Oracle 23ai SQL-Firewall – Datensicherheit in der Datenbank

Alexander Giesbrecht, Logicalis

Datensicherheit ist ein sensibles und wichtiges Thema in der heutigen Zeit. Ziel vieler Cyberangriffe sind die Daten von Unternehmen. Die Datenbank ist meist der zentrale Speicherort von Daten und mittels SQL werden die Daten von der Datenbank abgefragt. Daher lohnt sich der Blick auf Datensicherheit in der Datenbank. Oracle hat in die Datenbank 23ai eine SQL-Firewall integriert, um Angriffe mittels SQL-Injection, Abfragen von kompromittierten Accounts oder auch gefährliche Abfragen von privilegierten Benutzern zu erkennen und zu verhindern. Dieser Artikel befasst sich mit der grundlegenden Funktion und Steuerung der SQL-Firewall in der Oracle Database 23ai mittels PL/SQL-Package und grafisch über DataSafe.

Mit der Version Oracle 23ai kommt das Feature SQL-Firewall, um die Datensicherheit in der Datenbank zu erhöhen und Angriffe, wie SQL-Injections und die Ausnutzung von kompromittierten Accounts, zu erkennen und abwehren zu können *[1]*. Es ist in den Kernel der Datenbank integriert und benötigt somit keine zusätzliche Software-Installation. Die SQL-Firewall ist in der Enterprise Edition verfügbar und kann dort lizenziert und genutzt werden *[2]*. Es ist ebenfalls in den OCI-Datenbankangeboten wie Autonomous Database, ExaDB, ExaCC und BaseDB (EE-HighPerformance & EE-ExtremePerformance) inkludiert. Ausgenommen sind die BaseDB-Angebote für Standard Edition und Enterprise Edition in der Base- Performance-Ausprägung. Die SQL-Firewall kann auch in der Oracle 23ai Free genutzt werden.

Die Nutzung der SQL-Firewall muss jedoch lizenziert werden. Die Lizenz dafür ist in den beiden Lizenzpaketen „Oracle Database Vault“ und „Audit Vault and Database Firewall“ (AVDF) abgedeckt. In den OCI-Angeboten, in denen Database Vault inklusive ist, kann die SQL-Firewall somit genutzt werden.

Im Gegensatz zu „Audit Vault and Database Firewall“ (AVDF) ist die SQL-Firewall kein separates Produkt. AVDF ist eine zentrale Netzwerk-Lösung, welche sowohl Oracle als auch non-Oracle-Datenbanken überwachen kann. Dabei kann es auch mehrere Datenbanken gleichzeitig überwachen. Die SQL-Firewall überwacht nur die Datenbank, in der sie aktiviert ist. Dies vermeidet zusätzlichen Netzwerkload.

Funktionsweise

Wird die SQL-Firewall aktiviert, kann sie Anfragen an die Datenbank in zwei Kategorien überprüfen. Zum einen wird der Kontext der Ausführung überprüft. Dies beinhaltet Informationen über die Quelle der Anfrage wie IP-Adresse, OS-Benutzername und Programm. Zum anderen wird das SQL-Statement überprüft. Anhand der Policy für ein Schema oder für Benutzer in der Datenbank wird festgelegt, welcher Kontext und/oder welche SQLs erlaubt sein sollen. Wenn die Kriterien der Anfrage erfüllt sind, wird das SQL ausgeführt *(siehe Abbildung 1)*.

Beinhaltet die Anfrage Kriterien, die nicht erlaubt sind, wird die Anfrage entweder ausgeführt und in ein Violation Log geschrieben, oder die Anfrage wird geblockt und ebenfalls in das Violation Log geschrieben. Ob die nicht erwünschten Anfragen zugelassen oder geblockt werden sollen, wird bei der Erstellung der Policy eingestellt, kann aber jederzeit geändert werden. Das Violation Log kann in der View `DBA_SQL_FIREWALL_VIOLATIONS` eingesehen werden. Wird die Anfrage geblockt, erscheint die Fehlermeldung „ORA-47605: SQL Firewall violation“. Zusätzlich können die SQL-Firewall-Aktivitäten auch in das Audit Trail der Datenbank geschrieben werden.

Um eine Firewall Policy zu erstellen, sind folgende Schritte notwendig, die später noch genauer erläutert werden:

1. Aktivieren der SQL-Firewall
2. Erstellen einer Capture für einen User
3. Starten der Capture
4. Während der Capture den später erlaubten SQL-Workload ausführen
5. Stoppen der Capture und Erstellen einer Allow List aus der Capture
6. Aktivieren der Allow List

Das „Capture“, also das Erfassen des gewöhnlichen Workloads auf der Datenbank, muss nicht für jede Datenbank neu gemacht werden. Die Allow Lists können exportiert und in andere Datenbanken

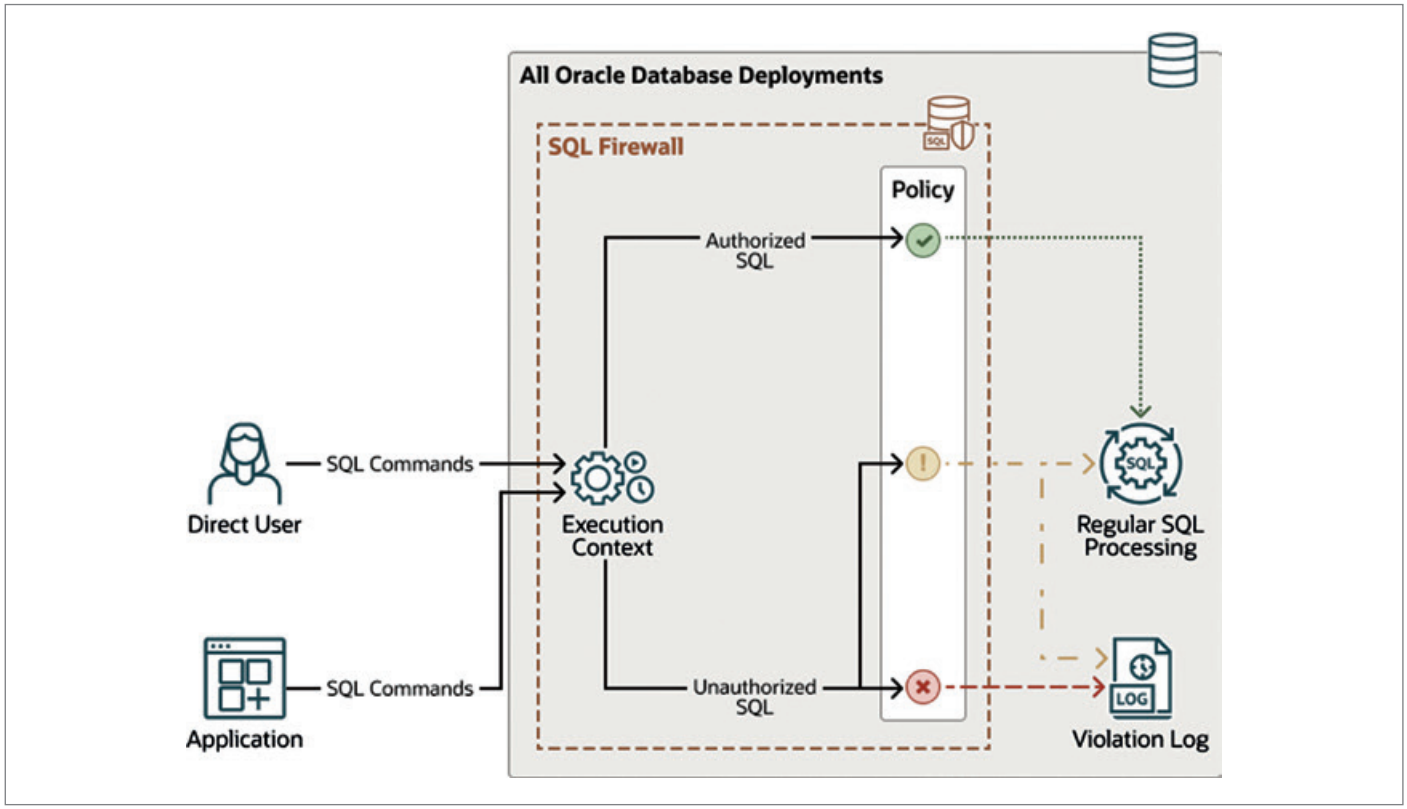

Abbildung 1: Funktionsweise der SQL-Firewall [3] (Quelle: Alexander Giesbrecht)

```
BEGIN
  DBMS_SQL_FIREWALL.CREATE_CAPTURE (
    username => 'APP_USER',
    top_level_only => TRUE,
    start_capture => FALSE );
END;
/
```

Listing 1: Anlegen einer Capture

```
DBA_SQL_FIREWALL_ALLOW_LISTS      → Liste der Allow lists
DBA_SQL_FIREWALL_ALLOWED_SQL      → Erlaubte SQL Pattern
DBA_SQL_FIREWALL_ALLOWED_OS_PROG  → Erlaubte Programme
DBA_SQL_FIREWALL_ALLOWED_OS_USER  → Erlaubte OS-User
DBA_SQL_FIREWALL_ALLOWED_IP_ADDR  → Erlaubte IP-Adresse/-
Adressbereich
```

Listing 2: Views für die Firewall Policy

```
BEGIN
  DBMS_SQL_FIREWALL.ENABLE_ALLOW_LIST (
    username => 'APP_USER',
    enforce => DBMS_SQL_FIREWALL.ENFORCE_SQL,
    block => TRUE
    );
END;
/
```

Listing 3: Aktivieren der Allow List

```
SQL> select * from user_tables;
select * from user_tables
              *
ERROR at line 1:
ORA-47605: SQL Firewall violation
Help: https://docs.oracle.com/error-help/db/ora-47605/
```

Listing 4: Fehlermeldung bei aktivierter Allow List und block=TRUE

```
execute bestellsum('123456');
select count(*) from bestellungen where kundennr='123456';
select kundennr, artikel, preis, menge from bestellungen where
kundennr='123456';
```

Listing 5: SQLs während der Aufzeichnung

```
@datasafe_privileges.sql DATASAFE_ADMIN GRANT ALL -VERBOSE
```

Listing 6: datasafe_privileges.sql-Aufruf

importiert werden. Bei einer Umgebung mit Testdatenbanken können somit Daten aus einer Testphase in der Testumgebung in die Produktionsdatenbank übernommen werden. Auch können dann Daten der Policy wie die erlaubten IP-Adressbereiche angepasst und geändert werden.

Die SQL-Firewall kann auf zwei Wegen administriert werden. Zum einen mittels PL/SQL-Package oder in der OCI mit Data-Safe auf grafischer Ebene.

Nutzung als PL/SQL-Package

Das Package DBMS_SQL_FIREWALL beinhaltet die Funktionen für die Administration der SQL-Firewall *[4]*. Um die SQL-Firewall zu aktivieren, wird der Befehl `EXEC DBMS_SQL_FIREWALL.ENABLE;` ausgeführt. Damit ist die SQL- Firewall aktiviert und kann konfiguriert werden. Die SQL-Firewall lässt sich sowohl auf PDB- oder auf CDB-Level aktivieren, je nachdem welches Schema überwacht werden soll. Das Erstellen einer Capture erfolgt wie in *Listing 1*.

Dabei wird der Name des zu überwachenden Datenbankbenutzers angegeben. Zusätzlich kann mit top_level_only angegeben werden, ob auch SQLs innerhalb von Functions aufgezeichnet werden sollen oder nur der Aufruf von Functions. Standardmäßig wird die Aufzeichnung gleich gestartet, was als optionaler Parameter jedoch verhindert werden kann. Soll das Capturing später gestartet werden, kann es mit `DBMS_SQL_FIREWALL.START_CAPTURE('APP_USER')` gestartet werden. Ab dann werden für alle Verbindungen des Users APP_USER der Kontext und die ausgeführten SQLs aufgezeichnet, bis die Aufzeichnung mit `DBMS_SQL_FIREWALL.STOP_CAPTURE('APP_USER')` wieder gestoppt wird. Während der Aufzeichnung kann der umfangreiche Applikationstest laufen, bei dem möglichst alle später relevanten Use Cases und Workloads ausgeführt werden sollten. Die aufgezeichneten Daten können während und nach der Aufzeichnung in den Views `DBA_SQL_FIREWALL_CAPTURE_LOGS`, für die ausgeführten SQLs, und `DBA_SQL_FIREWALL_SESSION_LOGS`, für die Kontextinformationen der Verbindungen, eingesehen werden. Nach Abschluss und

Abbildung 2: Allow List und Allowed SQL (Quelle: Alexander Giesbrecht)

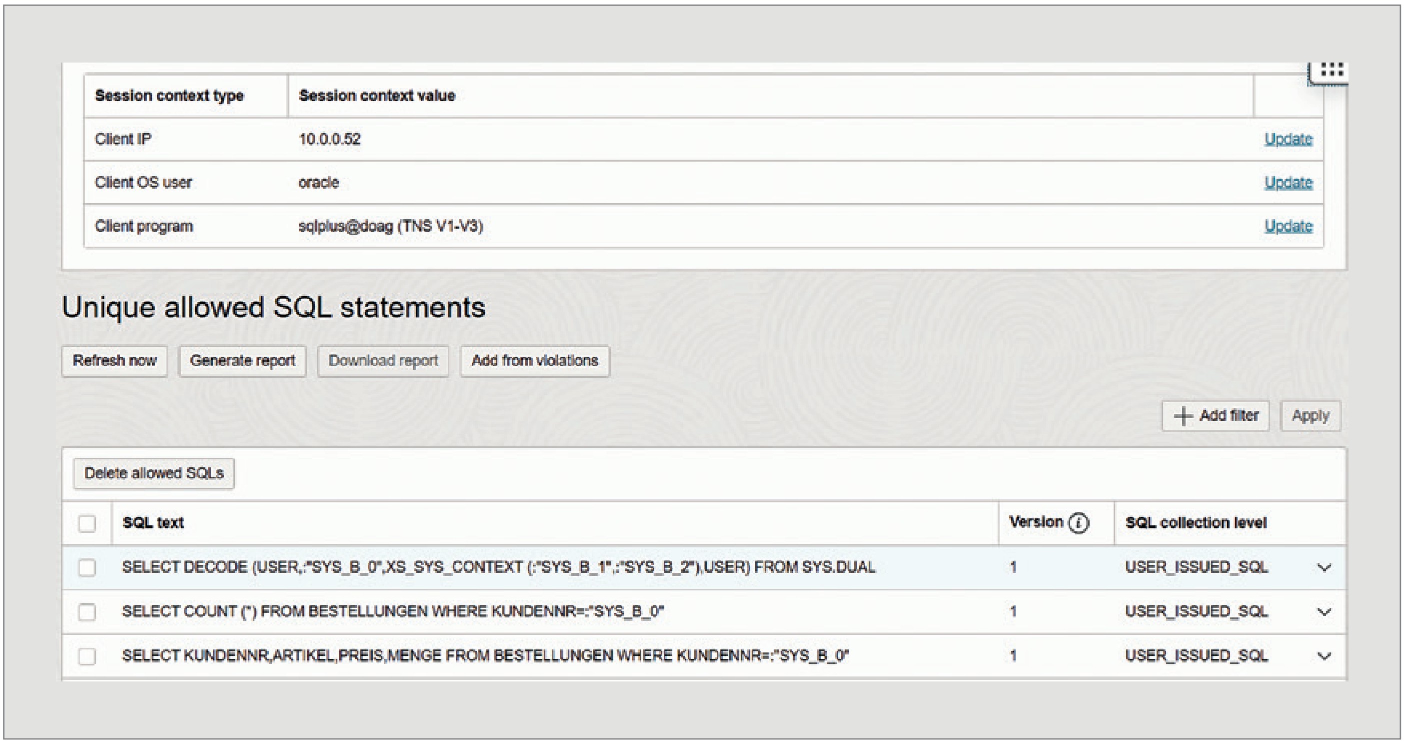

Abbildung 3: Allow List in DataSafe: Kontext und SQLs (Quelle: Alexander Giesbrecht)

Überprüfung der gesammelten Daten wird aus der Capture eine Allow List mit `DBMS_SQL_FIREWALL.GENERATE_ALLOW_LIST('APP_USER')` erstellt. Die Allow List ist die Firewall Policy, anhand welcher spätere Verbindungen und SQLs überprüft werden. In den Views aus *Listing 2* sind die Details der Allow List einsehbar.

Nach dem Erstellen der Allow List werden die Regeln noch nicht angewendet. Um die Allow List zu aktivieren, wird die Prozedur `ENABLE_ALLOW_LIST` *(siehe Listing 3)* ausgeführt. Für den Parameter „enforce" gibt es drei Möglichkeiten. Zum einen gibt es wie in *Listing 3* die Möglichkeit, nur die SQLs zu überprüfen. Ebenfalls kann mit `ENFORCE_CONTEXT` nur der Kontext der Verbindung überwacht werden. Sollen jedoch Kontext und SQL überwacht werden, kann dies mittels `ENFORCE_ALL` eingestellt werden.

Standardmäßig blockiert die Firewall beim Aktivieren der Allow List nicht unerwünschte Verbindungen, sondern protokolliert diese nur ins Violation Log. Sollen diese jedoch blockiert und protokolliert werden, kann beim Aktivieren der Allow List der Parameter `block` auf `TRUE` gesetzt werden, oder im Nachgang der Parameter mit der Prozedur `UPDATE_ALLOW_LIST_ENFORCEMENT` geändert werden. Die Allow List kann auch dann modifiziert werden, wenn diese aktiv ist. Dafür gibt es mehre Prozeduren, die es ermöglichen, Kontext oder SQLs zu entfernen oder hinzuzufügen. Ebenso kann jederzeit wieder eine Aufzeichnung gestartet werden und die Ergebnisse der Allow List können hinzugefügt werden.

Wird ein SQL ausgeführt, welches nicht in der Allow List enthalten ist, kommt es zu einer Fehlermeldung *(siehe Listing 4)*.

Praktischer Test

Um die Funktion der SQL-Firewall zu testen, wurden ein paar einfache Tests gemacht. Dabei wurde in einem Schema APP eine Tabelle „Bestellungen" mit fiktiven Beispieldaten und einer Prozedur „bestellsum" angelegt. Die Prozedur ermittelt die Summe aller Bestellungen zu einer mitgegebenen Kundennummer. Während der Capture wurden die SQLs aus *Listing 5* ausgeführt.

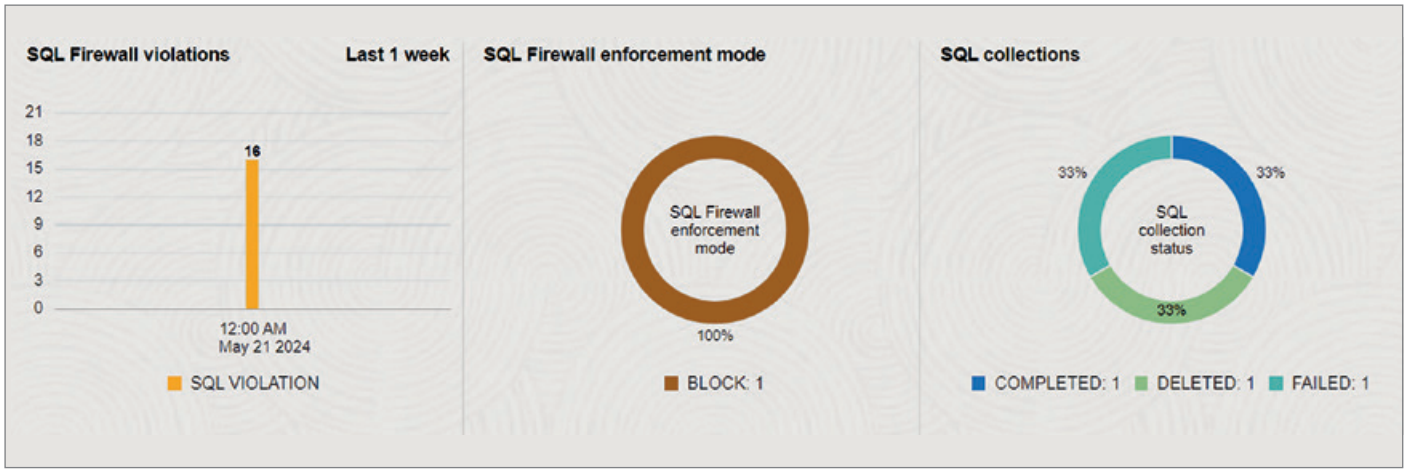

Abbildung 4: SQL Firewall Dashboad in DataSafe mit Violations (Quelle: Alexander Giesbrecht)

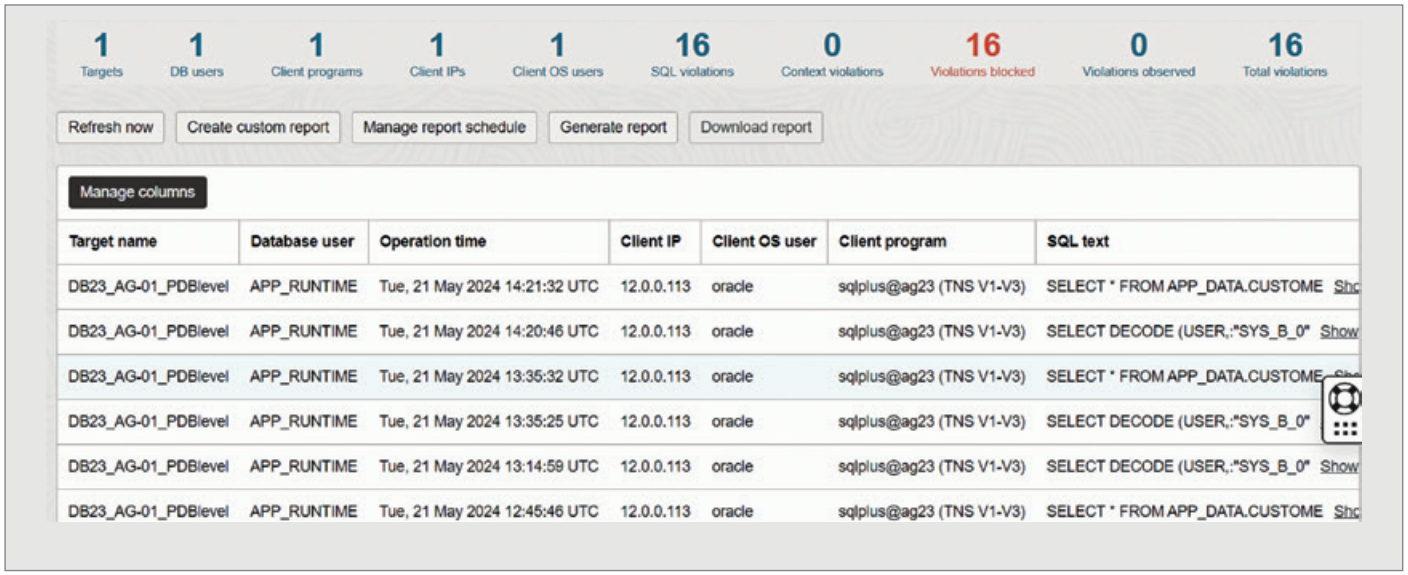

Abbildung 5: Violation Report in DataSafe (Quelle: Alexander Giesbrecht)

Die Aufzeichnung und die Allow List wurden mit *top_level_only=false* angelegt, was somit auch die Inhalte der Prozedur aufgezeichnet hat. *Abbildung 2* zeigt die SQLs der Allow List.

Bei den Tests wurde festgestellt, dass jegliche Veränderung des SQLs, wie zum Beispiel das Hinzufügen/Entfernen von Spalten oder WHERE-Bedingungen, zur Firewall Violation führt. Auch die Veränderung der Reihenfolgen der abgefragten Spalten oder das Ersetzen eines Wertes mit dem Spaltennamen wird von der SQL-Firewall abgefangen. Die Werte, wie in diesem Fall die Kundennummer, werden als Platzhalter gespeichert, was die Verwendung mit Eingabemasken, woraus eine SQL generiert wird, möglich macht. SQL-Injections können somit nicht mehr durchgeführt werden, weil das Muster der SQL-Abfrage dadurch verändert wäre und nicht mehr der Allow List entspräche.

OCI DataSafe

Die SQL-Firewall lässt sich in der OCI auch über DataSafe grafisch steuern. Dafür sind ein paar Vorarbeiten notwendig. Zum einen benötigt DataSafe einen eigenen Benutzer in der Datenbank. Dieser kann in der CDB oder PDB angelegt werden und muss dann mit notwendigen Berechtigungen versehen werden. In der Autonomous Database gibt es dafür einen vorgefertigten Benutzer DS$ADMIN mit allen notwendigen Berechtigungen. Um die Berechtigungen zu vergeben, stellt Oracle das Skript *datasafe_privileges.sql* in der OCI Console zur Verfügung, mit welchem der DataSafe-Benutzer alle notwendigen Berechtigungen erhält. Der Aufruf kann wie in *Listing 6* aussehen.

Im Anschluss muss das DataSafe Target angelegt werden. Dies kann eine PDB oder die CDB sein. Dafür gibt es mehrere Wege, am einfachsten ist jedoch der Register-Wizard in der OCI, welcher auch den notwendigen Private Endpoint anlegt und das Target registriert. Im Anschluss kann die Datenbank im DataSafe verwendet werden, der auch die SQL-Firewall beinhaltet. Der Ablauf für die Konfiguration ist identisch mit der Nutzung mittels PL/SQL-Package, jedoch können die Schritte zum Aktivieren der Firewall, das Anlegen und Starten einer Collection und das Erstellen und Bearbeiten der Allow List nun grafisch in der OCI Console vorgenommen werden *(siehe Abbildung 3)*.

Zusätzlich können die SQL- und Kontext-Violations auch grafisch angezeigt werden *(siehe Abbildung 4)*. Dies kann zusätzlich mit Notifications versehen werden, sodass bei Violations auch Benachrichtigungen für die Administratoren oder Entwickler erzeugt werden *(siehe Abbildung 5)*.

Fazit

Die SQL-Firewall ermöglicht es, die Datensicherheit der Datenbank zusätzlich zu erhöhen, indem unbekannte SQL-Muster bei SQL-Injections oder die Ausnutzung von kompromittierten Accounts über SQL und Verbindungskontext erkannt und blockiert werden können. Das Monitoring dabei ist ebenfalls möglich und wenn DataSafe verwendet wird, kann der Zuständige auch über Violations benachrichtigt werden. Jedoch ist der Einsatz der SQL-Firewall auch mit Aufwand verbunden. Zum einen muss das Testen der Applikation und des regulären Workloads auf der Datenbank möglichst detailliert und vollständig während der Aufzeichnung ausgeführt werden. Zwar kann die Allow List jederzeit angepasst werden, jedoch kann es dort zu unerwünschten Fehlern kommen, wenn in der Produktion Anwendungsfälle nicht vollständig in der Allow List enthalten sind. Außerdem muss auch immer in den Releases der Anwendung berücksichtigt werden, dass neue SQLs und vieles mehr dazu kommen können. Insgesamt bietet die SQL-Firewall einen zusätzlichen Schutz für Datenbanken mit kritischen und sensiblen Daten.

Quellen

[1] *https://blogs.oracle.com/cloudsecurity/post/sql-firewall-now-built-into-oracle-database-23c*

[2] *https://www.oracle.com/a/ocom/docs/security/oracle-SQL Firewall-faq.pdf*

[3] *https://docs.oracle.com/en/database/oracle/oracle-database/23/dbseg/using-oracle-SQL Firewall.html#GUID-B268CC0A-4FE5-4A50-9C20-FABC99B5C4AD*

[4] *https://docs.oracle.com/en/database/oracle/oracle-database/23/arpls/dbms_sql_firewall.html#GUID-4A41615C-D3D5-4386-A299-5C3D08350E8C*

Über den Autor

Alexander Giesbrecht ist Solution Engineer bei der Logicalis GmbH für Oracle-Datenbanken und -Architekturen. Während des dualen Studiums kam er mit Oracle- Datenbankadministration in Berührung und hat somit seit über 8 Jahren Erfahrungen in verschiedenen Oracle-Architekturen und -Bereichen gesammelt. Aktuell beschäftigt er sich unter anderem mit Migrationen nach Multitenant und in die OCI.

Alexander Giesbrecht
alexander.giesbrecht@logicalis.de