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

- 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
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 QUANTUMSQUEEEL)
    - 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.
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!
<table>
<thead>
<tr>
<th>DON'T</th>
<th>DO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passwordless accounts</td>
<td>Passwords for every user,</td>
</tr>
<tr>
<td></td>
<td>Delete the empty username account</td>
</tr>
<tr>
<td>old_passwords=1</td>
<td>PAM+LDAP, SHA256 in MySQL 5.6</td>
</tr>
<tr>
<td>Indiscriminate use of the root user,</td>
<td>Limited-access GRANTs</td>
</tr>
<tr>
<td>Blanket grants on <code>*.*</code> or <code>database</code>.*</td>
<td></td>
</tr>
<tr>
<td>Role accounts used by people</td>
<td>Named-user accounts</td>
</tr>
<tr>
<td>MySQL's encryption functions</td>
<td>Encrypt at the application layer</td>
</tr>
<tr>
<td>Connections over cleartext</td>
<td>SSL, VPN tunnels</td>
</tr>
</tbody>
</table>
## Don't Do THAT, Do THIS!

<table>
<thead>
<tr>
<th><strong>DON'T</strong></th>
<th><strong>DO</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL_SECURITY=DEFINER</td>
<td>SQL_SECURITY=DEFINER</td>
</tr>
<tr>
<td></td>
<td>SQL_SECURITY=INVOKER</td>
</tr>
<tr>
<td>VIEWs</td>
<td>Limited-access GRANTS</td>
</tr>
<tr>
<td>SQL_MODE=[unset]</td>
<td>SQL_MODE=STRICT_ALL_TABLES</td>
</tr>
<tr>
<td>sql_log_bin=OFF</td>
<td>binlog_format=ROW</td>
</tr>
<tr>
<td>LOAD DATA LOCAL INFILE</td>
<td>LOAD DATA INFILE</td>
</tr>
<tr>
<td>DNS / hostname-based ACLs</td>
<td>IP-based ACLs (skip_name_resolve)</td>
</tr>
</tbody>
</table>
Don't Do THAT, Do THIS!

<table>
<thead>
<tr>
<th>DON'T</th>
<th>DO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blind acceptance of user input</td>
<td>Sanitize, validate, whitelist, bind variables &amp; prepared statements</td>
</tr>
<tr>
<td>Simple passwords</td>
<td>Long random passwords, MySQL 5.6 password quality checking</td>
</tr>
<tr>
<td>Stale accounts, static credentials</td>
<td>Rotate usernames and passwords, Purge unused accounts</td>
</tr>
<tr>
<td>File-based replication credentials</td>
<td>Crash-safe replication, START SLAVE USER / PASSWORD</td>
</tr>
</tbody>
</table>
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
Empty username account leaks information; can be vector for a DoS.
Old passwords vs. new passwords:

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

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

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

```sql
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`
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'))
    ```
AES_ENCRYPT() / AES_DECRYPT(), continued

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

```sql
#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

```sql
BINLOG 'wNk3UXNkAAAEIAAAAACIAAAAAAAAAEABHRlc3QACmVuy3J5cHRpb24AAGP8AQIC
wNk3UXdkaAAANAAAAMHUIAAAAACIAAAAAAAAEAAY/8AAAAABAA0s0nrRsrFoUdZG1pM80=='
### INSERT INTO `test`.`encryption`
### SET
### @1=0 /* INT meta=0 nullable=0 is_null=0 */
### @2='003:z0661GY\xa1l6' /* BLOB/TEXT meta=2 nullable=1 is_null=0 */
```

- No binlog leakage with SBR and local variable.
AES_ENCRYPT() / AES_DECRYPT(), re-continued

SET @cipher := AES_ENCRYPT('plaintext', 'key');
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:**

```php
$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:**

```php
$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:**

```php
$stmt = $db->prepare('SELECT * FROM users WHERE username=?');
$stmt->execute($username);
```
Blind acceptance of user input, re-continued

**Stored XSS and script injection:**

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

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

```php
-some other script-
```

```php
$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.
Password quality and complexity

- Enforcement with PAM authentication plugin
  - Only option prior to MySQL 5.6
- Password quality plugin:
  ```sql
  mysql> INSTALL PLUGIN validate_password SONAME 'validate_password.so'
  ```

```sql
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)
```

```sql
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)
```
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

  $ mysql -u ews -pINCORRECT_PASSWORD
  Mar 30 06:03:42 pxc1 unix_chkpwd[4028]: password check failed for user (ews)
  Mar 30 06:03:42 pxc1 mysql: pam_unix(mysqld:auth): authentication failure; logname=uid=497 euid=497 tty= ruser=ews rhost=localhost user=ews
Auditing and Logging

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

```sql
mysql> install plugin server_audit SONAME 'server_audit.so';
mysql> show global variables like 'server_audit%';
```

<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>server_audit_events</td>
<td></td>
</tr>
<tr>
<td>server_audit_excl_users</td>
<td></td>
</tr>
<tr>
<td>server_audit_file_path</td>
<td>server_audit.log</td>
</tr>
<tr>
<td>server_audit_file_rotate_now</td>
<td>OFF</td>
</tr>
<tr>
<td>server_audit_file_rotate_size</td>
<td>1000000</td>
</tr>
<tr>
<td>server_audit_file_rotations</td>
<td>9</td>
</tr>
<tr>
<td>server_audit_incl_users</td>
<td></td>
</tr>
<tr>
<td>server_audit_logging</td>
<td>OFF</td>
</tr>
<tr>
<td>server_audit_mode</td>
<td>1</td>
</tr>
<tr>
<td>server_audit_output_type</td>
<td>file</td>
</tr>
<tr>
<td>server_audit_syslog_facility</td>
<td>LOG_USER</td>
</tr>
<tr>
<td>server_audit_syslog_ident</td>
<td>mysql-server_auditing</td>
</tr>
<tr>
<td>server_audit_syslog_info</td>
<td></td>
</tr>
<tr>
<td>server_audit_syslog_priority</td>
<td>LOG_INFO</td>
</tr>
<tr>
<td>server_audit_syslog_priority</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Time</th>
<th>User</th>
<th>Database</th>
<th>Event</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 30 08:07:39</td>
<td>raven</td>
<td>mysql</td>
<td>DISCONNECT</td>
<td>-</td>
</tr>
<tr>
<td>Mar 30 08:08:24</td>
<td>raven</td>
<td>mysql</td>
<td>FAILED_CONNECT</td>
<td>1045</td>
</tr>
<tr>
<td>Mar 30 08:08:24</td>
<td>raven</td>
<td>mysql</td>
<td>DISCONNECT</td>
<td>0</td>
</tr>
<tr>
<td>Mar 30 08:09:24</td>
<td>raven</td>
<td>mysql</td>
<td>FAILED_CONNECT</td>
<td>1045</td>
</tr>
<tr>
<td>Mar 30 08:09:24</td>
<td>raven</td>
<td>mysql</td>
<td>DISCONNECT</td>
<td>0</td>
</tr>
<tr>
<td>Mar 30 08:10:24</td>
<td>raven</td>
<td>mysql</td>
<td>FAILED_CONNECT</td>
<td>1045</td>
</tr>
<tr>
<td>Mar 30 08:10:24</td>
<td>raven</td>
<td>mysql</td>
<td>DISCONNECT</td>
<td>0</td>
</tr>
<tr>
<td>Mar 30 08:17:08</td>
<td>raven</td>
<td>mysql</td>
<td>DISCONNECT</td>
<td>0</td>
</tr>
<tr>
<td>Mar 30 08:26:54</td>
<td>raven</td>
<td>mysql</td>
<td>CONNECT</td>
<td>0</td>
</tr>
<tr>
<td>Mar 30 08:26:54</td>
<td>raven</td>
<td>mysql</td>
<td>DISCONNECT</td>
<td>0</td>
</tr>
<tr>
<td>Mar 30 08:27:12</td>
<td>raven</td>
<td>mysql</td>
<td>QUERY</td>
<td>'select * from mysql.user',0</td>
</tr>
</tbody>
</table>
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
  - Client: Fedora 20, SSD, 32GB RAM, 8 HT cores

```bash
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:

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

- 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

Questions?

Email: ernest.souhrada@percona.com
Twitter: @denshikarasu