive data

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

Such data patterns can be used to check one of the values in a column. As there can be data quality issues, a better approach is to check multiple values and flag a column as sensitive if a high fraction, say 80%, of the values match.

A sensitive data type (or sensitive type) is basically a container of these search patterns and defines what to search for. For example, the following sensitive type 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}
```

In addition, data types of columns can be used 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.

Good data discovery tools automate the discovery process by combining the discussed methods and by adopting other advanced techniques and optimizations. Such multi-pronged approach helps minimize false positives and false negatives. Oracle provides multiple sensitive data discovery tools described in the following sections.

Discovering sensitive data using DBSAT

Oracle Database Security Assessment Tool (DBSAT) is a standalone command-line tool that not only helps discover sensitive data but also 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 broader DBSAT capabilities.

The data discovery component of DBSAT helps identify sensitive columns in Oracle databases using sensitive types. DBSAT Discoverer collects the metadata (column names and comments) and matches it against the patterns defined as part of 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 is associated with "`Identification Info`" category and "`Public IDs`" subcategory.

```
[Full Name]
COLUMN NAME PATTERN = (CUSTOMER|CUST|CLIENT|FULL|PATIENT|PERSON).?(NAME|NM)
COLUMN 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 that provides information about the number of tables, columns, and rows identified as sensitive, grouped by sensitive category and subcategory. In the figure, * indicates the number of unique tables with sensitive data, and ** indicates the number of unique rows with sensitive data.

Sensitive category	# Sensitive tables	# Sensitive columns	# Sensitive rows
BIOGRAPHIC INFO - ADDRESS	9	36	6307209
BIOGRAPHIC INFO - EXTENDED PII	2	2	2000
FINANCIAL INFO - BANK DATA	2	2	830
FINANCIAL INFO - CARD DATA	7	7	3235
HEALTH INFO - PROVIDER DATA	1	1	149
IDENTIFICATION INFO - NATIONAL IDS	2	6	2000
IDENTIFICATION INFO - PERSONAL IDS	3	3	405
IDENTIFICATION INFO - PUBLIC IDS	9	26	2401125
IT INFO - USER DATA	13	15	13228
JOB INFO - COMPENSATION DATA	10	12	3380
JOB INFO - EMPLOYEE DATA	8	16	406
JOB INFO - ORG DATA	5	6	278
TOTAL	**29***	**132**	**8617644****

Figure 5.2: Sensitive category level summary

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

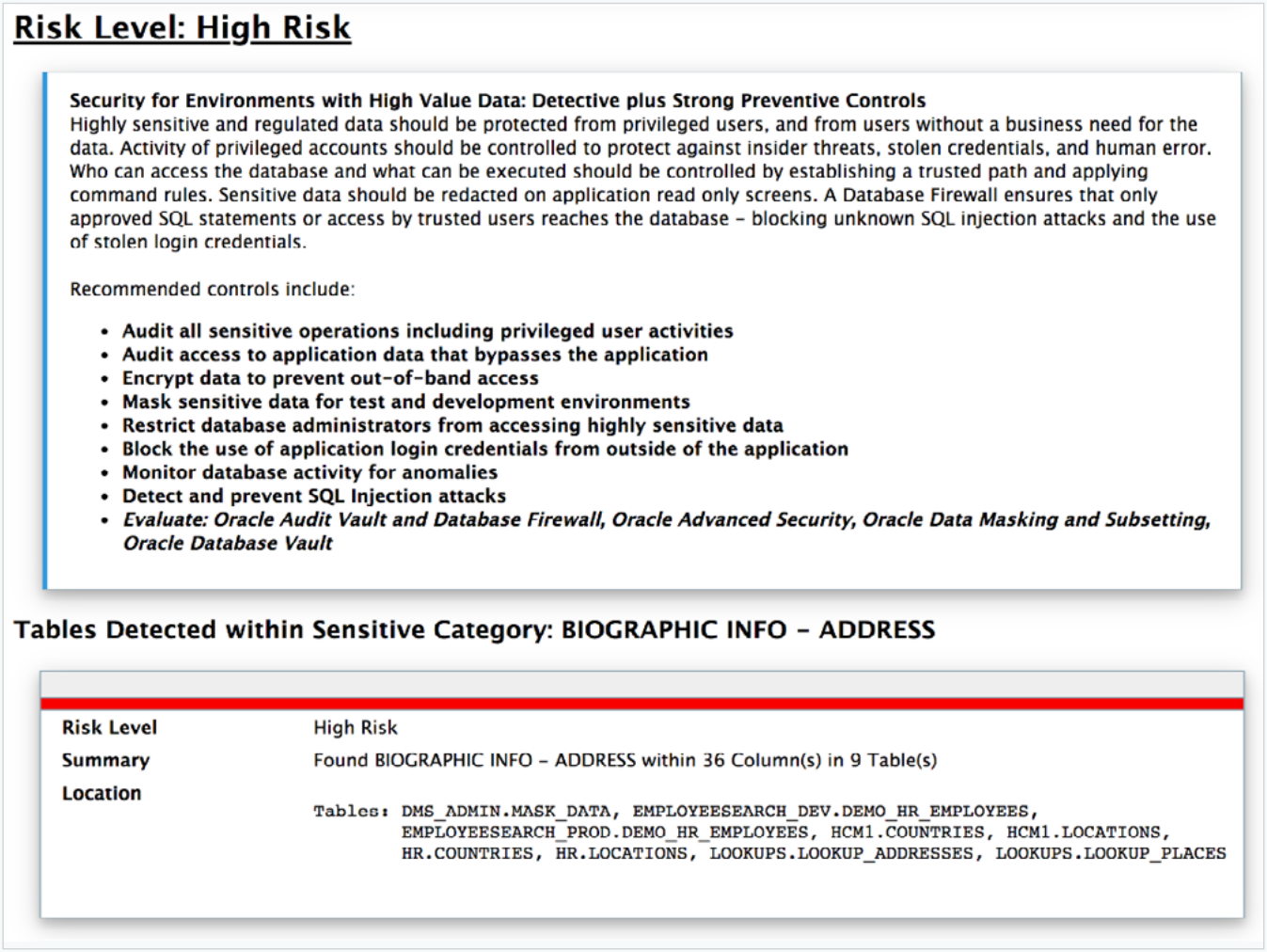


Figure 5.3: Recommendations for protecting data with high risk

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
DMS_ADMIN	MASK_DATA	9	7	10000	BIOGRAPHIC INFO - ADDRESS, IDENTIFICATION INFO - PUBLIC IDS, IT INFO - USER DATA
EMPLOYEES EARCH_DEV	DEMO_HR_ EMPLOYEES	34	16	1000	BIOGRAPHIC INFO - ADDRESS, BIOGRAPHIC INFO - EXTENDED PII, FINANCIAL INFO - CARD DATA, IDENTIFICATION INFO - NATIONAL IDS, IDENTIFICATION INFO - PUBLIC IDS, IT INFO - USER DATA, JOB INFO - COMPENSATION DATA
EMPLOYEES EARCH_DEV	DEMO_HR_ROLES	2	1	26	IT INFO - USER DATA
EMPLOYEES EARCH_DEV	DEMO_HR_ SUPPLEMENTAL_ DATA	6	4	415	FINANCIAL INFO - BANK DATA, FINANCIAL INFO - CARD DATA, IT INFO - USER DATA, JOB INFO - COMPENSATION DATA
EMPLOYEES EARCH_DEV	DEMO_HR_USERS	8	5	12	IDENTIFICATION INFO - PUBLIC IDS, IT INFO - USER DATA
EMPLOYEES EARCH_DEV	DEMO_HR_USER_ LABELS	2	1	12	IT INFO - USER DATA

Figure 5.4: Table level summary

Schema name	Table name	Column name	Column comment	Sensitive category	Sensitive type	Risk level
DMS_ADMIN	MASK_DATA	CITY	--	BIOGRAPHIC INFO - ADDRESS	CITY	High Risk
DMS_ADMIN	MASK_DATA	GIVENNAME	--	IDENTIFICATION INFO - PUBLIC IDS	FIRST NAME	High Risk
DMS_ADMIN	MASK_DATA	STREETADDRESS	--	BIOGRAPHIC INFO - ADDRESS	STREET	High Risk
DMS_ADMIN	MASK_DATA	SURNAME	--	IDENTIFICATION INFO - PUBLIC IDS	LAST NAME	High Risk
DMS_ADMIN	MASK_DATA	TELEPHONENUMBER	--	IDENTIFICATION INFO - PUBLIC IDS	PHONE NUMBER	High Risk
DMS_ADMIN	MASK_DATA	USERNAME	--	IT INFO - USER DATA	USER ID	High Risk
DMS_ADMIN	MASK_DATA	ZIPCODE	--	BIOGRAPHIC INFO - ADDRESS	POSTAL CODE	High Risk
EMPLOYEESEARCH_DEV	DEMO_HR_EMPLOYEES	ADDRESS_1	--	BIOGRAPHIC INFO - ADDRESS	FULL ADDRESS	High Risk
EMPLOYEESEARCH_DEV	DEMO_HR_EMPLOYEES	ADDRESS_2	--	BIOGRAPHIC INFO - ADDRESS	FULL 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 (ADM) feature in Oracle Enterprise Manager provides the capability 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, ADM examines the metadata, i.e., column names and comments, to discover sensitive columns. In addition, it 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, ADM 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 technologies.

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.

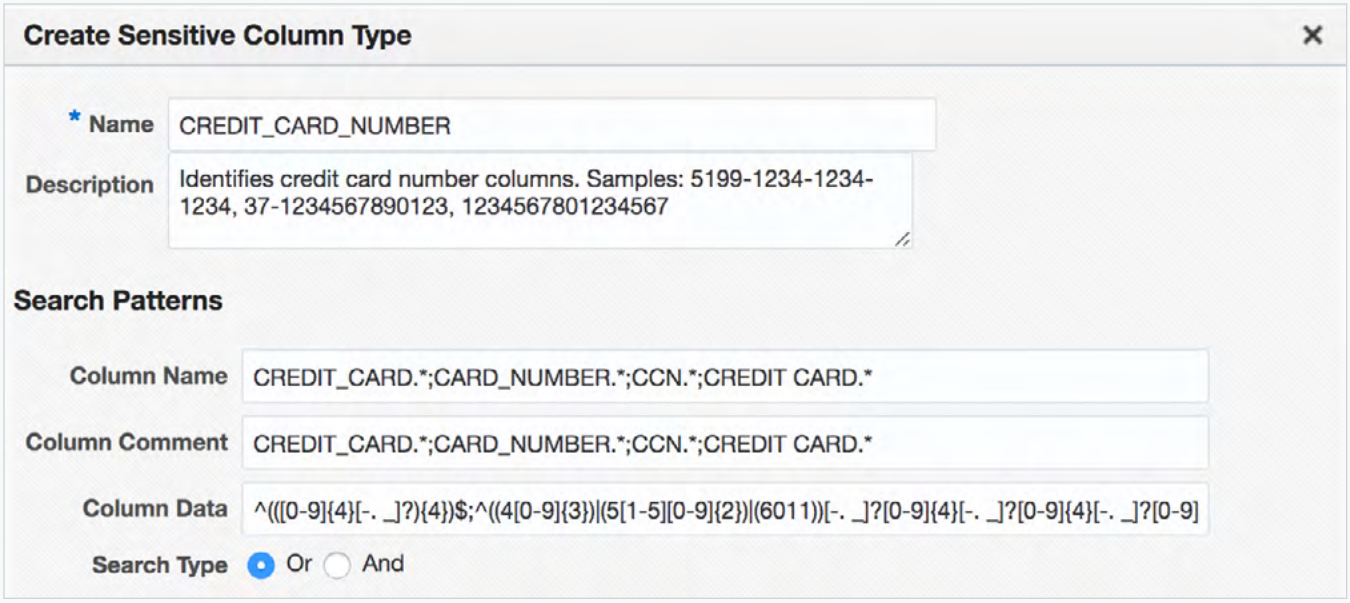


Figure 5.6: Sensitive column type definition

Oracle provides a set of predefined sensitive column types, and users can create new ones 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

ADM 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.

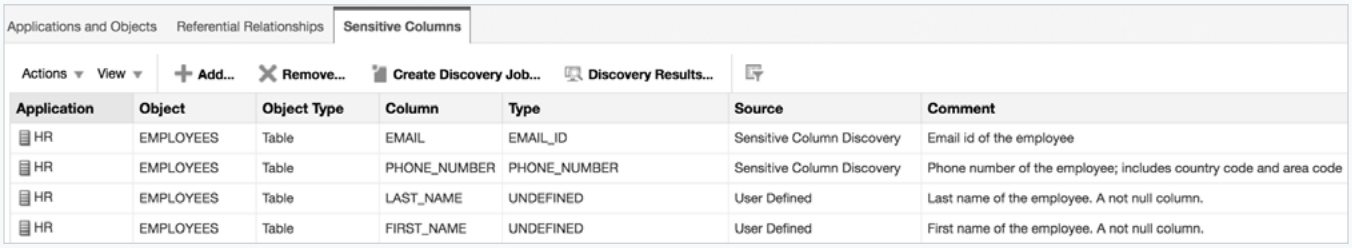


Figure 5.8: Sensitive columns

ADM 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 in the database. Users can review the discovered referential relationships and add additional relationships manually.

Understanding such dependencies helps preserve application integrity during data masking by ensuring that the data in the related columns is masked consistently. Figure 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: Referential relationships

Figure 5.10 presents the overall picture putting all this together. ADM 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.

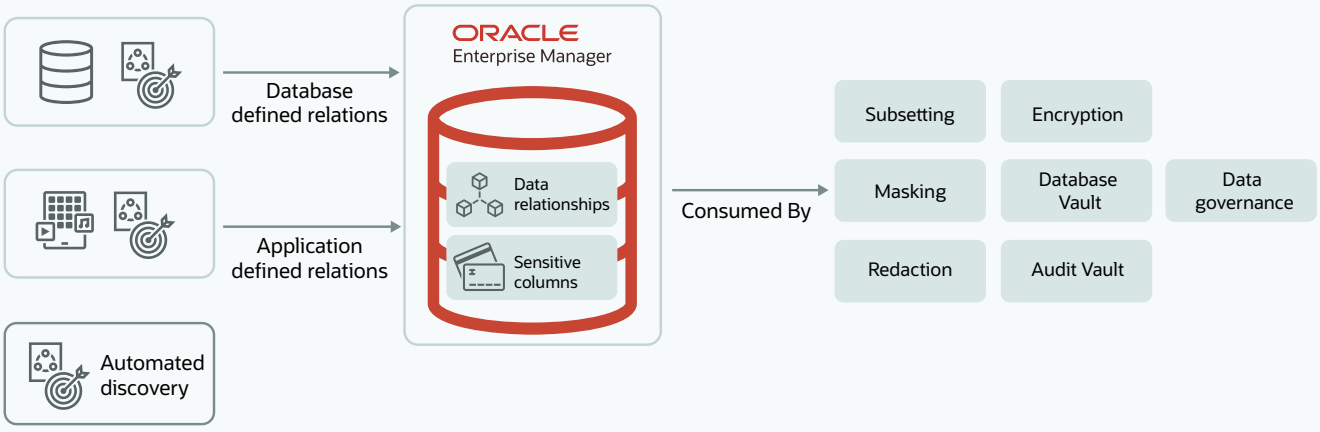


Figure 5.10: Overview of application data modeling

Discovering sensitive data using Oracle Data Safe

Many organizations have either already moved to the cloud or are planning to move in the near future. Security in the cloud follows a shared responsibility model. Depending on the cloud delivery model (IaaS, PaaS, or SaaS), some layers of the stack fall under the responsibility of the service provider, while others fall under the responsibility of the service consumer. One aspect that is common to both cloud and on-premises models is that customers should protect and manage their data, users, and security configurations. To better protect data with appropriate security controls, it is imperative to know what sensitive or regulated data is stored in databases.

Oracle Data Safe, a database security cloud service, provides a unified control center for protecting both cloud and on-premises databases. Data Safe provides an integrated set of security features, including sensitive data discovery. Using an extensible set of 130+ sensitive types, it helps discover and classify sensitive data. 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.

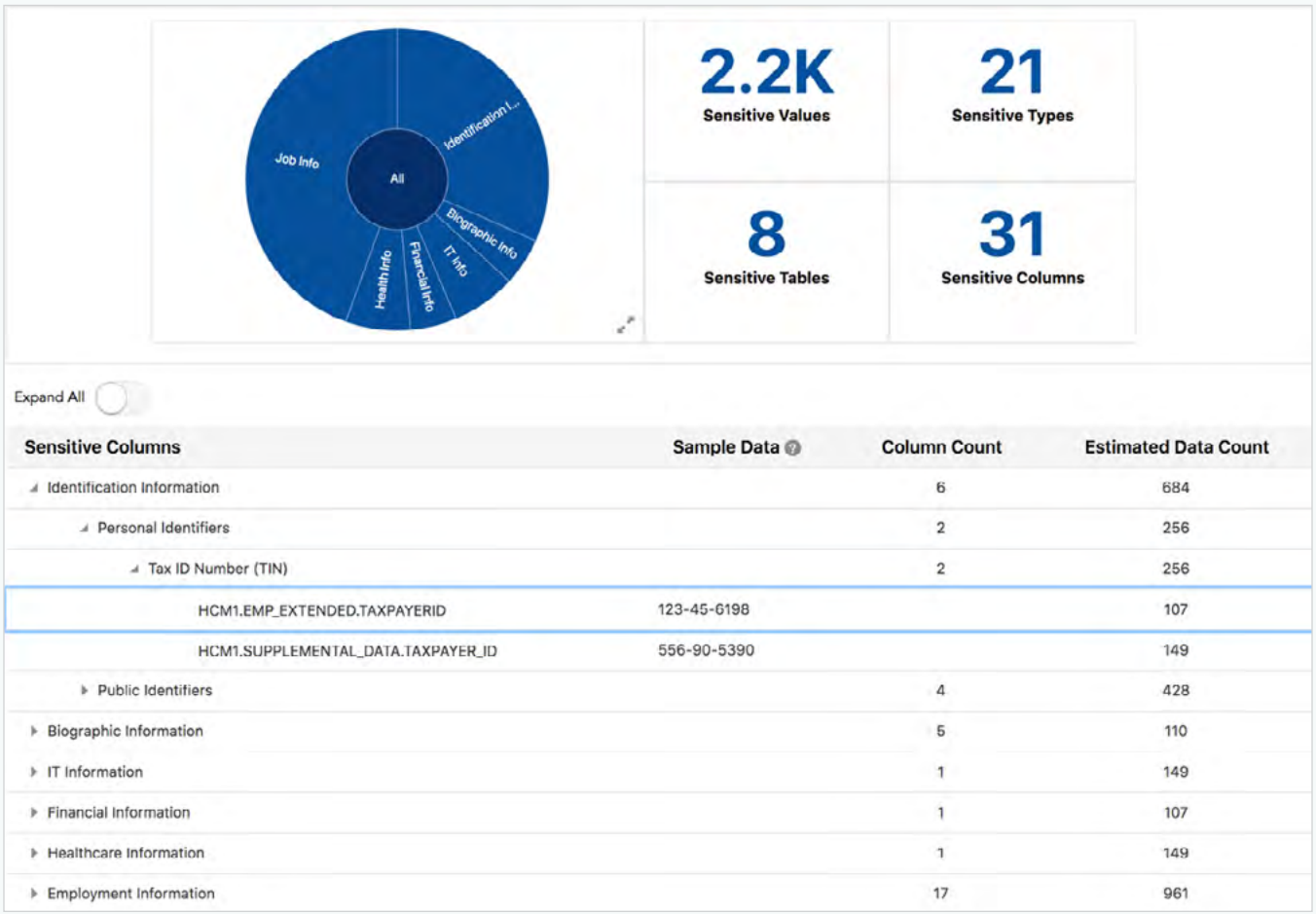


Figure 5.11: Data discovery report in Oracle Data Safe

Summary

Understanding sensitive data is the first step to implementing appropriate security controls to protect data. Oracle provides multiple sensitive data discovery tools: Database Security Assessment Tool (DBSAT), Enterprise Manager (EM) 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. The EM Application Data Modeling scans both metadata and data to discover sensitive columns and referential relationships. It helps understand the dependencies and preserve application integrity during data masking and subsetting. The Oracle Data Safe cloud service provides a comprehensive data discovery feature and is the recommended option for discovering sensitive data in Oracle Cloud and on-premises 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, PCI-DSS, and HIPAA.

06 Masking sensitive data

Reducing the exposure of sensitive data is a challenge faced by many organizations. This data can include credit card numbers, Social Security numbers (SSN), sales figures, and other personal or proprietary information. The problem is limited not just to production systems, but any system which has copies of the data. For example, snapshots of live production data are typically shared with development and data analytics teams so that they have access to realistic data to support their activities. Attackers have learned that valuable and sensitive data often ends up in non-production 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 SSN in order to verify their identity. While the system might require SSN for other purposes such as credit reporting, displaying SSN 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.

Multiple ways to mask sensitive data

Organizations have multiple ways to mask or anonymize sensitive data to minimize exposure and the associated risk.

Data masking, sometimes called "static masking", involves permanently altering data so that it is no longer sensitive. Masking sensitive data before it is handed over to development and analytics 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, sometimes referred to as "dynamic masking", obfuscates data on the fly as it is retrieved by a specific user but does not actually change the stored data. 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 data set. For example, the analytics team might need only 20% of the production data, data collected only over the last year, or data specific to a region. Data subsetting provides relevant data, 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 help facilitate compliance with various regulations 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 data masking, data subsetting and data redaction, and describes how these techniques can be applied to reduce the attack surface for sensitive data within applications.

Masking sensitive data

Most organizations 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 the EU GDPR also emphasize minimizing risk in non-production environments because these systems are typically not as protected or monitored as production systems.

Oracle Data Masking and Subsetting, an Oracle Enterprise Manager pack, masks and subsets data for non-production use such as development and analytics. After creating an application data model (ADM) consisting of sensitive columns and referential relationships, Oracle Data Masking is used to create a masking definition with rules to mask sensitive columns. For example, as shown in Figure 6.1, names can be replaced with realistic names and SSNs can be replaced with random SSN values. 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.

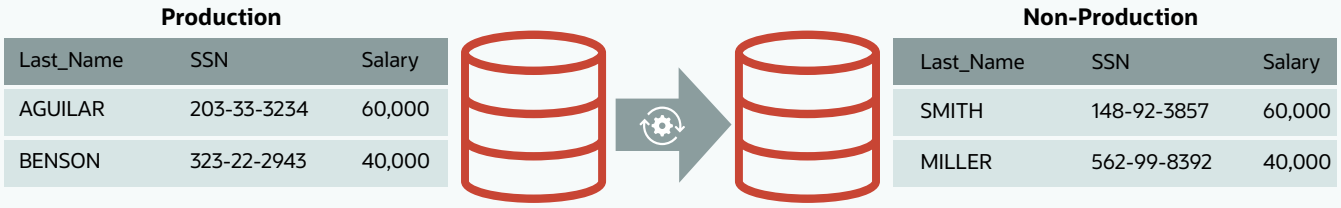


Figure 6.1: Oracle Data Masking overview

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, but realistic, credit card numbers for varieties of credit cards helps preserve diversity in the masked data set.

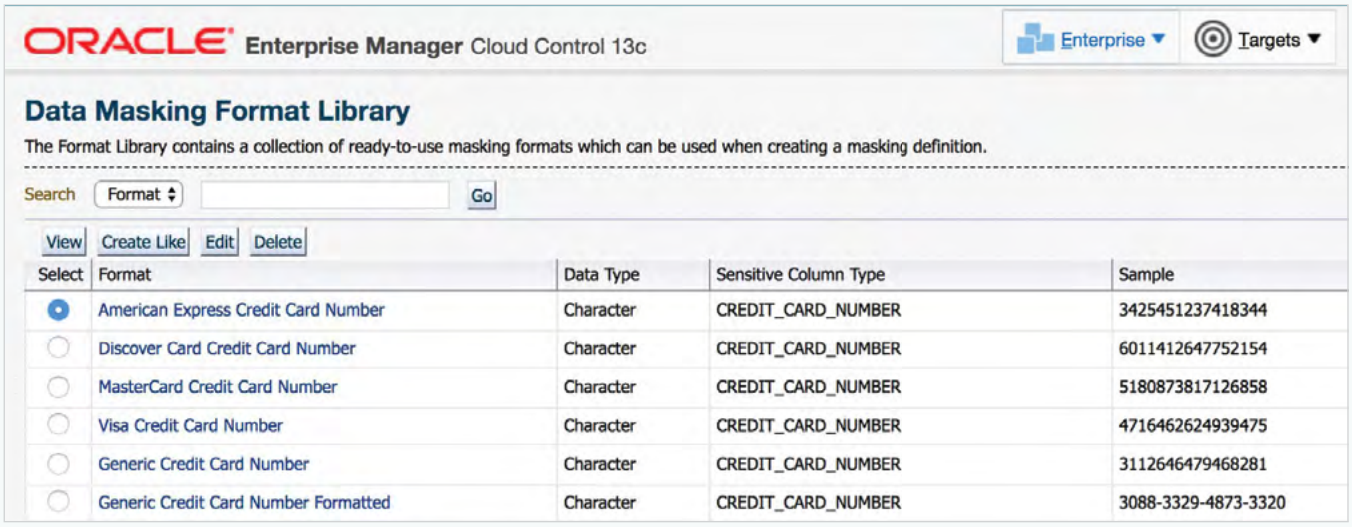


Figure 6.2: Credit card masking formats

By leveraging the masking format library, organizations can easily 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 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.

Powerful masking techniques

Oracle provides a comprehensive set of masking techniques, enabling complex applications to function with masked data in a variety of non-production environments. Figure 6.3 describes some common masking techniques.

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 shows some of the supported masking techniques in action. For example, national identifiers can be masked differently using conditions based on country codes, health records can be shuffled, names can be masked in a deterministic manner across databases and masking runs, and license plate numbers can be replaced with random characters while preserving the original format.

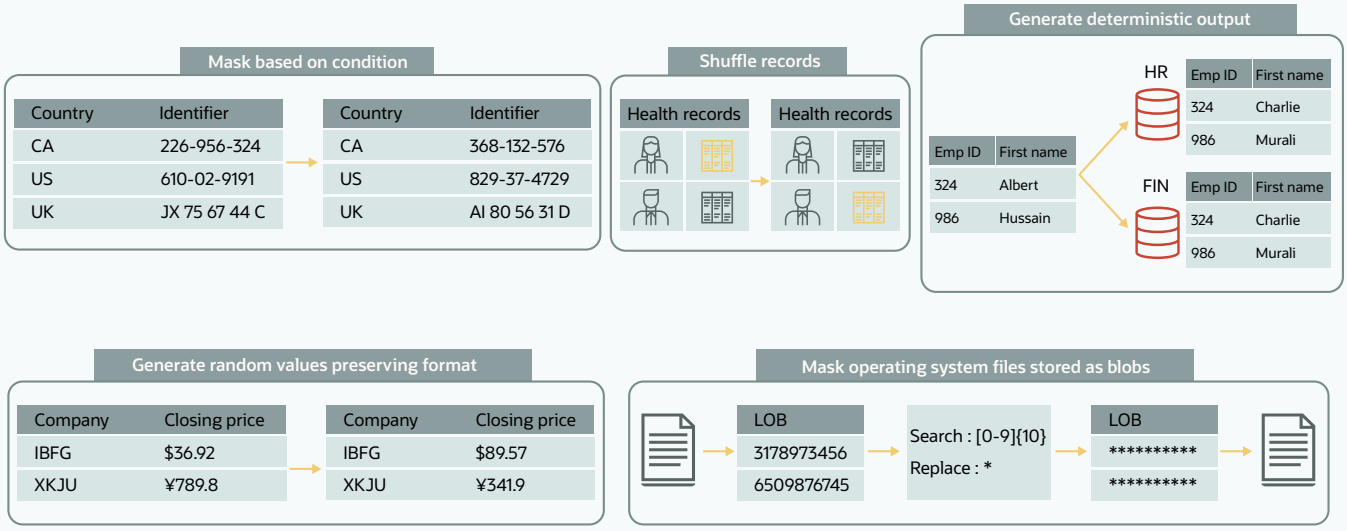


Figure 6.4: Data masking examples

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 development and analytics, 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.

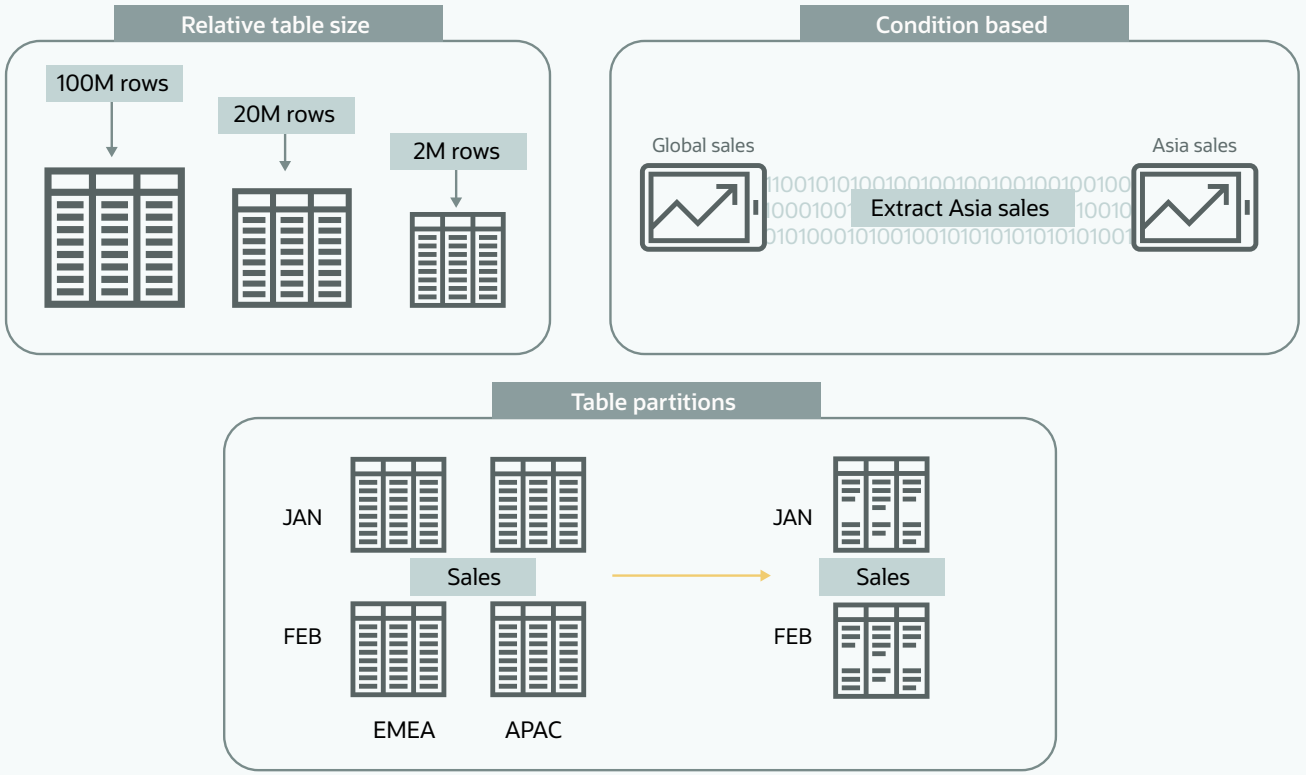


Figure 6.5: Data subsetting use cases

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 safely. 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, data scientists, and other third-party consumers.

But some organizations may have a 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.

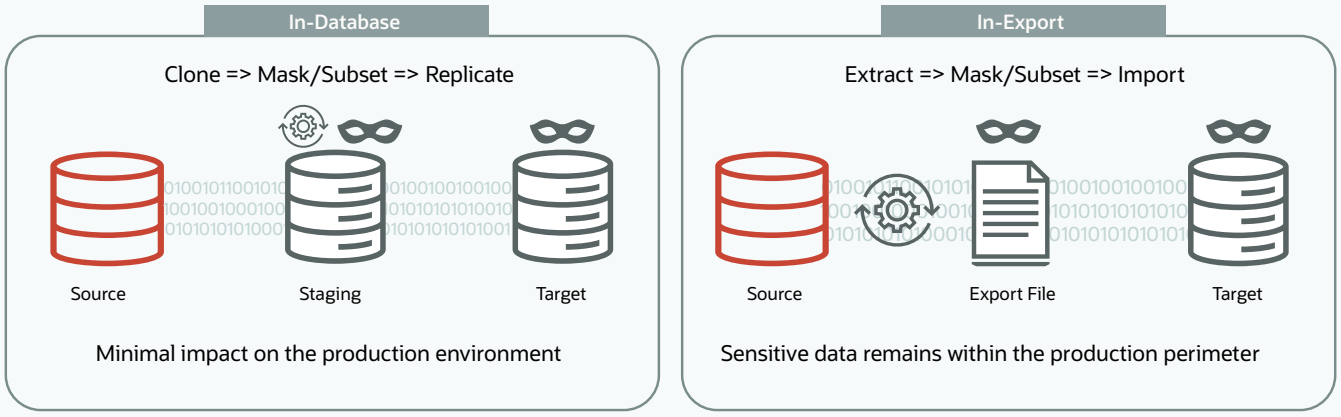


Figure 6.6: Data masking and subsetting modes

In-database mode allows masking and subsetting of data within a non-production database, with minimal or zero impact on the production environment. As this mode permanently changes the data stored in a database, it should never be done on a production instance.

Alternatively, the 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 files, which can be further imported into non-production databases. This mode eliminates the need for staging servers and helps ensure that sensitive data never leaves the production perimeter. However, this process should be done at a suitable time so that it doesn't interfere with the peak load time.

Masking sensitive data using Oracle Data Safe

Organizations are rapidly adopting cloud, with many preferring it for application development and analytics. It enables them to cost-effectively build, deploy, and manage applications. For continuous and quality development, teams prefer to use realistic data from production environments, but they need to ensure that there are no security and compliance risks.

The Oracle Data Safe cloud service provides tightly-integrated sensitive data discovery and data masking capabilities that help identify and mask sensitive data in both cloud and on-premises databases, all available through simple point-and-click interface. See *Chapter 13* for more information on how Oracle Data Safe can help meet security requirements, including data masking.

Redacting sensitive data

There is 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 modify the application access control logic to redact sensitive data. 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.

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. Further, organizations might not have access to the code such as in the case of packaged applications.

The best approach for controlling the exposure of sensitive data in applications is by enforcing redaction policies outside of the application control and from 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 prior to display by applications. This enables consistent redaction of database columns across multiple 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.

Oracle Data Redaction supports a number of different transformations as shown in Figure 6.7.

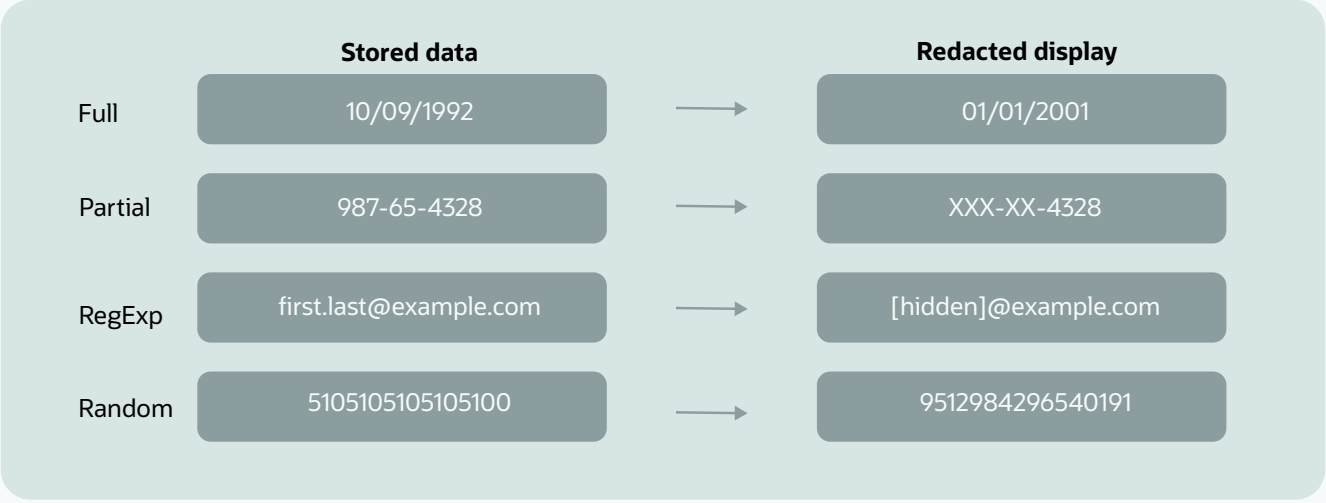


Figure 6.7: Examples of data redaction

Figures 6.8 and 6.9 help understand how the application users experience data redaction. Figure 6.8 shows HR data in original form without any data redaction. Figure 6.9 shows the redacted data in SSN, corporate card, position and salary fields using partial, regex, random and full transformations respectively.

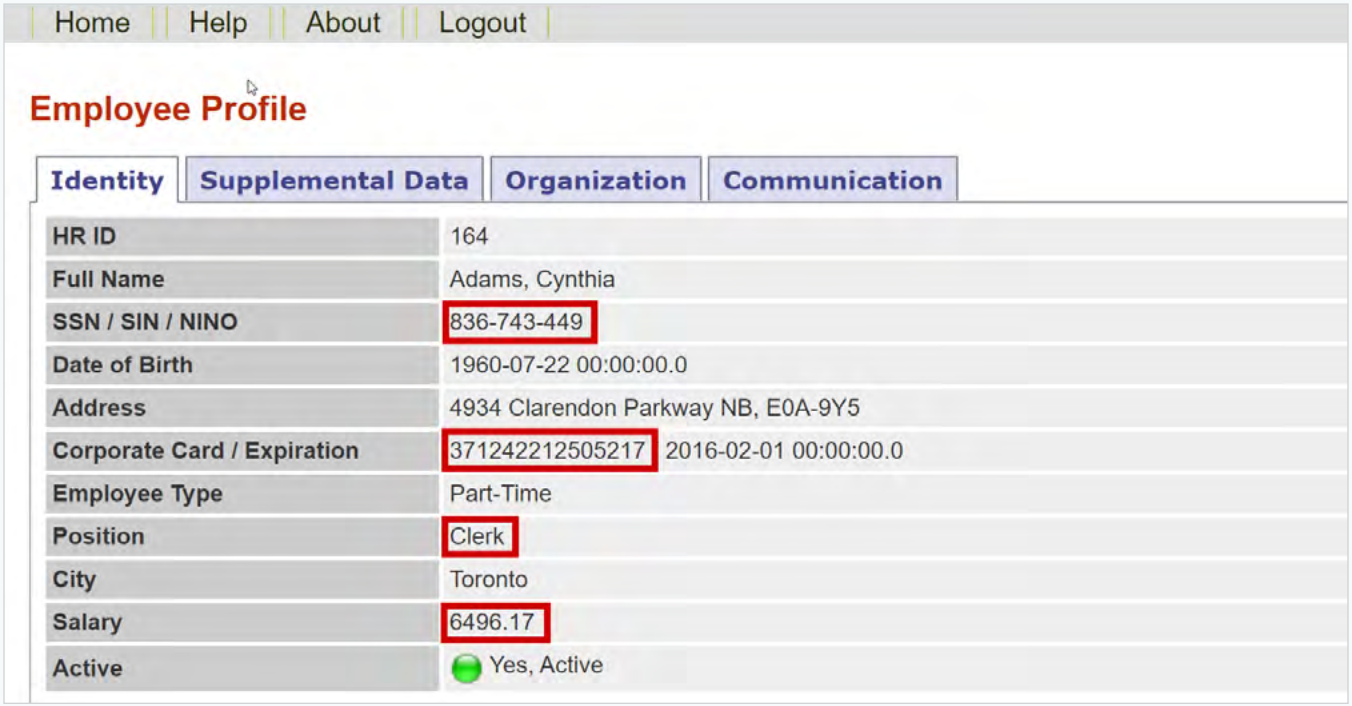


Figure 6.8: Non-redacted HR data

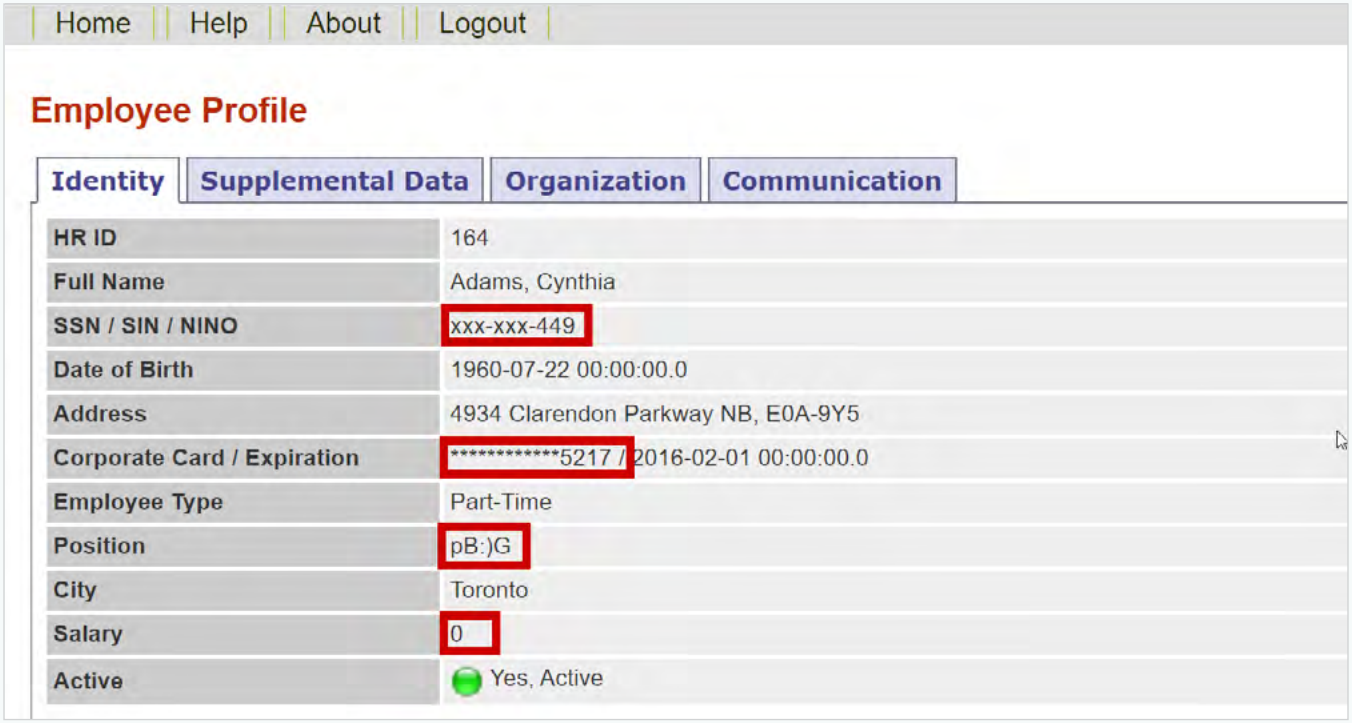
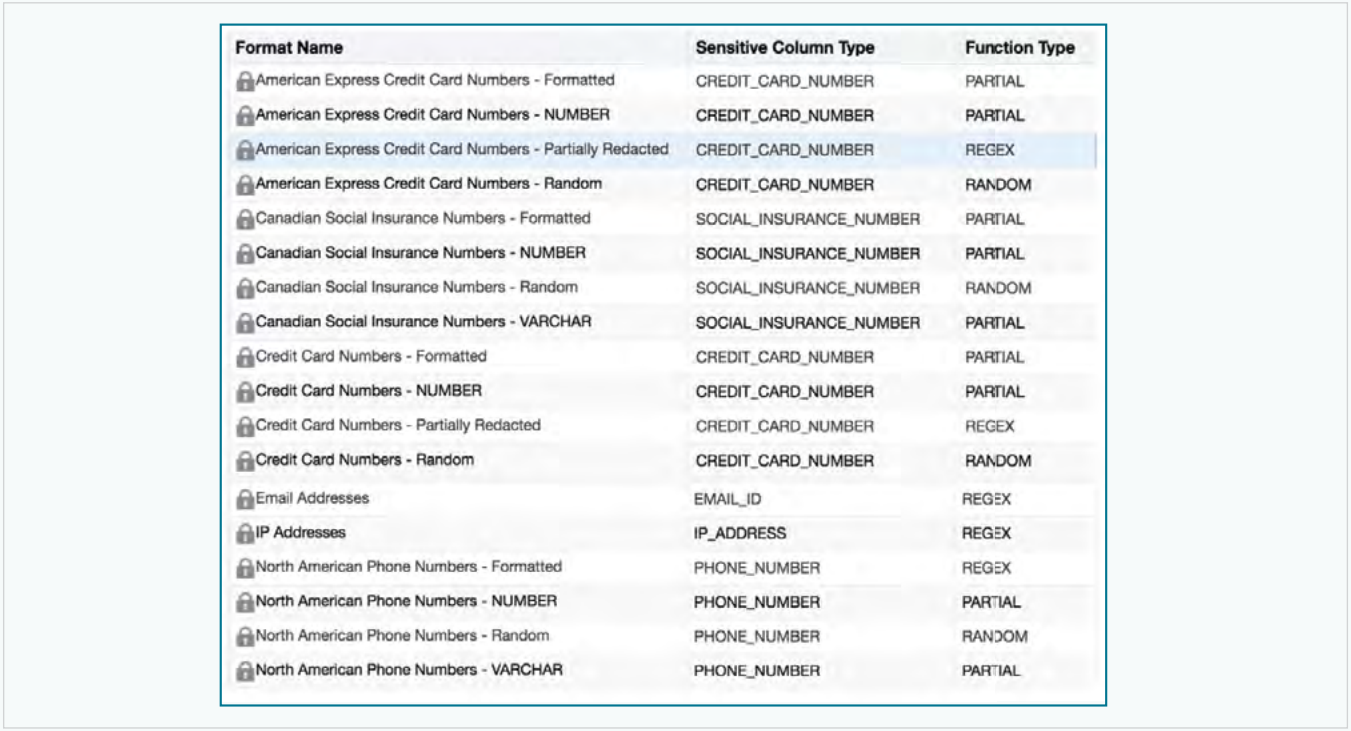


Figure 6.9: Redacted HR data

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. Figure 6.10 shows some predefined redaction formats provided by Oracle. Users can also create new redaction formats to meet specific requirements.



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 6.10: Predefined redaction formats

Oracle Data Redaction can be applied selectively based on policies that use the runtime contexts available from the database and 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 data before it is shared in non-production environments helps ensure that the sensitive information cannot be compromised even if those systems are hacked or those users are compromised. Redacting sensitive data, so that it can be displayed without risk and without altering the stored data, limits exposure of sensitive information to application users. These technologies enforce the principle of least privilege and play a key role in addressing anonymization and pseudonymization requirements associated with regulations such as PCI-DSS and the EU GDPR.

07 Database auditing and activity monitoring

Attackers that target your crown jewels in databases try multiple techniques. They try to bypass perimeter security, take advantage of trusted middle tiers, or even masquerade as privileged insiders. It is important to audit and monitor database activity to know what is happening so that appropriate actions can be taken.

Organizations today have hundreds or thousands of databases where user and administrator activities need to be audited and monitored for compliance with regulations and detect security breaches. This oversight requires continuous collection and analysis of massive amounts of activity data to run reports and generate alerts on anomalous activities.

Database auditing and network activity monitoring is about collecting native database audit and network-based SQL traffic data to monitor and report on database activity. Database auditing provides a record of database activities from users and applications, including those with high administrative privileges. Database firewalls monitor and evaluate incoming SQL traffic at the network level, identify and alert on anomalies or out-of-policy operations, and block out-of-policy SQL statements from reaching the database.

Oracle Audit Vault and Database Firewall (AVDF) incorporates both auditing and network monitoring technologies in a single product to provide a comprehensive view of database activity.

Use cases

There are three common use cases for Database auditing and network activity monitoring: implementing corporate security guidelines, ensuring regulatory compliance and conducting forensic analysis.

While corporate security guidelines vary, they require auditing privileged user activity, login events, sensitive data access, monitoring database network traffic, preventing SQL injection attempts, and other security related activities. This requires a solution that supports database auditing and network monitoring.

An organization's sensitive data could be subject to regulations such as GDPR, CCPA, PCI, HIPAA, IRS 1075, SOX, and UK DPA. These regulations require that access to sensitive or personal data be tracked, including access to data within the database. A solution that provides a rich set of pre-built compliance reports for implementing these regulations is needed.

Organizations need to support forensic analysis in the event of a breach or when they observe suspicious events. Forensic analysis requires the ability to not only collect and store vast amounts of audit and network event records, but also be able to search through it efficiently. In addition, it should support storing and retrieval of historical data, tracking user privilege changes and stored procedure changes to aid in analysis.

Database auditing? Network-based SQL traffic monitoring? Or both?

Database auditing involves creating and enabling database policies to track user actions, schema changes, login failures, etc. Each action generates an audit record that includes the specific database operation, the user who performed the operation, the database objects involved, and time of operation.

Database Firewall monitors and analyzes the SQL traffic on the network going to the database from an application or from a user/DBA connecting to the database. By monitoring and analyzing the SQL statements, the Database Firewall identifies and reports on the anomalous SQL statements generated from a SQL injection attack or as a result of out-of-policy violations and can block them. Additionally, corporate or regulatory policies may require enforcement of trusted path access to corporate databases ensuring that access requests are only coming from a list of trusted paths. These corporate or regulatory policies can be addressed using the Database Firewall.

Combining audit data collection with network-based activity monitoring is the best way to gain a complete picture of database activity. A solution focused solely on network monitoring wouldn't understand database synonyms or stored-procedure activity. It is not recommended to capture every activity in a database via auditing due to additional overhead. The combination of database auditing and network-based monitoring solves those issues and supports both security and regulatory compliance goals.

Customers frequently start with one capability and later expand their architecture to include both.

Introducing Oracle Audit Vault and Database Firewall

Oracle Audit Vault and Database Firewall (AVDF) is a scalable and flexible database auditing and network activity monitoring solution that consolidates audit data from databases, operating systems, directories, file systems, and applications into a single repository for analysis, alerting, and reporting on the audit data. AVDF also monitors SQL statements sent to the database over the network and determines whether to allow, log, or block the unauthorized SQL statements.

AVDF includes extensive reporting capabilities with an easy to use filter-based reporting interface with quick drill-downs for relevant information. With AVDF, a single instance of Audit Vault Server can monitor activity across hundreds of databases, providing a single console from which one can report and analyze security events throughout the database estate.

AVDF supports multiple databases: Oracle Database, MySQL, Microsoft SQL Server, SAP Sybase, IBM Db2 LUW, and PostgreSQL. Other databases and application audit data in tables are supported using the custom collector framework, which collects data via JDBC or REST API. Custom collection is also possible from systems that write audit data to XML or JSON files. A Java based software development kit (SDK) is included to accommodate those targets that cannot be accessed using any of the custom collector framework options.

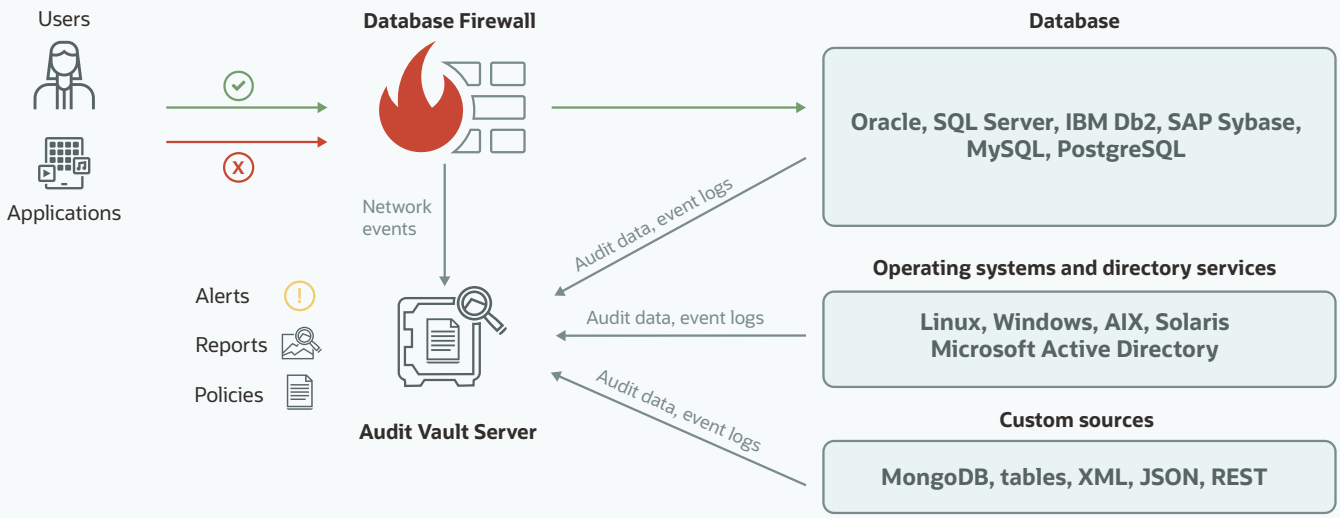


Figure 7.1: Oracle Audit Vault and Database Firewall

AVDF comprises of three key components: Audit Vault Server, Audit Vault Agent and Database Firewall.

Audit Vault Server

A central repository of audit records and network events captured by the audit collection agents and Database Firewall. Audit Vault Server performs four primary functions:

- Consolidation of audit data: Audit Vault Server captures audit and event logs from multiple sources, including audit data from application tables or files. Audit Vault Server maps the audit data and event logs to a normalized format, and includes them in a single report across all sources. For Oracle databases, it captures before/after values, changes to user entitlements, and stored procedure changes. Recommended policies for Oracle Database are provided in AVDF and can be enabled in the database with a single click.
- Out-of-the-box pre-defined reports: Audit Vault Server provides dozens of out-of-the-box reports relating to audit and network activity, privileged user activity, schema changes, entitlement changes, stored procedure modifications, before/after data value changes, login failures, and compliance reports for 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), and IRS Publication 1075. AVDF repository schema is documented and enables use of third party tools for specific reporting purposes.
- Configuring alert policies: Using the alert policy builder customers can specify rules for which they want to generate alerts, such as multiple failed login attempts, sensitive table access by unauthorized users, and data export operations.
- High Availability and archiving: Audit Vault Server supports a high-availability primary/standby architecture to ensure that audit record and network event collection does not stop in case of the failure of the primary node. Further, the collected audit data can be archived to lower cost storage and meet corporate compliance requirements for data retention.

Audit Vault Agent

Audit Vault Agent retrieves audit data from audit trails (sources of audit data) for various targets, such as databases and operating systems and sends the audit data to the Audit Vault Server. Periodically the Audit Vault Agent connects to the target to see if new audit records have been generated (since the last retrieval) and sends those records to the Audit Vault Server. Based on the type of audit trail (database table based audit trail, directory trail, transaction log trail, etc.) the Audit Vault Agents are deployed on a separate machine or on the same machine where the audit trail resides. A single Audit Vault Agent can support audit collection from multiple targets that are of different types.

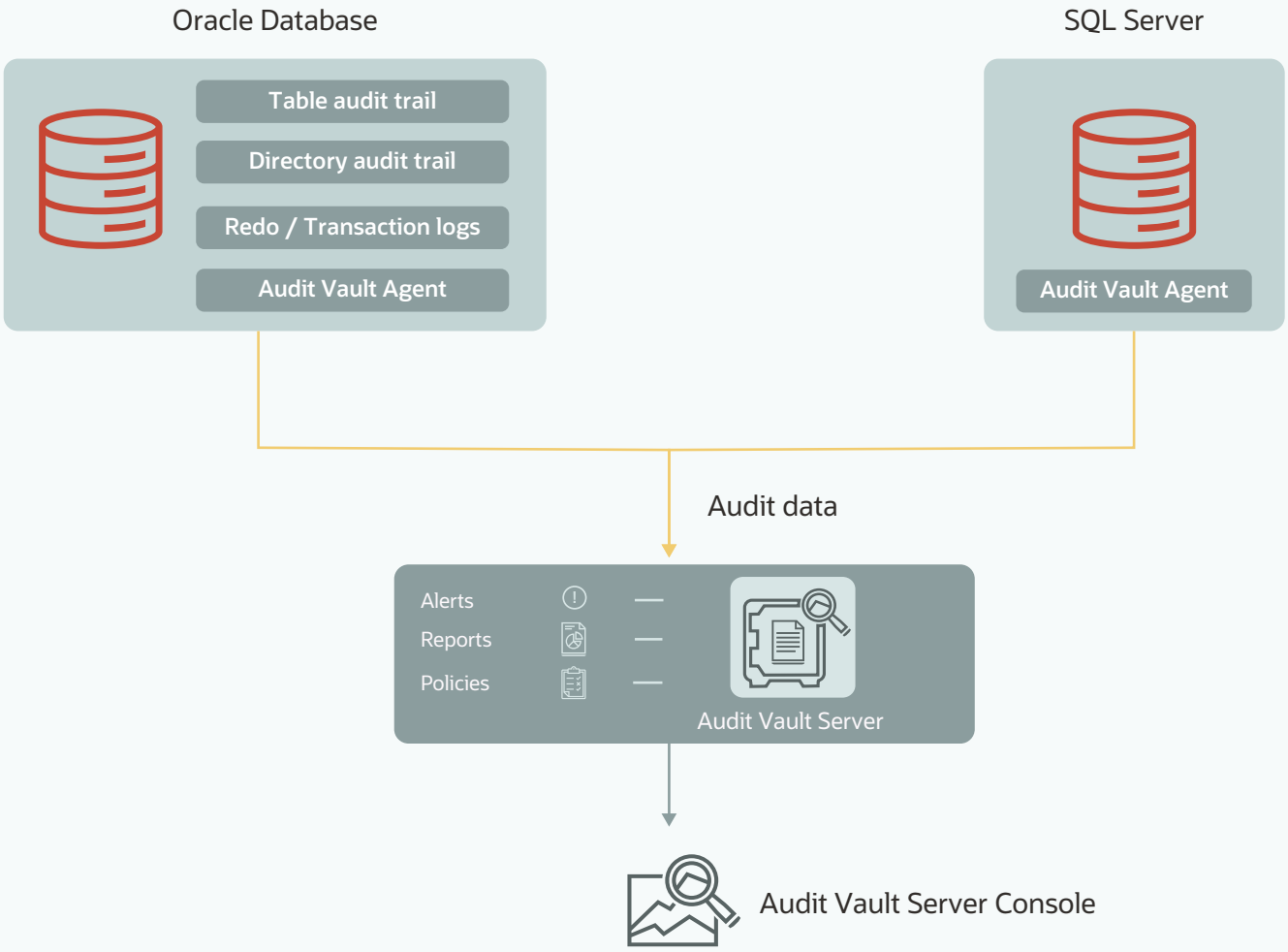


Figure 7.2: Database audit collection using AVDF

Database Firewall

The Database Firewall inspects the SQL traffic on the network going into the database and determines with high precision whether to allow, log, alert, substitute, or block the SQL statement before it reaches the database. Database Firewall events are stored in the Audit Vault Server and are consolidated with the audit data, giving customers a unified view into all activities. The Database Firewall is covered in detail in the next chapter.

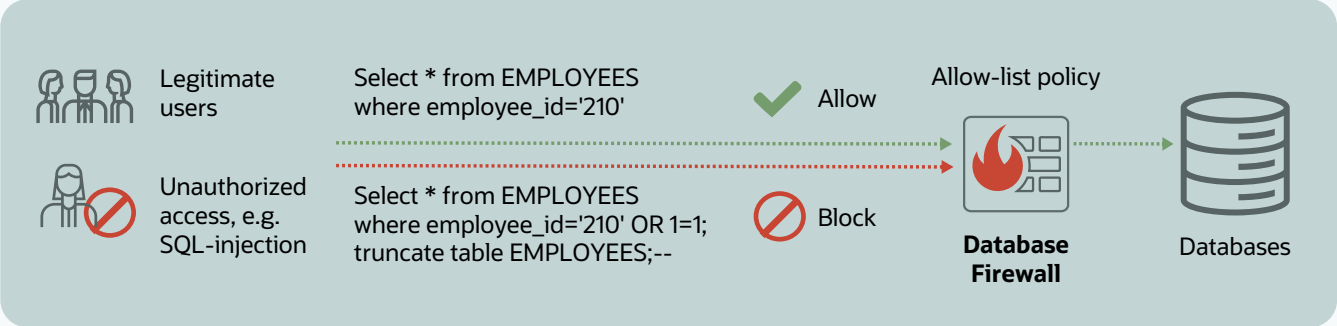


Figure 7.3: Monitoring network traffic with Database Firewall

Reports, alerts and notifications

AVDF reports are designed to provide detailed information on the audit activities and network events. The reports cover a wide-range of activities, including, all audit and network activity, privileged user activity, schema changes, entitlement changes, stored procedure modifications, before/after data value changes, sensitive data access and login failures. Table 7.1 below shows a summary of the different reports in AVDF followed by some sample reports.

Report type	Description
Activity	A set of reports that track database access activities such as audited SQL statements, application access activities, and user login activities. Some typical reports are: • All audit and network activity • Data Modifications • Before and after values of the modified data • DDL activity • Failed Logins
Alerts	Alert reports display the raised alerts and their status.

Report type	Description
Stored Procedure Audit	A set of reports that help you track the changes made to the stored procedures, such as creation, modification, and deletion.
Oracle Database Firewall	For database Targets that you are monitoring with the Database Firewall, this set of reports give detailed event information about the SQL traffic. For example, you can see details of statements that had warnings, or were blocked, according to the policy. You can also see information about the SQL statements sent to these databases.
User Entitlements	A set of reports that capture user access and privileges for Oracle Database targets. For example, reports provide information pertaining to user accounts, user privileges, object privileges and privileged users. Using AVDF, a baseline report can be created and changes from the baseline can be analyzed, making it easier for security and DBA personnel to monitor changes.
User Correlation	For Oracle Database targets running on Linux, these reports let you correlate events on the database with the original Linux OS user. This is useful in cases where this user executes a command on the database as another user by using su or sudo.
Database Vault Activity	If the Oracle Database targets have Database Vault enabled, the Database Vault Activity report shows Database Vault events, which captures policy or rule violations and unauthorized access attempts.
Anomalous Activity	Reports pertaining to users that are new users or dormant users accessing the system, or users accessing the system from IP addresses not seen before.

Table 7.1: Built-in reports in AVDF

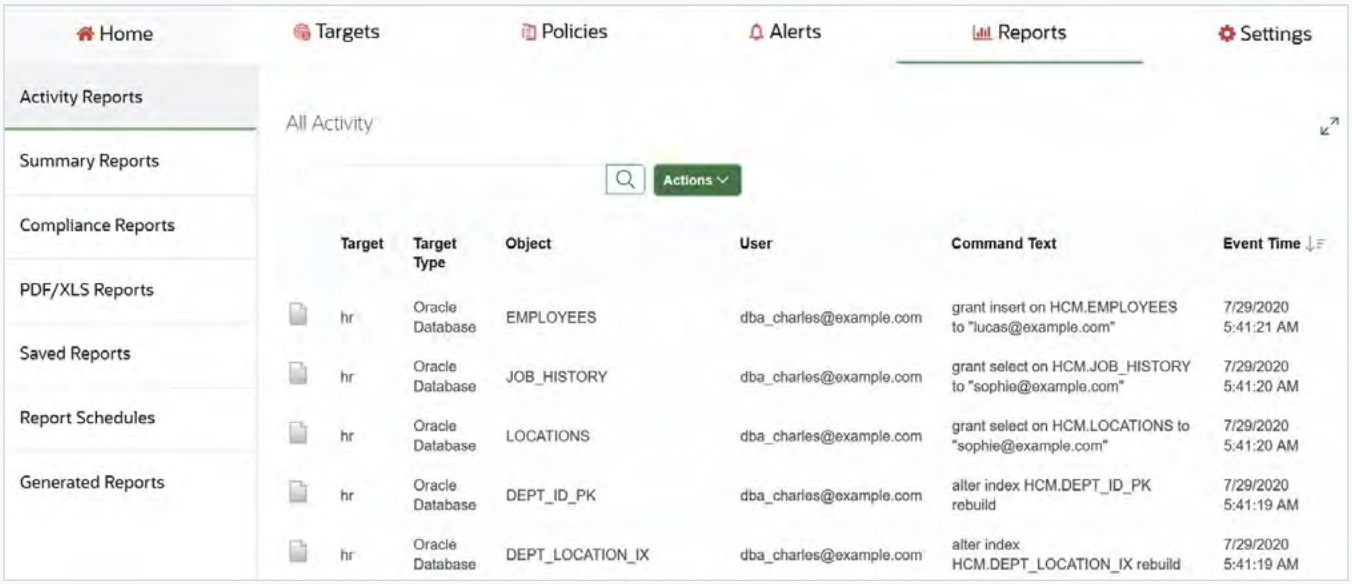


Figure 7.4: All activity report

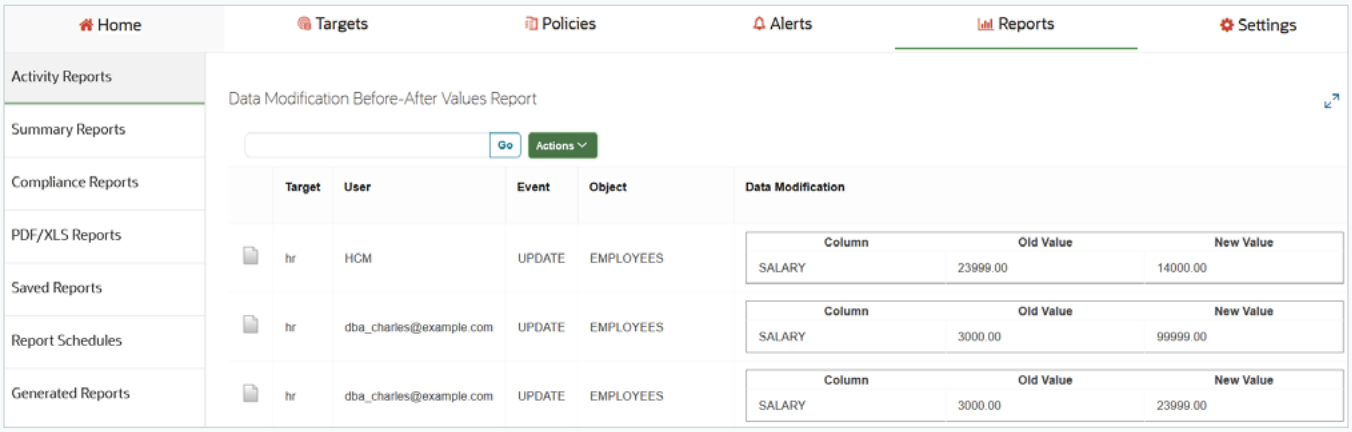


Figure 7.5: Data modification before/after values report

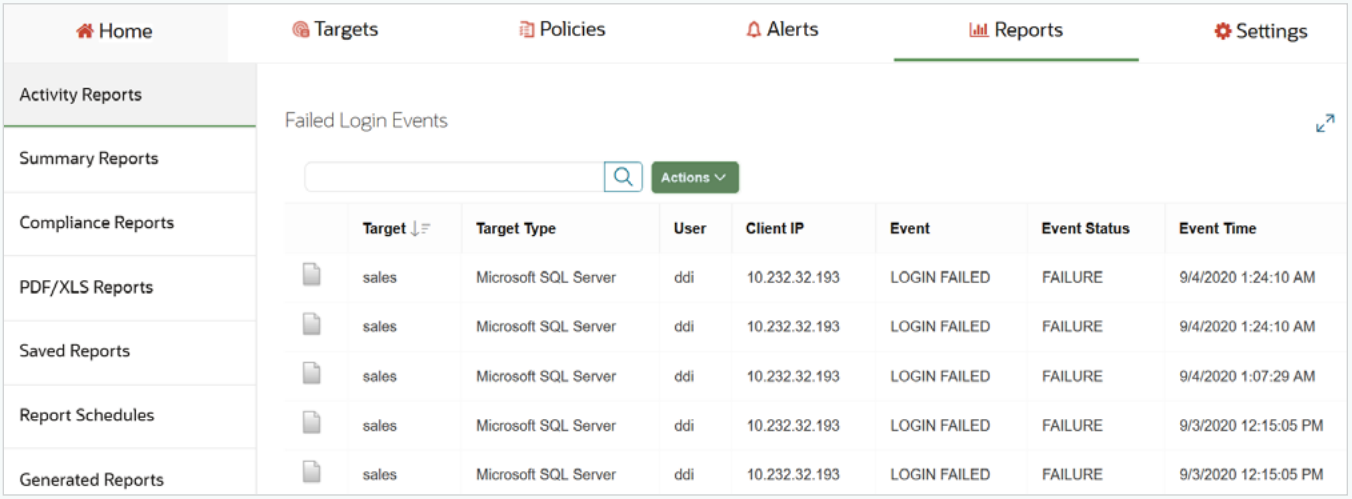


Figure 7.6: Failed login report

Auditors access reports interactively through a web interface, PDF, or XLS files. Report columns can be sorted, filtered, re-ordered, added, or removed. PDF and XLS reports can be scheduled to be generated automatically. Reports can also be defined to require signoff by multiple auditors. Users can create new or customize existing PDF and XLS report templates to meet customization requirements.

AVDF schema is open and documented, allowing users to retrieve the audit data from AVDF and use their reporting tool of choice.

Depending on the alert policies configured, AVDF can raise alerts based on the audit and network events that come from the Audit Vault Agent and Database Firewall. Alerts can be raised on multiple conditions. For example, there may be multiple failed login attempts occurring across multiple databases in a certain amount of time, indicating a possible brute force attack. Alerts can not only be displayed on the Audit Vault Server dashboard, but also can be sent to the user as an email, or to syslog server.

Additional support for Oracle databases

AVDF provides additional support for Oracle databases. This includes provisioning audit policies, monitoring entitlement changes, before/after data value changes and tracking stored procedure changes.

Audit policies for Oracle databases

Deciding what to audit depends upon an organization's policies. It is common to audit sensitive data access, privileged user activity, and security-relevant events. Database threats may also come from internal users. Oftentimes, certain database users are granted powerful system privileges. Their actions warrant greater scrutiny and constant monitoring. Audit policies need to be selective enough to reduce unnecessary audit records and effective enough to meet an organization's audit goals.

AVDF makes it easy to provision audit policies by providing recommended audit policies for Oracle, which the user can enable through single click. In addition to the AVDF recommended pre-defined policies, compliance related policies such as CIS and STIG are also provided.

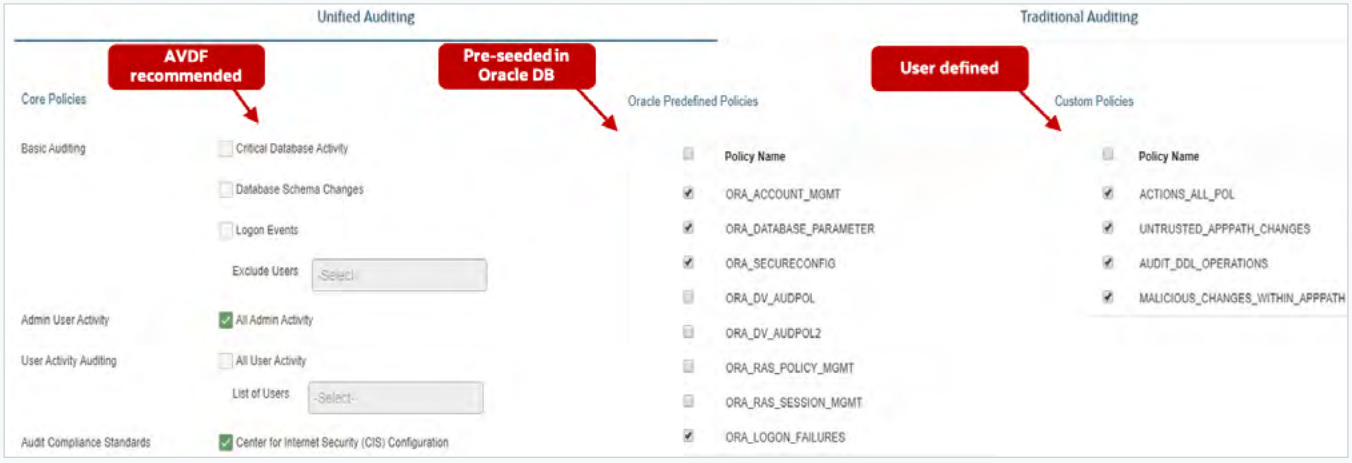


Figure 7.7: Easy audit policy provisioning

When the pre-seeded audit policies are enabled, these policies are created in the Target database without the user having to write SQL code for defining these policies.

Monitoring entitlements for Oracle Database

Tracking changes in entitlements is important for many reasons such as identifying users with improper access to certain tables, and rectifying improper grants of roles or privileges.

For Oracle databases, AVDF can be used to create a baseline of entitlements and to review any further changes.

Before/after data value changes

Before and after value collection is extensively used in the healthcare and financial services industry, as well as many other regulated industries. With before and after value collection, auditors can track the lifecycle of individual data attribute changes—an important component of many data governance requirements.

If a data value is changed, AVDF records the old value (before the change) and the new value (after the change), along with who changed it and when it was changed.

AVDF 20 includes a restricted license of Oracle GoldenGate and uses Oracle GoldenGate Integrated Extract process for before and after value collection.

Monitoring Oracle stored procedures

Stored procedures contain some of the important business and/or data access logic. Attackers sometimes modify these stored procedures to create backdoors to the database. Therefore, it is important to constantly monitor these stored procedures to identify any changes.

AVDF can periodically check the Oracle databases for any new stored procedures or any updates to stored procedures. The reports can be reviewed for any changes.

Hybrid cloud support

Organizations need a single pane of glass view into all database activities, regardless of where the database resides. Deploying AVDF for both on-premises and cloud database targets provides several advantages including consistent policies, unified reporting, and common alert management. Existing alert configurations and data retention policies can be applied to cloud databases. By monitoring the databases in the cloud, organizations can have an independent view of the events on their databases and ensure that the cloud vendors cannot modify their data.

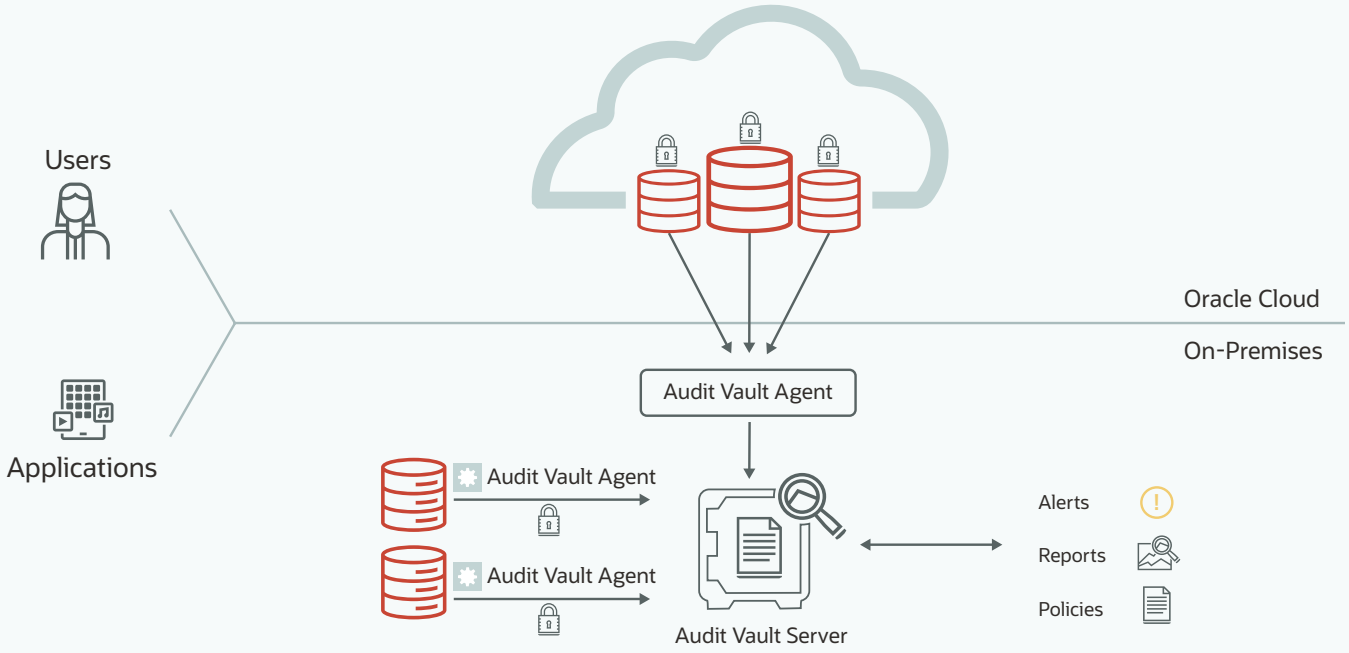


Figure 7.8: Hybrid cloud support

Summary

Oracle Audit Vault And Database Firewall is a complete database auditing and network activity monitoring solution that combines native audit data collection with network-based SQL traffic capture. It includes a highly scalable enterprise quality data warehouse, audit data collection agents, Database Firewall, powerful reporting and analysis tools and alert framework. Users can leverage the recommended policies to quickly enable database auditing with a single click.

The next chapter will discuss the Database Firewall and how you can use it to monitor and block SQL traffic before it is reaches the database.

08 Network-based SQL monitoring

Need for network-based SQL monitoring

The majority of data breaches are caused by misused insider credentials, compromised clients, or through SQL injection attacks coming through the web applications. SQL injection continues to be the number one application security risk on the Open Web Application Security Project Top 10 list for several years. There are many situations where we need to monitor, detect, and block such attacks before the SQL statement reaches the database. Network-based SQL traffic monitoring is ideal to deal with these threats.

This chapter describes common network attack vectors used to compromise the application data and how by using the Oracle Database Firewall, a component of Oracle Audit Vault and Database Firewall, organizations can monitor the SQL traffic to the database and prevent SQL injection. Database Firewall can also log the SQL traffic and raise alerts based on policies defined.

Need to monitor and block network attacks

Traditional perimeter-focused network firewalls used to be one of the ways to stop unwanted traffic to the databases. However, network firewalls are not adequate as they just allow or deny network communication to and from a certain port and IP address. They typically cannot evaluate the actual content of the packets and use full SQL context for access control. Moreover, the blurring network boundaries and the increasing insider threats have rendered traditional network firewalls inadequate.

Another approach to block network attacks is to add protection at the application level. This can be done by improving user authentication, enforcing least privilege, using prepared statements, avoiding dynamic queries, and performing input validation. Although these methods provide a stronger level of protection and should be practiced, minor gaps in development or configuration controls can easily introduce vulnerabilities. Organizations need a database security technology that can analyze the SQL and take context specific actions even if there are minor gaps in the application tier.

One of the biggest risk to databases is from attacks that come directly from the web. Hackers can use SQL injection techniques to insert a malicious SQL statement via the input data 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 on the database.

For example, if your application UI has a button called “Check Employee” along with an input text for providing employeeId as the parameter, the application tier may generate a query of the type “select * from `EMPLOYEES` where employee_id=‘`<InvalidatedInput>`’”. A normal web user may just type the employeeId ‘210’ but a malicious user can force the application to execute a different SQL query by injecting a doctored value such as:

```
210’ OR 1=1; truncate table EMPLOYEES;--
```

Instead of the simple SELECT query, we now see how the malicious input data can change the query parameters, resulting in the database returning all the data and truncating the table. This poses a massive risk to the organization because the malicious SQL statement is executed with all the privileges of the application account with full access.

Such SQL injection attacks are difficult to catch at the database level because the SQL looks like any other SQL that the user could be executing. Organizations need a simple but effective way to detect such attacks, raise alerts when needed, and block such attacks from reaching the database. Such monitoring should typically be used in conjunction with database auditing that captures local database activity and complete execution context.

Oracle Database Firewall

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 database access behavior, while helping prevent SQL injection, application bypass, and other malicious activities from reaching the database. A single Database Firewall can be used to protect multiple databases of different type from a central location. Oracle Database Firewall monitors enterprise databases including Oracle, MySQL, Microsoft SQL Server, SAP Sybase and IBM Db2.

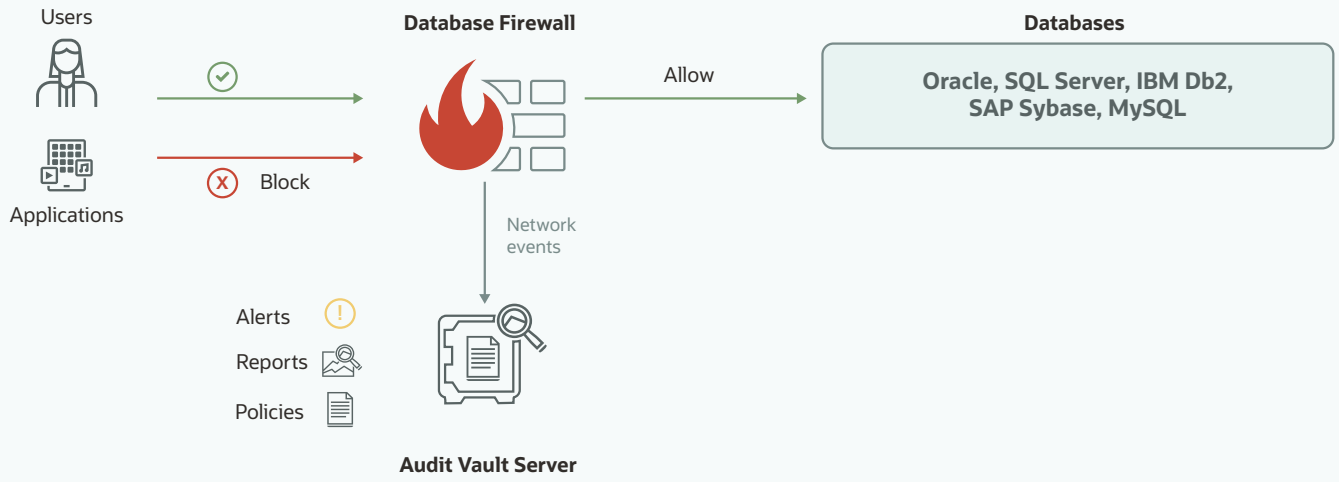


Figure 8.1: Oracle Audit Vault and Database Firewall

Oracle Database Firewall is a multi-stage analysis engine that inspects SQL traffic to the database and determines with high precision whether to allow, log, alert, substitute, or block the SQL as specified in the policy. The SQL traffic goes through different stages of analysis in the Database Firewall, including checks for originating IP address, database/OS user name, client program name, SQL statement category (DDL, DML, etc.), database tables being accessed and the actual SQL statements. All of this information can be used to determine whether the SQL statement should be logged, alerted, allowed to go through, or blocked.

One simple way to look for malicious SQL could be through regular expressions, but those are very easy to bypass. They can also lead to high rates of both false positives and false negatives, creating unnecessary alerts or letting bad traffic go in. Oracle Database Firewall uses a SQL grammar-based engine to parse the SQL and recognize the equivalent allowed statements. 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 and both these statements would belong to the same cluster. Understanding the similarity between different statements, filters millions of SQL statements down to just a few dozens. Using these SQL clusters, firewall policies can be defined that include both allowed SQL (allow-list) and disallowed SQL (deny-list). The SQL statements that do not belong to the allow-list or the ones that belong to the deny-list could indicate a possible attack.

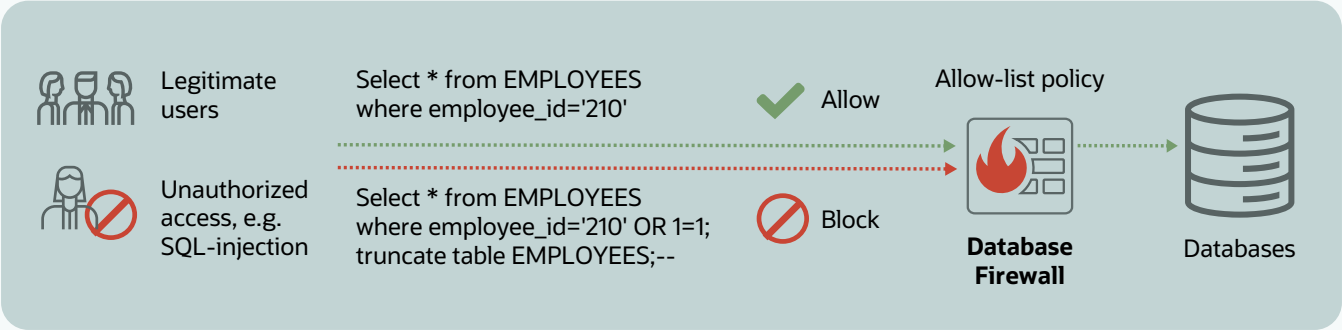


Figure 8.2: Monitoring SQL network traffic with Oracle Database Firewall

Oracle Database Firewall captures not only the SQL statement category and database tables being accessed, but also session context information such as database user name, OS user name, client IP address, client program name, table name, etc. Database Firewall policies can be built using this information.

Oracle Database Firewall policies

Policies would be driven by objectives such as the need to ensure trusted application path access to the database, creating an allow-list to pass specific types of SQL and blocking or alerting on everything else. At each stage of the Database Firewall policy engine, conditions defined in the policy are evaluated, and if there is a match, user specified actions such as alerting, logging, blocking or allowing the SQL statement to pass through to the database for execution, can be taken. If there is no match, the SQL statement is passed to the next evaluation stage.

At the 1st stage, rules relating to connection session context are evaluated, followed by SQL Statement rules and then the database object rules. Finally, the default rule is applied if there is no match found in the earlier stages.

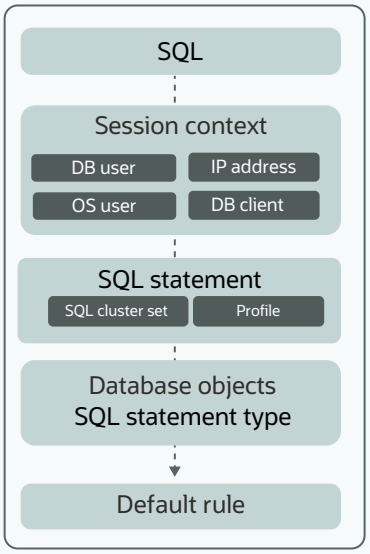


Figure 8.3: Multi-stage Database Firewall

Session context rule

Session context rules do not look at the specific SQL statement; instead it uses database session attributes such as IP address, database user name, OS user name, and database client program name to make a decision.

Session context rules allow SQL traffic from trusted application paths without requiring them to go through further processing. For example, you may only want to allow SQL requests from a trusted set of allow-list client IP address range, or block SQL requests originating outside of an expected IP address range.

Once matched, actions can be taken on the SQL traffic, such as blocking, alerting, or logging. SQL statements originating from sessions that do not match the rule are sent to the second stage of the Database Firewall for processing.

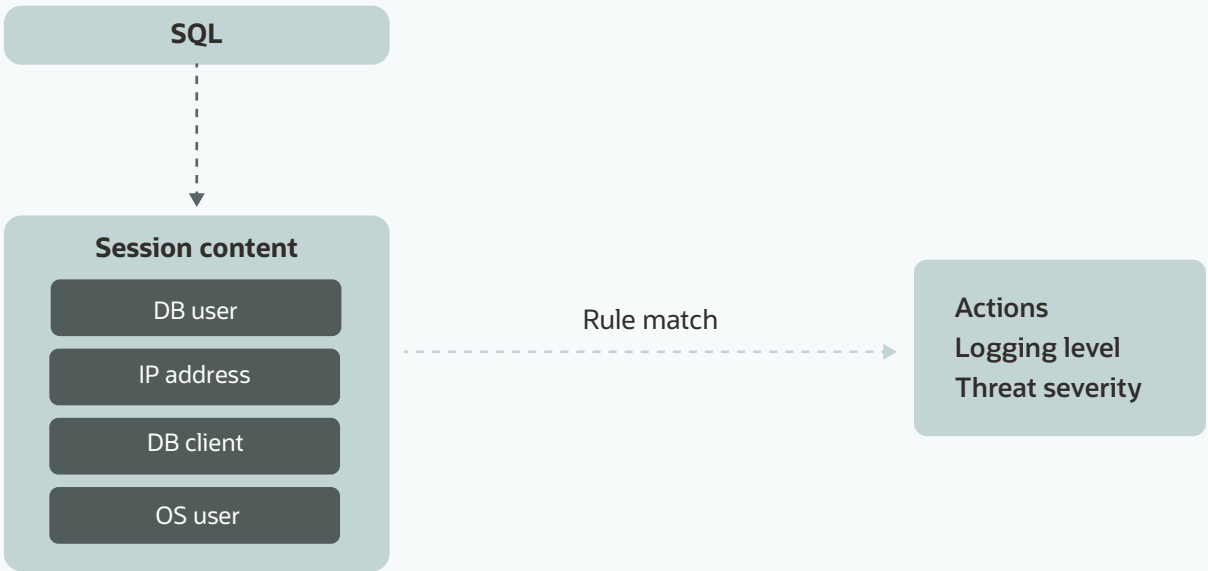


Figure 8.4: Session context rules

SQL statement rule

The second stage uses a SQL grammar based engine to parse the SQL statement, map it to a cluster, and take actions if the cluster set and profile match. Note: A cluster set is a user-defined group of clusters (a cluster is a group of SQL statements identified as being similar by the Database Firewall) and profile is a user-defined combination of IP address, database user names, OS user names and database client names.

If the incoming SQL statement matches with any of the SQL clusters and profile defined in the SQL statement rule, then actions as specified in the 'SQL Statement rule' are taken, if not, the SQL is sent to the next stage for processing. Using this rule, allow-list or deny-list based policies can be created. The allow-list of SQL clusters can be automatically created by sending known/expected SQL statements generated in a test or QA system, through the firewall. Database Firewall clusters these SQL statements into similar sets of SQL. For example, you could create an allow-list of SQL requests from a trusted set of application users. You could also allow database access for known set of SQL requests from trusted privileged users using a tool such as Oracle SQL Developer. SQL traffic that does not match is sent to the third stage, the Database Object rule.

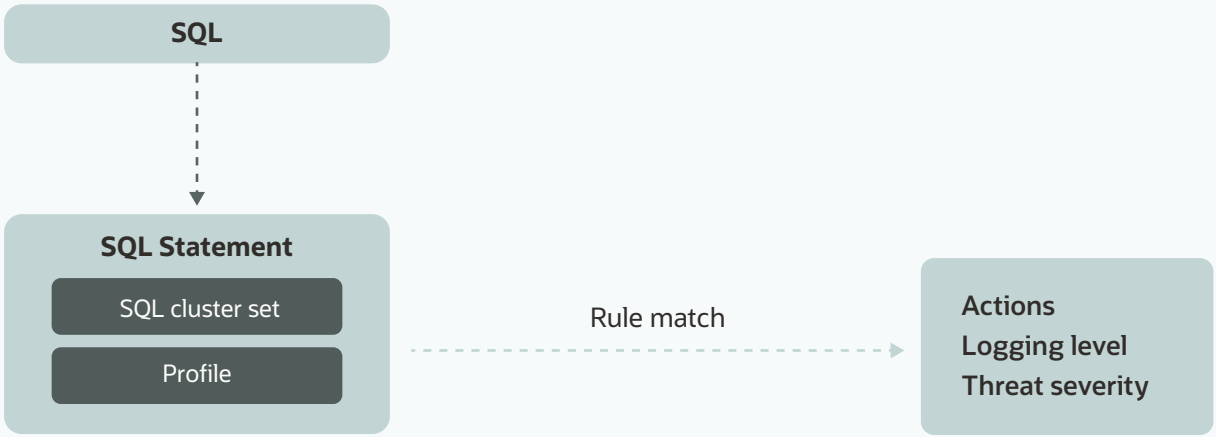


Figure 8.5: SQL statement rules

Database object rules

Database Object rules are used to block or allow specific types of SQL statements (DML, DDL, etc.) on specific database objects such as tables and views. These rules are often used for controlling access to sensitive application database objects. For example, you may want to allow only SELECTs on application tables, but block the SQL modifying sensitive tables.

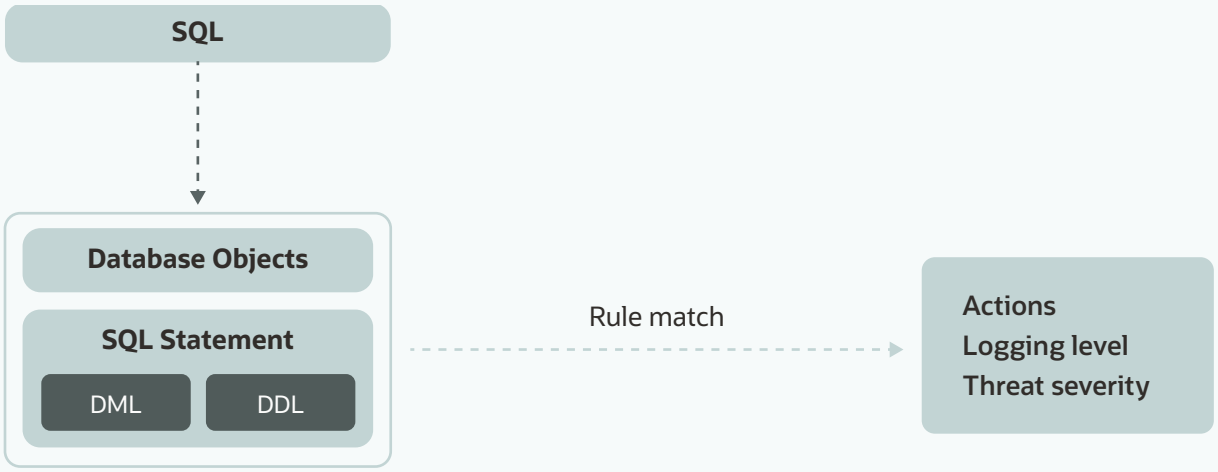


Figure 8.6: Database object rules

Default rule

The default rule is executed if the SQL statement does not match any of the other rules (Session context, SQL statement, Database object). It is useful to log these statements so you can analyze these statements and see why they did not match the Session content, SQL statement or Database object rules and if needed, modify the rules accordingly. In situations when the SQL statements change, or there are new users in the system or new applications, having a default rule ensures that you can capture any new traffic for which you have not yet created a firewall rule and decide how to process it.

Oracle Database Firewall deployment modes

The Database Firewall can be deployed in various modes, as shown in the table below.

Mode	Details	Supported functionality	
		Monitoring	Blocking
Proxy	All client connections go via firewall, including return traffic	✓	✓
Host Monitor	Agent runs on database host listening to incoming traffic and securely forwards the traffic to the Database Firewall	✓	
Out-of-band	Monitors copy of the database traffic sent to it	✓	

Table 8.1: Deployment modes

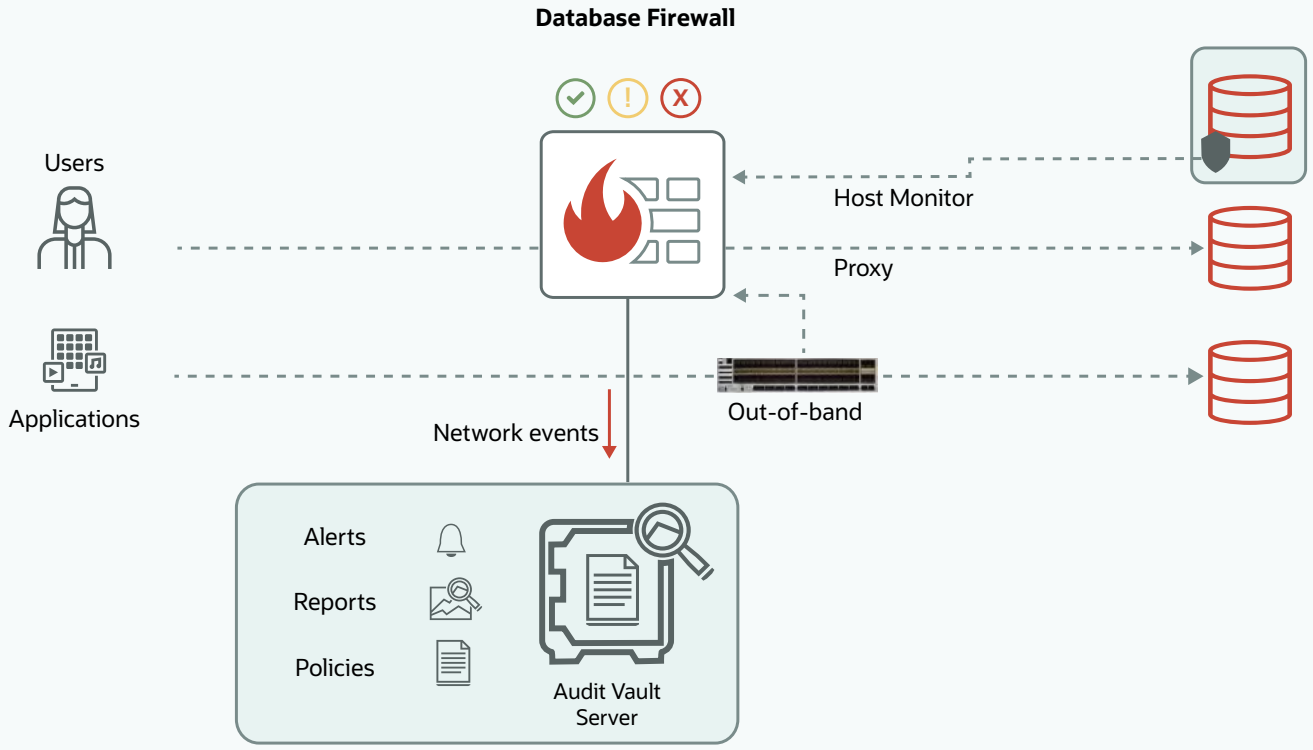


Figure 8.7: Oracle Database Firewall deployment architecture

The Database Firewall can be deployed in the following modes:

- **Proxy mode:** In this mode, all the traffic to database server is routed through the Database Firewall. Database clients need to be reconfigured so that they connect to the Database Firewall which processes the traffic and based on the policy, forwards it to the database, blocks it or raises alerts on it.
- **Monitoring in out-of-band mode:** In this mode, Database Firewall listens to the network traffic sent to the database. There are several technologies such as span ports, port replicators, etc. that can be used to send a copy of the database traffic to the Database Firewall. Database Firewall can monitor and alert on the SQL traffic, but it cannot block it.
- **Monitoring with Host Monitor:** Host Monitor is deployed on the same machine as the database and captures SQL traffic going to the database. The Host Monitor captures the traffic sent to the database and then securely forwards it to the Database Firewall. It is also easier to deploy the Host Monitor when there are several network paths to the database. It can monitor and alert, but not block the SQL statement.

Oracle Database Firewall reports

The Database Firewall sends network events and alerts to the Audit Vault Server, and this information can be viewed in the Database Firewall reports. For example, the Database Firewall reports display details of statements that raised alerts or were blocked. They also provide information on the SQL traffic to the databases such as SQL statement type (DDL, DML, etc.), database username, OS username, client application name, client IP address, Database Firewall action taken, and the SQL statement executed.

In addition, the Database Firewall policies can be configured to generate alerts and these alerts can be viewed in the AVDF reports, sent to an email recipient, or sent to the syslog server.

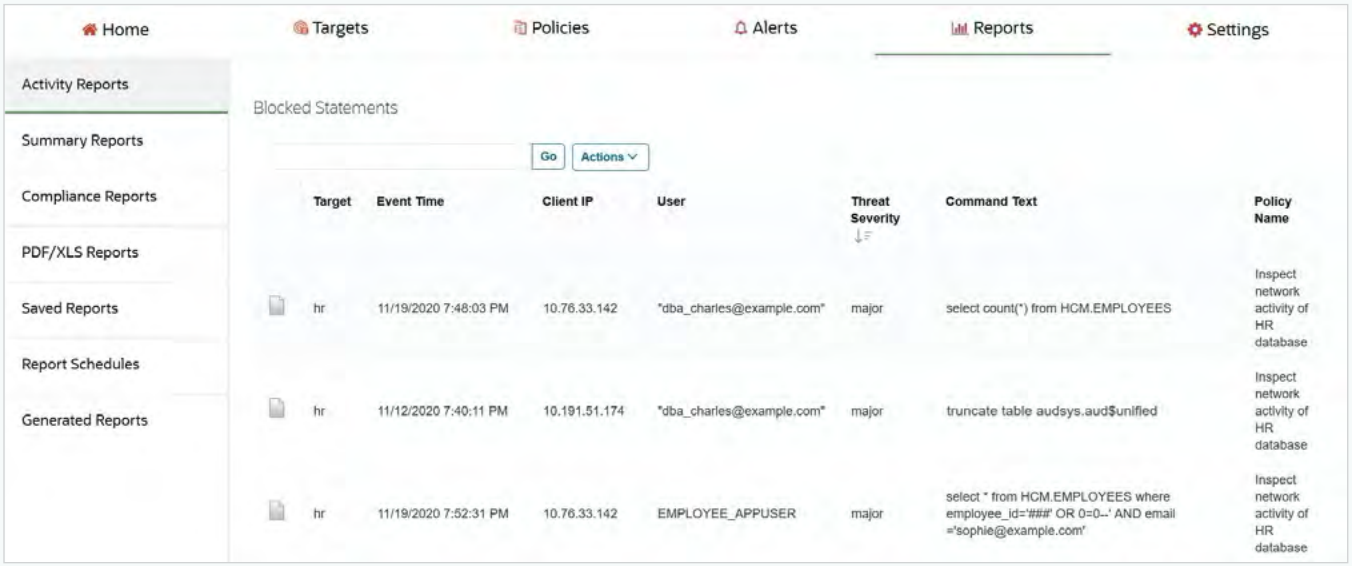


Figure 8.8: Database Firewall blocked statements

Summary

The increasing number of attacks on databases via SQL injection or misused insider credentials have made network-based SQL traffic monitoring an important component of an organization's overall security architecture. Oracle Database Firewall provides a sophisticated next-generation SQL grammar analysis engine that inspects the SQL statements on the network going to the database and determines with high accuracy whether to allow, log, alert, substitute, or block the SQL. Network-based SQL traffic monitoring introduces near-zero latency and has zero overhead on the monitored database. It requires no modification to the databases, and monitors multiple heterogeneous database systems.

Oracle Database Firewall supports allow-list and deny-list based policies to enforce trusted application path access to data. Policies can be applied on client program name, database user name, SQL types, table names, OS user name, and IP address.

The events captured by the Database Firewall events are sent to the Audit Vault Server, where users can view reports containing database audit events and network events to get a complete picture of the security and compliance status of their environment.

09 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 graphic 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		9000	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.

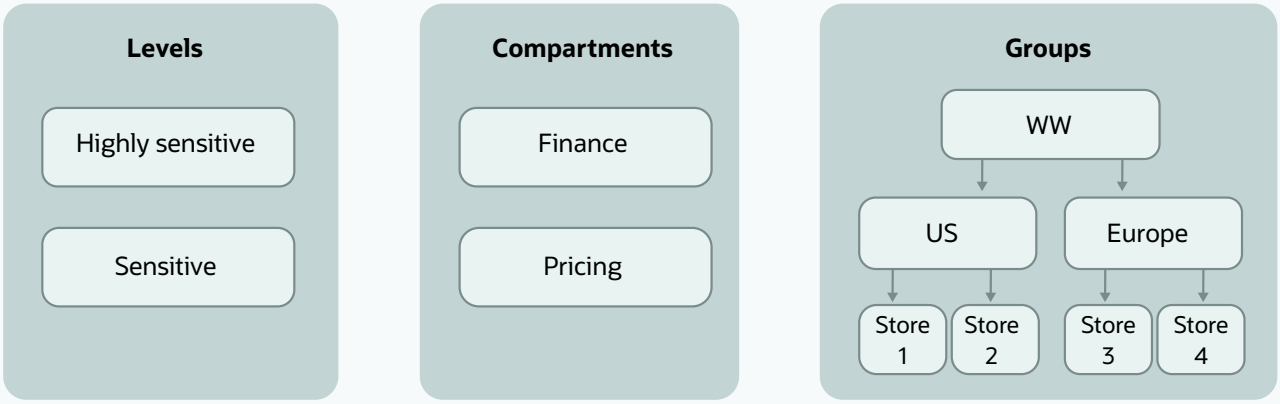


Figure 9.3: Oracle Label Security components: levels, compartments, groups

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, they will have the Pricing compartment as part of their user label. We will review how labels work by starting the example with the group component, then adding in compartment and level.

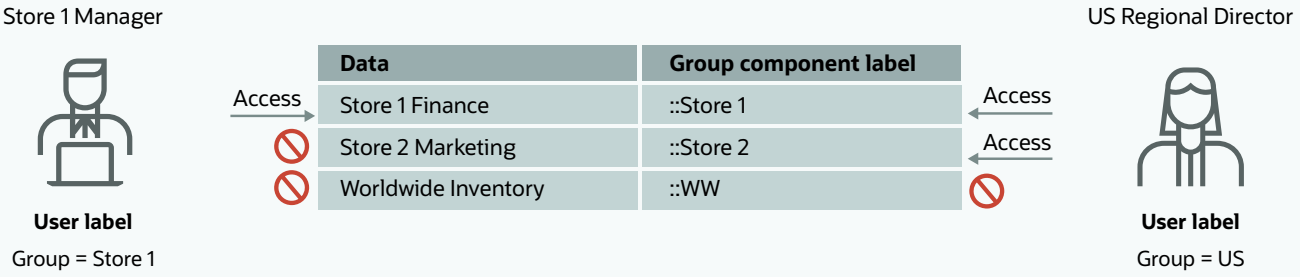


Figure 9.4: Using the group component

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.

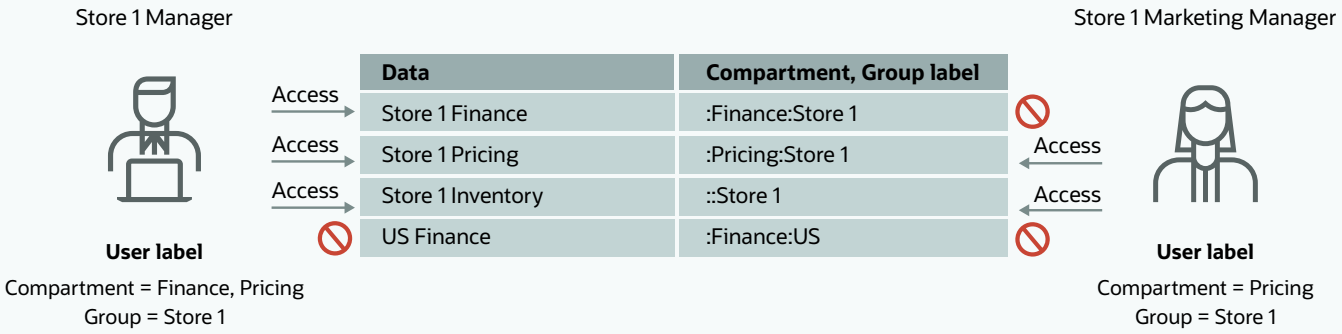


Figure 9.5: Using the compartment and group components

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.

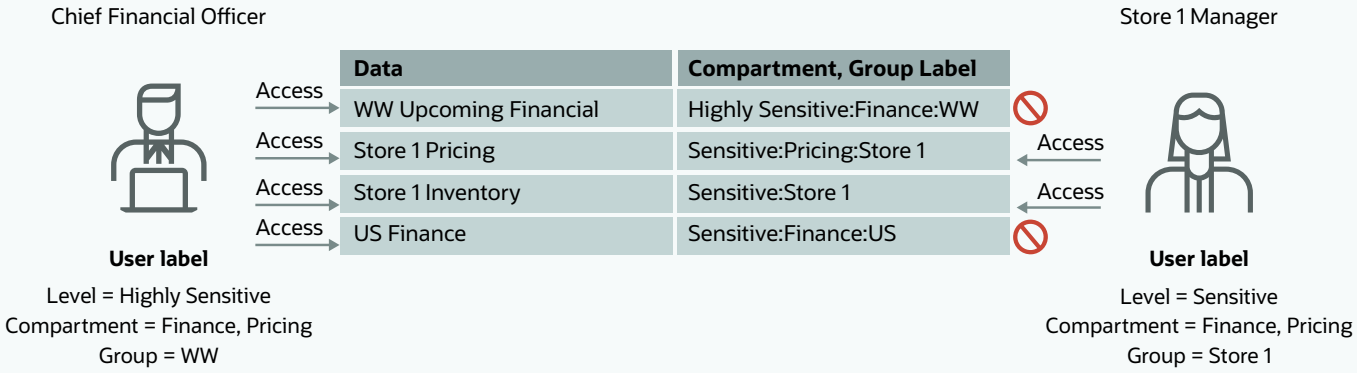


Figure 9.6: Using the level, compartment, and group components

While OLS 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. 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.

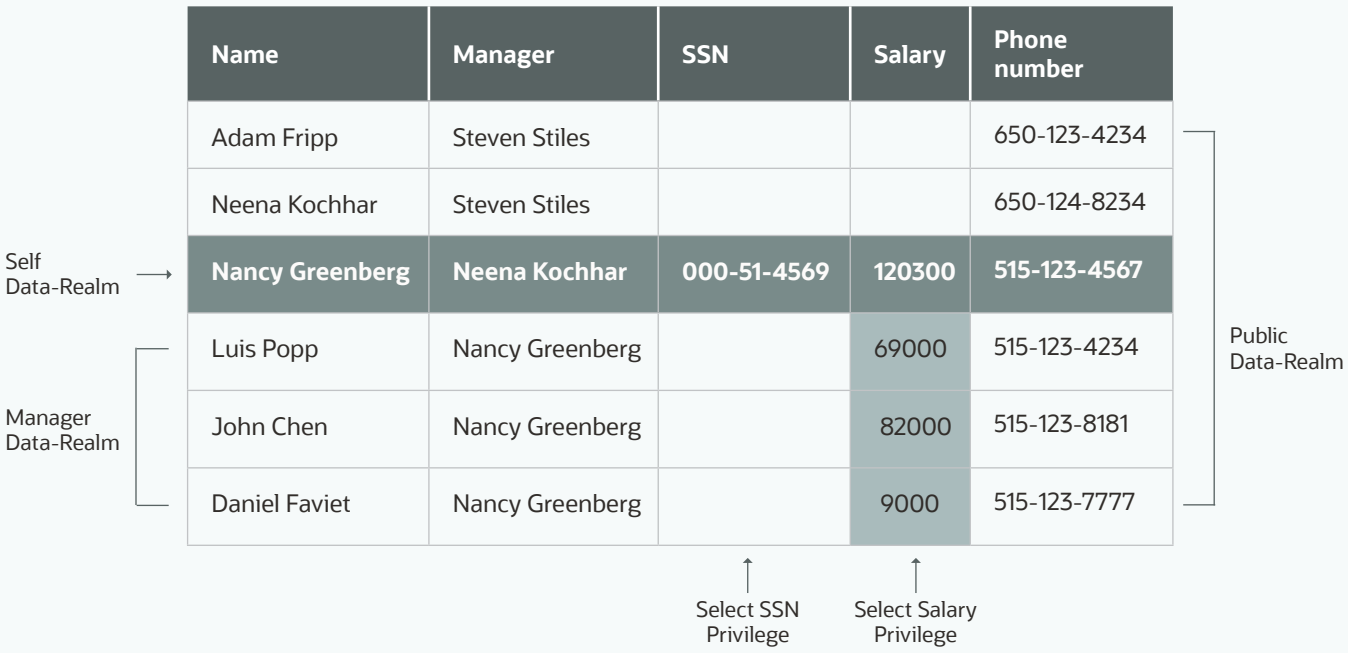


Figure 9.7: Controlling data access with RAS

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 the graphic, 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 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 periodically scan the database to ensure that it is securely configured, and to remediate the situation if there is a deviation.

Many regulations, such as EU GDPR, PCI DSS, Sarbanes-Oxley and various breach notification laws, promote security assessments as a key part of reducing IT risks. Various organizations such as the Center for Internet Security (CIS) and U.S. Department of Defense have recommendations for security configuration best practices. It is critical to have a solution that can perform an assessment, taking into consideration recommendations from different regulations and security frameworks, along with vendor best practices.

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 an attack and usually 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 assess 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?
- Which packaged applications are running on this database? Are those running with all-powerful privileges?
- Is auditing on? For whom? Which conditions?
- 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 steal the data. Organizations, as the owners of the data, need to do similar thinking to harden the database's 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 the database. Because the Oracle Database is highly customizable, assessing security also needs an understanding of the impact of configuration choices on the overall security.

Here are some key considerations for protecting your databases:

- Almost all databases hold sensitive data, but the level of importance may differ. For example, the 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.
- Common database vulnerabilities include unpatched systems, poor application design, weak user credentials, excessive privilege grants, lack of a trusted path to data, no separation of duties, no encryption, and inadequate auditing.
- Security configuration parameters are tightly related to how the database behaves and requires 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, several other actors/processes need to interact with your data through a database user account—the application itself, application administrators, security administrators, and others like service accounts, batch programs, etc. Clearly identifying the different types of database 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. Database design rarely takes 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 using default settings.

We now describe different software tools and services that can help you assess the security of your databases. Such tools include Database Security Assessment Tool, Oracle Database Life Cycle Management Pack, and Oracle Data Safe cloud service.

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 provides 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 all over the world by Oracle consultants and partners while executing Database security assessments.

DBSAT security assessment reports

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

The DBSAT Collector first gathers security configuration information from the database and underlying OS. The DBSAT Reporter then analyzes the collected data and generates detailed findings and recommendations. The output reports are in HTML, spreadsheet, text, and JSON formats. The DBSAT Discoverer (described 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 usages. The JSON output is convenient for data aggregation and integration purposes.

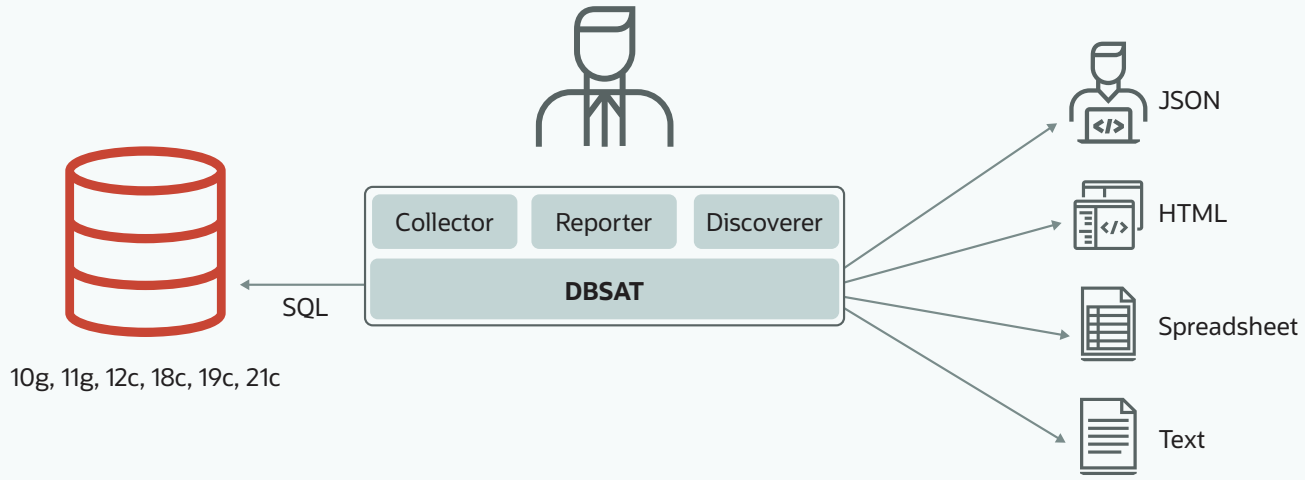


Figure 10.1: Database Security Assessment Tool

Assessment Date & Time

Date of Data Collection	Date of Report	Reporter Version
Wed Feb 03 2021 23:04:00	Wed Feb 03 2021 23:05:42	2.2.1 (May 2020) - f3a1

Database Identity

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

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	3	4	0	**12**
Privileges and Roles	4	17	1	0	0	0	**22**
Authorization Control	0	0	2	0	0	0	**2**
Fine-Grained Access Control	0	0	5	0	0	0	**5**
Auditing	0	10	2	0	0	0	**12**
Encryption	0	2	0	0	0	0	**2**
Database Configuration	7	3	0	2	1	1	**14**
Network Configuration	0	0	1	1	0	0	**2**
Operating System	1	2	0	1	1	0	**5**
Total	**17**	**34**	**11**	**7**	**6**	**2**	**77**

Figure 10.2: Database Security Assessment report

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 currently not in use. In cases where further analysis is needed, the status is shown as "Evaluate."
- Summary: Presents a 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 benchmark recommendation, Oracle Database STIG Rule, and the EU GDPR article/recital.

In the Finding below, DBSAT has 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.

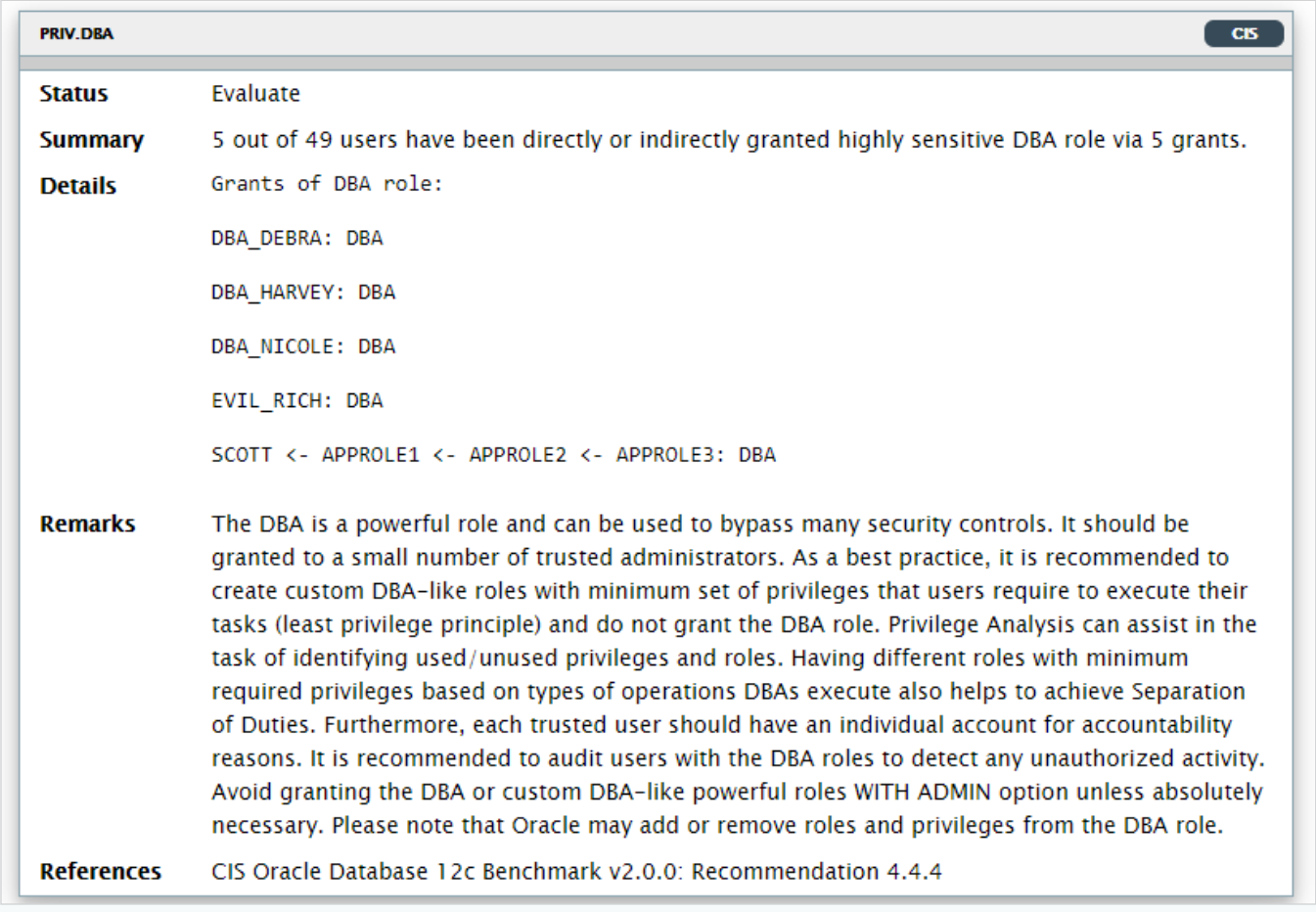


Figure 10.3: Sample finding: Users with DBA role

Discovering sensitive data with DBSAT

The Discoverer component of DBSAT helps identify tables and columns with sensitive data, along with the number of rows of sensitive data. To learn more about discovering sensitive data with DBSAT, please refer to "Discovering Sensitive Data", *Chapter 5*.

Database security assessment using Oracle Data Safe

Customers can use Oracle Data Safe cloud service to assess the security of their databases running on Cloud and on-premises. Data Safe is a database security cloud service that provides a comprehensive suite of security capabilities, including user and security assessments. Tightly integrated assessment capabilities provide the ability to simultaneously run assessments on multiple databases, schedule assessments, establish a security baseline, and get a comparison report that highlights the drift between that baseline and the current database security assessment.

See *Chapter 13* for more information on how Oracle Data Safe can help meet data security requirements 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.

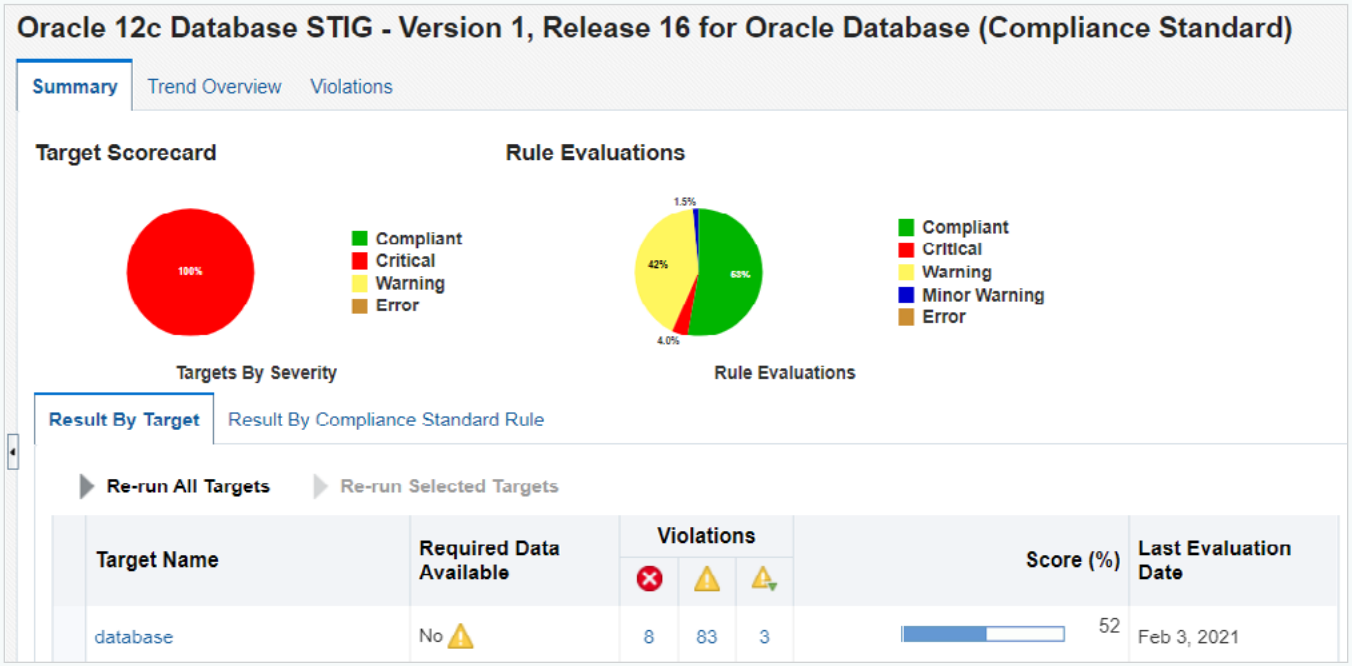


Figure 10.5: STIG compliance standard detail

Compliance frameworks, standards, and rules

DBLM provides an extensible framework that supports 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 table on next page 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 host 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

Every quarter Oracle routinely provides fixes in databases for functional, performance, or security issues discovered by internal testing, or reported by customers and external researchers. The security fixes can cover a wide range of issues including

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

When any vulnerability is found, fix involves evaluating the full data flow, constraints, and the flexibility inside the database so that the problem is fully fixed without breaking the applications. Some customers try temporary measures such as blocking a vulnerable SQL statement, but such measures are very easy to bypass, and cover a very small percentage of the documented fixes.

The timely application of patches from Oracle is necessary for organizations to maintain a proper security posture.

To give customers choice and flexibility, Oracle provides Release Updates (RU) and Release Update Revisions (RUR).

- RUs are proactive, tested bundles of critical fixes that enable customers to avoid known issues. RUs are cumulative, released quarterly, and contain security, optimizer, functional, and regression fixes.
- RURs are typically released every quarter after the RU release, and include the contents of a specific RU and fixes for newly identified security vulnerabilities and regressions that affect a lot of customers. RURs are specific to a particular RU train a Oracle's goal is to release two RURs on top of one RU at the same schedule as RUs. Please note that if some critical issue is found, Oracle may release an RUR earlier or even release a third RUR on top of an RU.

At the time of the first RUR release (after 3 months), customers can either progressively step forward by applying RUs, or pause with new optimizer and functional fixes for up to 6 months patch period by applying RURs and still get security fixes.

Oracle strongly recommends that customers apply RUs and RURs as needed every quarter.

For older releases (before 12.2), apply Patch Set Updates (PSU) or Bundle Patches (BP):

- 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 includes everything that is in the PSU plus optimizer and functional fixes. They are also delivered on a pre-defined schedule and are cumulative.

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

Patch deployements through database lifecycle management

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 securely is the database configured is the foundation for a defense-in-depth strategy. 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, Oracle Data Safe, and Oracle Database Lifecycle Management pack provide the tools to help you identify areas where your database configuration, operation, or implementation introduces risk.

11 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. Privacy laws regulate the collection, storage, sharing, control, and use of personal information about individuals, also called personal data or Personally Identifiable Information (PII).

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.

Globally, more than 130 different governments have enacted privacy legislation and many more have pending bills or initiatives. There are hundreds of data privacy regulations in the United States alone, at both the state and federal level, with 29 new state data privacy laws in US from July 2019 to July 2020.

Since the last edition of this book, we've seen the California Consumer Privacy Act (CCPA) kick off with dozens of lawsuits filed just within the first few months. EU GDPR is now widely enforced, with significant fines of hundreds of millions of Euros already levied.

Because EU GDPR is one of the most comprehensive privacy laws, it is of interest to many organizations. Several countries model their data privacy laws on the EU GDPR framework. 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. The details amongst regulations may vary, but all data privacy regulations essentially attempt to safeguard people's personal information by regulating security practices protecting that data.

There are numerous books, white papers, or other material such as this book that can help you get an overview of such regulations, 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. Note 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.

Introduction to EU GDPR

Before EU GDPR, each of the European Union countries had their own internal national Data Protection laws. Understanding the dozens of different laws and complying with their sometimes-conflicting requirements was a significant impediment to doing business in the EU.

EU GDPR provides a single common privacy framework for safeguarding EU residents from misuse of their data. This common privacy framework reduces the cost of compliance and simplifies the process of dealing with personal information for businesses serving residents of the European Union. 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, 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.

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.

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.

To understand the various actors and their roles, as well as how they relate to each other, consider a hypothetical company based in France selling good 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 is a Third Party. The company's Data Protection Officer is the liaison between the Supervisory Authority and the company.

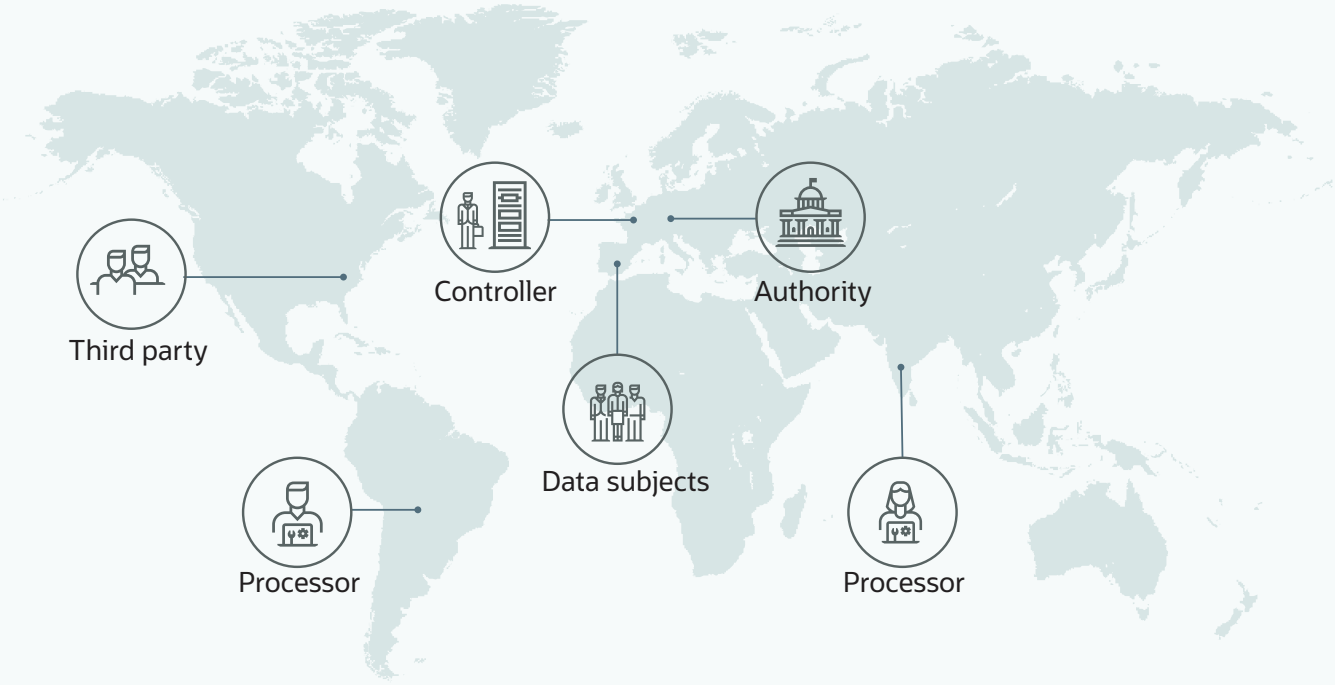


Figure 11.1: EU GDPR roles

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 and databases

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. 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 does not go into specifics on exactly what those technical measures and safeguards are–it is 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.

EU GDPR applies to all data processing and storage systems. Given the amount of personal data that tends to reside in databases, comprehensively protecting databases becomes a key part of compliance with EU GDPR and similar data privacy regulations. 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 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. From database perspective, the assessment should include:

- **Discover sensitive data**—organizations need 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. Data Safe, DBSAT Discovery and Enterprise Manager Application Data Modeling can help with this—remember, Enterprise Manager's Sensitive Data Discovery is included with ALL database security products and options.

- **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) or Data Safe can help with this. Refer to *Chapter 10* for more information on assessing database configuration.
- **Document risks** and the measures used to mitigate those risks—the findings and remediation recommendations from DBSAT or Data Safe can be made a part of the DPIA report.

Protect against database bypass

For Oracle databases, data should be 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 such attacks useless and forces attackers to use database commands to access data—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), a method of helping to ensure the security of processing (Article 31.1), 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). This is one of the core reasons why Transparent Database Encryption (TDE) is 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 or exports) of the database. Database encryption is now such a common requirement that all database services 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

Organizations must decide how to control access to data and how those controls should be configured to meet Article 25 obligations to protect data by default and by design. A few rules of thumb 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. 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 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 and 32.1, 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 such as taxpayer id, account number, and 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 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 (Articles 25.1, 47.1, and 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* or using Oracle Data Safe as discussed in *Chapter 13*.

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 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, 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 security 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

On-premises and cloud deployments are quite different, but from a security threat perspective many risks look the same. Both are concerned about network/OS infiltration, exploitation of configuration vulnerabilities including unpatched systems, compromised users, and vulnerable applications, however the infrastructure and administrative risks are quite different. On-premises systems benefit from known, trusted administrators and an inherently restricted threat environment as they operate inside the organization's firewall where they are protected, and access is limited. In comparison, cloud services benefit from dedicated security administrators, binding service level agreements, and an increased focus on process.

Cloud databases may be exposed to different threat vectors than those running on-premises. For example, attackers might attempt to reach the database through direct attack on the cloud provider's network, or through the networks of other customers. You also must trust the cloud administrators and depend upon their security policies. While database security is different for on-premises and cloud deployments, the broad industry consensus is that the cloud's security advantages outweigh the disadvantages.

This chapter describes how these differences change the security responsibilities for the consumer, and how they should secure their cloud databases.

Some threats remain the same

Some threats to your database in the cloud mirror the threats faced on-premises and cloud databases can be protected using similar controls:

- Encrypt data at rest and in motion to limit the attack surface on databases.
- Control what database users can do through proper privilege and role management, including Oracle Database Vault, Data Redaction, Label Security, and Real Application Security.
- Identify and mitigate any security configuration gaps with Oracle Data Safe.
- Monitor database activity to detect anomalies and block malicious activity with Oracle Data Safe.
- Minimize the proliferation of sensitive data in test and development with Oracle Data Masking and Subsetting or Oracle Data Safe.

Threats unique to the cloud

Some threats faced in the cloud are different, including difference in scope and emphasis. Let's 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. For on-premises, the organization knows their employees and contractors, and they can check employee backgrounds, even if it is just a credit check. Further, their physical controls supplement IT access controls, and an external attacker must first penetrate the internal network before attacking the database.

For cloud, organizations generally have no knowledge of who the cloud service providers' employees are, and have no ability to check their backgrounds. However, cloud being the business of cloud vendors, all reputable cloud vendors do deeper background checks, invest in controls to detect and prevent unwanted intrusions, and have fully-staffed security operations centers that monitor operations around the clock. They also have strong operational and separation-of-duty controls designed to minimize administrator access to user data.

Shared infrastructure

Another difference in the cloud is the use of shared resources. For on-premises, all resources including servers, networks, compute, and storage are deployed to support the company's goals. This dedicated infrastructure limits perceived risk because all applications belong to the same organization, operate within its boundaries, and follow its access policies to create a trusted zone. In attack scenarios, a server can be physically disconnected from the network or powered down while the situation is evaluated and remediated.

In the cloud, you can remove logical access to a service or de-provision it—but you have no physical access to power down a server or unplug the network cable. Customers should examine the cloud provider's architecture to identify how systems are isolated from one another, and what the process is to isolate and disconnect 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 your sensitive data. This makes encryption even **more important** in the cloud than on-premises. At Oracle, we encrypt databases in our cloud by default–we don't want to assume the risk of storing our customers' 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. Oracle Cloud Infrastructure Vault allows users to centrally manage and maintain control of the encryption keys that protect enterprise data and the secret credentials used to securely access resources.

However, some customers require that encryption keys for cloud-based systems be retained outside of the cloud service provider's environment. An on-premises deployment of Oracle Key Vault satisfies this need. In case of a loss of confidence in the security of a database in the cloud, you may simply suspend access to that database's master encryption key stored in Oracle Key Vault. Once those pending concerns are resolved, you can 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 a hybrid cloud security model, giving you best of both worlds.

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 monitor their activity. You can do this through Oracle Data Safe (recommended), or you could implement a hybrid cloud control with an on-premises monitoring system (for example, Oracle Audit Vault and Database Firewall) to monitor databases in the cloud. Oracle Audit Vault and Database Firewall in this mode could provide local monitoring of both on-premises and cloud resources.

Configuration management

Most organizations have configuration standards for hardware, operating systems, and databases. When a new database server is installed, those standards are followed, and the system can then move to production knowing that it is secured according to established risk acceptance.

With cloud, the service provider's configuration standards are followed, and they may or may not match perfectly with your on-premises standard. If a complete application stack is being deployed, 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 on the ground is usually quite alarming. 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 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.

Cloud service consumers must understand how their database is configured in the cloud. Consider using the Security Assessment feature in Oracle Data Safe to scan the database configuration and identify any gaps that could represent a vulnerability. Data Safe delivers actionable reports so you can start mitigating these risks immediately.

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 with 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 databases in its cloud by default, provides Oracle Data Safe, and includes security options like Oracle Database Vault, Label Security, and Data Redaction for the Oracle Database Cloud Service High and Extreme Performance editions, or with Oracle Autonomous Databases. This means consumers of those services get the advantage of the capabilities discussed in earlier chapters.

Shared responsibility

Security in the cloud imposes a shared responsibility model, with some tasks being the responsibility of the service provider, and others of the service consumer. The more access the service consumer has, the more security responsibility they need to assume. For on-premises, all responsibility falls upon the service consumer.

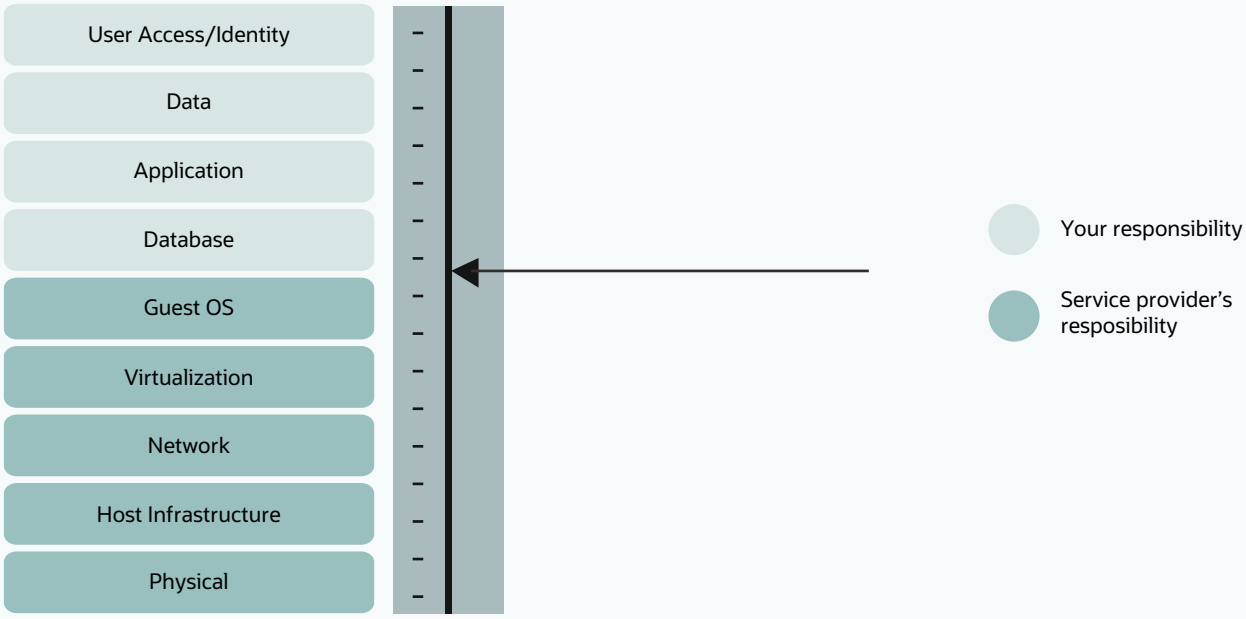


Figure 12.1: Shared responsibility model for PaaS

Infrastructure as a Service (IaaS) gives consumers access to the system at the operating system level, meaning the consumer assumes responsibility for everything above and including the OS. Conversely, Software as a Service (SaaS) usually only allows the consumer to access the application's user interface or published APIs. 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 (DBaaS) is an example of PaaS where the consumer has access at the database level and assumes responsibility for the database and higher in the stack. However, even within DBaaS offerings, the dividing line may vary. For example, Oracle's Autonomous Database shifts more of the security responsibility onto Oracle, bringing the Autonomous Database closer to the SaaS model. Responsibility for the database is split, with the majority of the responsibility (including application of security patches) lying with the provider, and a smaller part with the consumer. 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.

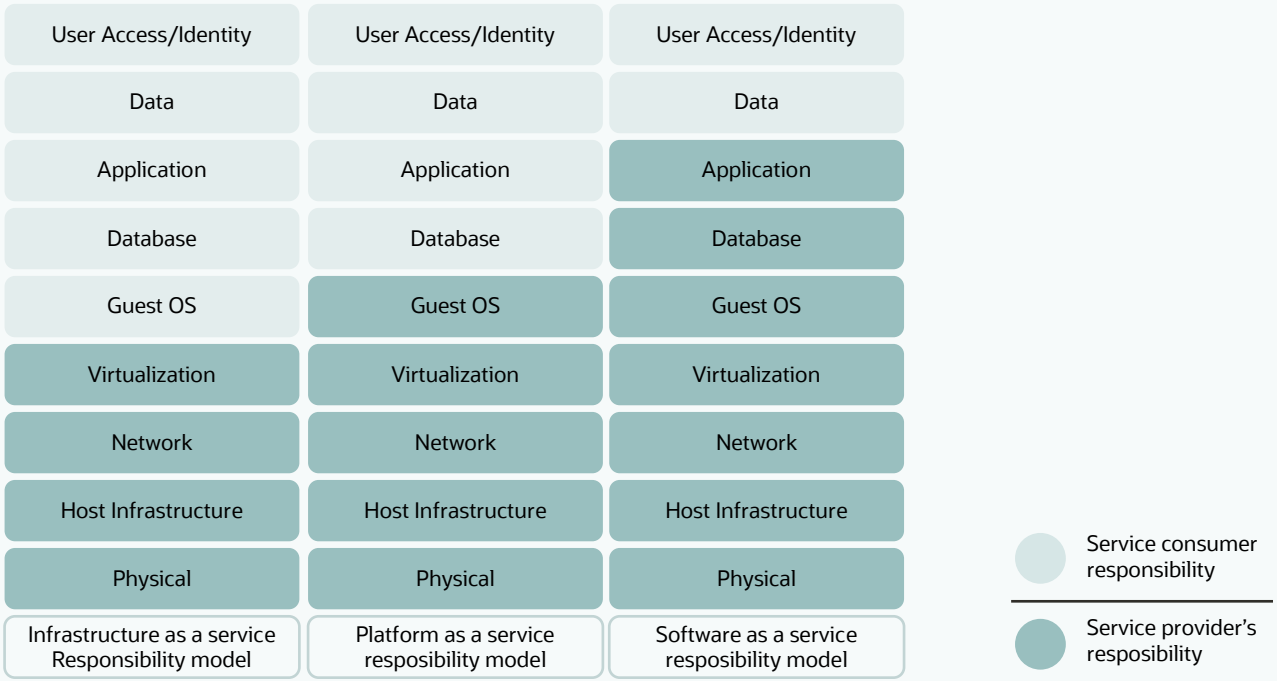


Figure 12.2: Cloud service responsibility model

The common DBaaS shared responsibility model changes if the service platform moves into the consumer's physical data center. For example, Oracle offers Exadata Cloud-at-Customer, where the cloud infrastructure is located inside the service consumer's data center. In this case, the consumer assumes responsibility for physical security of the system, most of the network and the infrastructure responsibility for power and 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.

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

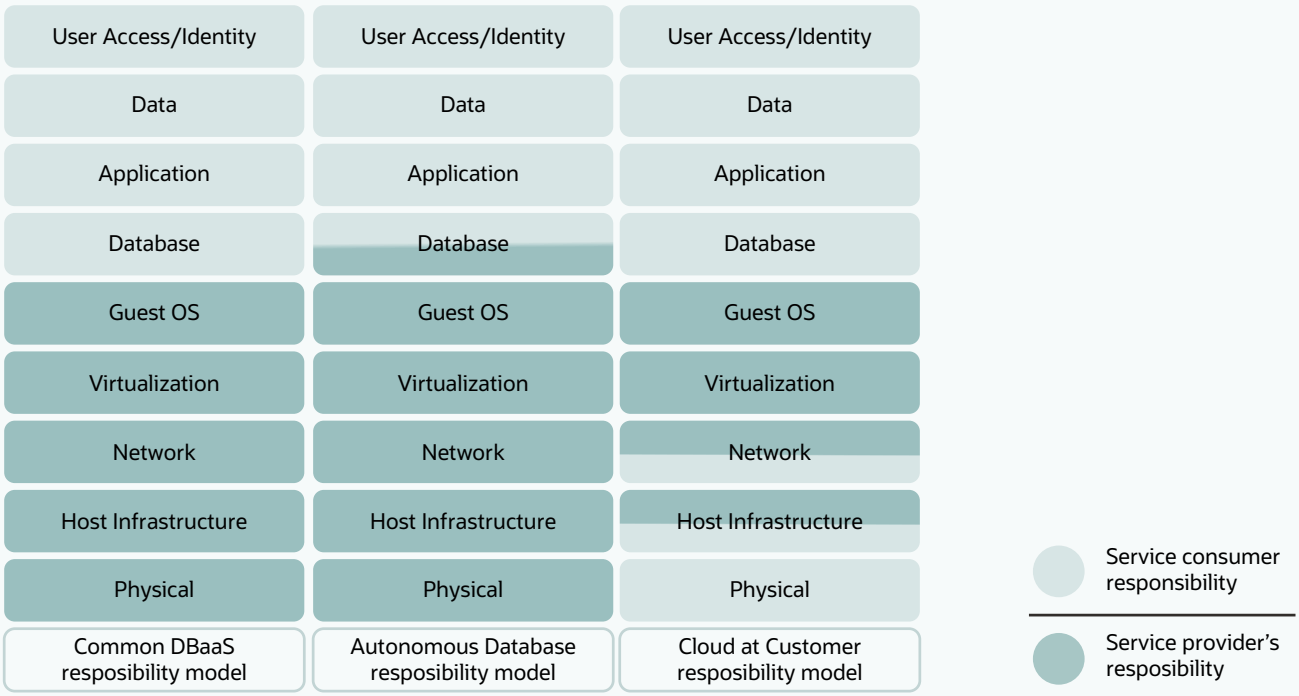


Figure 12.3: DBaaS shared responsibility model

So, what should we do?

The database cloud services used determine how and when periodic security assessments should be performed as part of the organization's security responsibilities. You should include some validation to confirm that the cloud service provider is also doing their part. Assessments may be procedural such 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 and how Oracle can help you with them:

- **Encryption:** DO NOT place unencrypted data in the cloud. This is a risk your organization should NOT take. Oracle encrypts Oracle Cloud Databases running on the Oracle Cloud Infrastructure automatically.
- **Data Minimization:** DO NOT leave sensitive data in test and development systems. Understand the sensitivity of the data in your database and anonymize that data using masking techniques, or use completely artificial data sets that present no security risk. You can easily mask your data in test and development systems (including substituting your data with artificial data sets) with Oracle Data Safe.
- **Access Control:** DO implement separation of duties. Plan as if the users have been compromised, 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. You can assess the risk level of your database users with the User Assessment feature in Data Safe. In addition, you can use Database Vault to further protect sensitive data.
- **Activity Monitoring:** DO create audit records for all critical activities and monitor the audit trail. Ensure the organization or cloud provider's audit facility creates alerts for unusual activities. You can review and enable recommended audit policies in your database with Data Safe. Data Safe can then collect your audit records and provide pre-defined audit reports to help you check for any suspicious activity.

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 different risks, including an extended network perimeter, expanded threat surface with an external administrative group, and shared infrastructure. Despite that, with the right procedures in place, Oracle 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 strong 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 applying appropriate security controls to protect their data based on how mission-critical or sensitive that data is. In some sense, this is no different from the responsibilities they have for their on-premises databases. To secure their users and data, database administrators need to be able to answer some fundamental questions:

- 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 the ability to secure both your cloud databases and your on-premises databases with an easy-to-use-interface?

Enter Oracle Data Safe

Data Safe 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, running in the Oracle Cloud.

Data Safe employs a simple "click-and-secure" model and is designed to be accessible to 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 immediately visible to database administrators.

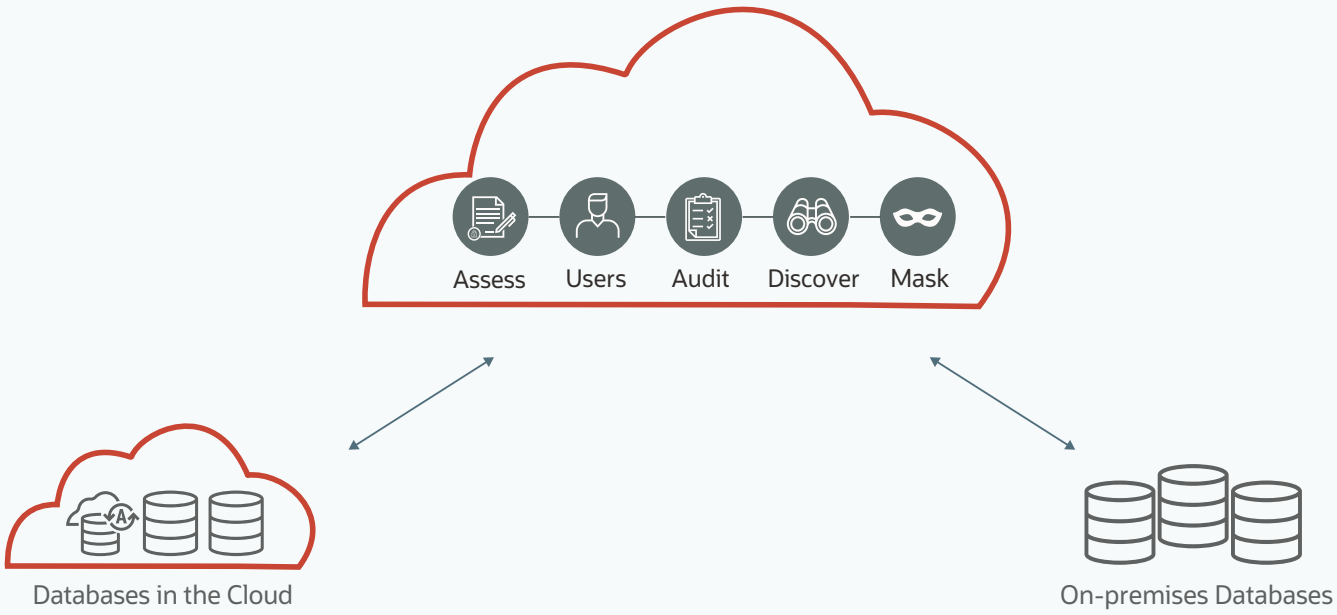


Figure 13.1: Oracle Data Safe provides essential security for Oracle databases, both in the cloud and on-premises

Assessing security

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 compared to 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.

The following snippets show a sample report of security assessment including a high-risk finding, and finding that needs to be looked at further. The findings include not just what the problem is, but also its severity and recommendations on how to remediate the problem.

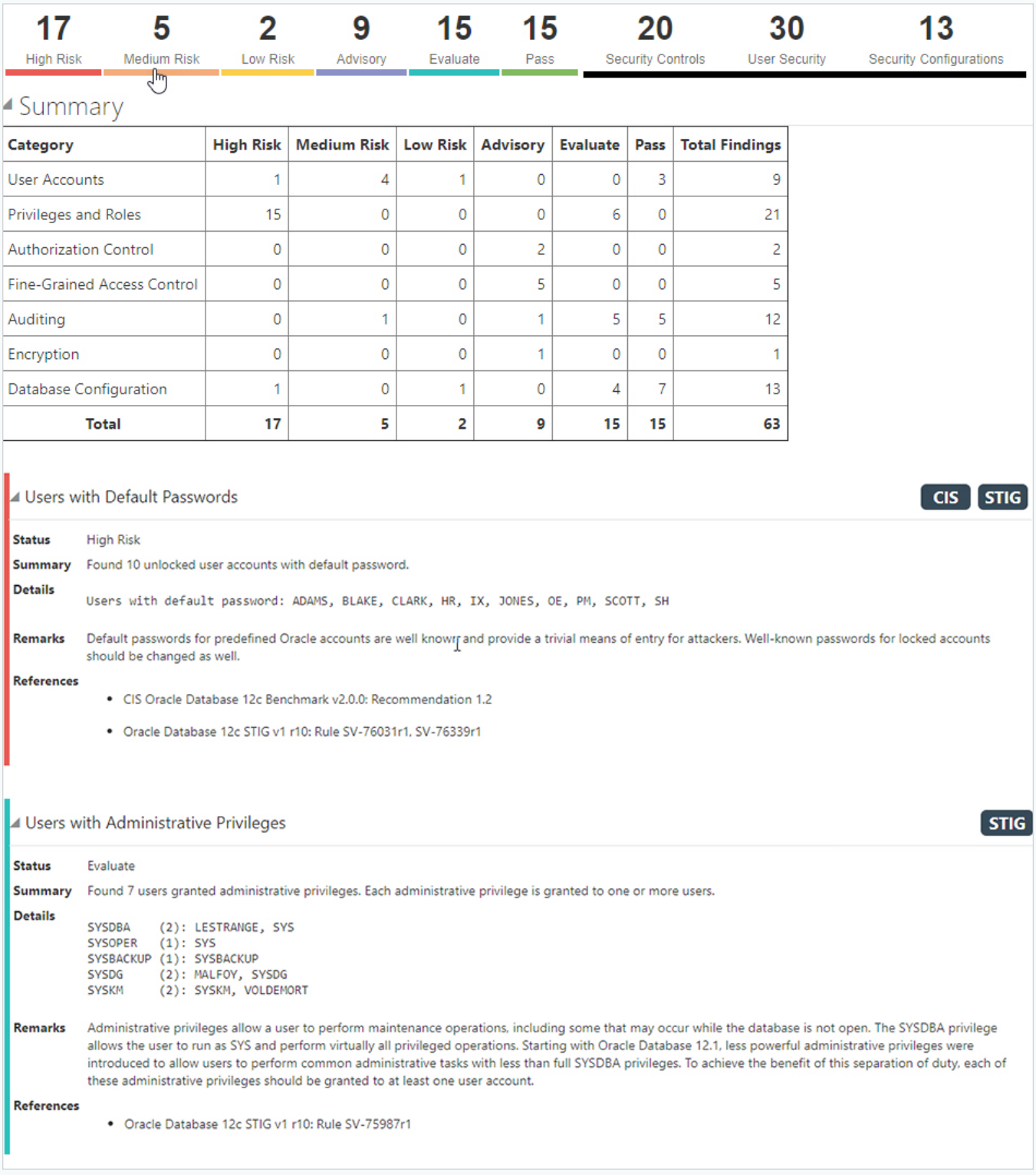


Figure 13.2: Oracle Data Safe: Security assessment

Understanding 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 controls such as auditing. It helps you understand, for example, how many users have not logged-in in the last 3 months or longer, or have not changed their passwords. If there is some suspicion about activity by a user, it can help you understand all details about the user including when they were created, 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.

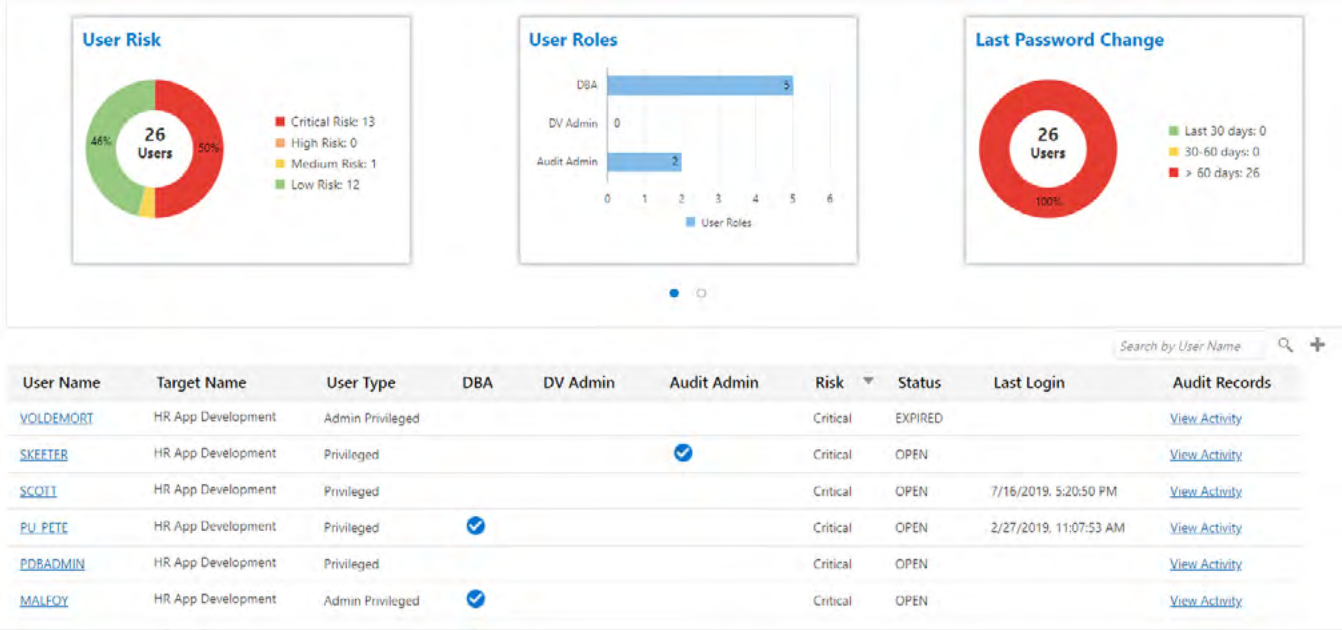


Figure 13.3: Oracle Data Safe: User risk assessment

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 enable them on the database. Data Safe can then start collecting and storing audit records obtained from the databases.

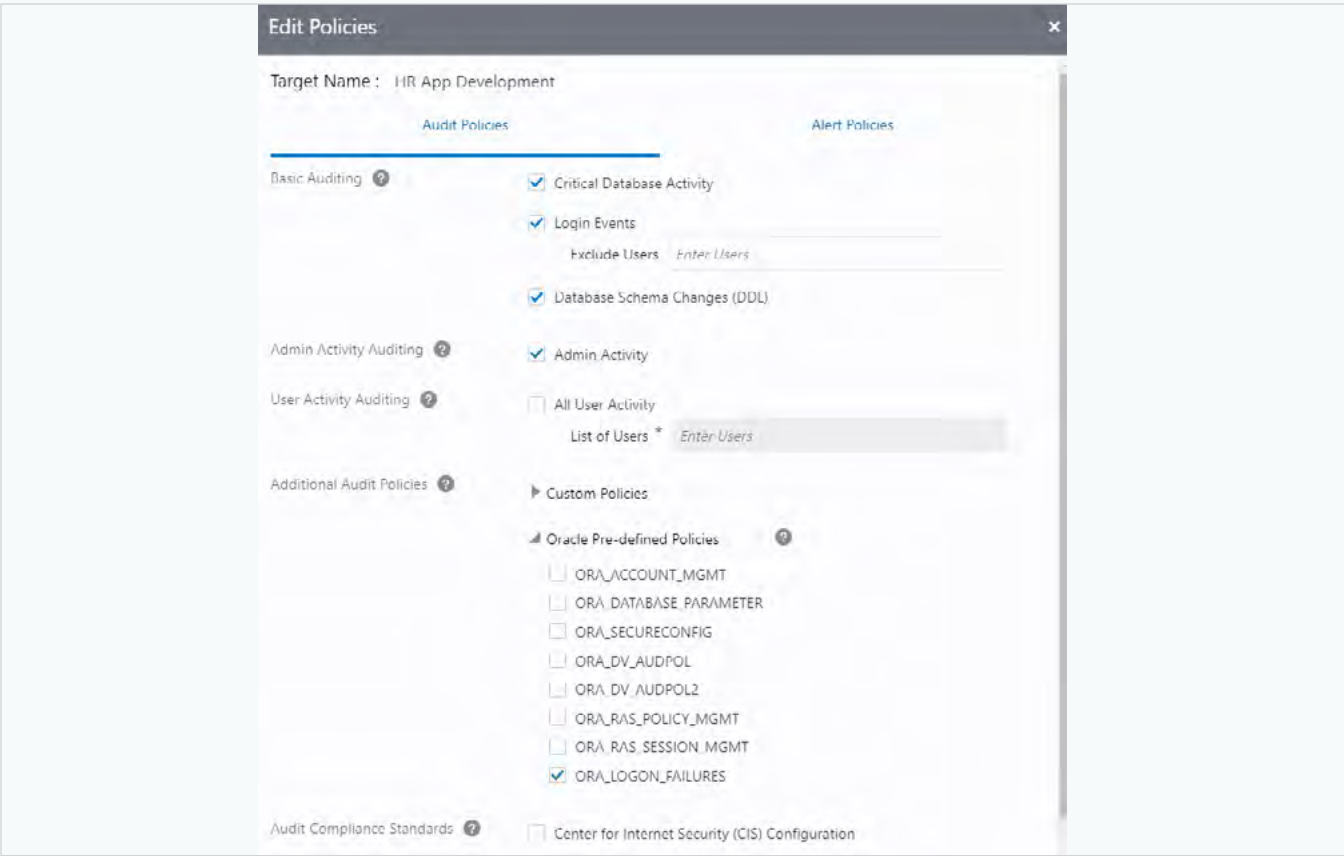


Figure 13.4: Oracle Data Safe: Audit policy provisioning

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?", "Where is it stored?", and "How much of it do I have?"

Data Safe discovers over 125 sensitive data types across categories including personally identifiable information, financial information, health information, IT and 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.

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

Figure 13.5: Oracle Data Safe: Predefined sensitive data types

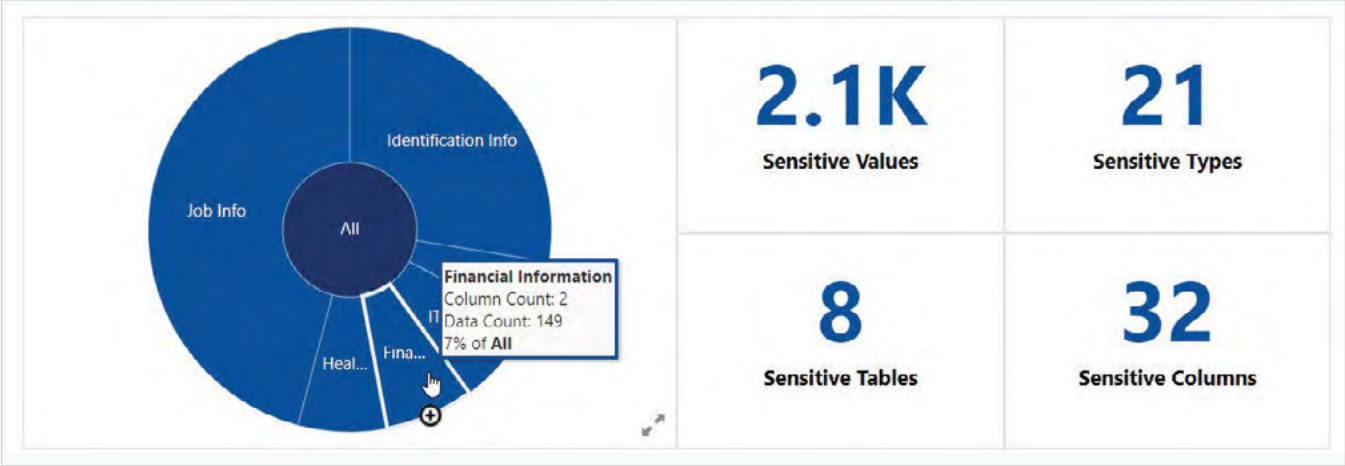


Figure 13.6: Oracle Data Safe: Sensitive data discovery

Masking sensitive data

In the earlier chapter on data masking, we learned how masking removes security risk from test and development systems and helps 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 Safe users can also add their own custom masking formats. 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. The results of each masking run are summarized with an interactive masking report.

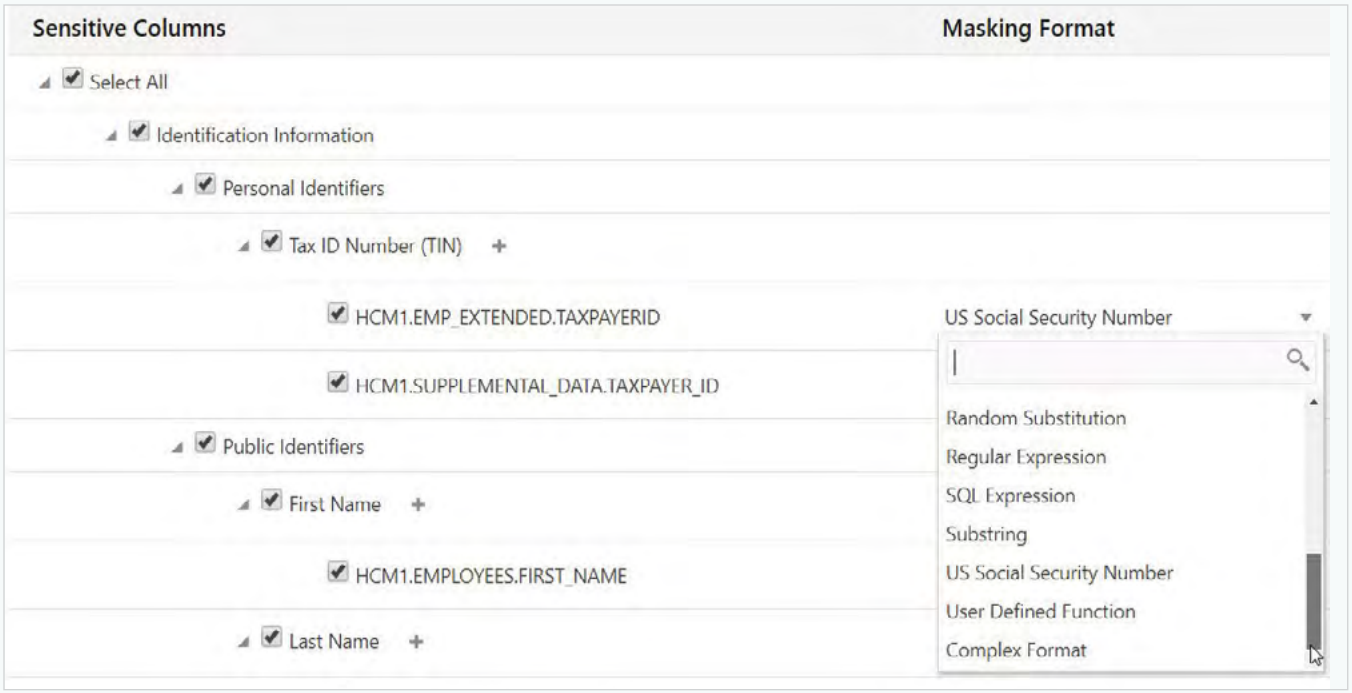


Figure 13.7: Oracle Data Safe: Sensitive data masking

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' overview that allows Data Safe users to visualize their configuration, user, and data risks, and identify open issues requiring immediate attention. By clicking into the graphs and charts on the dashboard users can drill down to view details by database or user, enabling them to identify opportunities to better secure their application data.

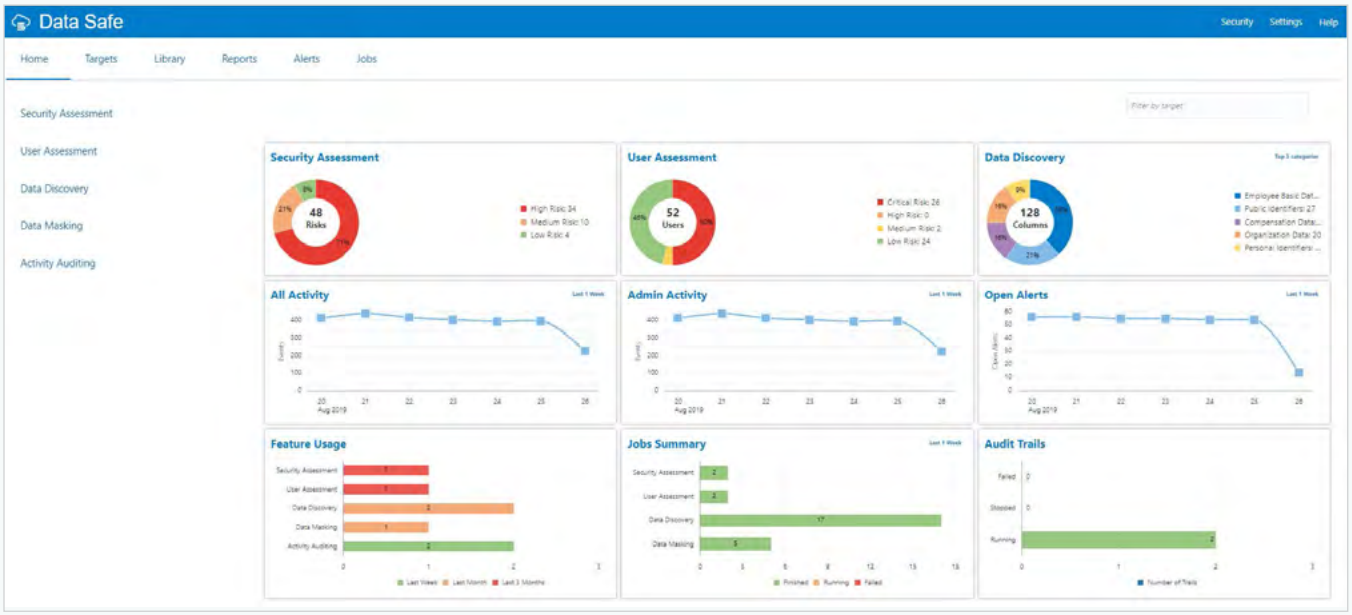


Figure 13.8: Oracle Data Safe: Security console

Protecting databases on-premises and in the cloud

Even with a managed database service like Oracle Autonomous Database, users have considerable latitude in how they configure their databases. Data Safe integrates seamlessly with Oracle Autonomous Databases, allowing cloud administrators to register new databases with a single click. For Oracle Cloud Databases, Data Safe supports a private endpoint feature which enables secure connectivity between the database and the Data Safe service in the Oracle Cloud without the need to expose the database's IP address publicly.

For customers who have network peering between the Oracle Cloud and their on-premises environment, this same private endpoint feature can be used to securely connect on-premises databases to Data Safe as well. This solution works whether the network connectivity uses a customer provided VPN solution or Oracle Fast Connect.

Finally, customers can download a lightweight on-premises connector through the Data Safe console and deploy it on a machine in their own network, or on a virtual machine in any other cloud network. The connector acts as a network proxy, maintaining a secure, encrypted network path to the Data Safe service. Each of these connectivity options is illustrated below.

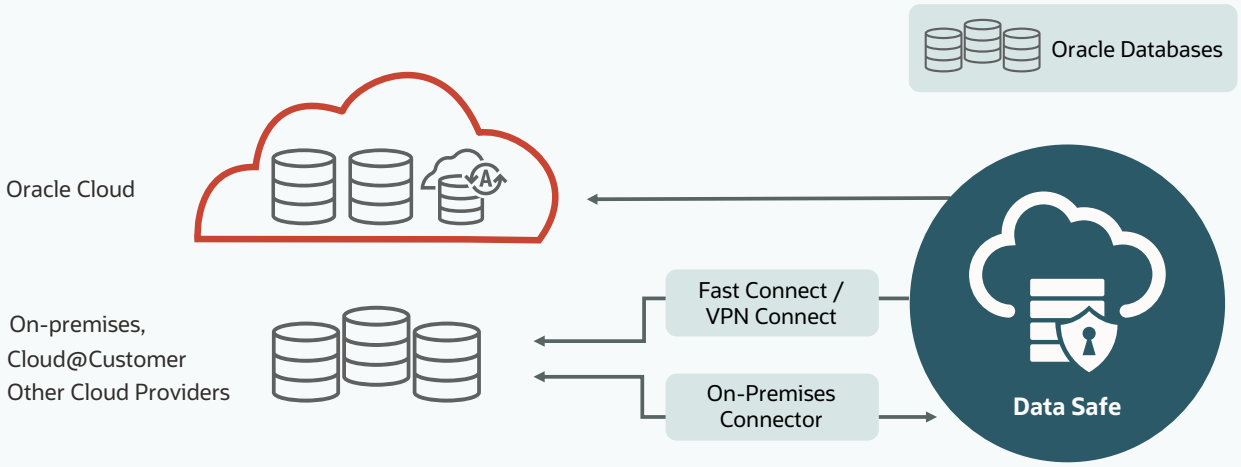


Figure 13.9: Oracle Data Safe connectivity

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, whether their databases are running on-premises or in the cloud. With Oracle Data Safe, critical functionalities for securing databases are instantly available through 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, whether they are on-premises or in any cloud.

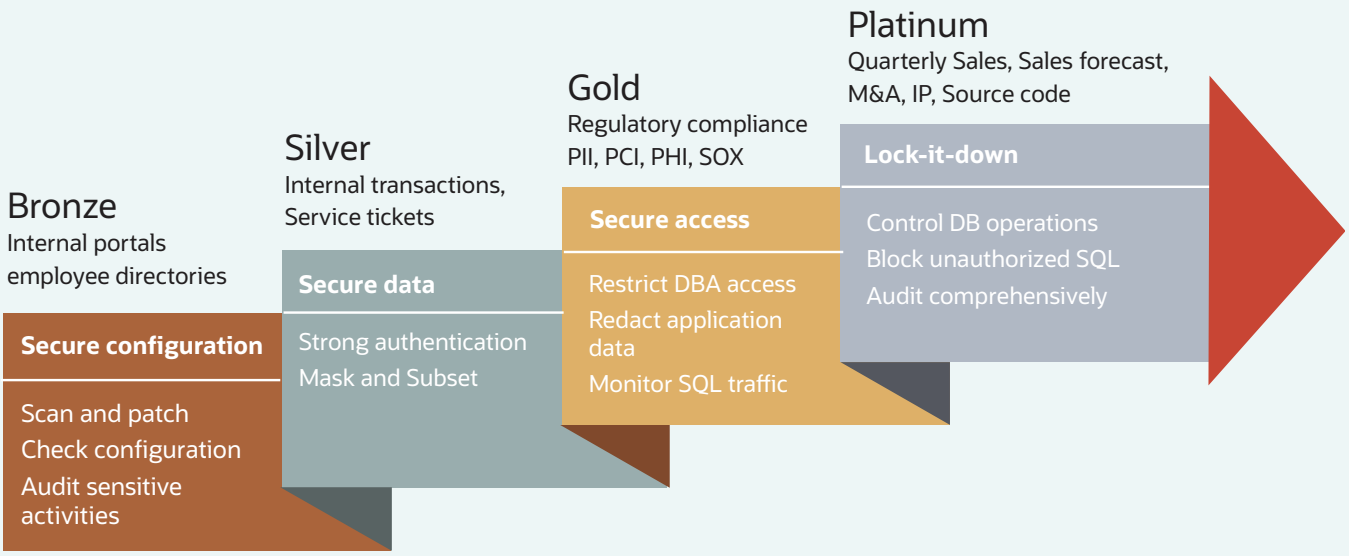
Conclusion: Putting it all together

> 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.



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, you 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.

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 the latest product data sheets, white papers, demo viewlets, and more.

https://www.oracle.com/security/database-security/

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 Data Safe—Get started: Learn about Oracle Data Safe, view links to descriptions of common tasks, and access video tutorials.

https://docs.oracle.com/en/cloud/paas/data-safe/

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/security-alerts/

Oracle Database Security blog: Articles on topics of interest to the database security community.

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

Oracle Corporate Security 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 week of most months, where we'll update you on the latest product announcements and provide a deep dive into a 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

Oracle Corporation

Worldwide Headquarters
500 Oracle Parkway, Redwood Shores, CA 94065, USA

Worldwide Inquiries
Tele + 1.650.506.7000 + 1.800.ORACLE1
Fax + 1.650.506.7200

oracle.com

Connect with us

facebook.com/oracle

youtube.com/oracle

linkedin.com/company/oracle

twitter.com/oracle