Mitigating Risks of SQL Injection

November, 2023, Version 1.0
Copyright © 2023, Oracle and/or its affiliates
Public

Purpose statement

This document provides an overview of features and enhancements included in the release Oracle Database 23c, and Oracle Audit Vault and Database Firewall (AVDF) 20.10. It is intended solely to help you assess the business benefits of upgrading to Oracle Database 23c or AVDF 20.10 and planning for the implementation and upgrade of the product features described.

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Copyright © 2023, Oracle and/or its affiliates / Public

Executive summary

SQL injection is one of the oldest and most frequently encountered database attack patterns. Despite years of research and efforts to solve the problem, it plagues data-driven web applications. The traditional efforts to use Web Application Firewalls (WAFs) or add protection at the application level fail to mitigate the SQL Injection risks. Oracle Database security provides two approaches to help mitigate the risk of SQL Injection against data-driven web applications.

1. Network-based Database Firewall in Oracle Audit Vault and Database Firewall (AVDF)
2. Oracle SQL Firewall built into Oracle Database

This technical report discusses how you can leverage and configure these solutions to mitigate the SQL Injection risks to your data-driven web applications and suggest strategies for selecting the approach that best fits your needs.

Copyright © 2023, Oracle and/or its affiliates / Public

Chapter contents

List of figures

Copyright © 2023, Oracle and/or its affiliates / Public

List of tables

Copyright © 2023, Oracle and/or its affiliates / Public

Introduction

SQL injection is one of the most popular attack patterns on the Open Web Application Security Project (OWASP) list of the top ten threats to application security and has been since 2017. SQL injection involves injecting malicious SQL code into input fields or parameters of data-driven applications, tricking the application into responding to the injected SQL. A successful SQL injection attack causes the application to expose data or take actions not intended by the application developer.

Most three-tier applications connect to the database as an application service account, which is a special type of non-human privileged account used to execute application SQL queries in the underlying database and run automated services and other processes. Application service accounts are privileged accounts with all necessary privileges to perform any action the application might be expected to perform. They typically have access to the entire application schema (including all data) and procedures. The application tier enforces access control in the middle tier based on the end user's identity and authorization. An SQL injection attack consists of the insertion or "injection" of malicious code into the input fields of the application. The attack forces the application to respond to the injected SQL, which could do anything to the database that the application service account is authorized to do. It is a perfect technique to bypass the authentication and authorization mechanisms of a web application and use the application service account's privileged access to retrieve the contents of the database - including unauthorized access to sensitive data: customer information, personal data, trade secrets, intellectual property, and more. Attackers may try to use SQL injection to add, modify, and delete records in the database.

SQL injection is often the entry point for a more extensive compromise. In addition to exfiltrating data from the compromised database, attackers may move laterally to other hosts on the internal network. There are many types of SQL injection vulnerabilities, attacks, and techniques. But all of them have a similar attack pattern:

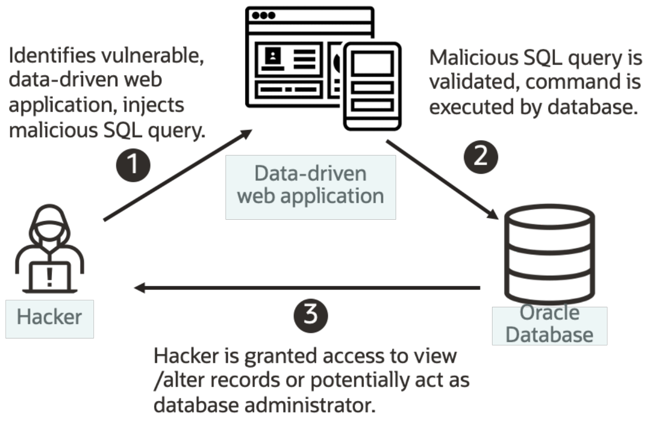

Identifies vulnerable, data-driven web application, injects malicious SQL query.
1
Data-driven web application
Malicious SQL query is validated, command is executed by database.
2
Hacker
3
Oracle Database
Hacker is granted access to view /alter records or potentially act as database administrator.

Figure 1: SQL Injection attack pattern

Preventing or mitigating SQL injection attacks is not easy. Traditional approaches incorporate protection at the application level, including improved user authentication, enforcing least privilege, using prepared statements, avoiding dynamic queries, and *performing input validation*. Although these methods provide more protection and should be practiced, even minor gaps in development or configuration controls can introduce vulnerabilities. For legacy applications, you need access to the source code to modify SQL injection vulnerabilities.

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

Copyright © 2023, Oracle and/or its affiliates / Public

Another approach to block SQL injection attacks is to *filter database traffic* before the database processes it. This is where a database firewall comes into the picture. Database firewalls offer a simple but effective way to analyze incoming SQL to detect injection attacks, raise alerts when required, and block injection attacks from reaching the database. Oracle Database security provides two approaches to filter database traffic to mitigate the risk of SQL injection attacks:

1. Network-based Database Firewall in Oracle Audit Vault and Database Firewall (AVDF)
2. SQL Firewall built into Oracle Database and included as part of both Database Vault and AVDF

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

SQL Firewall in Oracle Database 23c moves protection even closer to the data, shifting the point of enforcement from the network into the database kernel. Enforcing protection within the database kernel avoids the complexity of routing database traffic through an external Database Firewall. Applications can leverage this powerful new database security feature to reduce the risk of SQL injection attacks.

Let's look at both these solutions to understand how they mitigate the risk of SQL injection. Both network and built-in firewalls can address these three use cases:

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

Network-based Database Firewall

Database Firewall, an AVDF component, 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 protect multiple databases of different types from a central location. It monitors enterprise databases, including Oracle, MySQL, Microsoft SQL Server, SAP Sybase and IBM Db2.

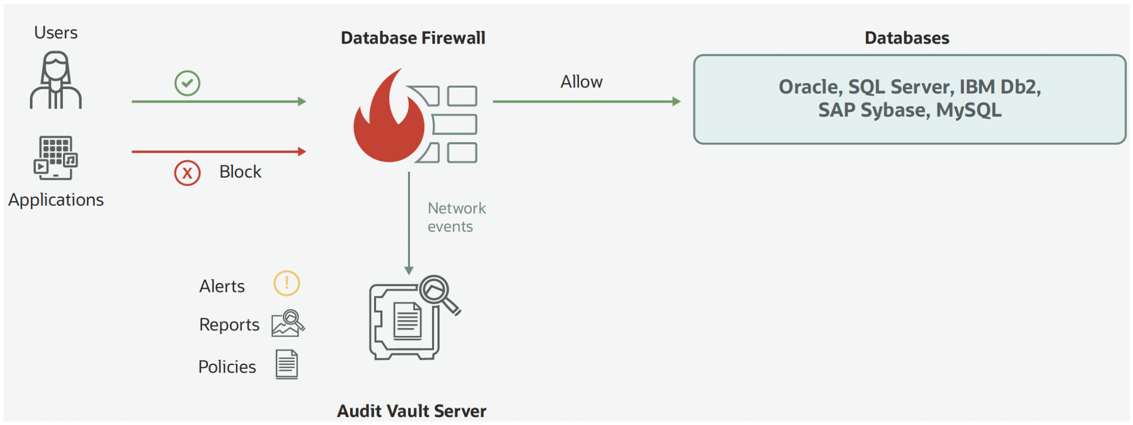


Figure 2: Network-based Database Firewall in AVDF

Database Firewalls can be deployed in various modes, as shown in the Table1:

Copyright © 2023, Oracle and/or its affiliates / Public

Table1. Deployment modes of Database Firewall

MODE	Details	Monitoring or blocking
Proxy	All the traffic to database server is routed through the Database Firewall, including return traffic.	• Monitoring • Blocking
Host monitor	Host Monitor is deployed on the same machine as the database and captures SQL traffic going to the database, and then securely forwards it to the Database Firewall.	• Monitoring
Out-of-band	Database Firewall listens to the network traffic sent to the database. 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.	• Monitoring

The choice of deployment mode depends on whether you want to detect and monitor potential SQL Injection attacks or if you wish to enforce trusted access patterns to the database by blocking un-authorized activity.

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

To block unauthorized SQL traffic, the firewall must be in-line with the incoming SQL traffic using a proxy deployment. Proxy enforcement can initially be set to monitor traffic (without blocking) and just alert on unauthorized activity. Once you are confident with the firewall policy, you can switch to the blocking mode to enforce the policy and block SQL statements that are not in your allow-list, as well as other unauthorized access.

Database Firewall inspects incoming SQL traffic and determines whether to allow, log, alert, substitute, or block SQL commands. The firewall evaluates SQL traffic through multiple stages, including checks for the IP address, database user, OS user, program name, SQL statement category, data definition language (DDL, data manipulation language (DML), and database tables accessed.

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

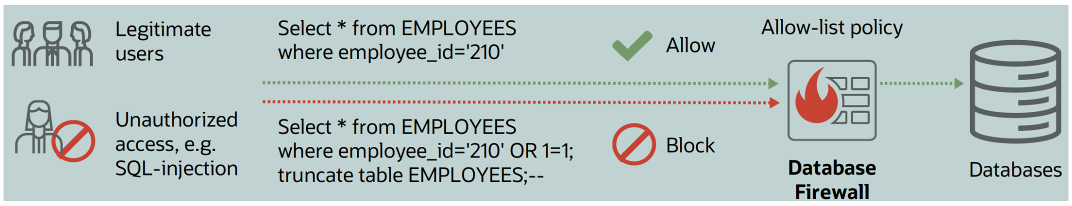

Legitimate users
Select * from EMPLOYEES where employee_id='210'
Allow
Allow-list policy
Unauthorized access, e.g. SQL-injection
Select * from EMPLOYEES where employee_id='210' OR 1=1; truncate table EMPLOYEES;--
Block
Database Firewall
Databases

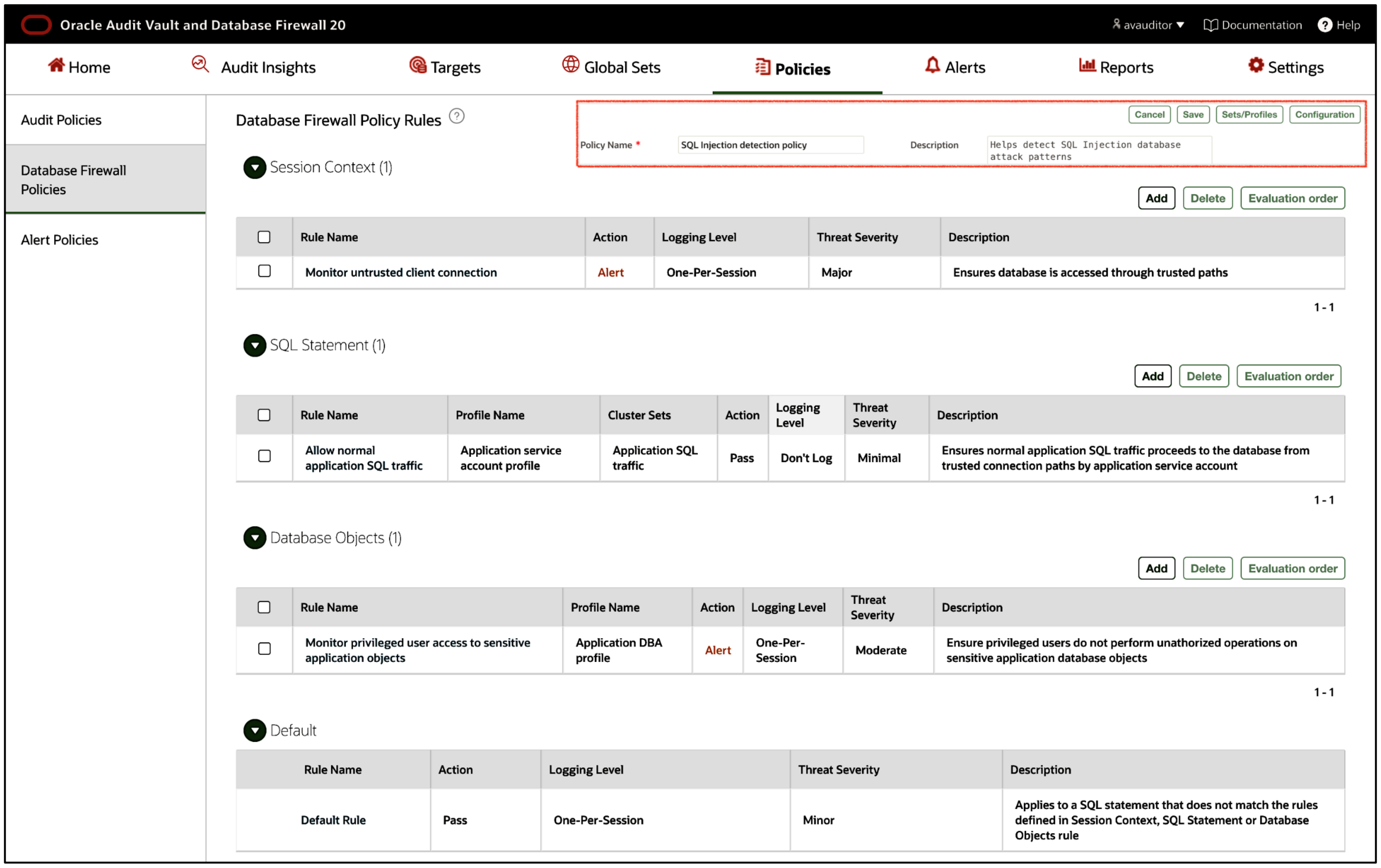
Figure 3: Mitigating SQL Injection risks with Database Firewall

Building allow-lists/acceptable baselines in a Database Firewall policy makes it an effective control to mitigate the risk of SQL Injection, including many zero-day exploits.

Copyright © 2023, Oracle and/or its affiliates / Public

Database Firewall policy for mitigating SQL Injection risk

A typical allow-list policy for mitigating SQL Injection risk looks like the following:



Database Firewall Policy Rules

Policy Name * SQL Injection detection policy

Description Helps detect SQL Injection database attack patterns

Session Context (1)

	Rule Name	Action	Logging Level	Threat Severity	Description
☐	Monitor untrusted client connection	Alert	One-Per-Session	Major	Ensures database is accessed through trusted paths

1 - 1

SQL Statement (1)

	Rule Name	Profile Name	Cluster Sets	Action	Logging Level	Threat Severity	Description
☐	Allow normal application SQL traffic	Application service account profile	Application SQL traffic	Pass	Don't Log	Minimal	Ensures normal application SQL traffic proceeds to the database from trusted connection paths by application service account

1 - 1

Database Objects (1)

	Rule Name	Profile Name	Action	Logging Level	Threat Severity	Description
☐	Monitor privileged user access to sensitive application objects	Application DBA profile	Alert	One-Per-Session	Moderate	Ensure privileged users do not perform unathorized operations on sensitive application database objects

1 - 1

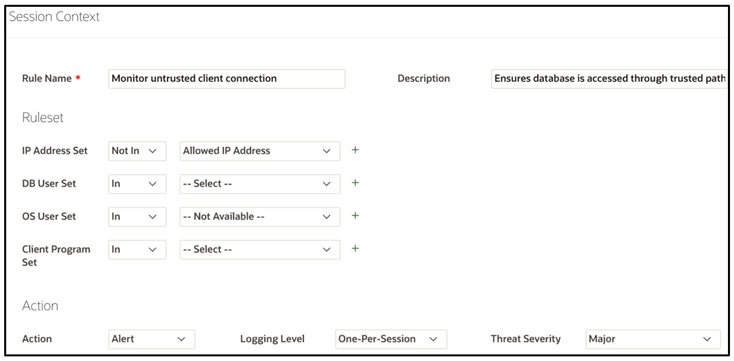
Default

Rule Name	Action	Logging Level	Threat Severity	Description
Default Rule	Pass	One-Per-Session	Minor	Applies to a SQL statement that does not match the rules defined in Session Context, SQL Statement or Database Objects rule

Figure 4: Database Firewall: Policy for detecting and alerting on SQL Injection attack patterns

Leverage the following sequential checks (aka rules) in Database Firewall policy to examine the incoming SQL traffic before it proceeds to the database:

1. Use Session Context rule(s) to help ensure database users (like administrators or application service accounts) access the database through trusted application paths as shown below. Alert (or block) when SQL traffic originates from untrusted client connections. The SQL traffic from trusted sources flows into subsequent rules for further processing.



Session Context

Rule Name * Monitor untrusted client connection

Description Ensures database is accessed through trusted path

Ruleset

IP Address Set	Not In	Allowed IP Address
DB User Set	In	-- Select --
OS User Set	In	-- Not Available --
Client Program Set	In	-- Select --

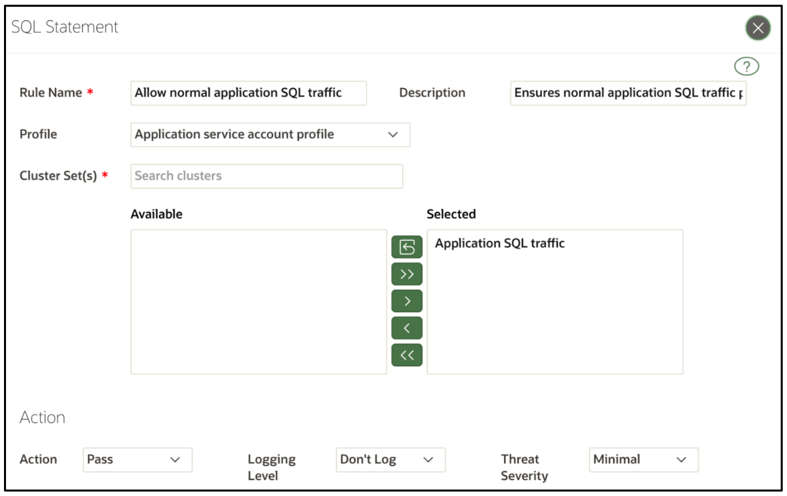
Action

Action: Alert | Logging Level: One-Per-Session | Threat Severity: Major

Figure 5: Database Firewall: Session Context rule

Copyright © 2023, Oracle and/or its affiliates / Public

2. Use SQL Statement rule(s) to help ensure only explicitly approved SQL statements are allowed to reach the database from trusted connection paths as shown below. The rest of the SQL traffic flows into subsequent rules for further processing.



SQL Statement

Rule Name *	Allow normal application SQL traffic	Description	Ensures normal application SQL traffic p
Profile	Application service account profile		
Cluster Set(s) *	Search clusters		

Available:

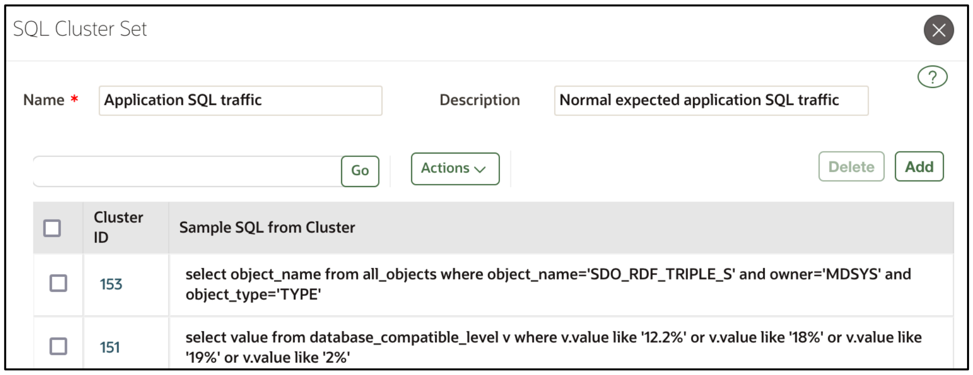
Selected: Application SQL traffic

Action

Action	Pass	Logging Level	Don't Log	Threat Severity	Minimal

Figure 6: Database Firewall: SQL Statement rule

Use SQL cluster set in the rule to represent an allow-list of approved SQL statements. Train the Database Firewall to learn the normal authorized SQL traffic from trusted connection paths. Database Firewall groups unique SQL statements in the SQL traffic to SQL clusters, which can then be used in the SQL cluster set as shown below.



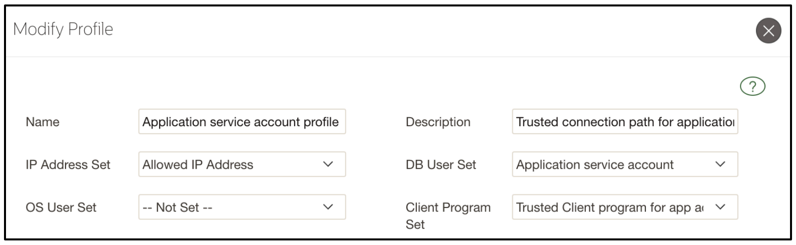
SQL Cluster Set

Name *	Application SQL traffic	Description	Normal expected application SQL traffic

Cluster ID	Sample SQL from Cluster
153	select object_name from all_objects where object_name='SDO_RDF_TRIPLE_S' and owner='MDSYS' and object_type='TYPE'
151	select value from database_compatible_level v where v.value like '12.2%' or v.value like '18%' or v.value like '19%' or v.value like '2%'

Figure 7: Database Firewall: SQL cluster set of normal application SQL traffic

Use profile in the rule to represent the authorized database users who have a valid business reason to access the approved SQL statements in the SQL cluster set from a trusted connection path as shown below.



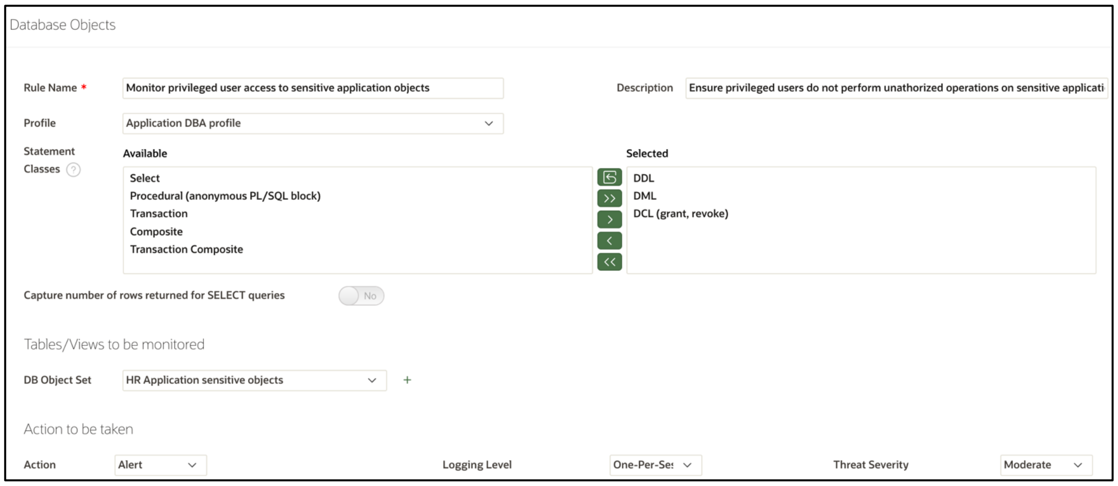
Modify Profile

Name	Application service account profile	Description	Trusted connection path for applicatio
IP Address Set	Allowed IP Address	DB User Set	Application service account
OS User Set	-- Not Set --	Client Program Set	Trusted Client program for app a

Figure 8: Database Firewall: application service account profile

Copyright © 2023, Oracle and/or its affiliates / Public

3. Use Database object rule(s) to ensure privileged users perform only authorized operations on sensitive application database objects over the network as shown below. Alert (or block) on any abnormal operations by privileged users. The rest of the SQL traffic flows into subsequent rules for further processing.



Database Objects

Rule Name *	Monitor privileged user access to sensitive application objects
Description	Ensure privileged users do not perform unauthorized operations on sensitive applicati
Profile	Application DBA profile

Statement Classes

Available	Selected
Select	DDL
Procedural (anonymous PL/SQL block)	DML
Transaction	DCL (grant, revoke)
Composite	
Transaction Composite	

Capture number of rows returned for SELECT queries: No

Tables/Views to be monitored

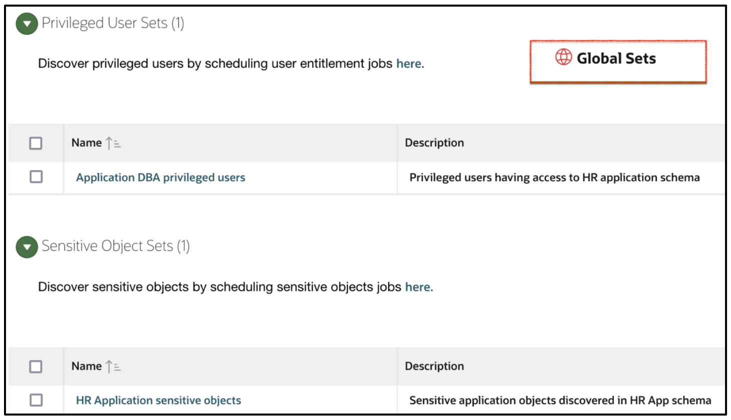
DB Object Set: HR Application sensitive objects

Action to be taken

Action: Alert   Logging Level: One-Per-Ses   Threat Severity: Moderate

Figure 9: Database Firewall: Database Object rule

Leverage Global Sets to identify privileged users and sensitive data as shown below. Use profile in the Database Object rule to represent the named set of privileged users like application DBAs and their trusted connection paths. Use the named set of discovered sensitive objects in the rule and choose the unauthorized operations that you wish to be alerted on.



Privileged User Sets (1)

Discover privileged users by scheduling user entitlement jobs here.

Name	Description
Application DBA privileged users	Privileged users having access to HR application schema

Sensitive Object Sets (1)

Discover sensitive objects by scheduling sensitive objects jobs here.

Name	Description
HR Application sensitive objects	Sensitive application objects discovered in HR App schema

Figure 10: Database Firewall: Global sets with privileged users and sensitive objects

4. Use the Default rule to help ensure that all newer SQL traffic proceeds to the database after being scrutinized. Newer SQL traffic might come in following application updates, the addition of new users or applications, or those not explicitly approved. The SQL statements are logged and are available in reports for viewing. Once the approved SQL baselines are well-established, you should switch the action to alert such traffic.

Alerts raised in the Database Firewall are sent to the Audit Vault server, where notifications are triggered according to the policies you set. Consider monitoring (and alerting) to detect any SQL Injection attack pattern occurrence. You might want to switch to blocking action in the Session Context, Database Object, and Default rule on a well-trained system to adopt a zero-tolerance and preventive strategy to SQL Injection risks (including zero-day attacks).

Copyright © 2023, Oracle and/or its affiliates / Public

Database kernel-resident SQL Firewall

SQL Firewall is a new feature built into Oracle Database 23c. Like the Database Firewall, SQL Firewall helps mitigate the risks of web application attacks such as SQL Injection on data-driven web applications. Unlike the Database Firewall, SQL Firewall does not require additional product installation or network configuration, greatly simplifying deployment. Because SQL Firewall is built into Oracle Database, it only works for Oracle Database 23c and higher.

SQL Firewall is part of the Oracle Database kernel, operating closer to where data resides, eliminating the possibility of bypassing the control. SQL Firewall inspects all incoming SQL statements and helps ensure the database executes only explicitly authorized SQLs coming from trusted database connection paths. It can examine all SQL statements - whether local or over the network, encrypted or clear text. Unlike regular expression-based pattern matching protection mechanisms, SQL Firewall cannot be bypassed by encoding the SQL statement, referencing synonyms, or using dynamically generated object names.

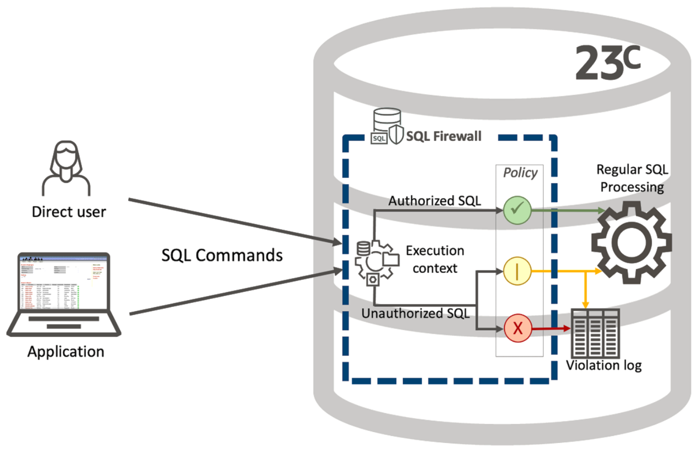


Figure 11: SQL Firewall built into Oracle Database kernel

While other database security features built into the Oracle Database provide different security controls to monitor/prevent web application attacks, SQL Firewall is the only one that inspects all incoming SQL statements and allows only authorized SQL. It logs and blocks unauthorized SQL from executing in the database, offering high levels of protection against SQL Injection database attacks for the Oracle Database.

SQL Firewall can also observe (or block) unusual access patterns such as connections not coming from trusted IP addresses, operating system usernames, or program names, helping ensure that any access to your databases comes exclusively from trusted endpoints. This function is useful when you want to put some protection in place immediately. At the same time, you create the allow-list of SQL statements for your applications, which requires you to extensively train the firewall to so that all possible SQL statements have been captured. The allow-list rules from session context attributes are easy to capture and, if desired, can be manually entered.

By building the SQL Firewall inside the database and streamlining its implementation, its performance overhead is negligible, making it suitable for all production workloads. SQL firewall is included with Oracle Database Vault. Use of SQL Firewall with AVDF is also included for Oracle Databases being monitored.

SQL Firewall policy for mitigating SQL Injection risk

SQL Firewall policies work at a database account level, whether of an application service account or a direct database user, such as a reporting user or a database administrator. In other words, you might have one SQL Firewall policy for the database user "HR" and another for the database user "JOE." This flexibility allows you to gradually build up the protection level of the database, starting from either the database administrators or the application service accounts.

SQL Firewall policy for every database account consist of two different allow-lists. An allow-list of authorized SQL statements, and an allow-list of associated trusted database connection paths. SQL Firewall's allow-lists help ensure that only authorized SQL statements from trusted database connections are permitted for execution inside the Oracle Database while alerting/blocking any unauthorized attempts to access sensitive data stored within them.

You train the SQL Firewall by capturing authorized SQL statements and trusted database connection paths for a database user. That training model is used to generate the SQL firewall policy with allow-lists from the captured SQL activities. After that, you enable the SQL firewall policy to prevent or detect potential SQL injection attacks.

SQL Firewall generates a violation log entry when it detects unauthorized activity (anything not in the allow-lists). If desired, you can configure SQL Firewall to block violating SQL traffic or connections. The process flow is captured here:

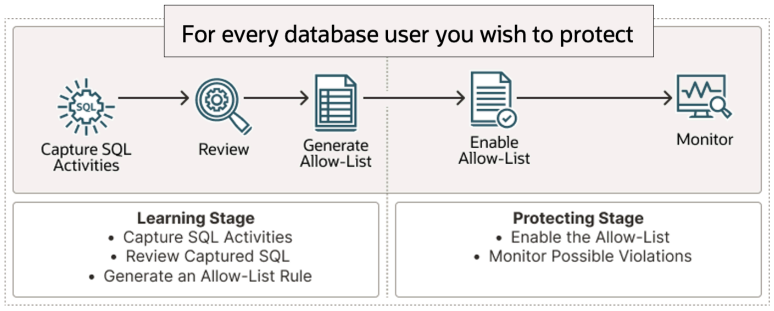


Figure 12: SQL Firewall process flow

Learning stage

Capture (or collect) SQL activities: For every database account you wish to protect with SQL Firewall, you let the SQL firewall learn the normal SQL traffic of the database user by capturing authorized SQL statements over trusted database connection paths. SQL Firewall captures the following categories of information to create the allow-lists of the firewall policy.

- Database session information—client IP address, OS program name, and the OS username
- Unique SQL statements identified by their SQL signature
- Execution context attributes:
  - Current user
  - Granularity of SQL query (top-level user-initiated statements, or SQL statements issued within PL/SQL procedures on their behalf)

SQL Firewall relies on an SQL signature to identify unique SQL statements. Every SQL statement is captured once per session, even if the statement is executed multiple times in the same session. The SQL statement is normalized, concatenated with database objects list, and subsequently hashed to generate the SQL signature.

Copyright © 2023, Oracle and/or its affiliates / Public

Review the capture (or collection): SQL Firewall lets you monitor the progress of the SQL capture to help ensure it has fully learned the application SQL traffic. Review `DBA_SQL_FIREWALL_CAPTURE_LOGS` (or Oracle Data Safe: Collection insights) and stop the collection once you are no longer capturing new unique statements.

Generate firewall policy with allow-lists: Generate the firewall policy with allow-lists that set the baseline for allowed SQL statements and allowed contexts.

Allowed SQL statements are a collection of unique combination pairs of <SQL signature, execution context>. Any incoming SQL queries that have a structure syntactically similar to the SQL signature in the policy allow-list will be passed for execution if the corresponding run-time execution context also matches the one in the allow-list.

Allowed contexts represent trusted database connection paths and consist of three distinct groups—client IP addresses, OS program names, and OS usernames. They help ensure that access to your databases comes exclusively from trusted endpoints defined in the allow-lists. Firewall policy with allow-lists are shown below:

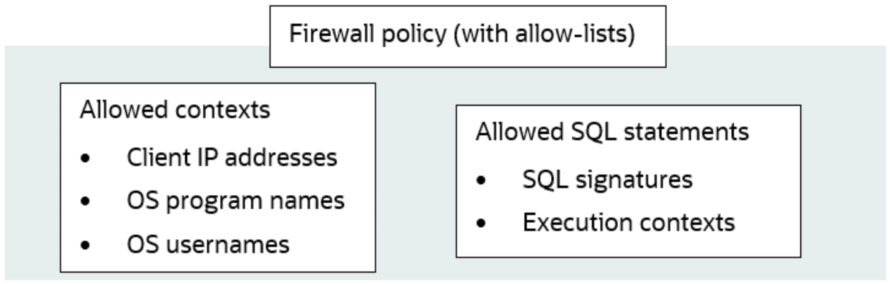


Figure 13: SQL Firewall policy

Protecting stage

Enable the firewall policy: Enable the generated firewall policy to protect the database user. The SQL Firewall enforces checks on the allow-lists when the user connects to the database and issues SQL statements. You can let the SQL Firewall know if you want to enforce checks on allowed contexts, allowed SQL statements, or both.

If the database connection paths and/or SQL statements in the incoming SQL traffic do not match the entries in the enabled and enforced allow-lists, a SQL Firewall violation is triggered and this incident is logged in the violation log. You can let the SQL Firewall know how to respond to SQL Firewall violation incident: allow the traffic to proceed to the database or block. Blocking raises ORA-47605: SQL Firewall violation, preventing anomalous database access without disrupting client connections. The significance of enforcement scope and firewall action is pictorially represented in Figure 14. SQL Firewall logs every violation and optionally blocks the database user from performing the actions.

Copyright © 2023, Oracle and/or its affiliates / Public

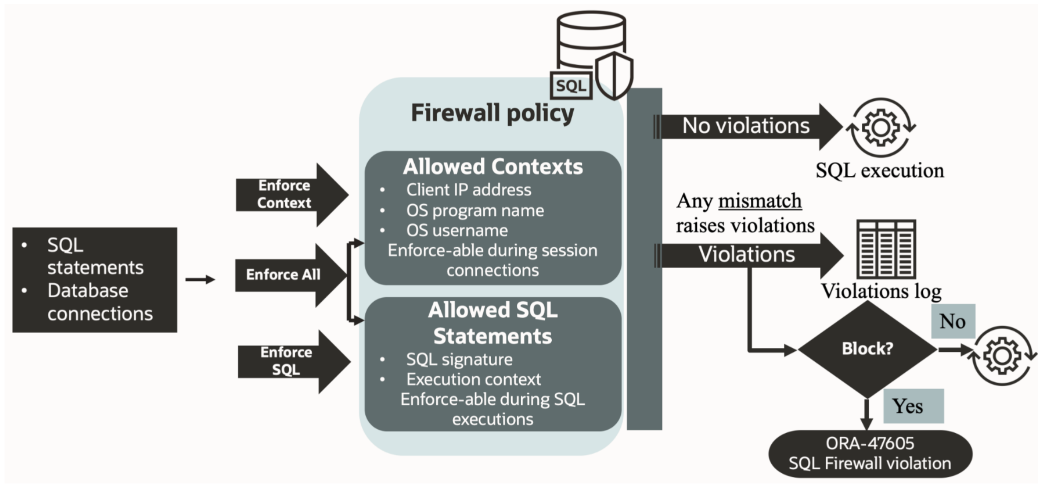


Figure 14: SQL Firewall policy enablement options: enforcement and action

SQL Firewall lets you append newer SQL statements and connection paths to the allow-lists in the enabled firewall policy; it is effective immediately.

Monitor violations: SQL Firewall raises and logs violations in real-time for every unmatched scenario of database connection or SQL command execution against the entries in the enabled allow-lists of the SQL Firewall policy. The security administrator can monitor the SQL Firewall violation log to detect the presence of these abnormalities. Data Safe can collect violation logs and visually present them in its reports. You might want to audit SQL Firewall violations (especially the blocked ones); their occurrence potentially indicates abnormal database access attempts including SQL Injection and credential theft/abuse. Auditing violations places a record of the violation in the database's audit trail, where it can be protected from tampering.

Managing SQL Firewall

You can manage the SQL Firewall in two ways:

- PL/SQL procedures in SYS.DBMS_SQL_FIREWALL package
- Data Safe

Use PL/SQL procedures in the SYS.DBMS_SQL_FIREWALL package if you wish to manage SQL Firewall within each database instance. Use Data Safe if you're looking for centralized violation reporting, UI-based management or to manage multiple SQL firewalls centrally. You can use the Data Safe REST APIs, software developer kits (SDKs), CLI, and Terraform for further automation and integration. You can also utilize the extensive Oracle Cloud Infrastructure (OCI) ecosystem for integrating SQL Firewall violations with OCI alerts and notifications.

Managing SQL Firewall with SYS.DBMS_SQL_FIREWALL package

PL/SQL procedures in SYS.DBMS_SQL_FIREWALL let you administer and manage the SQL Firewall configuration. The table below captures some common operations for configuring SQL Firewall for the sample application service account: EMPLOYEESEARCH_PROD.

Copyright © 2023, Oracle and/or its affiliates / Public

ORACLE

Table2. Important PL/SQL procedures in SYS.DBMS_SQL_FIREWALL package

Procedures	Values
Capture (or collect) SQL Activities	**DBMS_SQL_FIREWALL.CREATE_CAPTURE** • Creates (and starts) a SQL capture for the database user `DBMS_SQL_FIREWALL.CREATE_CAPTURE(‘EMPLOYEESEARCH_PROD’);`
	DBMS_SQL_FIREWALL.STOP_CAPTURE • Reviews the capture and then stops the capture `SELECT * FROM DBA_SQL_FIREWALL_CAPTURE_LOGS WHERE USERNAME = ‘EMPLOYEESEARCH_PROD’;` `DBMS_SQL_FIREWALL.STOP_CAPTURE(‘EMPLOYEESEARCH_PROD’);`
Generate firewall policy with allow-list rules	**DBMS_SQL_FIREWALL.GENERATE_ALLOW_LIST** • Generates a SQL Firewall policy with allow-list rules `DBMS_SQL_FIREWALL.GENERATE_ALLOW_LIST(‘EMPLOYEESEARCH_PROD’);`
	DBMS_SQL_FIREWALL.ADD_ALLOW_LIST/ DELETE_ALLOWED_CONTEXT • Modifies the allowed context values (i.e., client IP address, OS program name, and OS username) `DBMS_SQL_FIREWALL.ADD_ALLOWED_CONTEXT(USERNAME => ‘EMPLOYEESEARCH_PROD’, CONTEXT_TYPE => SYS.DBMS_SQL_FIREWALL.IP_ADDRESS, VALUE => ’10.0.0.0/24’);`
	DBMS_SQL_FIREWALL.APPEND_ALLOW_LIST • Appends newer SQL statements to existing allow-lists following application updates from capture log /violation log, and the changes will be effective immediately `DBMS_SQL_FIREWALL.APPEND_ALLOW_LIST(‘EMPLOYEESEARCH_PROD’, SYS.DBMS_SQL_FIREWALL.CAPTURE_LOG);`
Enable the firewall policy	**DBMS_SQL_FIREWALL.ENABLE_ALLOW_LIST** • Enables the SQL firewall policy for the user, and the change will be effective immediately. Choose your enforcement option (contexts/ SQL statements/both) and let the SQL Firewall know how to respond in case of mismatch (allow/block) while logging any violation. `DBMS_SQL_FIREWALL.ENABLE_ALLOW_LIST(USERNAME => ‘EMPLOYEESEARCH_PROD’, ENFORCE => SYS.DBMS_SQL_FIREWALL.ENFORCE_ALL, BLOCK => FALSE);`
Export/Import allow-list	**DBMS_SQL_FIREWALL.EXPORT_ALLOW_LIST/ IMPORT_ALLOW_LIST** • Exports (or imports) the firewall policy with an allow-list for the database user in JSON format. `DBMS_SQL_FIREWALL.EXPORT_ALLOW_LIST(USERNAME => ‘EMPLOYEESEARCH_PROD’, ALLOW_LIST => POLICY_CLOB)`

Copyright © 2023, Oracle and/or its affiliates / Public

SQL Firewall violation events are generated and written into the violations log (`DBA_SQL_FIREWALL_VIOLATIONS`) for every unmatched scenario of database connections and/or SQL statements execution. If the SQL Firewall policy is enabled in blocking mode, clients will receive the following ORA error when they attempt an action that is not permitted by the policy:

```
ORA-47605  SQL FIREWALL VIOLATION
```

You have two options to monitor SQL Firewall violations:

1. Read from the SQL Firewall violation log `DBA_SQL_FIREWALL_VIOLATIONS`.
2. Audit SQL Firewall violations of the database user by configuring audit policy for SQL Firewall, and read from audit trail `UNIFIED_AUDIT_TRAIL`.

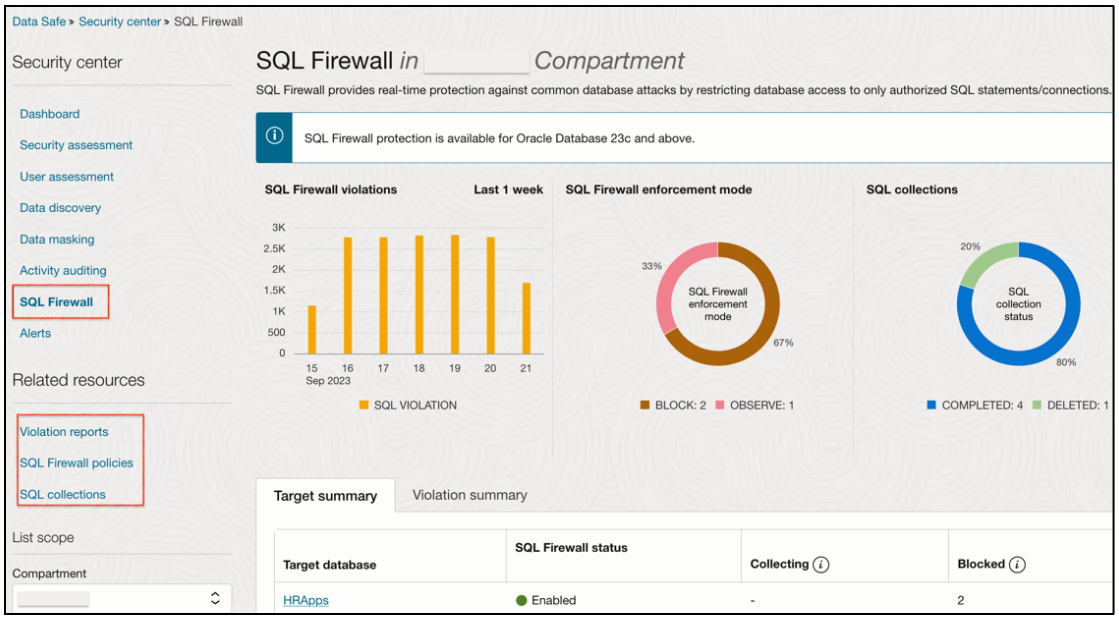
```
CREATE AUDIT POLICY EMPSEARCH_APPLICATION_AUDIT_POLICY ACTIONS COMPONENT =
SQL_FIREWALL ALL ON EMPLOYEESEARCH_PROD;

AUDIT POLICY EMPSEARCH_APPLICATION_AUDIT_POLICY;
```

Auditing of SQL Firewall enables tracking anomalous database activity of the user that has drifted from the typical expected access connection paths or approved SQL statements and can help demonstrate compliance. SQL Firewall violations are populated into UNIFIED_AUDIT_TRAIL with all contextual information, and the existing audit trail can be leveraged to monitor them from upstream applications like AVDF or Data Safe.

Managing SQL Firewall with Data Safe

With Data Safe you can manage multiple SQL firewalls centrally and get a comprehensive view of SQL Firewall violations across a fleet of Oracle databases as shown below. SQL Firewall administrators can use Data Safe to collect SQL activities of a database user with its associated database connection paths (IP address, OS program, OS user), and monitor the progress of the collection. Data Safe lets you generate and enable the SQL Firewall policy from the collected SQL traffic. Data Safe automatically collects the violation logs, and lets you monitor SQL Firewall violations from the console.



SQL Firewall *in* ________ *Compartment*

SQL Firewall provides real-time protection against common database attacks by restricting database access to only authorized SQL statements/connections.

SQL Firewall protection is available for Oracle Database 23c and above.

Target database	SQL Firewall status	Collecting	Blocked
HRApps	● Enabled	-	2

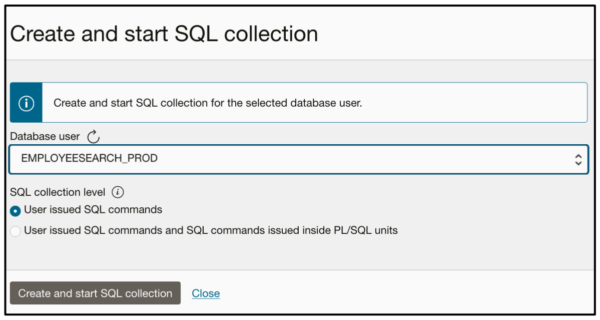
Figure 15: Data Safe: SQL Firewall dashboard

Copyright © 2023, Oracle and/or its affiliates / Public

The violation summary in the dashboard provides a comprehensive view of SQL Firewall violations from all the targets in the compartment that have SQL Firewall enabled for the chosen period. You can drill down into the violations for detailed analysis. A summary of SQL collection status and Firewall policy enforcement status for database users protected by SQL Firewall is also shown across the database estate. You can modify the SQL Firewall configuration for a target on drill down. Let us take a look into the workflow in Data Safe to configure SQL Firewall for the sample application service account: EMPLOYEESEARCH_PROD.

Select the database user from the dropdown list to create and start a collection of authorized SQL traffic for that user as shown here.

Once the collection starts, run the expected SQL application workload of the database user from all trusted database connection paths.


Create and start SQL collection

Create and start SQL collection for the selected database user.

Database user: EMPLOYEESEARCH_PROD

SQL collection level
[✓ User issued SQL commands]
User issued SQL commands and SQL commands issued inside PL/SQL units

Figure 16: Data Safe: create and start collection

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

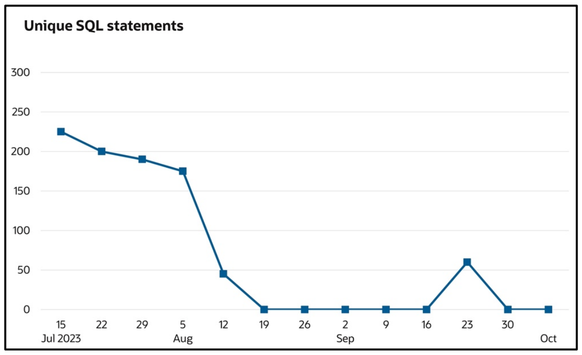
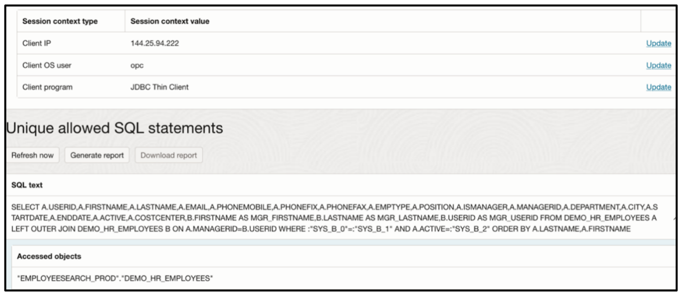


Figure 17: Data Safe: SQL collection insights

You can review the contents of allow-lists (allowed contexts and allowed SQL statements) within SQL Firewall policy in Data Safe as shown here.
You can modify allowed contexts to add new IP addresses, OS usernames and OR program names if required. You can generate a report of allowed SQL statements, along with a list of the database objects those SQL statements accessed for offline validation or change management.



Session context type	Session context value	
Client IP	144.25.94.222	Update
Client OS user	opc	Update
Client program	JDBC Thin Client	Update

Unique allowed SQL statements

SQL text

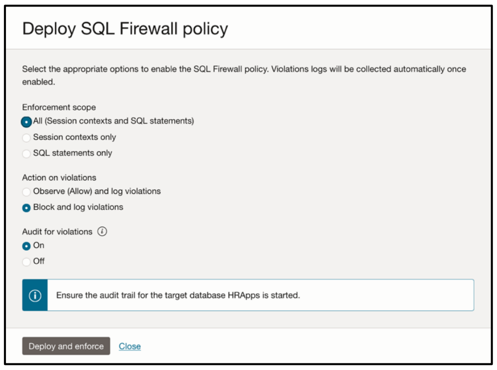
```
SELECT A.USERID,A.FIRSTNAME,A.LASTNAME,A.EMAIL,A.PHONEMOBILE,A.PHONEFIX,A.PHONEFAX,A.EMPTYPE,A.POSITION,A.ISMANAGER,A.MANAGERID,A.DEPARTMENT,A.CITY,A.S
TARTDATE,A.ENDDATE,A.ACTIVE,A.COSTCENTER,B.FIRSTNAME AS MGR_FIRSTNAME,B.LASTNAME AS MGR_LASTNAME,B.USERID AS MGR_USERID FROM DEMO_HR_EMPLOYEES A
LEFT OUTER JOIN DEMO_HR_EMPLOYEES B ON A.MANAGERID=B.USERID WHERE :"SYS_B_0"=:"SYS_B_1" AND A.ACTIVE=:"SYS_B_2" ORDER BY A.LASTNAME,A.FIRSTNAME
```

Accessed objects

"EMPLOYEESEARCH_PROD"."DEMO_HR_EMPLOYEES"

Figure 18: Data Safe: SQL Firewall policy with allow-lists

Data Safe lets you enable the SQL Firewall policy on a target with appropriate values for enforcement scope, action (observe or block) while logging, and audit as shown here.


Deploy SQL Firewall policy

Select the appropriate options to enable the SQL Firewall policy. Violations logs will be collected automatically once enabled.

Enforcement scope
[✓ All (Session contexts and SQL statements)]
Session contexts only
SQL statements only

Action on violations
Observe (Allow) and log violations
[✓ Block and log violations]

Audit for violations
[✓ On]
Off

Ensure the audit trail for the target database HRApps is started.

Figure 19: Data Safe: enabling SQL Firewall policy

Once the SQL Firewall policy is enabled, Data Safe automatically collects violations from the target's violation log and stores them in Data Safe's repository for 12 months. SQL Firewall violations stored in Data Safe are available for online analysis and reporting across your database fleet as shown here:

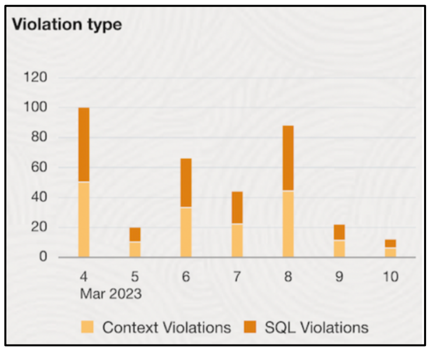
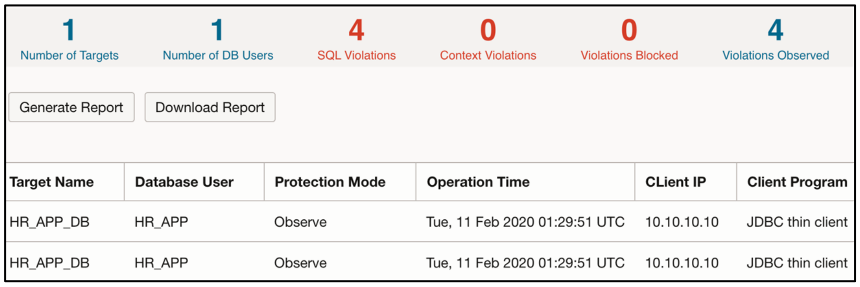


Figure 20: Data Safe: SQL Firewall violations (fleet view)



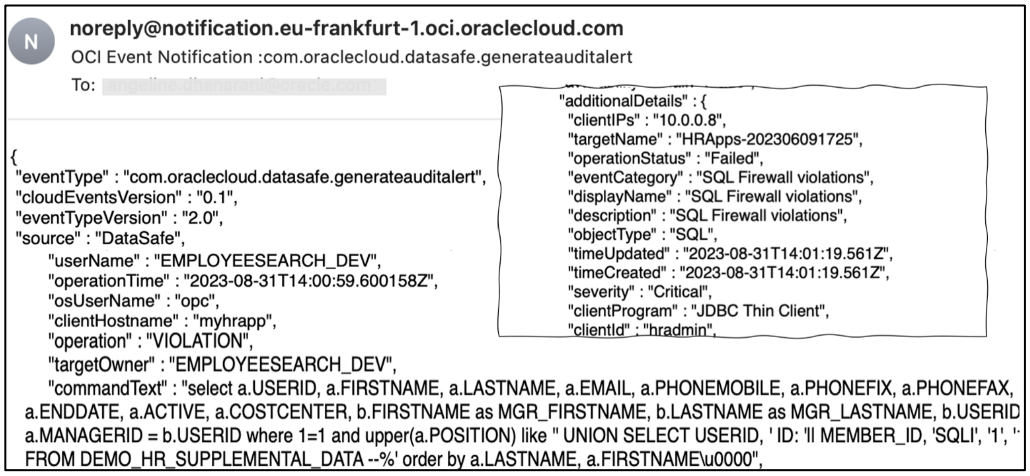
1	1	4	0	0	4
Number of Targets	Number of DB Users	SQL Violations	Context Violations	Violations Blocked	Violations Observed

Target Name	Database User	Protection Mode	Operation Time	CLient IP	Client Program
HR_APP_DB	HR_APP	Observe	Tue, 11 Feb 2020 01:29:51 UTC	10.10.10.10	JDBC thin client
HR_APP_DB	HR_APP	Observe	Tue, 11 Feb 2020 01:29:51 UTC	10.10.10.10	JDBC thin client

Figure 21: Data Safe: SQL Firewall violation reports

Data Safe lets you alert on audited SQL Firewall violations which you can further integrate with larger OCI ecosystem like OCI Events to pro-actively notify you by email or send to SIEM solutions. If you want to take advantage of alerts don't forget to turn on audit when you enable a SQL Firewall policy and start the unified audit trail for the target to let Data Safe collect the audited violations. Associate the SQL Firewall violations alert policy to the target.

Here, a sample OCI email notification proactively notifying the security administrator of potential SQL Injection attempts is shown below with all the relevant contextual details, including client connection, and the SQL command that triggered the violation.



noreply@notification.eu-frankfurt-1.oci.oraclecloud.com
OCI Event Notification :com.oraclecloud.datasafe.generateauditalert
To: [redacted]

```
{
 "eventType" : "com.oraclecloud.datasafe.generateauditalert",
 "cloudEventsVersion" : "0.1",
 "eventTypeVersion" : "2.0",
 "source" : "DataSafe",
      "userName" : "EMPLOYEESEARCH_DEV",
      "operationTime" : "2023-08-31T14:00:59.600158Z",
      "osUserName" : "opc",
      "clientHostname" : "myhrapp",
      "operation" : "VIOLATION",
      "targetOwner" : "EMPLOYEESEARCH_DEV",
      "commandText" : "select a.USERID, a.FIRSTNAME, a.LASTNAME, a.EMAIL, a.PHONEMOBILE, a.PHONEFIX, a.PHONEFAX,
  a.ENDDATE, a.ACTIVE, a.COSTCENTER, b.FIRSTNAME as MGR_FIRSTNAME, b.LASTNAME as MGR_LASTNAME, b.USERID
  a.MANAGERID = b.USERID where 1=1 and upper(a.POSITION) like '' UNION SELECT USERID, ' ID: '|| MEMBER_ID, 'SQLI', '1', '
  FROM DEMO_HR_SUPPLEMENTAL_DATA --%' order by a.LASTNAME, a.FIRSTNAME\u0000",
```

```
      "additionalDetails" : {
        "clientIPs" : "10.0.0.8",
        "targetName" : "HRApps-202306091725",
        "operationStatus" : "Failed",
        "eventCategory" : "SQL Firewall violations",
        "displayName" : "SQL Firewall violations",
        "description" : "SQL Firewall violations",
        "objectType" : "SQL",
        "timeUpdated" : "2023-08-31T14:01:19.561Z",
        "timeCreated" : "2023-08-31T14:01:19.561Z",
        "severity" : "Critical",
        "clientProgram" : "JDBC Thin Client",
        "clientId" : "hradmin",
```

Figure 22: Data Safe: Sample email notification on potential SQL Injection attempt

Copyright © 2023, Oracle and/or its affiliates / Public

Deciding which to use: Database Firewall or SQL Firewall

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

- **Performance Impact:** A network-based Database Firewall has no performance impact on the database server- it consumes no RAM, no additional CPU because all work is done off-server on the Database Firewall machine. For most deployment modes, Database Firewall introduces zero performance overhead. In proxy mode, the Database Firewall introduces minor additional network latency, with the degree of latency primarily controlled by the position of the Database Firewall on the network. SQL Firewall introduces no observable latency but does create minimal CPU overhead, not exceeding 1% to 1.5%.
- **Deployment Options:** Database Firewall is deployed on a dedicated virtual machine or hardware (separate from the database server). Database Firewall has multiple deployment options: Proxy, Host Monitor, and Out-of-Band. Proxy is the only inline configuration that supports blocking use cases, but using the proxy requires client-side connection changes to route traffic through the proxy. SQL Firewall is part of the database, requiring no client-side changes. SQL Firewall is the best solution for deployments where blocking is the predominant use case or if it is impractical or undesirable to change client-side configurations.
- **Scope of inspection:** Database Firewall can only monitor SQL traffic coming over the network. It lacks visibility into SQL traffic over local BEQ connections and is unaware of internal jobs/ stored procedure executions. The SQL Firewall runs inside the database and sees ALL traffic, regardless of the origination point.
- **Processing of encrypted traffic:** Handling encrypted SQL traffic requires additional configuration and processing in the Database Firewall but is seamless and straightforward in the SQL Firewall because the traffic has already been decrypted by the time it reaches the SQL Firewall.
- **Disruption in blocking:** Blocking SQL on the network (with Database Firewall) may disrupt transactions because each operation is handled individually as a stand-alone statement. In contrast, SQL Firewall understands database transactions and honors atomicity, consistency, isolation, and durability.
- **High Availability:** High availability for the Database Firewall is provided by redundant firewalls, with clients distributing connects/failing over based on client connect parameters or using a third-party load balancer. SQL Firewall is part of Oracle Database and takes advantage of high availability architectures like Real Application Clusters and Data Guard.
- **Separation of Duties:** Database Firewall can monitor database traffic without any input or control by the database administrators. SQL Firewall is part of Oracle Database and requires cooperation with the database administrators to deploy.

Summary

SQL injection is the most common database attack pattern for data-driven web applications. To mitigate the risk for Oracle Database, leverage Database Firewall in AVDF or SQL Firewall in Oracle Database. Neither relies on heuristic regex pattern matches typically used by most network firewall solutions. Instead, they rely on actual SQL signature and construct an allowed list of SQL statements. They combine allowed SQL with session context to learn the normal application SQL traffic and detect and alert/block the violations in real-time.

Copyright © 2023, Oracle and/or its affiliates / Public

Connect with us

Call **+1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle

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

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