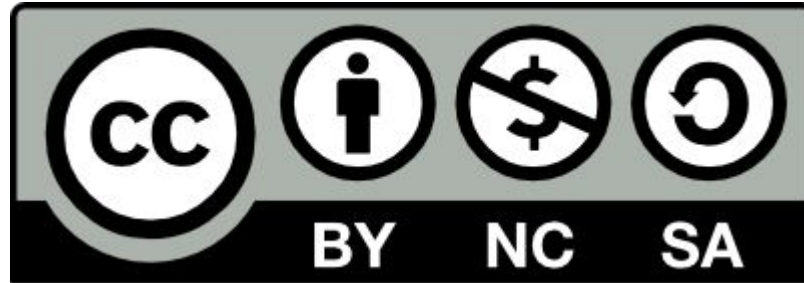


# Securing your MySQL/MariaDB Server Data

Colin Charles, Ronald Bradford  
Percona Live Santa Clara 2017  
(now as a webinar - July 2017)

# License

- Creative Commons BY-NC-SA 4.0
- <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>



# About: Colin Charles

- Chief Evangelist, Percona Inc
- Founding team of MariaDB Server (2009-2016), previously at Monty Program Ab, merged with SkySQL Ab, now MariaDB Corporation
- Formerly MySQL AB (exit: Sun Microsystems)
- Past lives include Fedora Project (FESCO), OpenOffice.org
- MySQL Community Contributor of the Year Award winner 2014
- <http://bytebot.net/blog/>

# About: Ronald Bradford

- MySQL Database Consultant at Pythian
- Author/Blogger/Speaker
- Oracle ACE Director 2010 - present
- MySQL Community Member of the Year Award winner 09, 13
- Formally MySQL Inc 06-08, Oracle Corporation 96-99
- <http://ronaldbradford.com/presentations/>
- <http://effectivemysql.com>

# Agenda

- Observed insecure practices
- Securing communications
- Securing connections
- Securing data
- Securing user accounts
- Securing server access

# Signs of Poor Security

- old\_passwords
- Users without passwords
- Anonymous users
- WITH GRANT privilege users
- ALL ON \*.\* privileged users
- '%' host user accounts
- Not using CREATE USER
- 'root' MySQL user without password
- 'root' MySQL user
- Generic OS DBA user e.g. 'dba'
- Disabled OS security e.g. Firewall/SELinux/Apparmor
- Open data directory privileges
- Default test database

# Easy Fixes

```
$ mysql_secure_installation
```

```
VALIDATE PASSWORD PLUGIN can be used to test passwords  
and improve security. It checks the strength of password  
and allows the users to set only those passwords which are  
secure enough. Would you like to setup VALIDATE PASSWORD plugin?
```

```
There are three levels of password validation policy:
```

```
LOW      Length >= 8
```

```
MEDIUM Length >= 8, numeric, mixed case, and special characters
```

```
STRONG Length >= 8, numeric, mixed case, special characters and dictionary  
file
```

```
Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG:
```

```
...
```


```
Estimated strength of the password: 25
```

```
Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No)
```

5.7 Functionality

# Current Insecure Practices

- Using password on command line
  - Command history
  - MySQL shell history
- Using simple passwords
  - It's just a test environment
- Using excessive permissions
  - GRANT, ALL, \*.\* , %



Very easy to  
improve poor  
practices



# Command Line options (non-interactive)

```
read MYSQL_USER
read -s MYSQL_PWD

mysql -u${MYSQL_USER} -p${MYSQL_PWD}
```

ps rewrite

```
ps -ef | grep mysql
.... mysql -udemo -px xxxxxxxx
```

```
mysql # using $HOME/.my.cnf
mysql --defaults-file=/path/to/.my.cnf
```

- What about gh-ost, pt-osc, other pt- tools etc

<https://dev.mysql.com/doc/refman/5.6/en/mysql-logging.html>

# Config Editor

```
$ mysql_secure_installation
$ mysql -uroot -p -e "CREATE USER demo@localhost IDENTIFIED BY 'passw0rd1';"
$ echo "[client]
user=demo
password=passw0rd1" > $HOME/.my.cnf
$ mysql -e "SELECT USER()"
$ rm $HOME/.my.cnf
$ mysql -e "SELECT USER()"

$ mysql_config_editor set --login-path=client --host=localhost --user=demo --password
$ ls -al $HOME/.mylogin.cnf
$ cat $HOME/.mylogin.cnf
$ mysql_config_editor print

$ mysql -e "SELECT USER()"
$ mysqldump .....
```



Since 5.6

# Why being SUPER is bad (GRANT ALL ON \*.\* )

- Bypasses read\_only (why we need super\_read\_only)
- Bypasses init\_connect
- Can disable binary logging
- Can change dynamic configuration
- Takes the reserved connection

<http://ronaldbradford.com/blog/why-grant-all-is-bad-2010-08-06/>

<http://effectivemysql.com/presentation/mysql-idiosyncrasies-that-bite/>

# Secure Communications

- SSL for replication
- SSL for client connections
- SSL for admin connections
- Encryption on the wire

<https://dev.mysql.com/doc/refman/5.6/en/secure-connections.html>

<https://dev.mysql.com/doc/refman/5.7/en/secure-connections.html>

# Secure Communications

```
[mysqld]  
ssl-ca=ca.pem  
ssl-cert=server-cert.pem  
ssl-key=server-key.pem
```

# SSL Protocols and Ciphers

```
mysql> SHOW SESSION STATUS LIKE 'Ssl_version';
```

```
+-----+-----+
```

```
| Variable_name | Value |
```

```
+-----+-----+
```

```
| Ssl_version   | TLSv1 |
```

```
+-----+-----+
```

```
mysql> SHOW SESSION STATUS LIKE 'Ssl_cipher';
```

```
+-----+-----+
```

```
| Variable_name | Value |
```

```
+-----+-----+
```

```
| Ssl_cipher    | DHE-RSA-AES128-GCM-SHA256 |
```

```
+-----+-----+
```

# SSL Client Connections

<https://dev.mysql.com/doc/connector-python/en/connector-python-connectargs.html>

```
import mysql.connector
from mysql.connector.constants import ClientFlag

config = {
    'user': 'ssluser',
    'password': 'asecret',
    'host': '127.0.0.1',
    'client_flags': [ClientFlag.SSL],
    'ssl_ca': '/opt/mysql/ssl/ca.pem',
    'ssl_cert': '/opt/mysql/ssl/client-cert.pem',
    'ssl_key': '/opt/mysql/ssl/client-key.pem',
}
```

<https://dev.mysql.com/doc/connectors/en/connector-net-tutorials-ssl.html>

# Secure Connections

- `mysql_ssl_rsa_setup` in MySQL 5.7
  - This program creates the SSL certificate and key files and RSA key-pair files required to support secure connections using SSL and secure password exchange using RSA over unencrypted connections, if those files are missing.
- uses the `openssl` command



# Secure Storage

- Encryption of data at rest
  - Data (table vs tablespace)
  - Binary Logs
  - Other Logs
- Key management

# Encryption in MariaDB Server

- Encryption: **tablespace** OR **table** level encryption with support for rolling keys using the AES algorithm
  - table encryption — PAGE\_ENCRYPTION=1
  - tablespace encryption — encrypts everything including log files
- file\_key\_management\_filename, file\_key\_management\_filekey, file\_key\_management\_encryption\_algorithm
- Well documented —  
<https://mariadb.com/kb/en/mariadb/data-at-rest-encryption/>
- Tablespace/logs scrubbing: background process that regularly scans through the tables and upgrades the encryption keys
- --encrypt-tmp-files & --encrypt-binlog

# Encryption in MariaDB Server II

```
[mysqld]
plugin-load-add=file_key_management.so
file-key-management
file-key-management-filename =
/home/mdb/keys.enc
innodb-encrypt-tables
innodb-encrypt-log
innodb-encryption-threads=4
aria-encrypt-tables=1 # PAGE row format
encrypt-tmp-disk-tables=1 # this is for Aria
```

```
CREATE TABLE customer (
    customer_id bigint not null primary
    key,
    customer_name varchar(80),
    customer_creditcard varchar(20))
ENGINE=InnoDB
page_encryption=1
page_encryption_key=1;
```

# Encryption in MariaDB Server III

- Use the preset! - `/etc/my.cnf.d/enable_encryption.preset`
- MariaDB Enterprise has a plugin for Amazon Key Management Server (KMS)
  - The reality is you can just compile this for MariaDB Server
- `mysqlbinlog` has no way to read (i.e. decrypt) an encrypted binlog
- This does not work with MariaDB Galera Cluster yet (gcache is not encrypted yet), and also xtrabackup needs additional work (i.e. if you encrypt the redo log)

# Encryption in MySQL

- MySQL 5.7.11 introduces InnoDB tablespace encryption
- `early-plugin-load=keyring_file.so` in `my.cnf`
- Must use `innodb_file_per_table`
- Convert via `ALTER TABLE table ENCRYPTION='Y'`
- Data is not encrypted in the redo/undo/binary logs
- Has external key management (Oracle Key Vault)

# Secure Accounts

- Privileges
- Passwords
- Password filesystem storage

# MySQL 5.6 improvements

- Password expiry - `ALTER USER 'foo'@'localhost' PASSWORD EXPIRE;`
- Password validation plugin - `VALIDATE_PASSWORD_STRENGTH()`
- `mysql_config_editor` - store authentication credentials in an encrypted login path file named `.mylogin.cnf`
  - <http://dev.mysql.com/doc/refman/5.6/en/mysql-config-editor.html>
- Random 'root' password on install
  - `mysql_install_db --random-passwords`
  - `cat $HOME/.mysql_secret`

# PASSWORD EXPIRE

```
$ export MYSQL_PS1="\u@\h [\d]> "  
  
$ mysql -uroot -p  
root@localhost [(none)]> CREATE USER demo IDENTIFIED BY 'passw0rd1';  
$ mysql -udemo -p #passw0rd1  
  
$ mysql -uroot  
root@localhost [(none)]> ALTER USER demo PASSWORD EXPIRE;  
  
$ mysql -udemo -p #passw0rd1  
demo@localhost [(none)]> # No issue connecting  
demo@localhost [(none)]> USE test;  
ERROR 1820 (HY000): You must reset your password using ALTER USER statement before  
executing this statement.  
  
# can reset password to current value  
demo@localhost [(none)]> ALTER USER demo IDENTIFIED BY 'passw0rd1';
```



# MySQL 5.7 improvements

- Improved password expiry — automatic password expiration available, so set `default_password_lifetime` in `my.cnf`
- You can also require password to be changed every n-days
  - `ALTER USER foo@localhost PASSWORD EXPIRE INTERVAL n DAY;`
- `PASSWORD EXPIRE DEFAULT | NEVER` options
- There is also account locking/unlocking now
  - `ALTER USER foo@host ACCOUNT LOCK | UNLOCK;`

<https://dev.mysql.com/doc/refman/5.7/en/alter-user.html>

# MariaDB Server passwords

- Password validation plugin
  - <https://mariadb.com/kb/en/mariadb/development/mariadb-internals-documentation/password-validation/>
- `simple_password_check` password validation plugin
  - can enforce a minimum password length and guarantee that a password contains at least a specified number of uppercase and lowercase letters, digits, and punctuation characters.
- `cracklib_password_check` password validation plugin
  - Allows passwords that are strong enough to pass CrackLib test. This is the same test that `pam_cracklib.so` does

# Authentication in MySQL / MariaDB Server

- `auth_socket` - Authenticates against the Unix socket file, using `so_peercred`
- `sha256_password` - `default-authentication-plugin=sha256_password`, passwords never exposed as cleartext when connecting; SSL or RSA auth
- `ed25519` - Elliptic Curve Digital Signature Algorithm, same as OpenSSH
- Kerberos/GSSAPI/SSPI - User principals: `<username>@<KERBEROS REALM>`
- Active Directory (Enterprise only)
- `mysql_no_login` ( MySQL 5.7 ) - prevents all client connections to an account that uses it

# PAM authentication

## Percona Server

```
INSTALL PLUGIN auth_pam SONAME 'auth_pam.so'; INSTALL SONAME 'auth_pam';
```

```
CREATE USER byte IDENTIFIED WITH auth_pam;
```

In `/etc/pam.d/mysqld`:

```
auth      required      pam_warn.so
auth      required      pam_unix.so audit
account   required      pam_unix.so audit
```

## MariaDB Server

```
CREATE USER byte IDENTIFIED via pam USING
'mariadb';
```

Edit `/etc/pam.d/mariadb`:

```
auth      required      pam_unix.so
account   required      pam_unix.so
```

Just use `—pam-use-cleartext-plugin` for MySQL to use `mysql_cleartext_password` instead of dialog plugin

# SQL Standard Roles

- Bundles users together, with similar privileges - follows the SQL standard
- MariaDB Server 10.0 (10.1 adds that each user can have a DEFAULT ROLE)
- MySQL 8.0 DMR

```
CREATE ROLE audit_bean_counters;  
GRANT SELECT ON accounts.* to audit_bean_counters;  
GRANT audit_bean_counters to ceo;
```

[https://mariadb.com/kb/en/mariadb/roles\\_overview/](https://mariadb.com/kb/en/mariadb/roles_overview/)  
<https://dev.mysql.com/doc/refman/8.0/en/roles.html>

# Auditing Capabilities

## MySQL

- Logging account access
- Logging SQL statements
- Logging uncommon SQL patterns

## OS

- Logging account logins
- Logging sudo commands

# Auditing - Percona

Since 5.5

```
INSTALL PLUGIN audit_log SONAME 'audit_log.so';  
SHOW PLUGINS;  
SHOW GLOBAL VARIABLES LIKE 'audit%';  
  
$ more /var/lib/mysql/audit.log
```

[https://www.percona.com/doc/percona-server/5.5/management/audit\\_log\\_plugin.html](https://www.percona.com/doc/percona-server/5.5/management/audit_log_plugin.html)

<https://www.percona.com/blog/2014/05/16/introduction-to-the-percona-mysql-audit-log-plugin/>

# Auditing - MariaDB

Since 5.5

```
INSTALL PLUGIN server_audit SONAME 'server_audit';  
SET GLOBAL server_audit_logging=on;  
SHOW GLOBAL VARIABLES LIKE '%audit%';  
  
$ more /var/lib/mysql/server_audit.log
```

<https://mariadb.com/kb/en/mariadb/about-the-mariadb-audit-plugin/>



# Auditing - Other

MySQL Enterprise

<https://dev.mysql.com/doc/refman/5.5/en/audit-log.html>



Since 5.5

Mcafee

<https://github.com/mcafee/mysql-audit/>



Since 5.1

# Auditing Implementation

- MariaDB Server
  - User filtering as an additional feature via audit API extensions
  - Query cache enabled? No table records
- Percona
  - Multiple output formats: OLD, NEW, JSON, CSV
  - Filter by user, SQL command type, database,
  - Auditing can be expensive, so asynchronous/performance/semisynchronous/synchronous modes for logging - e.g. log using memory buffers, drop messages if buffers are full, or log directly to file, flush and sync at events, etc.
- McAfee Audit plugin
  - Uses offsets
- MySQL Enterprise Audit Plugin (utility: `mysqldauditgrep`)

# Firewall Implementation

MariaDB - MariaDB MaxScale Firewall Filter

- <https://mariadb.com/resources/blog/maxscale-firewall-filter>

ProxySQL Firewall <http://www.proxysql.com/>

MySQL - MySQL Enterprise Firewall

- <https://www.mysql.com/products/enterprise/firewall.html>

Datasunrise firewall for Percona

- <https://www.datasunrise.com/firewall/percona/>

#PerconaLive @bytebot @RonaldBradford

# SQL Injection

- Always using bind variables
- Escape input content
- Restricted "least" privileges
  - Do not have GRANT ALL

# Improving Database Server Access

- Restricting user access to your database server (login accounts)
  - Every physical person
    - dedicated OS login
    - dedicated database login
  - sudo restrictions (e.g. `sudo su -`)
    - Setup sudo group
    - Grant only specific commands to execute
  - Never share account details
  - Password expiry
- MFA

# Improving Database Server Access

- Restricting traffic to your database server (open ports)
- Run a software firewall
  - iptables, ufw
- You should use OS software meant to benefit security
  - SELinux / Apparmor

# Improving Database Server Access

- If you can login, and stop MySQL, you can bypass security
  - Use `--skip-grant-tables`
- If you can edit `/etc/my.cnf` you can set
  - `--init-file=/path/to/my.sql`
- If you use `--init-file`, can user modify content of named file?

# Improving Data Access

- Restrict access to datadir
  - Don't put error log here
  - Don't put socket file here
  - Don't put pid file here
- Restrict access to view mysql.user table on filesystem
  - Check out the examples of how to Hack MySQL

<http://effectivemysql.com/downloads/MySQLSecurityEssentialsPerconaLive2015.pdf>



# Installation

- Using your Linux distribution... mostly gets you MariaDB when you ask for `mysql-server`
  - Except on Debian/Ubuntu
    - However, when you get `mariadb-server`, you get an authentication plugin — `auth_socket` for “automatic logins”
  - You are asked by `debhelper` to enter a password
- You can use the APT/YUM repositories from Oracle MySQL, Percona or MariaDB
- Don't disable SELinux: `system_u:system_r:mysqlld_t:s0`

# Update Cadence

A security patch is so named because it improves security and generally addresses a means of attack of your system

- OS
- Database
- Application Stack

Why are you still running MySQL 5.5 or older?

# Deployment Security

Who has control over running deployments?

i.e. deploying code that manipulates your data or structure

An application user SHOULD NOT have CREATE, ALTER, DROP privileges

- User to write data
- User to read data
- DBA to administer data (restricted to localhost)

# Use of Docker Containers

Docker shows a disregard for security with 'root' OS logins by default

- MySQL server installation approach via exposed passwords
  - See Giuseppe's MySQL Docker operations tutorial
- Configuration is contained with container
  - Can you access the container via SSH
  - Can you copy and use container

# Reference Material

<https://www.mysql.com/why-mysql/presentations/mysql-security-best-practices/>

- MySQL Authentication and Password Policies
- MySQL Authorization and Privilege Management
- MySQL Encryption to secure sensitive data
- MySQL Enterprise Firewall to block database attacks such as an SQL Injection
- MySQL Enterprise Audit to implement policy

# MySQL Manual Security Practices

<http://dev.mysql.com/doc/refman/5.5/en/security.html>

<http://dev.mysql.com/doc/refman/5.6/en/security.html>

<http://dev.mysql.com/doc/refman/5.7/en/security.html>

## Thanks/Q&A

- Ronald Bradford: @RonaldBradford / <http://ronaldbradford.com/blog>
- Colin Charles: @bytebot / <http://www.bytebot.net/blog/>