MySQL Security and Standardization at PayPal

Stacy Yuan & Yashada Jadhav • 29th May 2019
Stacy Yuan

Over 15 years of experience on various flavors of relational databases.

Focus on performance tuning, code reviews, database deployment and infrastructure management for MySQL

In her spare time, she enjoys reading books and doing some volunteer work.
Yashada Jadhav

7+ years of being in a relationship with databases

Presented previously at Percona Live and Oracle Open World

Prior to PayPal, worked at Amazon Web Services and Yahoo!

I share random database thoughts at -
https://mysql.dbgeekgirl.com
Agenda

MySQL Standardization at PayPal

MySQL Security

• Password Management

• Auditing for database objects

• Encrypted connections

• Migration to Percona XtraDB at PayPal
At PayPal, we put people at the center of everything we do.

PayPal is committed to democratizing financial services and empowering people and businesses to join and thrive in the global economy.

277 million account holders.

200+ markets around the world.

PayPal platform includes Braintree, Venmo and Xoom.
MySQL at PayPal

Internal Apps
Build, Test and Release Tools
Custom Monitoring

3rd Party
OSS Backend databases

Site
Site Facing Use Cases

DBaaS
Database as a Service
Standard Architecture

Application

Connection Cache

Database
Standard Architecture with Local DR

Application

Connection Cache

Master

Read Replica

AZ
Premium Architecture

AZ1

Application

Connection Cache

Master

Local Read Replica

Replica Scaling

AZ2

Application

Connection Cache

Remote Read Replica

Replica Scaling
MySQL Standardization at PayPal

Challenges

One DBA Team – Multiple Applications
• Automation is a challenge and an opportunity

Heterogenous Set ups
• Different hardware, software and architecture configurations

Operational Challenges
• Migrating "non-standard" setups to standard

Hardware

Operating System

MySQL Versions

Bare Metal Servers
VMs
Cloud VMs

Ubuntu 14.x, 16.x, 18.x
RHEL 6.x, 7.x
OEL 7.x

MySQL Community Edition 5.6, 5.7
Percona MySQL 5.7

© 2018 PayPal Inc. Confidential and proprietary.
MySQL Standardization at PayPal

### Journey

| Deployment | • Ansible for more scalable deployment  
• Deployment scripts for different OS versions |
| Best Practices | • Common standards across databases  
• Recommended MySQL parameter settings  
• MySQL design standards |
| Monitoring | • In house monitoring and alerting  
• TICK and PMM for OS and MySQL metrics |
| High Availability | • GitHub’s Orchestrator and VIP |
| Performance Tuning | • Performance Schema, PMM and Slow query log |
| Backups | • Percona Xtrabackup and mysqldump with onsite and offsite storage |
MySQL Standardization at PayPal

What’s next

Change Automation
Connection Multiplexer & Router
Benchmarking MySQL 8.0
InnoDB Encryption at Rest
Binary Log and Relay Log Encryption at Rest
Dual Password Support & Password Rotation
Role Based Access Control
MySQL Security at PayPal
MySQL Password Management

Password Management, Password Rotations and Methods to avoid Password Exposure
MySQL Password Management

- Password Strength
- Password Expiration
- Avoiding Password Exposure
- LDAP Authentication
Password Strength

Passwords can be your weakpoint.

MySQL’s Password Validation Plugin

Enabling the validate_password plugin

- Activate the plugin in my.cnf file and restart MySQL server
  
  ```
  plugin-load-add = validate_password.so
  validate-password = FORCE_PLUS_PERMANENT
  ```

- OR without restart
  
  ```
  mysql> install plugin validate_password soname 'validate_password.so';
  Query OK, 0 rows affected (0.03 sec)
  ```

- Why validate-password setting?
  
  ```
  mysql> uninstall plugin validate_password;
  ERROR 1702 (HY000): Plugin 'validate_password' is force_plusPermanent and can not be unloaded
  ```
### Password Strength

#### Plugin Variables

```
mysql> show global variables like 'validate_password%';
+--------------------------------------+
<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>validate_password_check_user_name</td>
<td>OFF</td>
</tr>
<tr>
<td>validate_password_dictionary_file</td>
<td></td>
</tr>
<tr>
<td>validate_password_length</td>
<td>8</td>
</tr>
<tr>
<td>validate_password_mixed_case_count</td>
<td>1</td>
</tr>
<tr>
<td>validate_password_number_count</td>
<td>1</td>
</tr>
<tr>
<td>validate_password_policy</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>validate_password_special_char_count</td>
<td>1</td>
</tr>
</tbody>
</table>
+--------------------------------------+
7 rows in set (0.00 sec)
```

#### `validate_password_policy`

<table>
<thead>
<tr>
<th>Policy</th>
<th>Test Performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or LOW</td>
<td>Length</td>
</tr>
<tr>
<td>1 or MEDIUM</td>
<td>Length; numeric, lowercase/uppercase, and special characters</td>
</tr>
<tr>
<td>2 or HIGH</td>
<td>Length; numeric, lowercase/uppercase, and special characters; dictionary file</td>
</tr>
</tbody>
</table>

```
mysql> select password("Abcdef1");
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements

mysql> select password("Abcdef1@");
<table>
<thead>
<tr>
<th>password(&quot;Abcdef1&quot;)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>*035C6A33B68D295FDE7BAA22AB7DC5733E136BD8</td>
<td></td>
</tr>
</tbody>
</table>
1 row in set, 1 warning (0.00 sec)
```
Password Expiration

Setting password to expire after N days

Setting password expiration on global level

- A password expiration policy for all accounts can be set using the MySQL setting `default_password_lifetime`
  
  ```sql
  SET GLOBAL default_password_lifetime = 180;
  ```

Setting password to never expire

- Passwords can be set to never expire on a global setting and password expiration policies can be set for individual accounts
  
  ```sql
  SET GLOBAL default_password_lifetime = 0;
  ```

Setting password expiration for individual accounts

  ```sql
  mysql> CREATE USER 'yashada'@'%' IDENTIFIED BY "Abcdef1@" PASSWORD EXPIRE INTERVAL 90 DAY;
  Query OK, 0 rows affected (0.03 sec)
  ```
Password Expiration

What happens when password expires?

- When password expires, the server restricts the client to a “sandbox” mode

```
mysql> SELECT 1;
ERROR 1820 (HY000): You must reset your password using ALTER USER statement before executing this statement.
```

```
mysql> ALTER USER USER() identified by 'Abcdef1@';
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> SELECT 1;
+---+
| 1 |
+---+
| 1 |
+---+
1 row in set (0.01 sec)
```
Password Expiration

Forcing users to change password at login

• Create users with expired password.

```sql
mysql> CREATE USER 'yashada'@'%' IDENTIFIED BY "Abcdef1@";
Query OK, 0 rows affected (0.03 sec)

mysql> alter user 'yashada'@'%' PASSWORD EXPIRE;
Query OK, 0 rows affected (0.03 sec)
```

• When user will try to login, the server will restrict client to “sandbox” mode and no operations can be performed till the user sets a password.

```sql
mysql> select 1;
ERROR 1820 (HY000): You must reset your password using ALTER USER statement before executing this statement.
```
Avoiding Password Exposure

Avoiding password exposure

• In the DBA operations world there are always challenges with passwords. How do you use the password for login, automation and operation scripts in a way that does not expose the password.

• One of the MySQL utilities that addresses some of these questions is mysql_config_editor

• mysql_config_editor enables you to store authentication credentials in a login path file named .mylogin.cnf in the current user's home directory.

Installing mysql_config_editor

• To install mysql_config_editor, all you need is the MySQL client installed.
Avoiding Password Exposure

Set up log in paths using `mysql_config_editor`

`~>` `mysql_config_editor` `set` `--login-path=monitor` `--user=monitor` `--password`

Enter password:

Using `login-path` to log in to MySQL

- To use this path to log in to MySQL –

  `~>` `mysql` `--login-path=monitor`

  Welcome to the MySQL monitor. Commands end with ; or \g.

  Your MySQL connection id is 554
Avoiding Password Exposure

How `mysql_config_editor` works?

- `mysql_config_editor` creates an encrypted file in the user's home directory called `.mylogin.cnf`

```bash
~> pwd
/root

~> ls -lhtr .mylogin.cnf
-rw-------. 1 root root 324 Apr 13 16:03 .mylogin.cnf

~> cat .mylogin.cnf
```

© 2019 PayPal Inc. Confidential and proprietary.
Avoiding Password Exposure

Using strong passwords without having to remember them

• The advantages of not having to remember or enter passwords are quite a few. For example, now a password can be a secure non dictionary string, that meets password validation rules.

• The password for monitor can be B*kA2aBntGYdvJaf. It can be set up in mysql_config_editor once and can be used without remembering the string using -

  mysql --login-path=monitor

Using the login path in utilities without exposing the password

• MySQL Utilities

  ~> mysqladmin --login-path=mysqlconn ping
  mysqld is alive

• Backup

  innobackupex --login-path=mysqlconn /backups
  xtrabackup: Transaction log of lsn (14148224090) to (14148224138) was copied.
  190413 16:32:06 completed OK!
Avoiding Password Exposure

Using the login path in utilities without exposing the password

- Automation Scripts

```
[root@ ~]# mysql --login-path=mysqlconn -e "show slave status \
G" | grep -i "Slave_IO_Running:" | awk -F: '{print $2 }'|
```
```
awk -F: '{print $2 }' | sed -e 's/^[ 	]*//'
```

Yes

Caveat – It’s not THAT secure

- We can read the contents of the encrypted log in file using the my_print_defaults utility.

```
[root@ ~]# my_print_defaults -s monitor
--user=monitor
--password=B*kA2aBntGYdvJaf
--host=XXXX
--port=3306
```

`my_print_defaults` is a part of standard MySQL install.
Storing and Retrieving Application Passwords

- Application passwords should not be hardcoded
- Application passwords and certificates are stored in a central key store
- Application passwords are retrieved via API calls to the key store
- What if key store becomes unreachable?

Keymaker software, generates a key with which password can be retrieved.

Application fetches password from keystore via keymaker API, and stores it in local encrypted cache.

DBA stores the password in central keystore

DBA shares the key with the developers who store the key in the application using a protected package.
LDAP Authentication

MySQL Server can be configured to use LDAP to authenticate users

Proxy user support

• LDAP authentication can return to MySQL a user name different from the operating system user, based on the LDAP group of the external user.

• For example, an LDAP user named stacy can connect and have the privileges of the MySQL user named dba_user, if the LDAP group for stacy is dba_users.

Percona’s PAM Authentication Plugin

• Percona PAM Authentication Plugin acts as a mediator between the MySQL server, the MySQL client, and the PAM stack.
LDAP Authentication

Installing the PAM plugin

- To install the PAM plugin –

```sql
mysql> INSTALL PLUGIN auth_pam SONAME 'auth_pam.so';
Query OK, 0 rows affected (0.08 sec)

mysql> INSTALL PLUGIN auth_pam_compat SONAME 'auth_pam_compat.so';
Query OK, 0 rows affected (0.05 sec)
```

Configure Percona PAM to authenticate to LDAP

- Configure Percona PAM to authenticate to LDAP by creating `/etc/pam.d/mysqld` with the following content –

```plaintext
auth        required pam_ldap.so audit
account     required pam_ldap.so audit
```
LDAP Authentication

Create a user with LDAP authentication

- Create a user that will use LDAP for authentication –

```sql
mysql> CREATE USER 'yashada'@'%' IDENTIFIED WITH auth_pam;
Query OK, 0 rows affected (0.04 sec)

mysql> GRANT SELECT ON ycsb.* TO 'yashada'@'%';
Query OK, 0 rows affected (0.02 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

mysql -u yashada -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \
g.
LDAP Authentication

Proxy user support

- If belonging to a group has certain MySQL privileges, setup proxy users instead to map a user's privilege to its defined group.

```sql
mysql> CREATE USER ''@'' IDENTIFIED WITH auth_pam as 'mysql,dba=dba_users';
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE USER dba_users@'localhost' IDENTIFIED BY 'somepassword';
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO dba_users@'localhost';
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> GRANT PROXY ON dba_users@'localhost' TO ''@'';
Query OK, 0 rows affected, 1 warning (0.02 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.05 sec)
```
LDAP Authentication

Proxy user support

- When any user belonging to the group logs in

  mysql -u yashada -p
  Enter password:
  Welcome to the MySQL monitor. Commands end with ; or \g.
  Your MySQL connection id is 8587
  mysql> show grants;
  +----------------------------------------------------------------------------------------+
  | Grants for @                                                                     |
  +----------------------------------------------------------------------------------------+
  | GRANT USAGE ON *.* TO ''@''                                                          |
  | GRANT PROXY ON 'dba_users'@'localhost' TO ''@''                                   |
  +----------------------------------------------------------------------------------------+
  2 rows in set (0.00 sec)
Audit Plugin
Tracking access and changes to your database
Audit Plugin
Installing the Audit Plugin

Install the plugin

```
mysql> INSTALL PLUGIN audit_log soname 'audit_log.so';
Query OK, 0 rows affected (0.06 sec)
```

Verify that the audit log plugin is installed

```
mysql> SELECT * FROM mysql.plugin WHERE name LIKE 'audit_log';
+-----------------------------+
| name           | dl             |
|-----------------------------|
| audit_log       | audit_log.so   |
+-----------------------------+
1 row in set (0.00 sec)
```
Audit Plugin
Audit Plugin Variables

Plugin Variables

```
mysql> show variables like 'audit%';
+--------------------------------------+---------+
| Variable_name                       | Value   |
+--------------------------------------+---------+
| audit_log_buffer_size               | 1048576 |
| audit_log_file                      | audit.log |
| audit_log_flush                     | OFF     |
| audit_log_format                    | OLD     |
| audit_log_handler                   | FILE    |
| audit_log_rotate_on_size            | 0       |
| audit_log_rotation                  | 14      |
| audit_log_file                      | /path/audit.log |
| audit_log_handler                   | FILE/SYSLOG |
| audit_log_strategy                  | ASYNCHRONOUS |
| audit_log_syslog_facility           | LOG_USER |
| audit_log_syslog_ident              | percona-audit |
| audit_log_syslog_priority           | LOG_INFO |
```

Audit Log file Management

<table>
<thead>
<tr>
<th>Variable</th>
<th>Recommended Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit_log_rotate_on_size</td>
<td>4M</td>
</tr>
<tr>
<td>audit_log_rotation</td>
<td>14</td>
</tr>
<tr>
<td>audit_log_file</td>
<td>/path/audit.log</td>
</tr>
<tr>
<td>audit_log_handler</td>
<td>FILE/SYSLOG</td>
</tr>
</tbody>
</table>
Audit Plugin

Audit Plugin Variables

Commands available

```sql
mysq1> SELECT name FROM performance_schema.setup_instruments WHERE name LIKE "statement/sql/%" ORDER BY name;
+---------------------------------------------+-----|
| name                                        |    |
+---------------------------------------------+-----|
| statement/sql/alter_db                      |    |
| statement/sql/alter_db_upgrade             |    |
| statement/sql/alter_event                  |    |
| statement/sql/alter_function               |    |
| statement/sql/alter_instance               |    |
| statement/sql/alter_procedure              |    |

For example –

Set `audit_log_include_commands` to a list of DDLs

`alter_db, alter_db_upgrade, alter_event, alter_function, alter_instance, alter_procedure, alter_server`

Audit Log Controls

<table>
<thead>
<tr>
<th>Policy</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>audit_log_policy</code></td>
<td>QUERIES (ALL, LOGIN, QUERIES, NONE)</td>
</tr>
<tr>
<td><code>audit_log_include_accounts</code></td>
<td>If you want to track specific accounts</td>
</tr>
<tr>
<td><code>audit_log_exclude_accounts</code></td>
<td>Monitoring / Tooling accounts</td>
</tr>
<tr>
<td><code>audit_log_include_databases</code></td>
<td>Databases like heartbeat</td>
</tr>
<tr>
<td><code>audit_log_exclude_databases</code></td>
<td></td>
</tr>
<tr>
<td><code>audit_log_include_commands</code></td>
<td>Only include commands of interest</td>
</tr>
</tbody>
</table>

© 2019 PayPal Inc. Confidential and proprietary.
Audit Plugin
Sample Audit Log

<AUDIT_RECORD
  NAME="Query"
  RECORD="2709669_2019-02-05T08:48:00"
  TIMESTAMP="2019-03-07T08:49:13 UTC"
  COMMAND_CLASS="alter_table"
  CONNECTION_ID="1793745"
  STATUS="0"
  SQLTEXT="ALTER TABLE actors CHANGE `last_update` `last_modified` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP"
  USER="yashada[yashada] @ [x.x.x.x]"
  HOST=""
  OS_USER=""
  IP="x.x.x.x"
  DB="sakila"
/>

© 2019 PayPal Inc. Confidential and proprietary.
Encrypted Connections

On wire data encryption for MySQL
Encrypted Connections

WHAT

• On-wire data encryption is the process of changing information that makes it unreadable except someone who has the key

WHY

• The world is not perfect!
• PayPal uses it to protect PII, credit card, etc to guard against things like identity theft

How?

At PayPal, an unencrypted DB connection is unacceptable
Encrypted Connections + Authentication

- **Public/private key pair** - encryption
- **CA Certificate** - verify the identity of the systems
- **Username and password** - DB authentication

MySQL

- Server private key
- Server public key
- CA cert
- AES256 ciphers

© 2019 PayPal Inc. Confidential and proprietary.
Set up Encrypted Connections

Configure MySQL using encrypted connections

Generate SSL Certs

Create database user with SSL enabled

Connect to database with encrypted connections in various way

Verify connection is encrypted

Troubleshooting Tips
Configure MySQL using Encrypted Connections

ssl option in server
• ssl-ca: The path name of the Certificate Authority (CA) certificate file in pem format.
• ssl-cert: The path name of the server public key certificate file in pem format.
• ssl-key: The path name of the server private key file. Never give this file to anyone.
• ssl-cipher: The list of permitted strong ciphers (AES256) for connection encryption
• require_secure_transport: OFF (by default)

my.cnf

```
[mysqld]
ssl-ca=ca.pem
ssl-cert=cert.pem
ssl-key=key.pem
ssl_cipher=DH-DSS-AES256-GCM-SHA384,DH-DSS-AES256-SHA,...
```

© 2019 PayPal Inc. Confidential and proprietary.
Client Configure for Encrypted Connections

ssl option in client

- **ssl-ca**: The path name of the Certificate Authority (CA) certificate file
- **ssl-cipher**: The list of permitted ciphers for connection encryption

```
my.cnf

[client]
ssl-ca=client-ca.pem
ssl_cipher=DH-DSS-AES256-GCM-SHA384,DH-DSS-AES256-SHA,...
```

Client Options for ssl-mode

- PREFERRED
- REQUIRED
- VERIFY_CA
- VERIFY_IDENTITY
Generate Certificate Service Request

- Using openssl creates CSR
- Private key remains local

Review CSR and order Certs

Sign and issue CA certs

MySQL

Security team

Certificate Authority

Verifications and send CA certs back to KC

Send CA certs back to KC

Using a vendor manages our intermediate CAs, which signs the CSRs of the MySQL servers
Form CA Certs

ssl-cert = cert.pem (user cert)

ssl-ca = ca.pem (root cert + intermediate cert)

client.ca.pem = root cert
   It will be distributed to the clients

ssl-key = key.pem
   It is private and held securely in the DB server
Create User with TLS Options

Create user without REQUIRE-SSL

CREATE USER 'ssluser'@'ip-address' IDENTIFIED BY '**********';
GRANT ALL PRIVILEGES ON ssldb.* TO 'ssluser'@'ip-address';

Connect to database with this user

mysql -u ssluser -p password -h host -P port#

- The server performs certificate and key file autodiscovery.
- If the server discovers valid certificate and key files, it enables support for encrypted connections by clients.

mysql -u ssluser -p password -h host -P port# --ssl-mode=disabled

- If the server does not find valid certificate and key files, it will use non-encrypted connections.

TLS_OPTIONS
SSL, X509, CIPHER, ISSUER, SUBJECT
Create User with SSL Requirement

**Alter user with SSL enabled**

```
ALTER USER 'ssluser'@'ip-address' REQUIRE SSL;
Or
GRANT ALL PRIVILEGES ON test.* TO 'ssluser'@'ip' REQUIRE SSL;
```

**Verify**

```
mysql> select user,host,ssl_type from mysql.user where user='ssluser';
+---------------+-------+--------+
<table>
<thead>
<tr>
<th>user</th>
<th>host</th>
<th>ssl_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssluser</td>
<td>*****</td>
<td>ANY</td>
</tr>
</tbody>
</table>
+---------------+-------+---------+
1 row in set (0.00 sec)
```

**Remove SSL constraint**

```
ALTER USER 'ssluser'@'ip-address' REQUIRE NONE;
```
### DB Encrypted Connections Support Verification

```sql
mysql> SHOW VARIABLES LIKE 'have_ssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_ssl      | YES   |
+---------------+-------+

mysql> SHOW SESSION STATUS LIKE 'Ssl%';
+-----------------------------+-------+
| Variable_name               | Value |
+-----------------------------+-------+
| Ssl_version                 | TLSv1.2|
| Ssl_cipher                  | ECDHE-RSA-AES256-GCM-SHA384 |
...```
Common Errors

Launch DB connection with --ssl-mode=disabled

Case 1:
User "require SSL":

ERROR 1045 (28000): Access denied for user 'ssluser'@'***' (using password: YES)

Case 2:
DB level "require_secure_transport":

ERROR 3159 (HY000): Connections using insecure transport are prohibited while --
require_secure_transport=ON.
SSL Vulnerability

By default, clients attempt to connect using encryption connection falling back to an unencrypted connection if an encrypted connection cannot be established.

```
mysql -u user -ppassword -h host -P port#
mysql> \s
--------------
mysql  Ver 14.14 Distrib 5.7.23-23, for Linux (x86_64) using 6.2
Connection id: 9051
Current database:
Current user: root@localhost
SSL: Cipher in use is ECDHE-RSA-AES256-GCM-SHA384
....
```

If the client does not use --ssl-mode=REQUIRED, an attacker can downgrade the client->attacker connection to plain text, then make an encrypted connection to the server.

If the client does use --ssl-mode=REQUIRED, an attacker can use their own self-signed certificate so the client->attacker is encrypted with the attacker's certificate, decrypt the client's traffic, then make an encrypted connection to the server.
SSL Vulnerability

Clients require an encrypted connection, and also perform verification against the server CA certificate

MySQL

- `u user`
- `p password`
- `h host`
- `-P port#`
- `--ssl-ca=/path/client-ca.pem`
- `--ssl-mode=VERIFY_CA`

```
mysql> \
----------------
```

MySQL Ver 14.14 Distrib 5.7.23-23, for Linux (x86_64) using  6.2
Connection id: 9051
Current database:
Current user: root@localhost
SSL: Cipher in use is ECDHE-RSA-AES256-GCM-SHA384
...

This will let the client verify that the server's certificate was signed by the CA they expect, which will prevent most MITM attacks.

The client would still be vulnerable to a MITM attack if the attacker has another certificate signed by the same CA (for example, a different server within the same enterprise).
Prevent MITM Attacks

Clients require an encrypted connection, and also perform verification against the server CA certificate and (with VERIFY_IDENTITY) against the server host name in its certificate.

```
Mysql -u user -p -h host -P port# --ssl-ca=/path/client-ca.pem -ssl-mode=VERIFY_IDENTITY
mysql> \s
--------------
mysql  Ver 14.14 Distrib 5.7.23-23, for Linux (x86_64) using 6.2
Connection id: 9051
Current database:
Current user: root@localhost
SSL: Cipher in use is ECDHE-RSA-AES256-GCM-SHA384
...
```

This will let the client verify that the server’s "Common Name" field matches the server’s hostname the client uses. This should prevent all MITM attacks.

But it is difficult to manage so many certs since we have so many database servers.
Practical Way

Clients require an encrypted connection, also perform verification against the server CA certificate

Create user using IP

```
GRANT USAGE ON *.* TO 'myuser'@'100.10.10.100' IDENTIFIED BY '****';
GRANT ALL PRIVILEGES ON `ssldb`.* TO 'myuser'@'100.10.10.100';
```

```
Mysql -u myuser -p -h host -P port# --ssl-ca=/path/client-ca.pem -ssl-mode=VERIFY_CA
mysql> \

--------------
mysql  Ver 14.14 Distrib 5.7.23-23, for Linux (x86_64) using 6.2
Current user: myuser@100.10.10.100
SSL: Cipher in use is ECDHE-RSA-AES256-GCM-SHA384
......
Connection: 100.10.10.100 via TCP/IP
```

This will let the client verify that the server's certificate was signed by the CA they expect, and from server side only accept db connection request from client 100.10.10.100

© 2019 PayPal Inc. Confidential and proprietary.
Prevent MITM Attacks

Client

Require SSL

Certificate Verification

Identity Verification

DB account user + password + individual host

MySQL

CA cert

Private key
Public key
CA cert
Verify Live DB Connections are Encrypted

Check live running database

```sql
mysql> SELECT sbt.variable_value AS tls_version, t2.variable_value AS cipher, processlist_user AS user, processlist_host AS host
    -> FROM performance_schema.status_by_thread AS sbt
    -> JOIN performance_schema.threads AS t ON t.thread_id = sbt.thread_id
    -> JOIN performance_schema.status_by_thread AS t2 ON t2.thread_id = t.thread_id
    -> WHERE sbt.variable_name = 'Ssl_version' and t2.variable_name = 'Ssl_cipher' ORDER BY tls_version;
```

<table>
<thead>
<tr>
<th>tls_version</th>
<th>cipher</th>
<th>user</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSv1.2</td>
<td>DHE-RSA-AES256-GCM-SHA384</td>
<td>userapp1</td>
<td><em>.</em>.<em>.</em></td>
</tr>
<tr>
<td>TLSv1.2</td>
<td>DHE-RSA-AES256-GCM-SHA384</td>
<td>userapp2</td>
<td><em>.</em>.<em>.</em></td>
</tr>
<tr>
<td>TLSv1.2</td>
<td>DHE-RSA-AES256-GCM-SHA384</td>
<td>userapp</td>
<td><em>.</em>.<em>.</em></td>
</tr>
<tr>
<td>TLSv1.2</td>
<td>DHE-RSA-AES256-GCM-SHA384</td>
<td>userapp1</td>
<td><em>.</em>.<em>.</em></td>
</tr>
<tr>
<td>TLSv1.2</td>
<td>DHE-RSA-AES256-GCM-SHA384</td>
<td>userappb</td>
<td><em>.</em>.<em>.</em></td>
</tr>
<tr>
<td>TLSv1.2</td>
<td>DHE-RSA-AES256-GCM-SHA384</td>
<td>userapp2</td>
<td><em>.</em>.<em>.</em></td>
</tr>
<tr>
<td>TLSv1.2</td>
<td>DHE-RSA-AES256-GCM-SHA384</td>
<td>userapp2</td>
<td><em>.</em>.<em>.</em></td>
</tr>
<tr>
<td>TLSv1.2</td>
<td>DHE-RSA-AES256-GCM-SHA384</td>
<td>userappb</td>
<td><em>.</em>.<em>.</em></td>
</tr>
<tr>
<td>TLSv1.2</td>
<td>DHE-RSA-AES256-GCM-SHA384</td>
<td>userapp3</td>
<td><em>.</em>.<em>.</em></td>
</tr>
<tr>
<td>TLSv1.2</td>
<td>DHE-RSA-AES256-GCM-SHA384</td>
<td>userappb</td>
<td><em>.</em>.<em>.</em></td>
</tr>
</tbody>
</table>
```
Verify Encrypted Connections from User

Check table `Information_schema.USER_STATISTICS`

```sql
Use information_schema;
sel...
Set up Replication with Secure Connections

Create replication user require SSL using ip

- ip1 is master and ip2 and ip3 are slaves
- GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'repl'@'ip1' require SSL;
- GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'repl'@'ip2' require SSL;
- GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'repl'@'ip3' require SSL;

Enable replication

- Change master to
- MASTER_HOST=master-host,
- MASTER_USER='repl',
- MASTER_PASSWORD='***',
- MASTER_SSL_CA=/path/client-ca.pem,
- MASTER_SSL=1,
- MASTER_AUTO_POSITION=1;
- START SLAVE;
- SHOW SLAVE STATUS;
Encrypted Connections

Troubleshooting Tips

MySQL Client could be community version
• ERROR 2059 (HY000): Authentication plugin 'dialog' cannot be loaded: dlopen(/usr/local/mysql/lib/plugin/dialog.so, 2): image not found

CA certs file permission
• ERROR 2026 (HY000): SSL connection error: SSL_CTX_set_default_verify_paths failed

Errors Received With Cipher Mismatch
Please check ssl_ciphers

Errors Received Without A Complete Truststore
Please generate complete truststore file.

Add “-Djavax.net.debug=all” JVM arguments to debug TLS connections
• -jvm-args="-Djavax.net.debug=all -Djavax.net.ssl.trustStore=/tmp/truststore -Djavax.net.ssl.trustStorePassword=*** -Djava.security.egd=file:///dev/urandom"
Security-Enhanced Linux (SELINUX)

What is SELinux

- It is a Linux kernel security module for supporting access control security policies
- SELinux defines the access and transition rights of every user, application, process, and file on the system
- SELinux then governs the interactions of these entities using a security policy
Security-Enhanced Linux (SELINUX)

Two modes
- SELINUX=enforcing
- SELINUX=permissive
  SELinux does not enforce its policy, but only logs what it would have blocked (or granted)

Change SELinux mode to permissive
- Edit the file /etc/selinux/config, reboot the server
- setenforce 0  #online change, it will be lost after the server reboots
- Recommended way: semanage permissive -a mysql_t
- The best way:
  Create policies
  load policies, run for a while in permissive state
  Change it to enforcing after no denied message in audit.log
Challenges while DB migration from Others to Percona

Audit log

• MySQL commercial version uses function/procedures to handle
• Remove audit related functions
  • MySQL> delete from mysql.func where name like 'audit%';
  • MySQL> INSTALL PLUGIN audit_log SONAME 'audit_log.so';
• Add audit related settings into my.cnf, restart mysql.

LDAP

• Drop all existing ldap accounts and re-create them

SSL

• MySQL community version is incompatible with Percona, need to remove all packages.

Missing library packages

• Need to install Perl related packages as well as other lib packages.

© 2019 PayPal Inc. Confidential and proprietary.
Questions?