# Securing customer data (PII) in MySQL

Alexander Rubin
VirtualHealth

VIRTUALHEALTH

# About me

Working with MySQL for 10-15 years

- Started at MySQL AB 2006
  - Sun Microsystems, Oracle (MySQL Consulting)
  - Percona since 2014
- Recently joined Virtual Health

# Protecting data in MySQL: Requirements

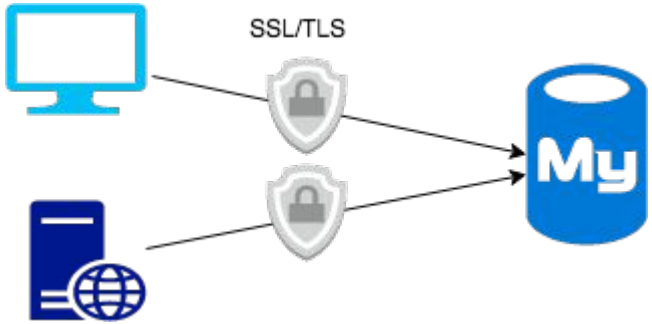
1. Encryption
   a. Data in Flight: SSL/TLS
   b. Data at Rest
2. Audit trail: logging actions
3. User auth
4. De-identification (for development and research)

# Data in Flight Encryption: SSL/TLS

# Data in Flight Encryption - client connection

## SSL/TLS: Default in MySQL 5.7+

- If SSL is enabled (default) on the server client will use it
- No need to generate keys and send it to the client
    - Server key - will be generated when MySQL starts
    - Client key will be generated on demand



# Data in Flight Encryption - client connection

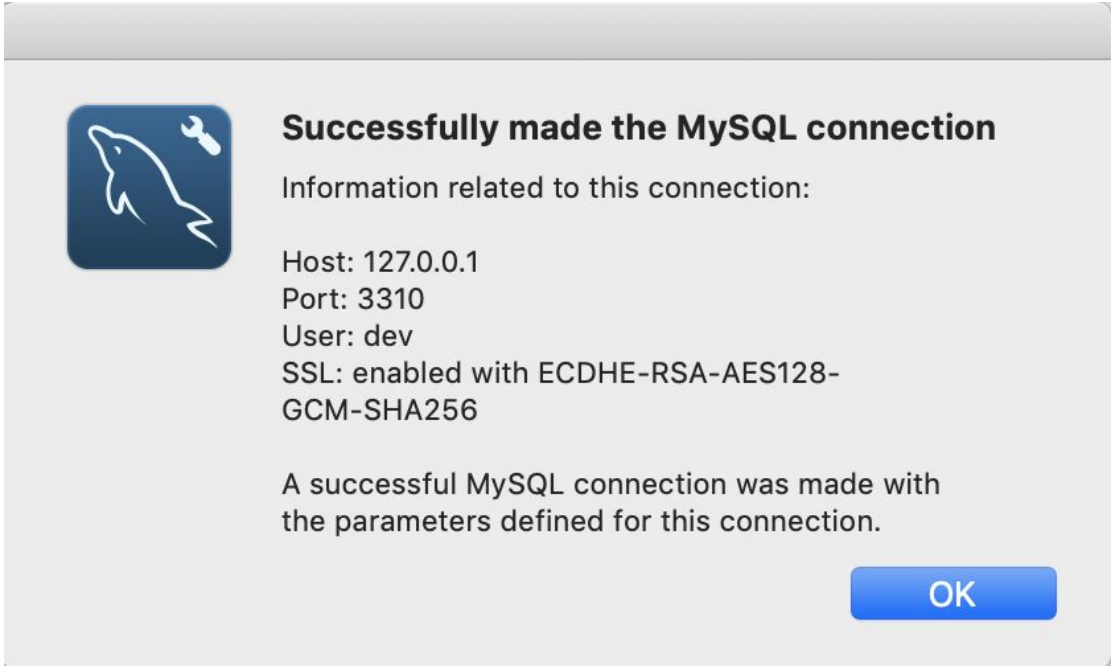
## SSL/TLS: Default in MySQL 5.7+

```
$ mysql -h db
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
Server version: 5.7.25-28-57-log Percona XtraDB Cluster (GPL)

mysql> \s
--------------
Connection id:          799621
...
SSL:                    Cipher in use is ECDHE-RSA-AES128-GCM-SHA256
```

# Data in Flight Encryption - client connection

MySQL workbench:




# Data in Flight Encryption - client connection

## Create user and force ssl/tls

```
CREATE USER 'user'@'<host>' IDENTIFIED BY '<pass here>' REQUIRE SSL;

$ mysql> alter user dev@'10.0.0.1' require ssl;
Query OK, 0 rows affected (0.00 sec)

$ mysql -u dev -h 10.0.0.1 -e '\s' | grep SSL
SSL:                    Cipher in use is TLS_AES_256_GCM_SHA384

$ mysql -u dev -h 10.0.0.1 --skip-ssl
ERROR 1045 (28000): Access denied for user 'dev'@'10.0.0.1' (using password: YES)
```

# Data in Flight Encryption - client connection

## Convert your app to use SSL/TLS

```
$ mysql> alter user dev@'10.0.0.1' require ssl;
Query OK, 0 rows affected (0.00 sec)
```

Change the connection parameters in your language:

PHP: `mysqli::ssl_set ( string $key , string $cert , string $ca , string $capath , string $cipher ) : bool`

Python:

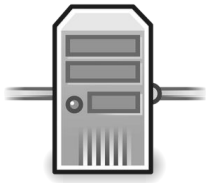
- Connector/Python:
  - As of Connector/Python 2.2.2, if the MySQL server supports SSL connections, Connector/Python attempts to establish a secure (encrypted) connection by default, falling back to an unencrypted connection otherwise.
- SQLAlchemy: `ssl_args = {'ssl': {'cert':'/path/to/client-cert', 'key':'/path/to/client-key', 'ca':'/path/to/ca-cert'}`

# Data in Flight Encryption - server to server

## Protecting communications: master -> slave

Setup is simple:

1. Copy keys from master to slave
2. Setup slave with:



```
mysql> CHANGE MASTER TO
    -> MASTER_HOST='master_hostname',
    -> MASTER_USER='repl',
    -> MASTER_PASSWORD='password',
    -> MASTER_SSL=1;
```

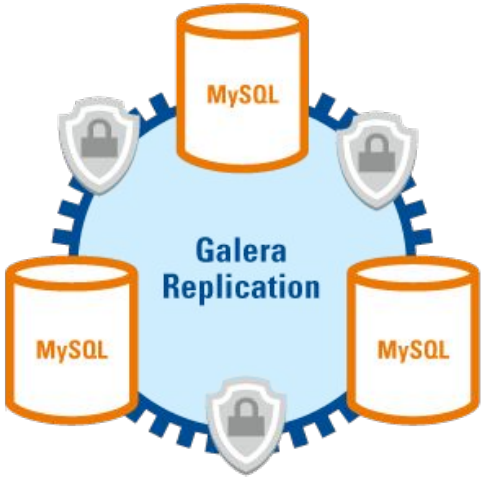
https://dev.mysql.com/doc/refman/5.7/en/replication-solutions-encrypted-connections.html

# Data in Flight Encryption - server to server

## Protecting communications: XtraDB Cluster

Need to protect
(in addition to client connections):

1. Communication between nodes
2. Synchronization (SST/IST)




https://www.percona.com/doc/percona-xtradb-cluster/LATEST/security/encrypt-traffic.html

# Data in Flight Encryption - server to server

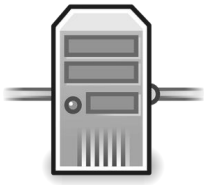
## Protecting communications: XtraDB Cluster

Setup is super-simple:

1. Copy the same certificates to all nodes (take all from 1 node)
2. Add to to my.cnf

```
pxc-encrypt-cluster-traffic=ON
ssl-key=server-key.pem
ssl-ca=ca.pem
ssl-cert=server-cert.pem
```

3. Restart all nodes



https://www.percona.com/doc/percona-xtradb-cluster/LATEST/security/encrypt-traffic.html

# Data at Rest Encryption: disk

# Data at Rest Encryption

Options:

1. Full disk encryption, i.e. Luks --- OK
2. Transparent Database Encryption (TDE) --- BETTER
3. Field level encryption --- BEST & WORST

# Data at Rest Encryption

Full disk encryption options:

1. Luks:

   https://www.percona.com/blog/2017/06/06/mysql-encryption-at-rest-part-1-luks/

   https://www.percona.com/resources/technical-presentations/mysql-disk-encryption-luks-percona-technical-webinar
2. AWS/Cloud - Encrypting volume (EBS) with custom key
3. Shared storage encryption, etc

# Data at Rest Encryption

Full disk encryption:

1. Only protect from physical access to disk (or reusing images)
2. If MySQL is running: data in MySQL files are not encrypted

# Data at Rest Encryption

Transparent Database Encryption (TDE): MySQL implementation

1. Create master key and store it
2. Use master key to encrypt table (tablespace) key
3. Use tablespace key to encrypt table data

When a tablespace is encrypted, a tablespace key is encrypted and stored in the tablespace header. When an application or authenticated user wants to access encrypted data, `InnoDB` uses a master encryption key to decrypt the tablespace key. The decrypted version of a tablespace key never changes, but the master encryption key can be changed as required. This action is referred to as *master key rotation*.
https://dev.mysql.com/doc/refman/5.7/en/innodb-tablespace-encryption.html#innodb-tablespace-encryption-about

# Data at Rest Encryption

Transparent Database Encryption (TDE): encrypting db files

1. **InnoDB files:** tablespaces, redo logs, undo logs:
    - Available since MySQL 5.7
2. **Binary logs, relay logs:** for MySQL replication:
    - Available in MySQL 8.0 and Percona Server 5.7 & 8.0
3. **Tmp files:** Available in Percona Server 5.7 & 8.0

https://www.percona.com/doc/percona-server/5.7/management/data_at_rest_encryption.html

https://dev.mysql.com/doc/refman/8.0/en/innodb-tablespace-encryption.html

https://mariadb.com/kb/en/library/data-at-rest-encryption-overview/

# Data at Rest Encryption

TDE: full config and testing (percona server 5.7)

```
mysql> create table a(s varchar(255)) engine=InnoDB;
Query OK, 0 rows affected (0.01 sec)

mysql> insert into a values ('82cU1JPgGMk2wtPj1MjnFkdeAJdhlPZiqGEMVtH8sAU');
Query OK, 1 row affected (0.00 sec)

mysql> insert into a values ('qqqqqq');
Query OK, 1 row affected (0.00 sec)

mysql> update a set s = '82cU1JPgGMk2wtPj1MjnFkdeAJdhlPZiqGEMVtH8sAU';
Query OK, 1 row affected (0.00 sec)
Rows matched: 2  Changed: 1  Warnings: 0
```

# Data at Rest Encryption

TDE: full config and testing (percona server 5.7)

```
/data/mysql# grep -r '82cU1JPgGMk2wtPj1MjnFkdeAJdhlPZiqGEMVtH8sAU' *
Binary file ib_logfile0 matches
Binary file log-bin.000004 matches
Binary file test/a.ibd matches
Binary file xb_doublewrite matches
```

# Data at Rest Encryption: add encryption options

```
[mysqld]
early-plugin-load=keyring_file.so
keyring_file_data=/mount/mysql/mysql-keyring/keyring
innodb_sys_tablespace_encrypt=1
innodb_parallel_dblwr_encrypt=1
innodb_temp_tablespace_encrypt=1
innodb_encrypt_tables=FORCE
innodb_encrypt_online_alter_logs=1
innodb_undo_log_encrypt=1
innodb_redo_log_encrypt=1
innodb_scrub_log=1
master_verify_checksum=1
binlog_checksum=1
encrypt_binlog=1
encrypt_tmp_files=1
```

# Data at Rest Encryption

TDE: full config and testing (percona server 5.7)

```
mysql> create table a(s varchar(255)) engine=InnoDB /* encrypted='y' */;
Query OK, 0 rows affected (0.01 sec)

mysql> insert into a values ('82cU1JPgGMk2wtPj1MjnFkdeAJdhlPZiqGEMVtH8sAU');
Query OK, 1 row affected (0.00 sec)

mysql> insert into a values ('qqqqqq');
Query OK, 1 row affected (0.00 sec)

mysql> update a set s = '82cU1JPgGMk2wtPj1MjnFkdeAJdhlPZiqGEMVtH8sAU';
Query OK, 1 row affected (0.00 sec)
Rows matched: 2  Changed: 1  Warnings: 0
```

# Data at Rest Encryption: add encryption options

```
/data/mysql# grep -r '82cU1JPgGMk2wtPj1MjnFkdeAJdhlPZiqGEMVtH8sAU' *
/data/mysql#
```

# Data at Rest Encryption

TDE: key rotation

```
mysql> ALTER INSTANCE ROTATE INNODB MASTER KEY
```

https://dev.mysql.com/doc/refman/5.7/en/alter-instance.html

https://www.percona.com/doc/percona-server/5.7/management/data_at_rest_encryption.html

Rotating the master encryption key only changes the master encryption key and re-encrypts tablespace keys. It does not decrypt or re-encrypt associated tablespace data.

# Data at Rest Encryption: Options

TDE: Storing keys - hashicorp vault plugin

```
$ mysqld --early-plugin-load="keyring_vault=keyring_vault.so" \
 --loose-keyring_vault_config="/home/mysql/keyring_vault.conf"
```

https://www.percona.com/doc/percona-server/5.7/management/data_at_rest_encryption.html#id38

# Data at Rest Encryption: field level encryption

- Application level encryption
  - Application code encrypt needed PII fields
- Issues:
  - Key storage and rotation
    - Hashicorp Vault or AWS secrets manager potentially solves that
  - Searches in MySQL - range search
  - Order by
  - Indexes

# Audit log

# Audit logs

## Percona Server: Audit Log Plugin

### Log ALL queries on MySQL server

Example of the Audit event:

```
<AUDIT_RECORD
 "NAME"="Audit"
 "RECORD"="1_2014-04-29T09:29:40"
 "TIMESTAMP"="2014-04-29T09:29:40 UTC"
 "MYSQL_VERSION"="5.6.17-65.0-655.trusty"
 "STARTUP_OPTIONS"="--basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib/mysql/plugin --user=mysql
--log-error=/var/log/mysql/error.log --pid-file=/var/run/mysqld/mysqld.pid --socket=/var/run/mysqld/mysqld.sock --port=3306"
 "OS_VERSION"="x86_64-debian-linux-gnu",
  />
```

https://www.percona.com/doc/percona-server/5.7/management/audit_log_plugin.html

# User authentication

# Internal users access

- Database access
  - Developers
  - Support
  - Business Analysts
- Challenges
  - Security
  - Performance impact

# Internal users access: options

- Use shared account
  - Big NO
- Create and manage 30+ MySQL user accounts on all MySQL servers
  - Not easy to manage
- Use LDAP Auth on MySQL server (Percona Server, MariaDB, etc)
  - Requires setup for Linux / PAM
  - Need to reconfigure 100s of servers
- Use Hashicorp vault to create MySQL user dynamically
  - MySQL “Bastion”
  - https://medium.com/hootsuite-engineering/mysql-bastion-streamlined-db-access-with-proxysql-vault-and-ad-aa79877247b4

# Internal users access: options

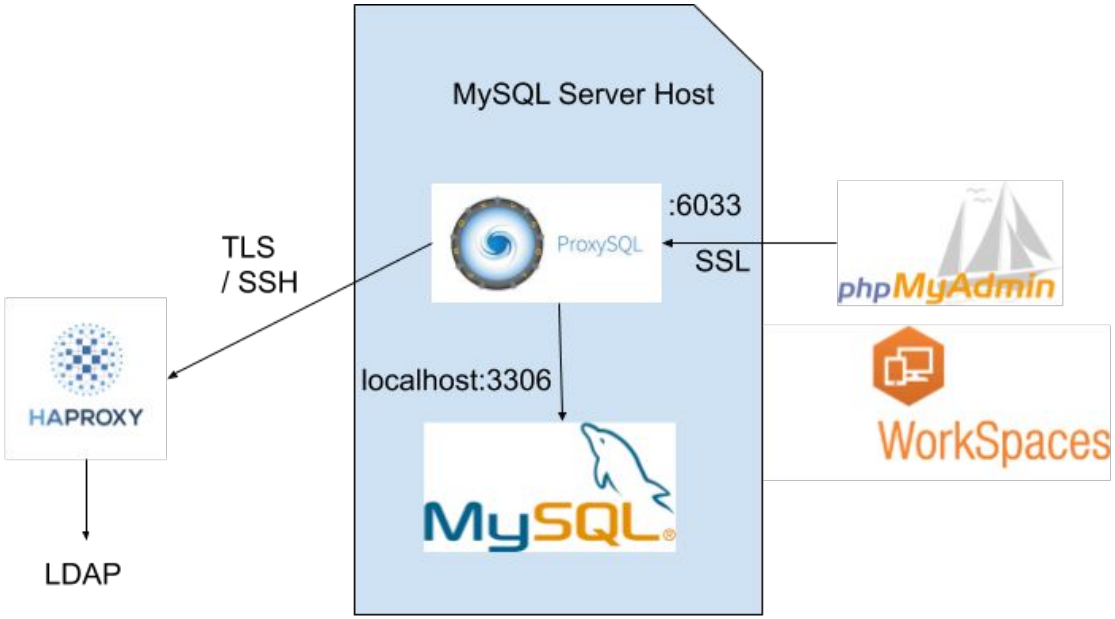
- Can we use a Proxy + LDAP ?

**ProxySQL**

High-performance MySQL proxy

https://proxysql.com/

# Internal users access: architecture



# Internal users access: components

- LDAP / Active Directory (Samba)
- ProxySQL 2.0 + LDAP plugin

*LDAP authentication in ProxySQL 2.0 is implemented as a plugin that is not part of the core of ProxySQL itself.*

*To load the plugin, it is required to specify in ProxySQL config file (`proxysql.cfg`) in the global section, using option `ldap_auth_plugin` :*

```
ldap_auth_plugin="/path/to/MySQL_LDAP_Authentication_plugin.so"
```

# Internal users access: config

```
update global_variables set variable_value = 'ldap://localhost:10389'
        where variable_name = 'ldap-uri';
update global_variables set variable_value = 'dc=example,dc=com'
        where variable_name = 'ldap-root_dn';
update global_variables set variable_value = '@example.com'
        where variable_name = 'ldap-bind_dn_suffix';
update global_variables set variable_value = "@example.com"
        where variable_name = 'ldap-bind_dn_suffix';
LOAD LDAP VARIABLES TO RUNTIME;
SAVE LDAP VARIABLES TO DISK;
```

# Internal users access: Credentials mapping

- CN=arubin is a member of mysqlro group
- Map mysqlro group to mysql_readonly account called 'ldapro'

```
INSERT INTO mysql_ldap_mapping (priority, frontend_entity, backend_entity)
VALUES
(10, 'mysqlro', 'ldapro');
```

*Need to create 'ldapro'@'127.0.0.1' with select privilege in BOTH MySQL and ProxySQL*

# Internal users access: Credentials mapping

```
$ mysql -h 127.0.0.1 -P 6033 -uarubin -p
```


```
2019-06-12 14:50:24 [INFO] LDAP: search sAMAccountName string:
ldap://localhost:10389/?sAMAccountName?sub?(member:1.2.840.113556.1.4.1941:=CN=arubin,CN=Users,..)
2019-06-12 14:50:24 [INFO] ldap search sAMAccountName completed after 4563us.
2019-06-12 14:50:24 [INFO]  arubin@mysql.virtualhealth.com:  sAMAccountName: mysqlro
2019-06-12 14:50:24 [INFO] LDAP: f_e: mysqlro, FE: mysqlro, BE: ldapro
2019-06-12 14:50:24 [INFO] LDAP: Adding user arubin in cache
2019-06-12 14:50:24 [INFO] LDAP: user arubin found in cache
2019-06-12 14:50:24 [INFO] LDAP: user arubin found in cache and is not expired
```

# ProxySQL: query fingerprint

```
mysql> set global general_log=1;
Query OK, 0 rows affected (0.00 sec)

2019-06-15T02:43:25.205634Z      9004 Query
select /* mysql_user=arubin */ * from mysql.user
```

# De-identification / anonymization

# De-identification / anonymization

Goal: create a database version without PII / PHI

- Change the PHI data to FAKE data

```
Alexander Rubin          ->        John Smith
<real phone number>      ->        555-555-05-55
```

# De-identification / anonymization

Faker lib: https://github.com/joke2k/faker

1. Generated mysql table with faker:

```
mysql> select * from faker.fake_clients limit 1\G
*************************** 1. row ***************************
        id: 1
   address: 27680 Anthony Fields Anthonyville, NE 22423
  birthday: 2001-01-21
     email: Luke.Hensley.1@fakemailvh.com
first_name: Luke
 last_name: Hensley
     phone: 1-555-205-7146
       txt: Every development say. Throughout beautiful instead ahead despite measure. Current
practice nation. Operation speak according south.
   zipcode: 00501
```

# De-identification / anonymization

Faker lib: https://github.com/joke2k/faker

Implementation:

1. Generated mysql table faker.fake_clients with faker
2. Created a mapping table, mapping by id (randomly)
3. A script to copy data from sourceDB to destinationDB (insert … select )
4. … Backup and package de-identified db into Docker container

# De-identification / anonymization: Other options

1. On the fly
   - Use views: select rand_phone(phone) from …
2. File bases anonymizers:
   - Clickhouse Obfuscator:

     https://github.com/yandex/ClickHouse/blob/master/dbms/programs/obfuscator/Obfuscator.cpp
   - Gonymizer (PostgreSQL): https://github.com/smithoss/gonymizer

# Thank you!

Alexander Rubin

https://www.linkedin.com/in/alexanderrubin