MySQL Enterprise Security

Introduction

MySQL Enterprise offers important security extension, we now works with some of them.

MySQL Enterprise Firewall guards against cyber security threats by providing real-time protection against database specific attacks. It protects the data by monitoring, alerting, and blocking unauthorized database activity without any changes to applications. In this lab you will work with group profiles, which are used to create a composite list of allowed queries for a group of users and enforces firewall protection across all in the group profile.

MySQL Enterprise Audit provides an easy to use, policy-based auditing solution that helps implement stronger security controls and satisfy regulatory compliance.

MySQL Enterprise Masking and De-identification provides an easy to use, built-in database solution that helps creating views that protect sensitive data from unauthorized uses by hiding and replacing real values with substitutes. Before 8.0.33 Data Masking was a plugin (deprecated), now is available as component with additional options. To use the component in existing installations, please remove the plugin and all of its loadable functions to avoid conflicts (see here).
We test here only a subset of the Data Masking functions.

Estimated Lab Time: 25 minutes

Objectives

In this lab, you will work with:

- Enterprise Firewall
- Enterprise Audit
- Enterprise Data Masking

> **Note:**

- Server: mysql1

Task 1: MySQL Enterprise Firewall

1. If not already connected, connect to mysql1 through app-srv

   ```
   <span style="color:green">shell-app-srv$</span> <copy>ssh -i
   $HOME/sshkeys/id_rsa_mysql1 opc@mysql1</copy>
   ```

2. Install MySQL Enterprise Firewall on mysql-advanced using CLI

   ```
   <span style="color:green">shell-mysql1></span> <copy>mysqlsh
   admin@mysql1:3307 -D mysql < /mysql/mysql-
   ```

```
latest/share/linux_install_firewall.sql</copy>
```

3. Connect to the instance with administrative account first SSH connection - administrative

```
<span style="color:green">shell-mysql1></span> <copy>mysqlsh
admin@mysql1:3307</copy>
```

4. Check if firewall is enabled

```
<span style="color:blue">mysql></span> <copy>SHOW GLOBAL VARIABLES LIKE
'mysql_firewall_mode';</copy>
```

5. Create a new user 'fwtest' and assign full privileges to database world

```
<span style="color:blue">mysql></span> <copy>CREATE USER 'fwtest'@'%'
IDENTIFIED BY 'Welcome1!';</copy>
```

```
<span style="color:blue">mysql></span> <copy>GRANT ALL PRIVILEGES ON world.*
TO 'fwtest'@'%';</copy>
```

6. Grant firewall management privilege to admin and exclude it from firewall checks

```
<span style="color:blue">mysql></span> <copy>GRANT FIREWALL_ADMIN,
FIREWALL_EXEMPT on *.* to admin@'%';</copy>
```

7. Create a firewall group profile (called 'fwgrp') and assign fwtest user

```
<span style="color:blue">mysql></span> <copy>CALL
mysql.sp_firewall_group_enlist('fwgrp', 'fwtest@%');</copy>
```

8. Set the new group in recording mode

```
<span style="color:blue">mysql></span> <copy>CALL
mysql.sp_set_firewall_group_mode('fwgrp', 'RECORDING');</copy>
```

9. Open a second SSH connection and use it to connect as “fwtest” from app-srv

```
<span style="color:green">shell-mysql1></span> <copy>mysqlsh
fwtest@mysql1:3307</copy>
```

```
<span style="color:blue">mysql></span> <copy>USE world;</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT * FROM city limit 25;
</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT Code, Name, Region FROM
country WHERE population > 200000;</copy>
```

10. Administrative connection: Return to admin session (first SSH connection terminal) and verify that there are now rules in allowlist

```
<span style="color:blue">mysql></span> <copy>SELECT RULE FROM
performance_schema.firewall_group_allowlist WHERE NAME = 'fwgrp';</copy>
```

11. Administrative connection: switch Firewall mode from **'recording'** to **'protecting'** and verify the presence of rules in allowlist

```
<span style="color:blue">mysql></span> <copy>CALL
mysql.sp_set_firewall_group_mode('fwgrp', 'PROTECTING');</copy>
```

12. fwtest connection: run these commands. Which one’s work? Which ones fail and why?

```
<span style="color:blue">mysql></span> <copy>USE world;</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT * FROM city limit 25;
</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT Code, Name, Region FROM
country WHERE population > 200000;</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT * FROM countrylanguage;
</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT Code, Name, Region FROM
country WHERE population > 500000;</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT Code, Name, Region FROM
country WHERE population > 200000 or 1=1;</copy>
```

13. Administrative connection: Set the firewall to **detecting** mode now.

```
<span style="color:blue">mysql></span> <copy>CALL
mysql.sp_set_firewall_group_mode('fwgrp', 'DETECTING');</copy>
```

14. Firewall messages require an increase in the default log level.

```
<span style="color:blue">mysql></span> <copy>SET PERSIST
log_error_verbosity=3;</copy>
```

15. fwtest connection: Repeat a blocked command (it works? Why?)

```
<span style="color:blue">mysql></span> <copy>SELECT Code, Name, Region FROM
country WHERE population > 200000 or 1=1;</copy>
```

16. Administrative connection: Now exit from administrative session on mysql1 and search the error in the error log.

```
<span style="color:blue">mysql></span> <copy>\q</copy>
```

```
<span style="color:green">shell-mysql1></span> <copy>grep "MY-011191"
/mysql/log/err_log.log</copy>
```

17. Administrative connection: Error log can also be interrogated also from the client.

```
<span style="color:green">shell-mysql1></span> <copy>mysqlsh
admin@mysql1:3307</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT * FROM
performance_schema.error_log WHERE ERROR_CODE='MY-011191';</copy>
```

18. Administrative connection: Disable now the firewall and exit from the client

```
<span style="color:blue">mysql></span> <copy>CALL
mysql.sp_set_firewall_mode('fwtest@%', 'OFF');</copy>
```

```
<span style="color:blue">mysql></span> <copy>\q</copy>
```

Task 2: MySQL Enterprise Audit

1. If not already connected, connect to mysql1 through app-srv
2. Enable Audit using the pre-configured script in the share directory of your MySQL installation, and verify that the script return the message "OK"

```
<span style="color:green">shell-mysql1></span> <copy>mysqlsh
admin@mysql1:3307 -D mysql < /mysql/mysql-
latest/share/audit_log_filter_linux_install.sql</copy>
```

3. Connect to the instance with administrative account

```
<span style="color:green">shell-mysql1></span> <copy>mysqlsh
admin@mysql1:3307</copy>
```

4. Verify that the Audit plugin is loaded and active. If it's not active, repeat previous step or ask help to the instructor

```
<span style="color:blue">mysql></span> <copy>SELECT PLUGIN_NAME,
PLUGIN_STATUS FROM INFORMATION_SCHEMA.PLUGINS WHERE PLUGIN_NAME LIKE
'audit%';</copy>
```

5. By default, rule-based audit log filtering logs no auditable events for any users. To log all auditable events for all users, activate these filters

```
<span style="color:blue">mysql></span> <copy>SET PERSIST
audit_log_rotate_on_size=20971520;</copy>
```

6. Activate the firewall rules Various variables can be set for log management. As minimal configuration, set now automatic rotation at 20MB, or the audit.log becomes soon not manageable

```
<span style="color:blue">mysql></span> <copy>SELECT
audit_log_filter_set_filter('log_all', '{ "filter": { "log": true } }');
</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT
audit_log_filter_set_user('%', 'log_all');</copy>
```

7. Exit from admin connection and login with the user “appuser2” to submit some commands

```
<span style="color:blue">mysql></span> <copy>\q</copy>
```

```
<span style="color:green">shell-mysql1></span> <copy>mysqlsh
appuser2@mysql1:3307</copy>
```

```
<span style="color:blue">mysql></span> <copy>USE world;</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT * FROM city limit 25;
</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT Code, Name, Region FROM
country WHERE population > 200000;</copy>
```

```
<span style="color:blue">mysql></span> <copy>\q</copy>
```

8. Open the audit.log file in the datadir and verify the content

```
<span style="color:green">shell-mysql1></span> <copy>nano
/mysql/data/audit.log</copy>
```

9. For additional and more complex rules, you can check the documentation

Task 3: MySQL Enterprise Data Masking

1. Install the Data Masking components

    - Login to mysql instance with administrative

    ```
    <span style="color:green">shell></span> <copy>mysqlsh
    admin@mysql1:3307</copy>
    ```

    - Create the masking_dictionaries table

    ```
    <span style="color:blue">mysql></span> <copy>CREATE TABLE IF NOT EXISTS
        mysql.masking_dictionaries(
        Dictionary VARCHAR(256) NOT NULL,
        Term VARCHAR(256) NOT NULL,
        UNIQUE INDEX dictionary_term_idx (Dictionary, Term),
        INDEX dictionary_idx (Dictionary)
        ) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;</copy>
    ```

    - Load and install the components

    ```
    <span style="color:blue">mysql></span> <copy>INSTALL COMPONENT
    'file://component_masking';</copy>
    ```

    ```
    <span style="color:blue">mysql></span> <copy>INSTALL COMPONENT
    'file://component_masking_functions';</copy>
    ```

2. Check if the components are loaded

```
<span style="color:blue">mysql></span> <copy>SELECT * FROM mysql.component;
</copy>
```

3. Use some data masking functions

    - Mask the name of the cities between first and last character

```
<span style="color:blue">mysql></span> <copy>SELECT mask_inner(NAME,
1,1) FROM world.city limit 10;</copy>
```

- Mask first and last character of the cities

```
<span style="color:blue">mysql></span> <copy>SELECT mask_outer(NAME,
1,1) FROM world.city limit 10;</copy>
```

- Generate a random email address with a specified number of digits for the name and surname parts in the specified domain, mynet.com

```
<span style="color:blue">mysql></span> <copy>SELECT gen_rnd_email(6, 8,
'mynet.com');</copy>
```

4. Create a masked view for city table

```
<span style="color:blue">mysql></span> <copy>USE world;</copy>
```

```
<span style="color:blue">mysql></span> <copy>DESC city;</copy>
```

```
<span style="color:blue">mysql></span> <copy>CREATE VIEW masked_city AS
SELECT id, CAST(MASK_INNER(NAME, 1,1) AS CHAR(35)) AS name, CountryCode,
District, CAST(GEN_RANGE(100000, 10000000) AS SIGNED) AS population FROM
world.city;</copy>
```

```
<span style="color:blue">mysql></span> <copy>SELECT * FROM masked_city limit
5;</copy>
```

5. You can now close the mysqlsh connection

```
<span style="color:blue">mysql></span> <copy>\q</copy>
```

Learn More

- https://dev.mysql.com/doc/refman/8.4/en/firewall-installation.html
- https://dev.mysql.com/doc/refman/8.4/en/audit-log-installation.html

- https://dev.mysql.com/doc/refman/8.4/en/data-masking-components-installation.html

Acknowledgements

- **Author** - Marco Carlessi, Principal Sales Consultant
- **Contributors** - Perside Foster, Principal Sales Consultant, Selena Sánchez, MySQL Solutions Engineer
- **Last Updated By/Date** - Perside Foster, Partner Solutions Engineer, March 2025