Securing MySQL
With a Focus on SSL
Chris Conlon
Software Developer at yaSSL
Bozeman, MT
SSL Statistics

Ivan Ristic: Internet SSL Survey 2010
http://www.ssllabs.com

- Sample of 119 Million Domain Names
  0.60%, Certificate Name Matches
  **0.42%**, Counting only valid ones

- Alexa Top 1M Sites
  120,000 Use SSL (**12%**)
Part I: MySQL Security
   1. Common Attacks & Vulnerabilities
   2. Good Security Practices for MySQL

Part II: SSL/TLS
   1. Overview of SSL and TLS
   2. Configuring and Building MySQL with SSL
   3. MySQL SSL Command Options
   4. SSL Certificate Creation
   5. Performance Comparison

Part III: Additional Security Concerns
   1. Data Storage and Encryption

Part IV: Wrap-Up
   1. Licensing
   2. yaSSL
   3. Conclusion

© Copyright 2011 yaSSL
Part I

MySQL Security

MySQL Updates
Account Passwords
Test Databases
mysqlld
Privileges
Common Attacks and Vulnerabilities

Do we really need to secure our MySQL database?

YES!

MySQL is Susceptible to Many Attacks:

- Basic Attacks (empty password, etc.)
- SQL Injection Attacks
- Known MySQL Bugs and Vulnerabilities
- Trojanning MySQL
A. Keeping MySQL Version Up to Date

An easy way to stay better protected:

- New MySQL Patches, Bug Fixes, etc.
- You should take advantage of updates
'MySQL' Vulnerabilities By Year
cvedetails.com (nvd.nist.gov)
Good Security Practices for MySQL

• yaSSL Vulnerabilities affecting MySQL in the past:

  - CVE-2005-3731: Certificate Chain Processing
  - CVE-2008-0227: Denial of Service (crash)
  - CVE-2008-0226: Allowed Execution of Arbitrary Code
  - CVE-2009-4484: Allowed Execution of Arbitrary Code, Denial of Service Possible
B. Passwords: Root Accounts

- They are empty by default

Quick Check: `mysql -u root
("Welcome to the MySQL monitor" = Not Good)

```
shell> mysql -u root
mysql> UPDATE mysql.user SET Password = PASSWORD('newpwd')
   -> WHERE User = 'root';
mysql> FLUSH PRIVILEGES;
```
Good Security Practices for MySQL

B. Passwords: Anonymous Accounts

Assign anonymous accounts passwords:

shell> mysql -u root -p
   Enter password: (enter root password here)
mysql> UPDATE mysql.user SET Password = PASSWORD('newpwd')
   -> WHERE User = "";
mysql> FLUSH PRIVILEGES;

Or remove them:

shell> mysql -u root -p
   Enter password: (enter root password here)
mysql> DROP USER '@localhost';
mysql> DROP USER '@host_name';
B. Passwords: Strength is Key

Use strong passwords

- Combine letters and numbers
  - `mhallwltpic++ = "mary had a little lamb who liked to program in C++"
- `uuidgen`, `pwgen` tools
C. Securing Test Databases

- By default, anyone can access test databases
  - Convenient for testing - not production

- Delete databases or restrict privileges

```
shell> mysql -u root -p
    Enter password: (enter root password here)
mysql> DELETE FROM mysql.db WHERE Db LIKE 'test%';
mysql> FLUSH PRIVILEGES;
```
D. Securing mysqld

- Don't run MySQL as root user
  
  ```shell
  shell> mysqld --user=mysql
  ```

- Disable Remote Access (`--skip-networking`)
  - Only allows access from local machine
Good Security Practices for MySQL

E. *mysql_secure_installation* script

Allows you to:

- Set a password for root account
- Remove root accounts that are accessible from outside of the local host
- Remove anonymous user accounts
- Remove the test database that can be accessed from all users
- Reload privilege tables so that above take effect

* Not available on Windows
F. Notes about Privileges

• Don't grant all users **PROCESS** or **SUPER** privilege
  - Can see text of currently-executing queries
    (SHOW processlist; )

• Don't grant all users the **FILE** privilege
  - Enables reading/writing to file system wherever mysqld process has access
G. Additional Measures

These depend on your unique situation:

• Restrict access to **log files**
  - Ensure only ‘root’ and the mysqld user can access

• Restrict MySQL **data directory** access only to server account
G. Additional Measures

These depend on your unique situation:

- Add **Application-specific** Users
  - Each user only has required privileges
    (Ex: Ruby/PHP/etc. Application)

- Restrict where MySQL **listens**
  - You might only need to listen on localhost
    bind-address = 127.0.0.1
G. Additional Measures

These depend on your unique situation:

- Can disable **LOAD DATA LOCAL INFILE** command
  - Can allow reading of local files

- Remove Content of MySQL **History File**
  - All executed SQL commands are stored
    ```
    cat /dev/null > ~/.mysql_history
    ```
Part II

SSL / TLS

Overview
X.509 Certificates
Handshake
MySQL and SSL
By default, MySQL uses unencrypted connections between the client and server!
A. What is SSL / TLS?

- Originally developed by Netscape
- Layered between Transport (TCP/IP) and Application layers:

```
Network Layer
  ▼
  Network Access
  ▼
  Internet Layer
  ▼
  Transport Layer
  ▼
  Application Layer

SSL Record Layer
  ▼
SSL Alert Protocol
  ▼
SSL Change Cipher Spec Protocol
  ▼
SSL Handshake Protocol
```

Protocols Secured by SSL/TLS:

- HTTP
- SMTP, etc.
- HTTP
- LDAP, etc.

© Copyright 2011 yaSSL
A. What is SSL / TLS?

• Provides secure client/server communication, including:

- **Privacy**
  - Prevent eavesdropping

- **Authentication**
  - Prevent impersonating

- **Integrity**
  - Prevent modification

• Can be implemented on almost any OS that support TCP/IP
A. What is SSL / TLS?

- Uses a variety of encryption algorithms to secure data:

  - Hashing Functions: MD2, MD4, MD5, SHA-1, SHA-2, RIPEMD
  - Block and Stream Ciphers: DES, 3DES, AES, ARC4, RABBIT, HC-128
  - Public Key Options: RSA, DSS, DH
A. What is SSL / TLS?

- These algorithms are negotiated during the SSL handshake
- Are combined into a "Cipher Suite":

Examples:

```plaintext
SSL_RSA_WITH_DES_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
```
B. X.509 Certificate Concepts

- Elements in "Public Key Infrastructure (PKI)"
- Acts as a container for public key (used to verify/validate end entities)
- Digitally-signed by a trusted authority
- Buy (CA) vs. Create Your Own (Self-Sign)

Verisign, DigiCert, Thawte, etc.
- Costs Money
- Trusted

Created Yourself
- Free
- Trusted (if you control both sides)
C. SSL Handshake

- Simplified Diagram

1. Client Hello
   - Cryptographic Info
     - (SSL version, supported ciphers, etc.)

2. Server Hello
   - Cipher Suite
   - Server Certificate
   - Server Key Exchange (public key)
   - (Client Certificate Request)
   - Server Hello Done

3. Verify server cert, check crypto parameters

4. Client Key Exchange
   - (Certificate Verify)
   - (Client Certificate)

5. Verify client cert (if required)

6. Change Cipher Spec
   - Client Finished

7. Change Cipher Spec
   - Server Finished

8. Exchange Messages (Encrypted)
D. SSL is Everywhere

SSL is used in a wide range of applications
- Browsers
- Email
- Routers
- Factory Automation
- VoIP
- Automobile Communications
- Sensors
- Smart Power Meters

And much more!!
E. SSL in MySQL?

- Your system must support either OpenSSL or yaSSL
- MySQL must be built with SSL support

Note: MySQL is bundled with yaSSL
Checking for SSL

- Confirm that user in 'mysql' database includes SSL-related columns:
  - Beginning with: `ssl_`, `x509_`

- Check if binary is compiled with SSL support:

  ```
  shell> mysqld --ssl --help
  060525 14:18:52 [ERROR] mysqld: unknown option '--ssl'
  ```

- `mysqld`: Check for 'have_ssl' system variable
Configure MySQL to use the built-in SSL (yaSSL):

shell> cmake . -DWITH_SSL=bundled

-DWITH_SSL options:

  no: No SSL support (default)
  yes: Use system SSL library if present, else bundled library
  bundled: SSL library bundled with distro (yaSSL)
  system: Use the system SSL library

** yaSSL on Unix requires /dev/urandom and /dev/random to be available
Overview of SSL / TLS

To allow client connections through SSL, start MySQL with the appropriate options:

```
shell> mysqld_safe --user=mysql \\
    --ssl-ca=ca-cert.pem \\
    --ssl-cert=server-cert.pem \\
    --ssl-key=server-key.pem
```

---

**--ssl-ca**: Identifies the certificate authority certificate

**--ssl-cert**: Identifies the server public key

**--ssl-key**: Identifies the server private key

© Copyright 2011 yaSSL
Client connecting securely:

I. Account created with GRANT statement including REQUIRE_SSL:

```
shell> mysql -u user -p --ssl-ca=ca-cert.pem
```

II. Account created with REQUIRE_X509 in addition:

```
shell> mysql -u user -p --ssl-ca=ca-cert.pem \
    --ssl-cert=client-cert.pem \
    --ssl-key=client-key.pem
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Cmd-Line</th>
<th>Option File</th>
<th>System Var</th>
<th>Var Scope</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>have_openssl</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Global</td>
<td>No</td>
</tr>
<tr>
<td>have_ssl</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Global</td>
<td>No</td>
</tr>
<tr>
<td>skip-ssl</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl-ca</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl-capath</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl-cert</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl-cipher</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl-key</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl-verify-server-cert</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MySQL SSL Command Options

have_openssl

have_ssl

YES = mysqld supports SSL connections
DISABLED = server was compiled with SSL support, not enabled (--ssl-xxx)

Check:
SHOW VARIABLES LIKE 'have%ssl';
MySQL SSL Command Options

skip-ssl

Indicate that SSL should not be used
Same as using --ssl=0

ssl

Server: Specifies that the server permits SSL connections
Client: Permits a client to connect to server using SSL
MySQL SSL Command Options

ssl-ca

The path to the file containing list of trusted CAs

ssl-capath

The path to a directory containing trusted CAs (PEM format)
MySQL SSL Command Options

**ssl-cert**

Name of the SSL certificate to be used

**ssl-cipher**

A list of permissible ciphers to use for SSL

--ssl-cipher=AES128-SHA
--ssl-cipher=DHE-RSA_AES256-SHA:AES128-SHA
MySQL SSL Command Options

ssl-key

Name of the SSL key file

ssl-verify-server-cert

- Clients only
- Server's Common Name verified against server host name
- Connection rejected if no match

© Copyright 2011 yaSSL
SSL Certificate Creation

A. Generating Certificates

1. Create CA certificate (private key, public cert)
2. Create server key
3. Create server certificate
4. Create client key
5. Create client certificate
A. Generating Certificates

Create CA certificate (private key, public cert)

```bash
shell> openssl genrsa 2048 > ca-key.pem

shell> openssl req -new -x509 -nodes -days 1000 -key ca-key.pem > ca-cert.pem
```
A. Generating Certificates

Create server key and certificate

shell> openssl req -newkey rsa:2048 -days 1000 \  -nodes -keyout server-key.pem > server-req.pem

shell> openssl x509 -req -in server-req.pem -days 1000 \  -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 > server-cert.pem
A. Generating Certificates

Create client key and certificate

shell> openssl req -newkey rsa:2048 -days 1000 \   -nodes -keyout client-key.pem > client.req.pem

shell> openssl x509 -req -in client.req.pem -days 1000 \   -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 > client-cert.pem
A. Generating Certificates

Remove passphrase from client/server key:

```shell
shell> openssl rsa -in client-key.pem -out client-key.pem
shell> openssl rsa -in server-key.pem -out server-key.pem
```
Performance Comparison of SSL

Test Machine

MacBook Pro

2.33 GHz
2 GB 667 MHz DDR2 SDRAM
Mac OS X 10.6.6 (Snow Leopard)
Performance Comparison of SSL

Footprint Size
Performance Comparison of SSL

Command:
du -sh .

Result:
5.3% Difference
Performance Comparison of SSL

MySQL Footprint Comparison (Detail)
SSL vs. No SSL

<table>
<thead>
<tr>
<th></th>
<th>SSL</th>
<th>No SSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin</td>
<td>86</td>
<td>79</td>
</tr>
<tr>
<td>lib</td>
<td>13</td>
<td>9.2</td>
</tr>
</tbody>
</table>

Command:
```
du -sh *
```
Performance Comparison of SSL

Average Query Times

(SELECT Queries, sysbench)
Performance Comparison of SSL

MySQL Average SELECT Query Times
No SSL vs. SSL
100,000 Requests
sysbench

Average Query Time (ms)

Concurrency (# of Client Connections)
Performance Comparison of SSL

MySQL Average SELECT Query Times (ms)

No SSL vs. SSL
100,000 Requests
sysbench

<table>
<thead>
<tr>
<th>Concurrency (# of Client Connections)</th>
<th>No SSL</th>
<th>SSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>0.14</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>0.14</td>
</tr>
<tr>
<td>4</td>
<td>0.21</td>
<td>0.29</td>
</tr>
<tr>
<td>8</td>
<td>0.65</td>
<td>0.76</td>
</tr>
<tr>
<td>16</td>
<td>1.33</td>
<td>1.62</td>
</tr>
<tr>
<td>32</td>
<td>2.67</td>
<td>3.32</td>
</tr>
</tbody>
</table>

© Copyright 2011 yaSSL
Performance Comparison of SSL

MySQL Average SELECT Query Times
No SSL vs. SSL
100,000 Requests
sysbench

Client Concurrency = 8

<table>
<thead>
<tr>
<th>Average Query Time (ms)</th>
<th>No SSL</th>
<th>SSL</th>
<th>16.9% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.65</td>
<td>0.76</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Copyright 2011 yaSSL
Part III

Additional Security Concerns

Data Encryption
A. Why should you be interested in this?

- Corporate networks are becoming increasingly open to the outside
- Network is regarded as being inherently insecure
- Encrypting data is the best option
  
  "Last Line of Defense"

- Data exposure can be costly, damaging, embarrassing
B. Client Side Encryption

- Encrypt data in code before it is passed to MySQL
- Many encryption modules available (PHP, Perl, etc.)

Advantages
- Data encrypted between code & MySQL
- Allows the use of bin logging (MySQL backup/replication)

Disadvantages
- What to do with the key?
C. Server Side Encryption

- **AES_ENCRYPT()**, **AES_DECRYPT()** functions
  - AES-128 Default
  - AES-256 w/ source-code change

- Entire Disk Encryption

- Transparent Data Encryption (Gazzang ezNcrypt)
Gazzang ezNcrypt

- ezNcrypt sits between your storage engine and file system to encrypt your data before it hits the disk.
- Traditionally called - Transparent Data Encryption (TDE)
  - The data is encrypted transparently, no changes are needed to your application, code or MySQL.

Application SQL
“insert into orders (number, credit card,...)
Values (20090101,4307,...)”

File System
orders.myd
9f7c7d77a87
7fg8e78s09ab

Table Orders
20090101,4307
Gazzang ezNcrypt

Addresses Problems such as

- Unauthorized attempts to read data off the database files, like SSN, Credit Card numbers or phone numbers
- Theft of the data files and intellectual property
- Tampering of data, directly modifying values in files
- Protection of tapes, backups and Data at Rest
- Protecting disks in the case the physical hardware is stolen or incorrectly disposed
Gazzang ezNcrypt

ezNcrypt Database Protection

- The database is protected from all OS users
- Any user including root that does not have the key cannot unlock the data.

- The MySQL process is the only authorized to retrieve the Key to unlock the database data

© Copyright 2011 yaSSL
Data Storage and Encryption

Gazzang ezNcrypt

- Gazzang Key Storage System (KSS)
C. Server Side Encryption

**Advantages:**
- Data is stored encrypted
- Easy to use

**Disadvantages:**
- bin logging (all queries are shown in plain text)

  *Exception*: Gazzang can protect the bin logs

- What to do with the key?
Part IV

Wrap-Up

Licensing Concerns
About yaSSL
Licensing Concerns

A. yaSSL vs. OpenSSL

- OpenSSL uses BSD-style license with announcement clause
- Makes it incompatible with GPL
- yaSSL = dual licensed (GPL, Commercial)

B. FLOSS Exception

- Permits GPL library to be used with FLOSS application
  
  “Free/Libre and Open Source Software”
yet another SSL (yaSSL)

- **Founded:** 2004
- **Location:** Bozeman, Seattle, Portland
- **Our Focus:** Open Source Embedded Security for Devices
- **Products:**
  - CyaSSL, yaSSL
  - yaSSL Embedded Web Server

© Copyright 2011 yaSSL
Part I: MySQL Security
  1. Common Attacks & Vulnerabilities
  2. Good Security Practices & Policies for MySQL

Part II: SSL/TLS
  1. Overview of SSL and TLS
  2. Configuring and Building MySQL with SSL
  3. MySQL SSL Command Options
  4. SSL Certificate Creation
  5. Performance Comparison

Part III: Additional Security Concerns
  1. Data Storage and Encryption
Thanks!

http://www.yassl.com

Email: info@yassl.com
     chris@yassl.com

Phone: (206) 369-4800
Helpful Sources

MySQL Manual:
http://dev.mysql.com/doc/refman/5.5/en/

MySQL Security Resources around the Internet
http://www.symantec.com/connect/articles/secure-mysql-database-design

SSL/TLS
https://www.ssllabs.com/