MySQL Security for Fun and Profit

Ernie Souhrada

Senior Consultant, Percona

03 April 2014

Topics Du Jour

- Introduction
- Thinking (in)Securely
- The 30-Second Guide to Network Security
- Don't Do THAT, Do THIS !
- Security + 1 = Performance – x
- Questions?

Introduction

- Who?
  - 2 Years at Percona, 15 with MySQL, 20 in Technology
  - Digital Renaissance Man (sorry, I know it sounds pompous)
- What?
  - Security / Information Assurance in a MySQL Context
- Why?
  - Target, TJX, etc.
  - Ed Snowden
  - The NSA
  - Lessons Never Learned (Injection STILL the #1 webapp vulnerability[1])

Thinking (in)Securely

Thinking (in)Securely

- The Traditional Risk Analysis Model
  - Who wants my data?
  - What are they willing to do/spend to get it?
  - What happens if they're successful?
  - What does it cost me to protect it?
  - Trusted vs. untrusted resources
- Assume You're Already Owned

Thinking (in)Securely

BECAUSE YOU ARE.

Thinking (in)Securely

- Consider everything as potentially hostile.
  - SQL injections
  - Packet sniffing and other MITM attacks
  - Malicious data
  - Bad/malicious code
  - MySQL direct packet injection (NSA's QUANTUMSQUEEL)
    - Just because the NSA has something doesn't mean that they are the only ones that have it.

- What are the implications of this mindset?

Thinking (in)Securely

- Jimmy Hoffa runs the only secure server in the world.

Thinking (in)Securely

The threats are real, but our responses still need to be realistic.

Thinking (in)Securely

- If the NSA is interested in you, you're screwed.
- There's more to “Security” than just security.

- Performance and usability still matter.
- We don't have infinite resources.
- Not all data are of equal value.
- Maximize Security and minimize risk at minimum cost.
  - Traditional risk analysis model is still relevant.

The 30-Second Guide to Security

The 30-Second Guide to Security

- This is not a system / network security presentation, but …
  - SELinux, AppArmor. Use them.
  - Network isolation, firewalling in **both** directions.
  - Keep software patched and current.
  - Two-factor authentication, random passwords, public-key cryptography.
  - Verify data integrity.
  - Maintain tight access controls.
  - Log everything.
  - Encrypt everything.

Don't Do THAT, Do THIS!

Don't Do THAT, Do THIS!

DON'T	DO
Passwordless accounts	Passwords for every user, Delete the empty username account
old_passwords=1	PAM+LDAP, SHA256 in MySQL 5.6
Indiscriminate use of the root user, Blanket grants on *.* or `database`.*	Limited-access GRANTs
Role accounts used by people	Named-user accounts
MySQL's encryption functions	Encrypt at the application layer
Connections over cleartext	SSL, VPN tunnels

Don't Do THAT, Do THIS!

DON'T	DO
SQL_SECURITY=DEFINER	SQL_SECURITY=DEFINER SQL_SECURITY=INVOKER
VIEWs	Limited-access GRANTs
SQL_MODE=[unset]	SQL_MODE=STRICT_ALL_TABLES
sql_log_bin=OFF	binlog_format=ROW
LOAD DATA LOCAL INFILE	LOAD DATA INFILE
DNS / hostname-based ACLs	IP-based ACLs (skip_name_resolve)

Don't Do THAT, Do THIS!

DON'T	DO
Blind acceptance of user input	Sanitize, validate, whitelist, bind variables & prepared statements
Simple passwords	Long random passwords, MySQL 5.6 password quality checking
Stale accounts, static credentials	Rotate usernames and passwords, Purge unused accounts
File-based replication credentials	Crash-safe replication, START SLAVE USER / PASSWORD

Don't Do THAT, Do THIS!

DO:

- Encrypt backups.
- Checksum / verify important data.
  - In-row verification
  - Replication data integrity checking (pt-table-checksum)
- Track and audit user activity, log it elsewhere.
  - PAM + auditd + syslog
  - MariaDB audit plugin
  - Enterprise audit plugin (commercial licensees)

Don't Do THAT, Do THIS : Examples

Don't Do THAT, Do THIS : Examples

Empty username account leaks information; can be vector for a DoS.

```
mysql> show grants;
+---------------------------------------+
| Grants for @localhost                 |
+---------------------------------------+
| GRANT USAGE ON *.* TO ''@'localhost' |
+---------------------------------------+
1 row in set (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| test               |
+--------------------+
2 rows in set (0.00 sec)
```

```
mysql> use information_schema;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> pager md5sum
PAGER set to 'md5sum'
mysql> select * from tables t1;
9ff9aacbd4db78a17c83bf7ab71c71f9  -
70 rows in set (0.02 sec)

mysql> select * from tables t1 join tables t2;
1293da79477078cdc82a4c701838e6e6  -
4900 rows in set (0.06 sec)

mysql> select * from tables t1 join tables t2 join tables t3;
941ae6ac863b3672a26e973d41f3215a  -
343000 rows in set (2.37 sec)
```

Don't Do THAT, Do THIS : Examples

Old passwords vs. new passwords:

```
mysql> select old_password('foobar') AS oldpw\G
oldpw: 4655c05b05f11fab
```

This algorithm is broken[2]. See http://www.sqlhack.com

```
mysql> select password('foobar') AS newpw\G
newpw: *9B500343BC52E2911172EB52AE5CF4847604C6E5

mysql> select sha1(unhex(sha1('foobar'))) AS newpw\G
newpw: 9b500343bc52e2911172eb52ae5cf4847604c6e5
```

- Under 5.6, old passwords are disallowed by default.
  If needed, override this with **`skip_secure_auth`** in my.cnf

Don't Do THAT, Do THIS : Examples

AES_ENCRYPT() / AES_DECRYPT()

- Information leakage in log files
  - Slow query log:

```
# Time: 140330  1:36:21
# User@Host: root[root] @ localhost []
# Thread_id: 11  Schema: test  QC_hit: No
# Query_time: 0.005958  Lock_time: 0.000189  Rows_sent: 0  Rows_examined: 0
SET timestamp=1396168581;
insert into encryption (i, data) values (4, aes_encrypt('plaintext','key'));
```

  - General log:

```
140330  1:52:45    14 Query     insert into encryption (i, data) values (4, aes_encrypt('plaintext','key'))
```

Don't Do THAT, Do THIS : Examples

AES_ENCRYPT() / AES_DECRYPT(), continued

- Binary log where binlog_format IN ('STATEMENT', 'MIXED')

```
#140330  1:36:21 server id 100  end_log_pos 1967        Query   thread_id=11
SET TIMESTAMP=1396168581/*!*/;
insert into encryption (i, data) values (4, aes_encrypt('plaintext','key'))
```

- No binlog leakage with RBR

```
BINLOG '
wNk3UxNkAAAAMwAAAEEIAAAAACIAAAAAAAEABHRlc3QACmVuY3J5cHRpb24AAgP8AQIC
wNk3UxdkAAAANAAAAHUIAAAAACIAAAAAAAEAAv/8AAAABAAOOszOnrRsrfOaUdZGipM8Q==
'/*!*/;
### INSERT INTO `test`.`encryption`
### SET
###   @1=0 /* INT meta=0 nullable=0 is_null=0 */
###   @2='▯▯3:z0▯▯iGY\x1a*L▯' /* BLOB/TEXT meta=2 nullable=1 is_null=0 */
```

- No binlog leakage with SBR and local variable.

Don't Do THAT, Do THIS : Examples

AES_ENCRYPT() / AES_DECRYPT(), re-continued

```
SET @cipher := AES_ENCRYPT('plaintext', 'key');
INSERT INTO encryption (i, data) VALUES (3, @cipher);
```

```
# at 1071
#140330  1:27:15 server id 100  end_log_pos 1126       User_var
SET @`cipher`:=_binary X'4AF7B9C80CAA006CAA48935386D983C6' COLLATE `binary`/*!*/;
# at 1126
#140330  1:27:15 server id 100  end_log_pos 1241       Query   thread_id=9     exec_time=0     error_code=0
SET TIMESTAMP=1396168035/*!*/;
insert into encryption (i, data) values (3, @cipher)
```

- Slow log and general log still leak information.

Don't Do THAT, Do THIS : Examples

Blind acceptance of user input

SQL Injection:

```
$sql = 'SELECT FROM users WHERE username=' + $username + "'";
```

Suppose `$username` is:

```
A'; DROP TABLE users; SELECT 1 FROM DUAL WHERE 't'='t
```

Then the SQL becomes:

```
SELECT FROM users WHERE username='A';
DROP TABLE users;
SELECT 1 FROM DUAL WHERE 't'='t'
```

Don't Do THAT, Do THIS : Examples

Blind acceptance of user input, continued

Whitelisting input characters:

```
$username = "A'; DROP TABLE users; SELECT 1 FROM DUAL WHERE 't'='t";
$username =~ s/\W//g;
$sql= 'SELECT * FROM users WHERE username=' + $username + "'";
    -> SELECT * FROM users WHERE
        username='ADROPTABLEusersSELECT1FROMDUALWHEREt=t'
```

Parameterized statements and bind variables:

```
$stmt = $db->prepare('SELECT * FROM users WHERE username=?');
$stmt->execute($username);
```

Don't Do THAT, Do THIS : Examples

Blind acceptance of user input, re-continued

Stored XSS and script injection:

```
$comment_text = '<script language=”javascript”>
                    alert(“gotcha!”);
                    </script>';

$stmt = dbh->prepare('INSERT INTO comments (id, data) VALUES (?,?)');
$stmt->execute(1, $comment_text);
```

– some other script –

```
$comment_text = $dbh->selectrow_array(“SELECT data FROM comments WHERE id=1”);
print $comment_text;
```

Marginally acceptable: Sanitize/escape at output time.
Much better: Sanitize/escape at input time.
Optimal: Sanitize on both ends.

Don't Do THAT, Do THIS : Examples

Password quality and complexity

- Enforcement with PAM authentication plugin
  - Only option prior to MySQL 5.6
- Password quality plugin:

```
mysql> INSTALL PLUGIN validate_password SONAME 'validate_password.so'
```

```
mysql> pager grep password
PAGER set to 'grep password'
mysql> show plugins;
| mysql_native_password       | ACTIVE   | AUTHENTICATION     | NULL                | GPL     |
| mysql_old_password          | ACTIVE   | AUTHENTICATION     | NULL                | GPL     |
| sha256_password             | ACTIVE   | AUTHENTICATION     | NULL                | GPL     |
| validate_password           | ACTIVE   | VALIDATE PASSWORD  | validate_password.so | GPL     |
48 rows in set (0.00 sec)

mysql> show variables like 'validate_password%';
| validate_password_dictionary_file    |        |
| validate_password_length             | 8      |
| validate_password_mixed_case_count   | 1      |
| validate_password_number_count       | 1      |
| validate_password_policy             | MEDIUM |
| validate_password_special_char_count | 1      |
6 rows in set (0.00 sec)
```

Don't Do THAT, Do THIS : Examples

Auditing and Logging

- PAM authentication plugin with local accounts (login only):

```
$ sudo cat /etc/pam.d/mysqld
auth       required     pam_warn.so
auth       required     pam_unix.so audit
account    required     pam_unix.so audit

mysql> INSTALL PLUGIN auth_pam SONAME 'auth_pam.so';
mysql> CREATE USER 'ews'@'localhost' IDENTIFIED WITH auth_pam;

$ sudo chgrp mysql /etc/shadow
$ sudo chmod g+r /etc/shadow
$ mysql -u ews -pCORRECT_PASSWORD
Mar 30 06:03:35 pxc1 mysqld: pam_warn(mysqld:auth): function=[pam_sm_authenticate]
service=[mysqld] terminal=[<unknown>] user=[ews] ruser=[ews] rhost=[localhost]

$ mysql -u ews -pINCORRECT_PASSWORD
Mar 30 06:03:42 pxc1 mysqld: pam_warn(mysqld:auth): function=[pam_sm_authenticate]
service=[mysqld] terminal=[<unknown>] user=[ews] ruser=[ews] rhost=[localhost]
Mar 30 06:03:42 pxc1 unix_chkpwd[4028]: password check failed for user (ews)
Mar 30 06:03:42 pxc1 mysqld: pam_unix(mysqld:auth): authentication failure; logname=
uid=497 euid=497 tty= ruser=ews rhost=localhost  user=ews
```

Don't Do THAT, Do THIS : Examples

Auditing and Logging

- User/Query Auditing with MariaDB Audit Plugin (Windows version available! Not currently 5.6 compatible.)

```
mysql> install plugin server_audit SONAME 'server_audit.so';
mysql> show global variables like 'server_audit%';
+-------------------------------+-----------------------+
| Variable_name                 | Value                 |
+-------------------------------+-----------------------+
| server_audit_events           |                       |
| server_audit_excl_users       |                       |
| server_audit_file_path        | server_audit.log      |
| server_audit_file_rotate_now  | OFF                   |
| server_audit_file_rotate_size | 1000000               |
| server_audit_file_rotations   | 9                     |
| server_audit_incl_users       |                       |
| server_audit_logging          | OFF                   |
| server_audit_mode             | 1                     |
| server_audit_output_type      | file                  |
| server_audit_syslog_facility  | LOG_USER              |
| server_audit_syslog_ident     | mysql-server_auditing |
| server_audit_syslog_info      |                       |
| server_audit_syslog_priority  | LOG_INFO              |
+-------------------------------+-----------------------+
```

Don't Do THAT, Do THIS : Examples

Auditing and Logging

- MariaDB Audit Log Plugin, continued (log to remote syslog server)

```
mysql> SET GLOBAL server_audit_output_type='syslog';
mysql> SET GLOBAL server_audit_syslog_facility='LOG_LOCAL6';
mysql> SET GLOBAL server_audit_logging='ON';

On MySQL server:
root# echo 'local6.*    @@10.10.10.18:514' >> /etc/rsyslog.conf

On syslog server:
root# echo 'local6.*    /var/log/mysql-audit.log' >> /etc/rsyslog.conf
```

```
Mar 30 08:07:39 raven mysql-server_auditing:  raven.deadbunny.lan,root,localhost,421,17,QUERY,,'set global server_audit_logging=\'ON\'',0
Mar 30 08:08:24 raven mysql-server_auditing:  raven.deadbunny.lan,repl,10.10.10.130,432,0,FAILED_CONNECT,,,1045
Mar 30 08:08:24 raven mysql-server_auditing:  raven.deadbunny.lan,repl,10.10.10.130,432,0,DISCONNECT,,,0
Mar 30 08:08:56 raven mysql-server_auditing:  raven.deadbunny.lan,root,localhost,421,17,QUERY,,'show slave status',0
Mar 30 08:09:24 raven mysql-server_auditing:  raven.deadbunny.lan,repl,10.10.10.130,433,0,FAILED_CONNECT,,,1045
Mar 30 08:09:24 raven mysql-server_auditing:  raven.deadbunny.lan,repl,10.10.10.130,433,0,DISCONNECT,,,0
Mar 30 08:10:24 raven mysql-server_auditing:  raven.deadbunny.lan,repl,10.10.10.130,434,0,FAILED_CONNECT,,,1045
Mar 30 08:10:24 raven mysql-server_auditing:  raven.deadbunny.lan,repl,10.10.10.130,434,0,DISCONNECT,,,0
Mar 30 08:17:08 raven mysql-server_auditing:  raven.deadbunny.lan,root,localhost,421,0,DISCONNECT,,,0
Mar 30 08:26:54 raven mysql-server_auditing:  raven.deadbunny.lan,root,localhost,435,0,CONNECT,,,0
Mar 30 08:26:54 raven mysql-server_auditing:  raven.deadbunny.lan,root,localhost,435,30,QUERY,,'select @@version_comment limit 1',0
Mar 30 08:26:54 raven mysql-server_auditing:  raven.deadbunny.lan,root,localhost,435,31,QUERY,,'select USER()',0
Mar 30 08:27:12 raven mysql-server_auditing:  raven.deadbunny.lan,root,localhost,435,32,QUERY,,'select * from mysql.user',0
```

Don't Do THAT, Do THIS : Examples

Encryption in flight

- Traditional SSL connections
  - Certificate acquisition process identical to SSL for websites
  - Varying levels of certificate specificity requirements (lowest to highest):
    - `CREATE USER … REQUIRE SSL (encryption required)`
    - `CREATE USER … REQUIRE X509 (valid certificate required)`
    - `CREATE USER … REQUIRE CIPHER (specific cipher or set of ciphers)`
    - `CREATE USER … REQUIRE ISSUER (specific CA)`
    - `CREATE USER … REQUIRE SUBJECT (specific entity definition)`
    - `CREATE USER … REQUIRE ISSUER AND SUBJECT AND CIPHER`
- SSH tunnel
  - Monitor/restart SSH sessions - autossh[3]
- Performance....

Security + 1 = Performance – x

Security + 1 = Performance – x

- Audit Mechanisms
- To SSL or Not to SSL?

Security + 1 = Performance – x

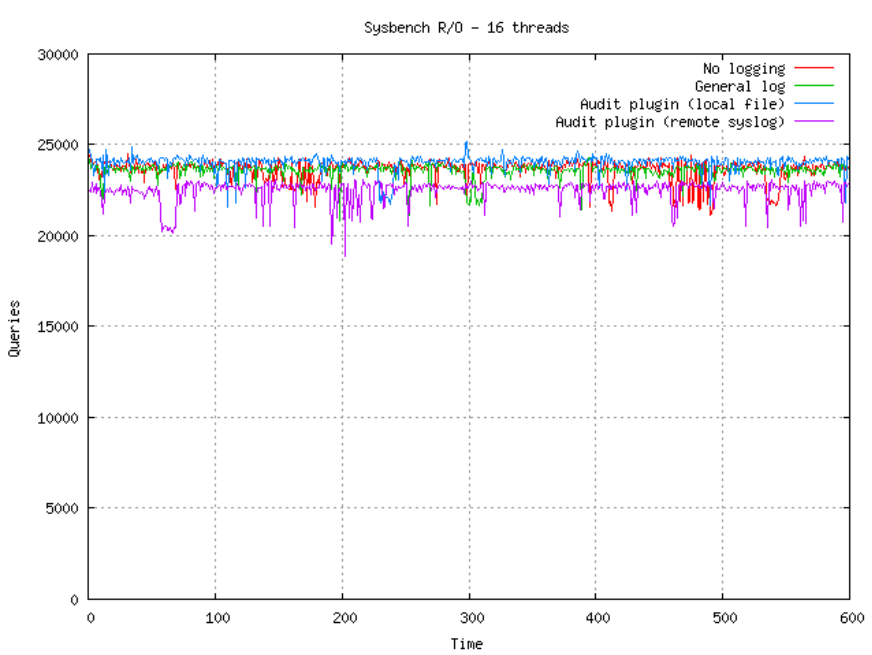
- MariaDB Audit Plugin vs. General Log
  - 10 minute sysbench OLTP read-only test with 16 threads
  - 16 tables of 1M rows each; 16M rows total, 3.9GB of data (8GB buffer pool)
  - Server: PS 5.5.36, CentOS 6.5, SSD RAID-0, 32GB RAM, 8 HT cores[4]
  - Client: Fedora 20, SSD, 32GB RAM, 8 HT cores

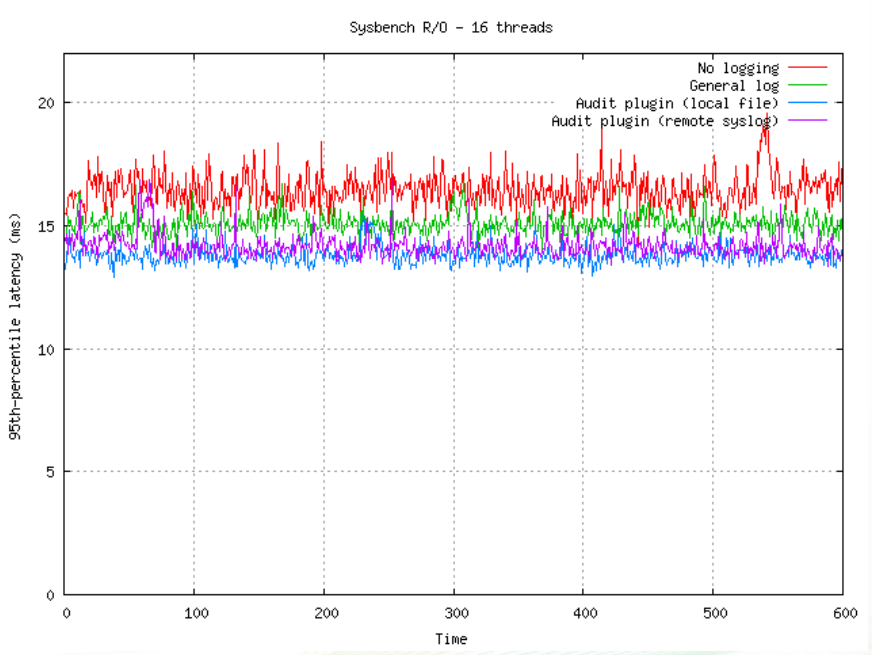
```
sysbench --test=/usr/share/sysbench/db/oltp.lua --mysql-host=10.10.10.4 \
--mysql-port=3306 --mysql-user=sbtest --mysql-password=sbtest \
--num-threads=16 --oltp-read-only=on --max-time=600 --rand-seed=31337 \
--oltp-tables-count=16 --max-requests=0 --forced-shutdown=2 \
--report-interval=1 --oltp-skip-trx=on run
```

  - Test 1: No logs of any kind (baseline)
  - Test 2: General log enabled (writing to local file)
  - Test 3: Audit plugin enabled (writing to local file)
  - Test 4: Audit plugin enabled (writing to remote file via syslog)

Security + 1 = Performance – x

- MariaDB Audit Plugin vs. General Log





Security + 1 = Performance – x

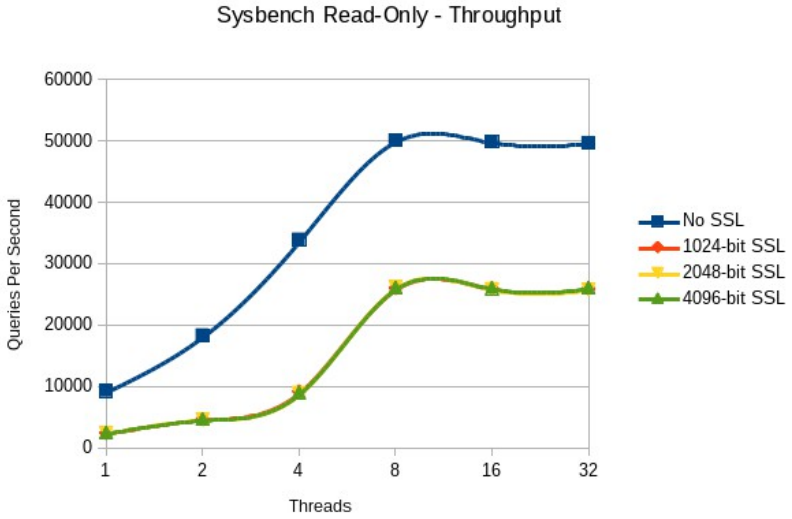
- MariaDB Audit Plugin vs. General Log:

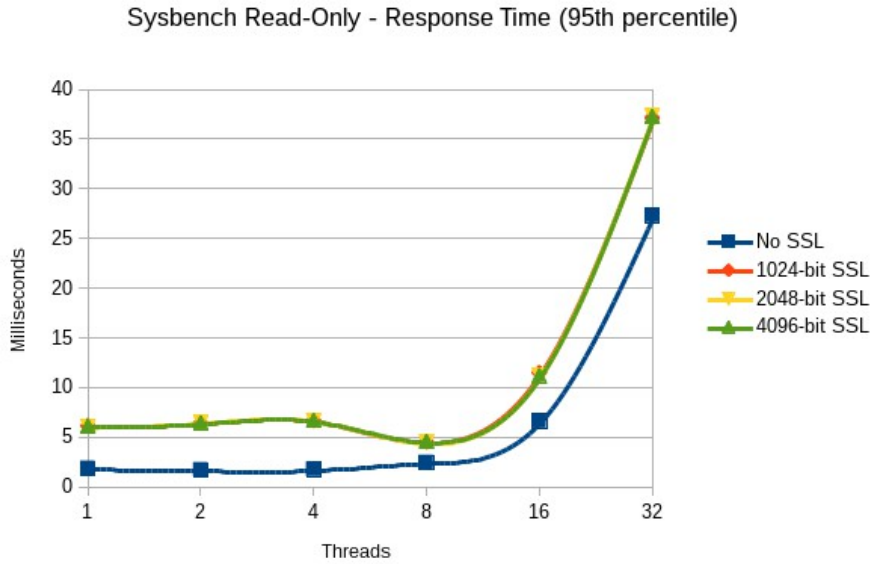
Test	QPS	Latency(ms): Min | Avg | 95th% | Max
No Log	23619.91	4.58 | 9.48 | 16.49 | 123.89
General Log	23573.93	4.65 | 9.50 | 15.09 | 134.71
Audit Plugin (Local File)	24011.38	4.34 | 9.33 | 13.81 | 117.12
Audit Plugin (Remote Syslog)	22497.01	4.64 | 9.95 | 14.20 | 117.65

- Limiting factor for remote logging is rsyslogd performance.
  - Disable rate limiting on rsyslogd or events will get lost.
  - Materially slower than the other tests.
- For local options, throughput numbers are within 2%; difference in average throughput is in the same range. I don't consider these variances statistically significant.

Security + 1 = Performance – x

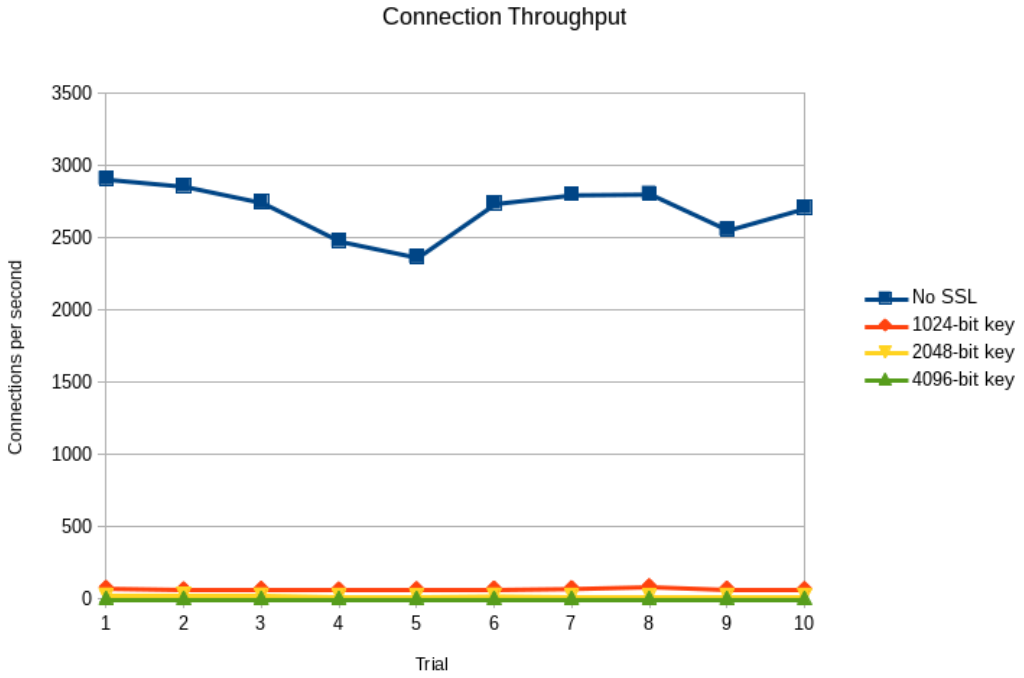
- SSL Encryption[4]
  - Sysbench R/O from 1 to 32 thread





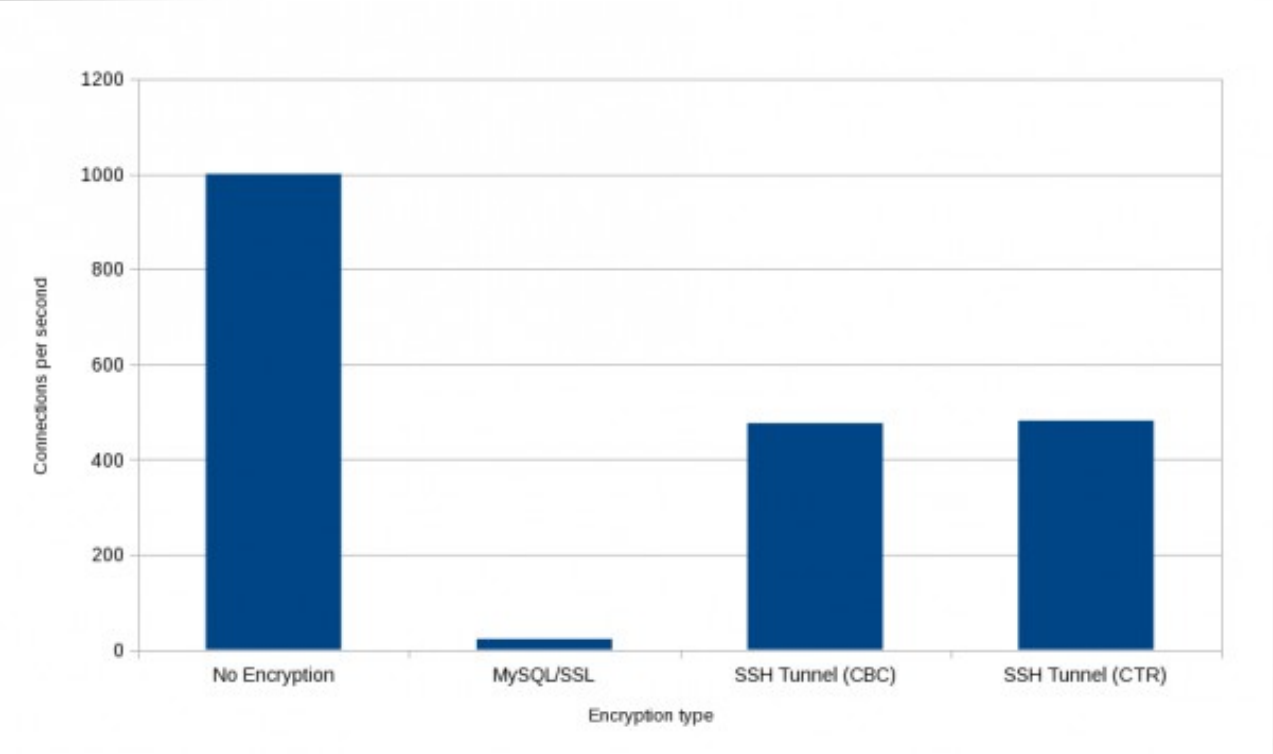
Security + 1 = Performance – x

- SSL Encryption:
  Connection overhead[4,5]



Security + 1 = Performance – x

- SSL Encryption[5]
  SSH tunnel vs. SSL
- You do what you have to do...



References

1. https://www.owasp.org/index.php/Top_10_2013-Top_10
2. http://www.sqlhack.com
3. http://www.harding.motd.ca/autossh/
4. http://www.mysqlperformanceblog.com/2013/10/10/mysql-ssl-performance-overhead/
5. http://www.mysqlperformanceblog.com/2013/11/18/mysql-encryption-performance-revisited/

Questions?

Email: ernest.souhrada@percona.com

Twitter: @denshikarasu