



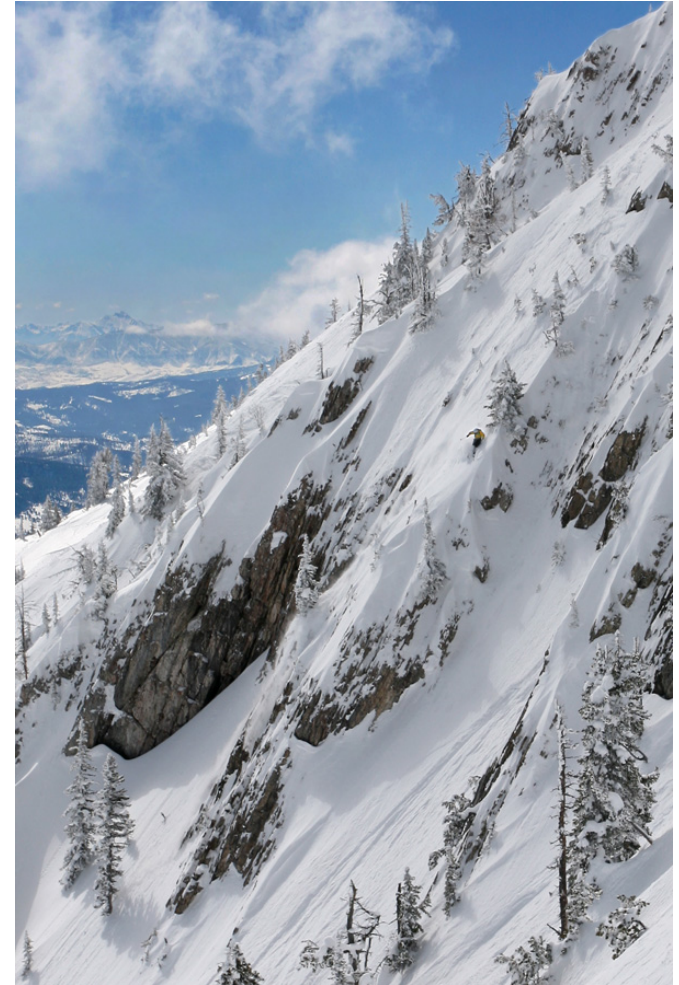
# Securing MySQL

With a Focus on SSL

# About Me

## Chris Conlon

Software Developer at yaSSL  
Bozeman, MT



© Copyright 2011 FishEyeGuyPhotography

# SSL Statistics

## Ivan Ristic: Internet SSL Survey 2010

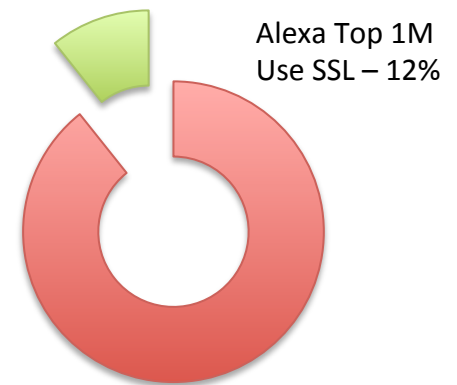
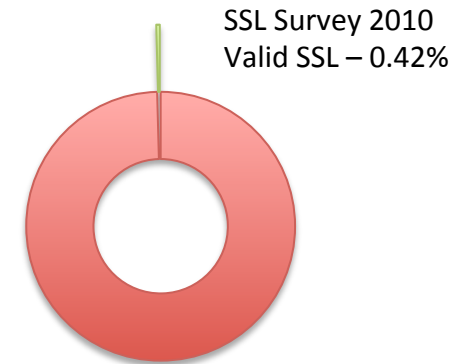
<http://www.ssllabs.com>

- 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%)



# Presentation Outline

## 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

# Part I

## MySQL Security

MySQL Updates  
Account Passwords  
Test Databases  
mysqld  
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

# Good Security Practices for MySQL

## A. Keeping MySQL Version Up to Date

An easy way to stay better protected:

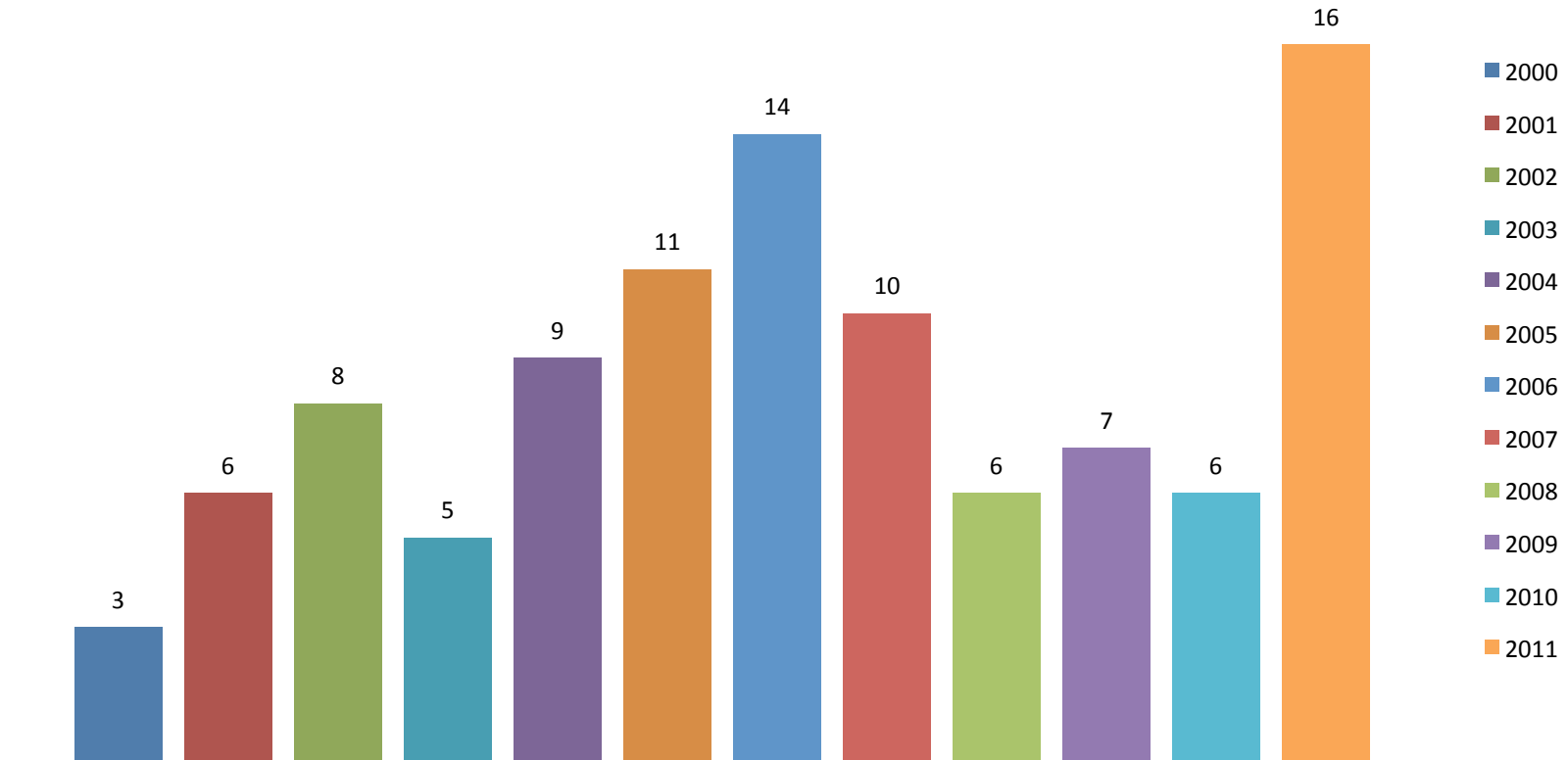
- New MySQL Patches, Bug Fixes, etc.
- You should take advantage of updates



# Good Security Practices for MySQL

## '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

# Good Security Practices for MySQL

## 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';
```

# Good Security Practices for MySQL

## 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

# Good Security Practices for MySQL

## 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;
```

# Good Security Practices for MySQL

## D. Securing mysqld

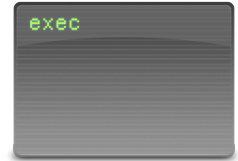
- Don't run MySQL as root user

```
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*

# Good Security Practices for MySQL

## 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



# Good Security Practices for MySQL

## 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



# Good Security Practices for MySQL

## 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`



# Good Security Practices for MySQL

## 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`

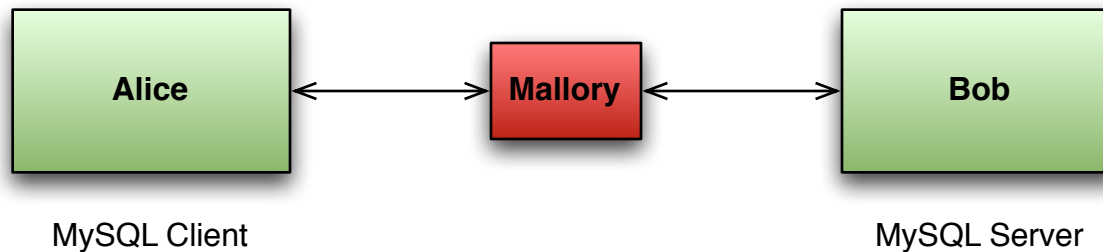
# Part II

## SSL / TLS

Overview  
X.509 Certificates  
Handshake  
MySQL and SSL

# Overview of SSL / TLS

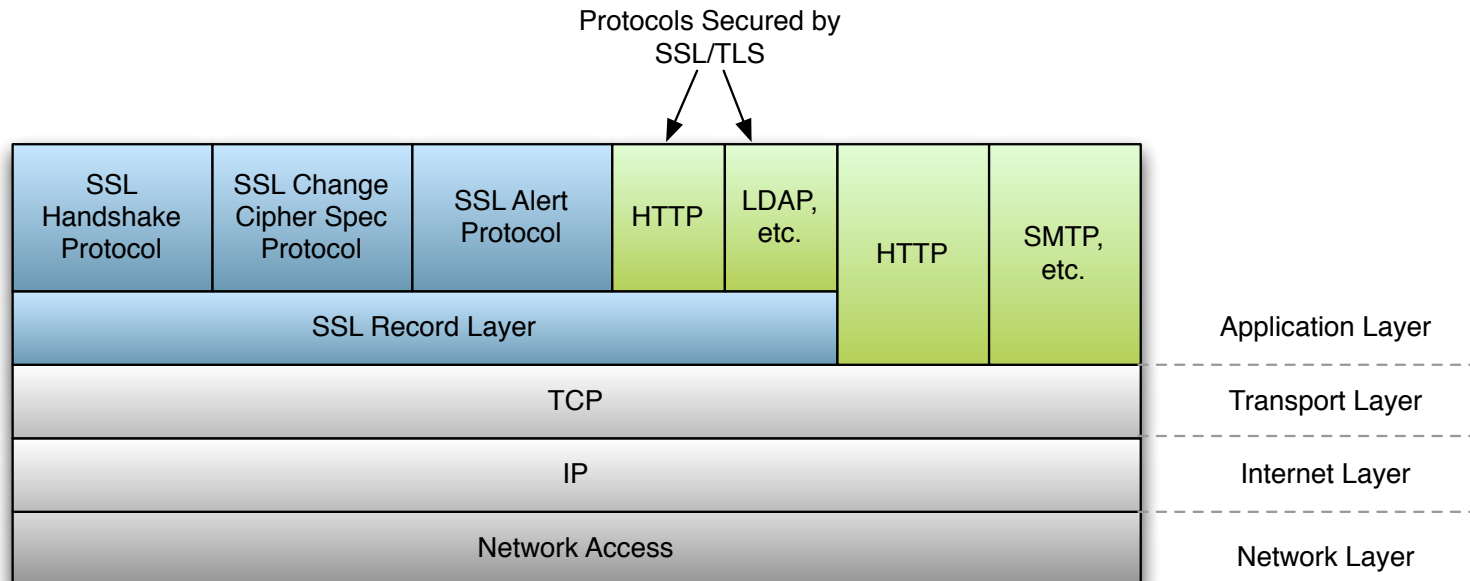
By default, MySQL uses **unencrypted connections** between the client and server!



# Overview of SSL / TLS

## A. What is SSL / TLS?

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



# Overview of SSL / TLS

## A. What is SSL / TLS?

- Provides secure client/server communication, including:

<b>Privacy</b>	Prevent <b>eavesdropping</b>
<b>Authentication</b>	Prevent <b>impersonating</b>
<b>Integrity</b>	Prevent <b>modification</b>

- Can be implemented on almost any OS that support TCP/IP

# Overview of SSL / TLS

## A. What is SSL / TLS?

- Uses a variety of encryption algorithms to secure data:

MD2, MD4, MD5, SHA-1, SHA-2, RIPEMD -----

DES, 3DES, AES, ARC4, RABBIT, HC-128 -----

RSA, DSS, DH -----

Hashing Functions

Block and Stream Ciphers

Public Key Options



# Overview of SSL / TLS

## A. What is SSL / TLS?

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


### Examples:

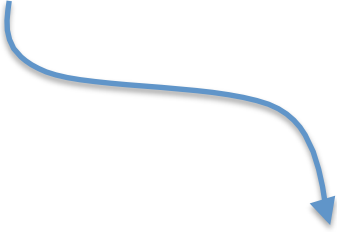
```
SSL_RSA_WITH_DES_CBC_SHA  
SSL_DHE_RSA_WITH_DES_CBC_SHA  
TLS_RSA_WITH_AES_128_CBC_SHA  
TLS_DHE_DSS_WITH_AES_128_CBC_SHA  
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
```

# Overview of SSL / TLS

## 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)

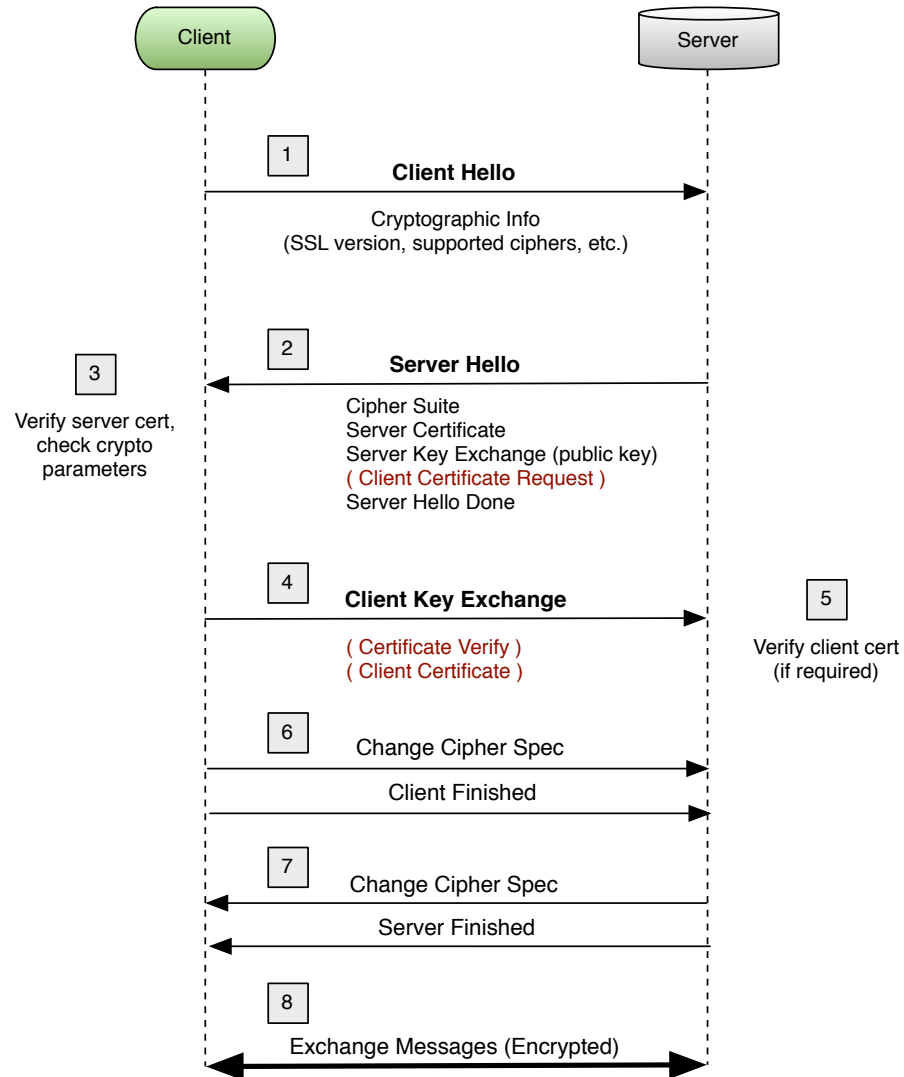
- 
- VeriSign, DigiCert, Thawte, etc.
  - Costs Money
  - Trusted

- 
- Created Yourself
  - Free
  - Trusted (if you control both sides)

# Overview of SSL / TLS

## C. SSL Handshake

### - Simplified Diagram



# Overview of SSL / TLS

## 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!!**



# Overview of SSL / TLS

## 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



# Overview of SSL / TLS

## 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

# Overview of SSL / TLS

## 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



# Overview of SSL / TLS

## 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
```

# MySQL SSL Command Options

Name	Cmd-Line	Option File	System Var	Var Scope	Dynamic
have_openssl			Yes	Global	No
have_ssl			Yes	Global	No
skip-ssl	Yes	Yes			
ssl	Yes	Yes			
ssl-ca	Yes	Yes		Global	No
ssl-capath	Yes	Yes		Global	No
ssl-cert	Yes	Yes		Global	No
ssl-cipher	Yes	Yes		Global	No
ssl-key	Yes	Yes		Global	No
ssl-verify-server-cert	Yes	Yes			

<http://dev.mysql.com/doc/refman/5.5/en/ssl-options.html>

# 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

# 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



# SSL Certificate Creation

## A. Generating Certificates

Create CA certificate (private key, public cert)

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

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

# SSL Certificate Creation

## 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
```

# SSL Certificate Creation

## 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
```

# SSL Certificate Creation

## A. Generating Certificates

Remove passphrase from client/server key:

```
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)

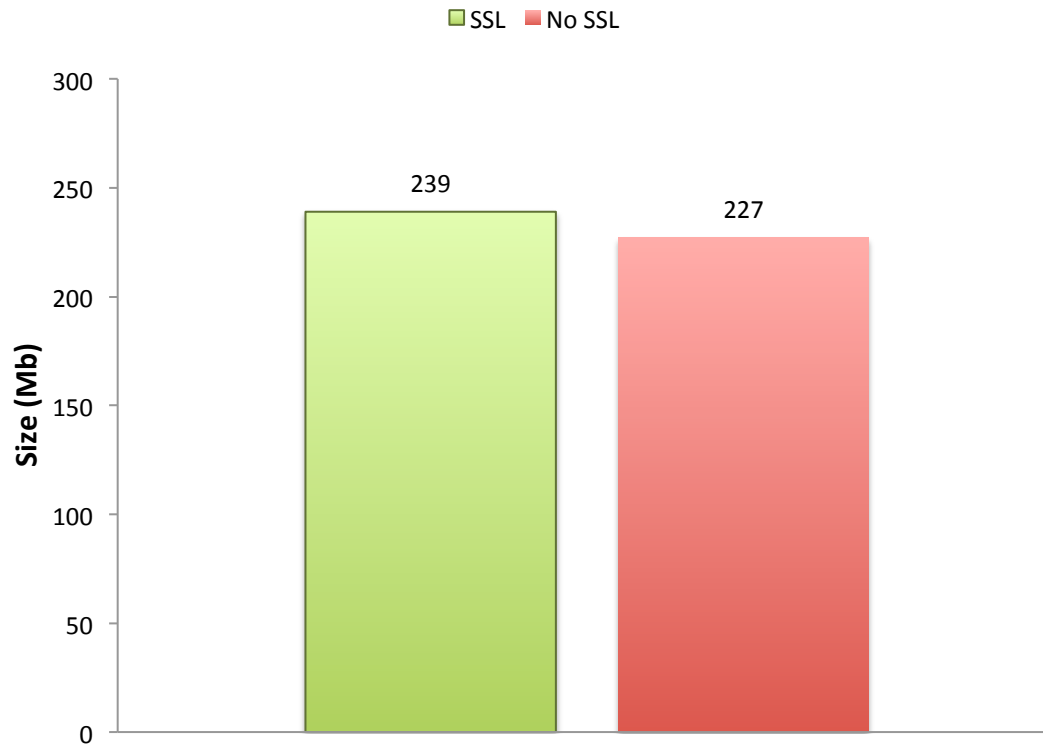
# Performance Comparison of SSL

## Footprint Size

# Performance Comparison of SSL

## MySQL Footprint Size

SSL vs. No SSL



**Command:**

`du -sh .`

**Result:**

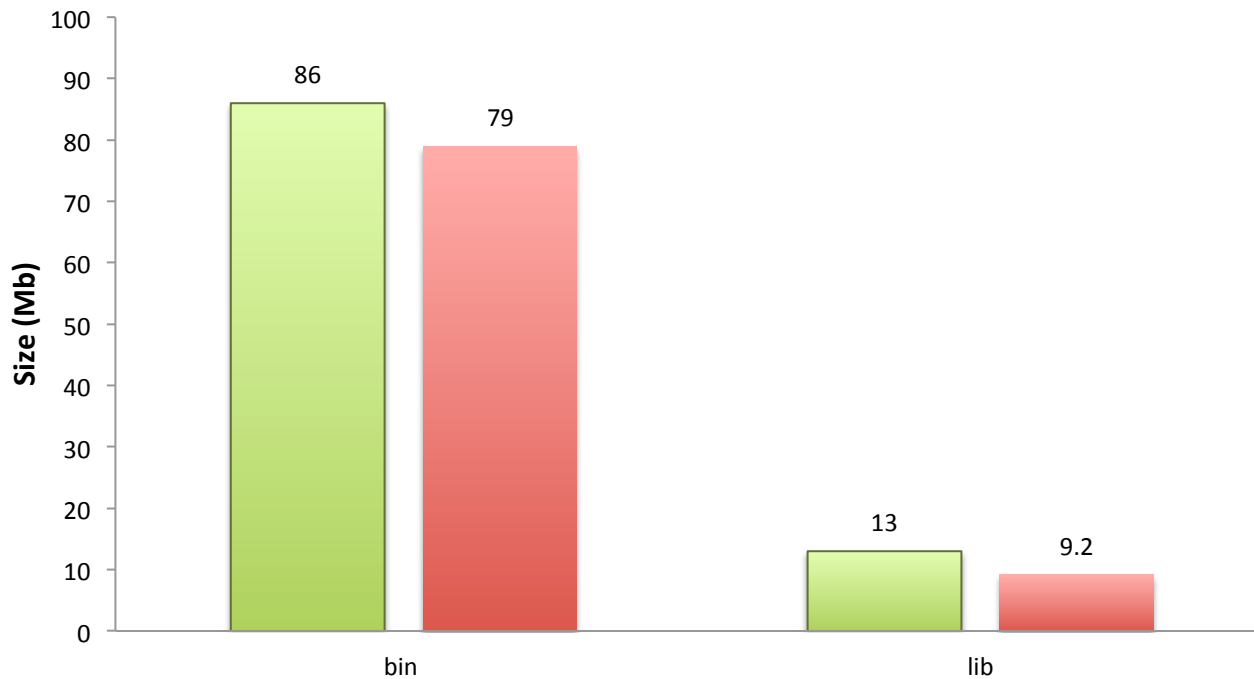
5.3% Difference

# Performance Comparison of SSL

## MySQL Footprint Comparison (Detail)

SSL vs. No SSL

SSL No SSL



**Command:**  
`du -sh *`



# Performance Comparison of SSL

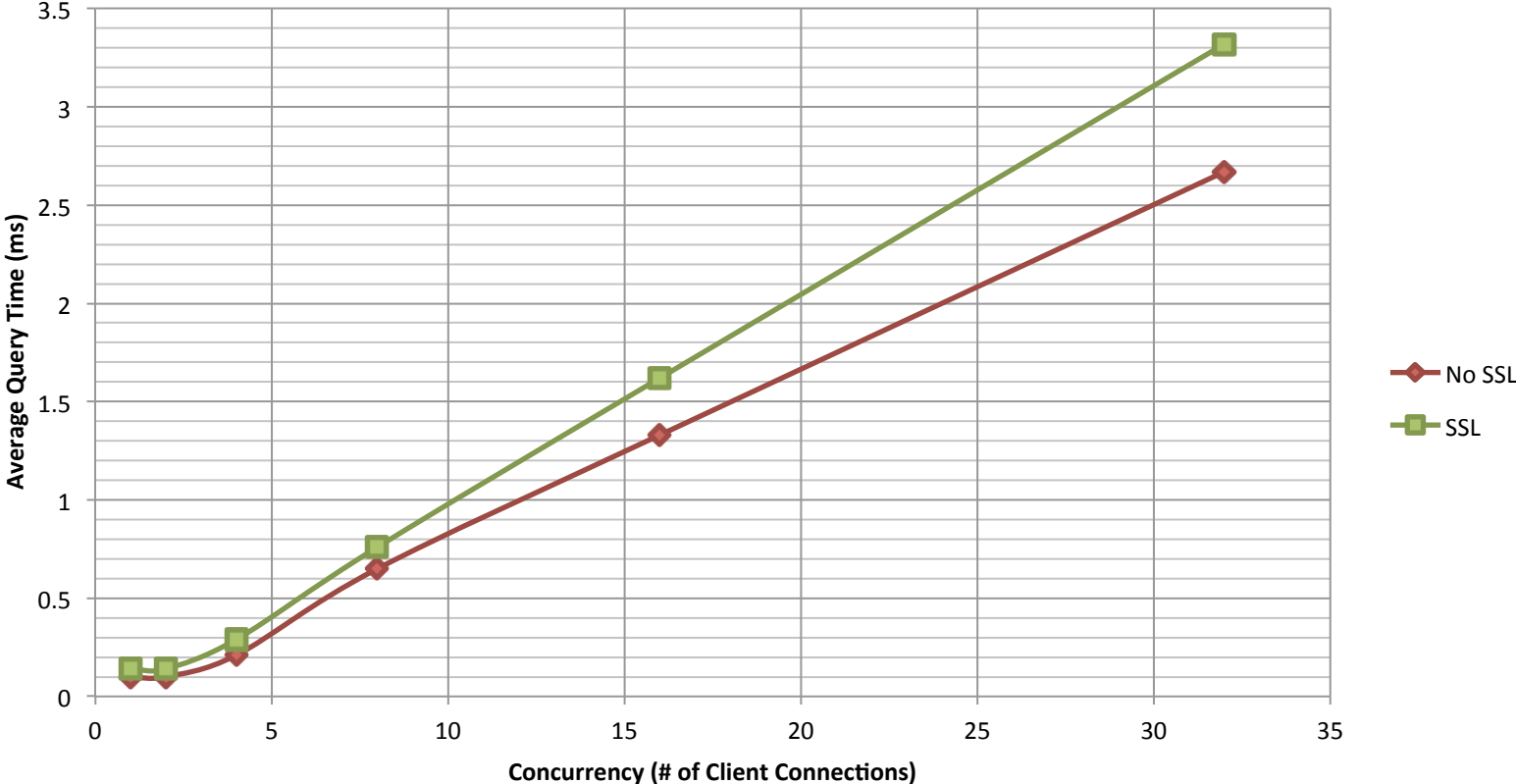
## Average Query Times

(SELECT Queries, sysbench)

# Performance Comparison of SSL

## MySQL Average SELECT Query Times

No SSL vs. SSL  
100,000 Requests  
sysbench

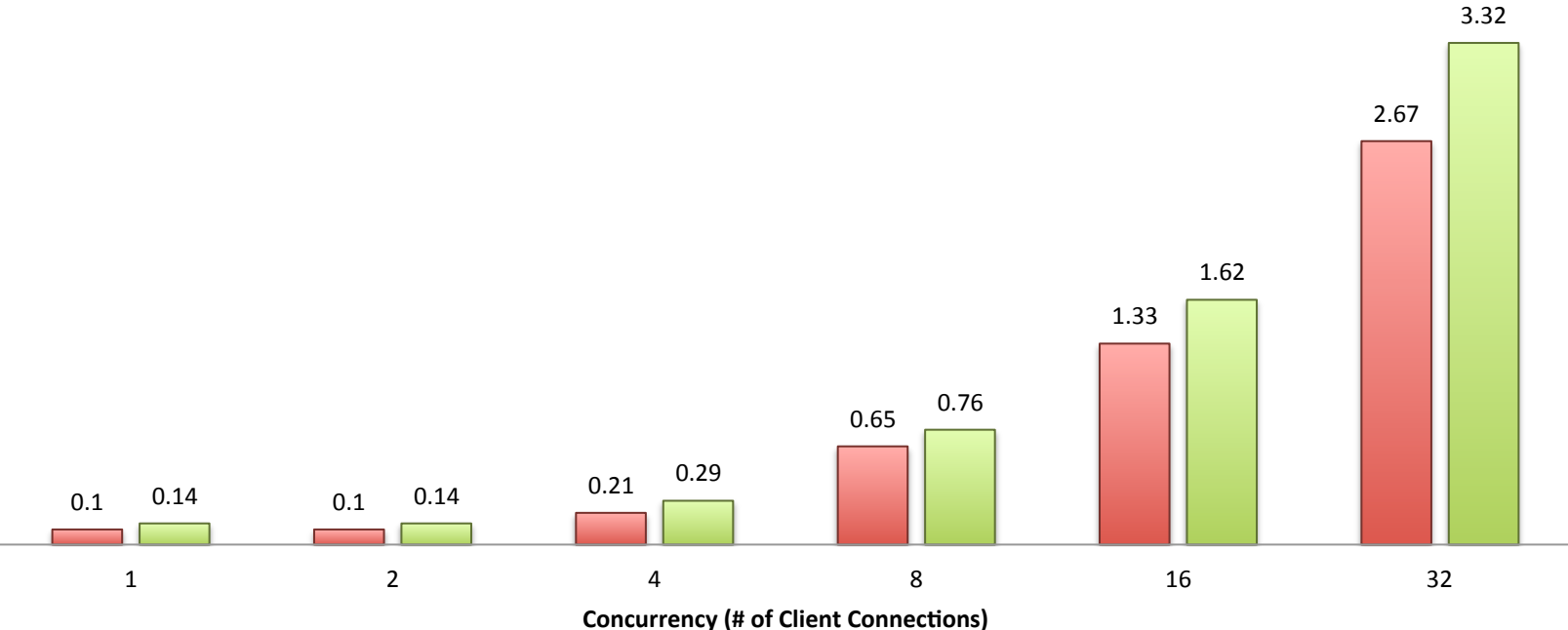


# Performance Comparison of SSL

## MySQL Average SELECT Query Times (ms)

No SSL vs. SSL  
100,000 Requests  
sysbench

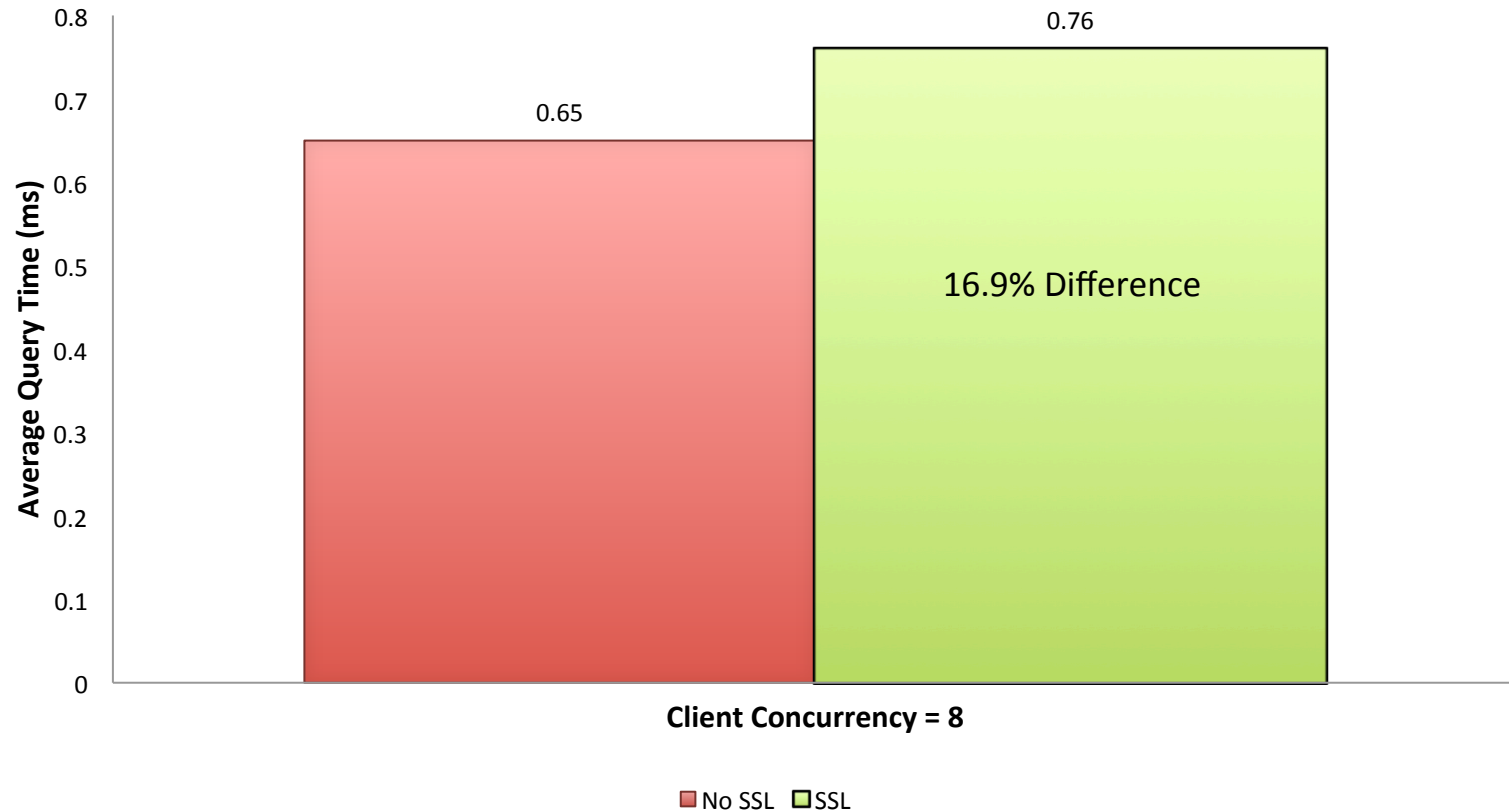
No SSL SSL



# Performance Comparison of SSL

## MySQL Average SELECT Query Times

No SSL vs. SSL  
100,000 Requests  
sysbench



# Part III

Additional Security  
Concerns

Data Encryption

# Data Storage and 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

# Data Storage and Encryption

## 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?

# Data Storage and Encryption

## C. Server Side Encryption

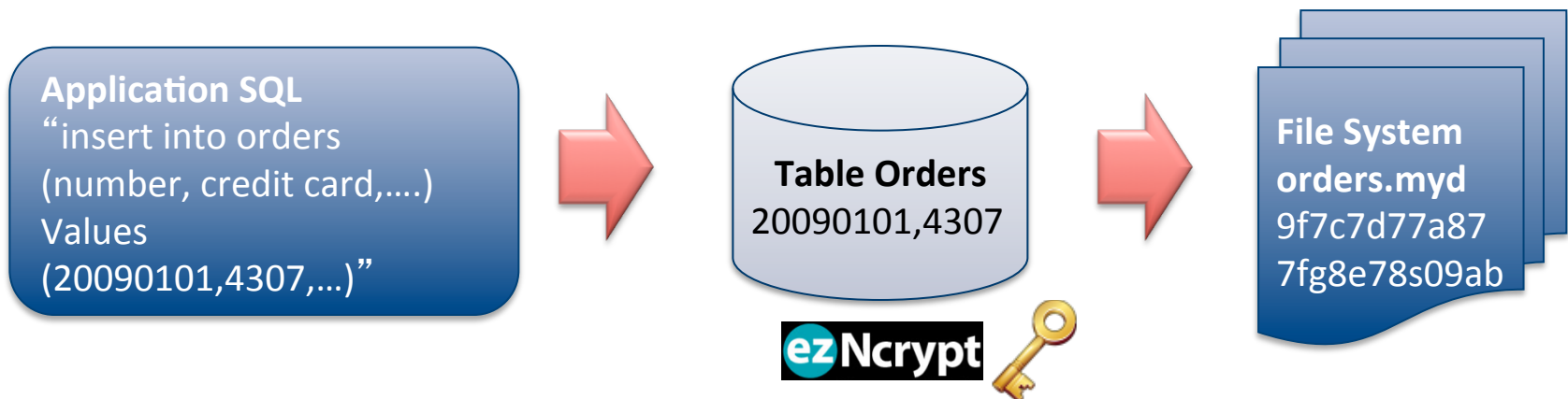
- **AES\_ENCRYPT()**, **AES\_DECRYPT()** functions
  - AES-128 Default
  - AES-256 w/ source-code change
- Entire Disk Encryption
- Transparent Data Encryption (*Gazzang ezNcrypt*)



# Data Storage and Encryption

## 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.



# Data Storage and Encryption

## Gazzang ezNcrypt

Addresses Problems such as

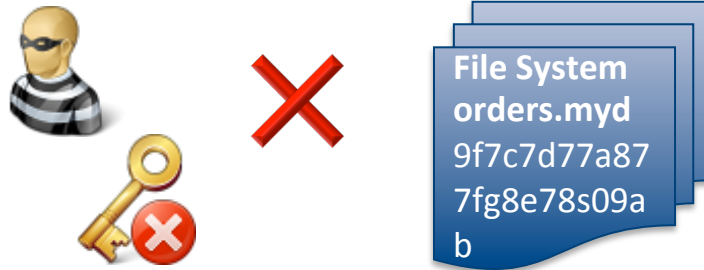
- Unauthorized attempts to read data off the database files, like SSN, Credit Card numbers or phone numbers
- Theft of the data files and intellectual property
- Tampering of data, directly modifying values in files
- Protection of tapes, backups and Data at Rest
- Protecting disks in the case the physical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

# Data Storage and Encryption

## 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.

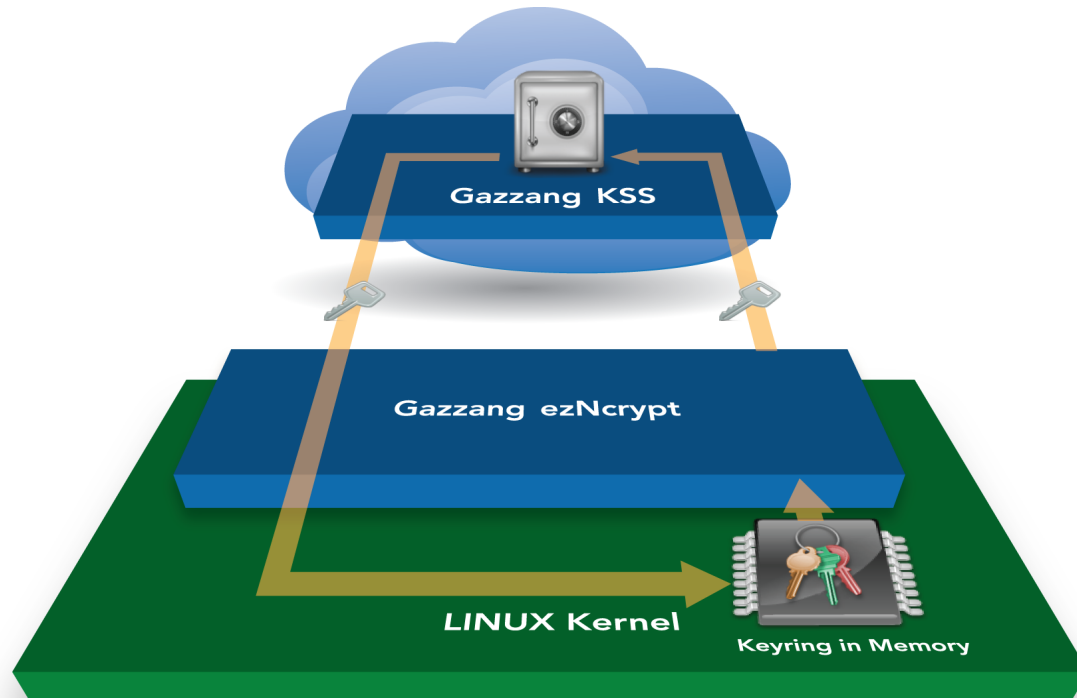


- The MySQL process is the only authorized to retrieve the Key to unlock the database data

# Data Storage and Encryption

## Gazzang ezNcrypt

- Gazzang Key Storage System (KSS)



# Data Storage and Encryption

## 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



# Conclusion

## 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/>

<http://dev.mysql.com/doc/refman/5.5/en/default-privileges.html>

<http://dev.mysql.com/doc/refman/5.5/en/mysql-secure-installation.html>

<http://dev.mysql.com/doc/refman/5.5/en/secure-connections.html>

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

## MySQL Security Resources around the Internet

<http://www.symantec.com/connect/articles/secure-mysql-database-design>

## SSL/TLS

<https://www.ssllabs.com/>

[http://en.wikipedia.org/wiki/Transport\\_Layer\\_Security](http://en.wikipedia.org/wiki/Transport_Layer_Security)