Revisiting MySQL Access Control

NebraskaCERT, Omaha
August 11, 2005

Mat Caughron, CISSP
PHP Consulting

Public Service Announcement

Please silence all pagers

and cell phones now.

Thanks!

Disclaimers

Only You Know Completely What Your Risks Are.

Cryptography is a Powerful Tool.

All Tools Have Limitations.

Remember the IISSCC's Mantra?

Let’s start here.

Security Transcends Technology.

The philosophical basis for access control can be learned from the Greeks.

Other things include a portion of everything, but mind is infinite and self-powerful and mixed with nothing, but it exists alone itself by itself.

- Anaxagoras, circa 250BC

(London: K. Paul, Trench, Trubner, 1898), 235-262

1. In order to rule, there must be separation.

Lots of examples in the military chain of command, uniforms and separate barracks for example, or traveling by Air Force One versus in a humvee.

In biology, the differentiation of species is presupposed to the food chain. If everything was still algae there could be no order. Something as subtly as a difference in position, could give rise to a hierarchy of ruling.

In corporate structure, note the cars people drive. Successful CEO has a Lexus versus the wheels of the average sysadmin. CEO's and powerful people intuit the need for this difference. I'm not one of them.

Positions of the desks and offices in a university. Students are all homogenously seated, grad students share a room and desks, assistant profs. get their own desk, full profs have their own office, department heads have their own secretary, corner office etc.

"He is overcome with difficulty, who knows how to recognize his forces and those of the enemy." - Machiavelli, circa 1500AD

2. Self-rule therefore also requires separation.

- If the mind can rule itself, it must be separated from itself. Socrates exemplified this by questioning people to see whether they knew the difference between what they knew and didn't know.

- Three branches of government separate powers for a self-governing people.

- Self-control is achieved by having the mind rule the emotions, so activities that separate these are important.

Separation of Roles in Information Science

- Security transcends technology?
- Data, or information, is passive, and receives separation from mind. In itself data is homogenous. *
- A separation of duties or roles is frequently required for a given system to achieve its goal.
- Information science depends on other disciplines for its distinctions, which are invariably derived elsewhere.

Metrics Alone Are Worthless

- Information is readily measured.
- What counts in risk metrics isn't information as an unseparated commodity.
- Qualitative risk evaluations are necessary and inevitable.
- Metrics alone get you nowhere fast. (snort logs?) If you measure without a determinate purpose, you'll only wind up wasting time or deceiving yourself, just as the risk management pros.

Fundamental basis for separation.

people who need to know, versus who shouldn't know
people who modify, versus people who shouldn't modify

data that can be seen, versus data that should not be seen
data that can be written, versus data that should not be written

public at large, versus those interested

Access control exists to propagate and enforce a separation.

- How to distinguish among users?
- How to distinguish data? On what basis?
- Access control typically implies both.
- Principle of least privilege requires all this.

Relational Databases are Highly Flexible and Can Accomodate:

- Discretionary access control model.
- Mandatory access control model. (WURD)
- Role based access control model requires visual labels everywhere.

c.f. Rold-based access controls. David Ferraiolo and Richard Kuhn, 1992

the MySQL (webapp) model

Multi-layer discretionary access control...

1. operating and file system

2. database user-entity access control

3. application layer access control (in database typically)

Sometimes #2 and #3 are mixed.

Agenda for Today

- Provide a theoretical foundation of the reasons for access control and models.
- Review some lessons from past MySQL vulnerabilities.
- Provide a best-practices lock-down procedure with a step-by-step walk-through.
- Anticipate and discuss some upcoming features.

Why are we talking about MySQL?

- MySQL has a strong prevalence on the public network... more on this in a bit.
- As an open source project, MySQL has lacked the economic resources for research that Oracle or Microsoft can afford.
- Best practices advocacy seems to be especially needed in this arena.
- Many of these lessons are applicable elsewhere.

Why is MySQL so prevalent?

- comes built in with a variety of systems
- comes as a default requirement
- easy to deploy (installers for many OS's)
- no up-front license cost
- aggressively cross-platform
- favorable press: "everybody is doing it"

Where is MySQL being deployed? All over.

- for centralized logging with syslog-ng
- nessus vulnerabilities are stored in mysql
- snort - raw insert speed is good in mysql
- Apache access with mod_mysql
- Serves as a backend for wikis, weblogs, discussion boards, etc.
- A tool of choice for spammers and hackers.
- PHP,perl,Ruby,jBoss,apache application projects etc. use MySQL as a cheap and easy default setup.

Most recent vulnerability

- July 25, 2005
- not specific to MySQL
- flaw in zlib's inftrees.c
- remote execution capable (!)
- US-CERT VU#680820, CAN2005-2596

zlib Patch Code

```
Index: lib/libz/inftrees.c
===================================================================
RCS file: src/lib/libz/inftrees.c,v
retrieving revision 1.5
diff -u -p -r1.5 inftrees.c
--- lib/libz/inftrees.c 11 May 2005 03:47:48 -0000
   1.5
+++ lib/libz/inftrees.c 2 Jul 2005 19:29:56 -0000
@@ -134,7 +134,7 @@ unsigned short FAR *work;
         left -= count[len];
         if (left < 0) return -1;        /* over-subscribed */
     }
-    if (left > 0 && (type == CODES || (codes - count[0] != 1)))
+    if (left > 0 && (type == CODES || max != 1))
         return -1;                      /* incomplete set */

     /* generate offsets into symbol table for each length for sorting */
```

Learning from History

- Our focus here will be on server-side vulnerabilities, not so much the client or affiliated scripts.
- Nessus database lists 69 total tests.
- Let's review the more significant exposures and vulnerabilities.

Types of Attacks on MySQL

in order of severity

- tie-up 99% cpu (reduction of service)
- crash the mysql daemon (denial of service)
- use mysql daemon to crash the OS (DoS)
- unauthorized read-level access to structure or data
- manipulate/delete data, table structure, files
- gain remote access to the file system with escalated privileges of the mysql daemon (worst case)

Denial of Service Most Common

- CVE-2003-0073 Double-free vulnerability in mysqld for MySQL before 3.23.55 allows attackers with MySQL access to cause a denial of service (crash) via mysql_change_user.

- CAN-2004-0836 Lukasz Wojtow discovered a buffer overrun in the mysql_real_connect function. In order to exploit this issue an attacker would need to force the use of a malicious DNS server ().

- CAN-2004-0837 Dean Ellis discovered that multiple threads ALTERing the same (or different) MERGE tables to change the UNION could cause the server to crash or stall.

- CAN-2004-0956 MySQL before 4.0.20 allows remote attackers to cause a denial of service (application crash) via a MATCH AGAINST query with an opening double quote but no closing double quote.

2002

- CAN-2002-1373 A signed integer vulnerability in the COM_TABLE_DUMP package for MySQL 3.x to 3.23.53a, and 4.x to 4.0.5a, allows remote attackers to cause a denial of service (crash or hang) in mysqld by causing large negative integers to be provided to a memcpy call.

  CAN-2002-1374 The COM_CHANGE_USER command in MySQL 3.x to 3.23.53a, and 4.x to 4.0.5a, allows a remote attacker to gain privileges via a brute force attack using a one-character password, which causes MySQL to only compare the provided password against the first character of the real password.

  CAN-2002-1375 The COM_CHANGE_USER command in MySQL 3.x to 3.23.53a, and 4.x to 4.0.5a, allows remote attackers to execute arbitrary code via a long response.

  CAN-2002-1376 The MySQL client library (libmysqlclient) in MySQL 3.x to 3.23.53a, and 4.x to 4.0.5a, does not properly verify length fields for certain responses in the read_rows or read_one_row routines, which allows a malicious server to cause a denial of service and possibly execute arbitrary code.

Unwanted File Manipulation

- CAN-2004-0957 Sergei Golubchik discovered that if a user is granted privileges to a database with a name containing an underscore ("_"), the user also gains the ability to grant privileges to other databases with similar names.

- CAN-2004-0381 MySQL allows local users to overwrite arbitrary files via a symlink attack on the failed-mysql-bugreport temporary file

- 2004 Debian Security Advisory warnings: The script mysqlbug in MySQL allows local users to overwrite arbitrary files via a symlink attack. The script mysqld_multi in MySQL allows local users to overwrite arbitrary files via a symlink attack.

More High Profile

- CAN-2003-0780 any attacker who has the credentials to connect to this server may execute arbitrary code on this host with the privileges of the mysql database by changing his password with a too long one containing a shell code. This buffer overflow condition could also be exploited by a user who has permission to execute "ALTER TABLE" commands on the tables in the "mysql" database. If successfully exploited, this vulnerability could allow the attacker to execute arbitrary code with the privileges of the mysqld process.

- MySQL 3.23.55 and earlier creates world-writeable files and allows mysql users to gain root privileges by using the "SELECT * INFO OUTFILE" operator to overwrite a configuration file and cause mysql to run as root upon restart, as demonstrated by modifying my.cnf.

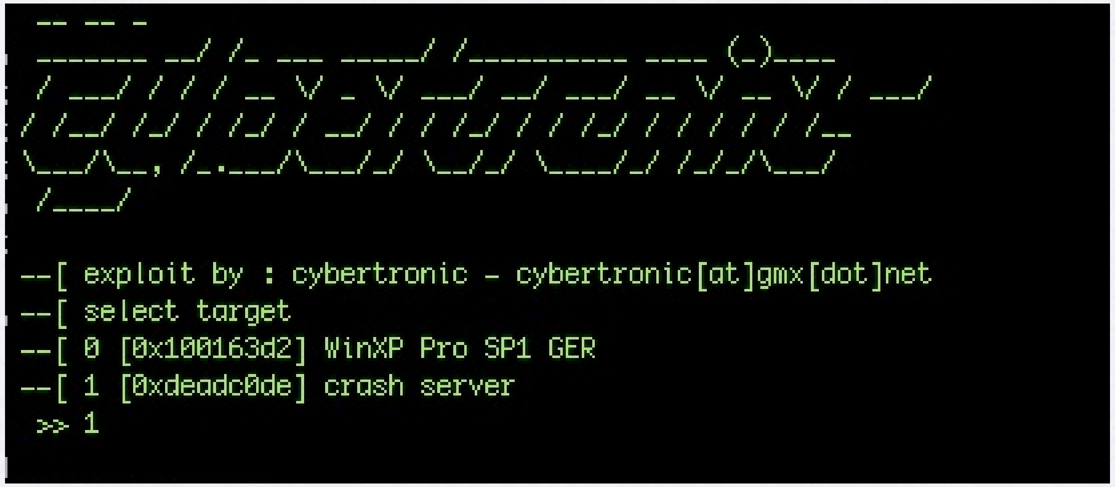
2005

CAN-2005-0709 MySQL 4.0.23 and earlier, and 4.1.x up to 4.1.10, allows remote authenticated users with INSERT and DELETE privileges to execute arbitrary code by using CREATE FUNCTION to access libc calls, as demonstrated byusing strcat, on_exit, and exit.

CAN-2005-0710 MySQL 4.0.23 and earlier, and 4.1.x up to 4.1.10, allows remote authenticated users with INSERT and DELETE privileges to bypass library path restrictions and execute arbitrary libraries by using INSERT INTO to modify the mysql.func table, which is processed by the udf_init function.

CAN-2005-0711 MySQL 4.0.23 and earlier, and 4.1.x up to 4.1.10, uses predictable file names when creating temporary tables, which allows local users with CREATE TEMPORARY TABLE privileges to overwrite arbitrary files via a symlink attack.

Cybertronic DoS tool



```
               __              __                   _
  _______  __/ /_  ___  _____/ /__________  ____  (_)____
 / ___/ / / / __ \/ _ \/ ___/ __/ ___/ __ \/ __ \/ / ___/
/ /__/ /_/ / /_/ /  __/ /  / /_/ /  / /_/ / / / / / /__
\___/\__, /_.___/\___/_/   \__/_/   \____/_/ /_/_/\___/
    /____/

--[ exploit by : cybertronic - cybertronic[at]gmx[dot]net
--[ select target
--[ 0 [0x100163d2] WinXP Pro SP1 GER
--[ 1 [0xdeadc0de] crash server
 >> 1
```

[In]Famous Weak Hash

- Can be useful to bootstrap harder-to-get hashes.
- Brute force 7 chars in less than 1 hour, 8 chars in < 24 hours
- Replaced by SHA1 in MySQL 4.1

```
/ MySQL brute force password attack
//
// to compile : g++ -omysqlpassword mysqlpassword.c -O6 -lm
//
// Written by : term@rmci.net, current version http://term.rmci.net/mysqlpassword.cpp
//
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h> // memset
#include <unistd.h> // usleep

using namespace std;

struct rand_struct {
  unsigned long seed1,seed2,max_value;
  double max_value_dbl;
};

void make_scrambled_password(char *,const char *);
char *scramble(char *,const char *,const char *, int);

int brute(const char *password) {
  // Tune stuff here, change min / max for the char range to crack and width for max password width.
  unsigned int min=32,max=122,pos=0,width=11,max_pos=0;
  unsigned char data[255];
  register unsigned long long loops=0;
  char *encrypted_password = new char[255];
  memset(encrypted_password, 0, 255);
  memset((char*)&data, min, 255);
  while(width) {
    loops++;
    if(data[pos] != max) {
      data[pos]++;
    } else {
      for(register int i=pos; i<max; i++) {
        if(data[i] != max) {
          data[i]++;
          pos=i;
          break;
        }
```

Don’t Panic!

- As far as systems go, MySQL-based applications seem to be fairly resilient.
- Is MySQL simply winning the war of attrition by being free or open. (?!)
- There is strength in numbers. More eyeballs and configuration flexibility do offset risk aggregation.
- The database is not the only component of online applications, it just has to do what databases do best: NOT be outward facing.
- Best practices go a long way.
- Start with defense in depth!

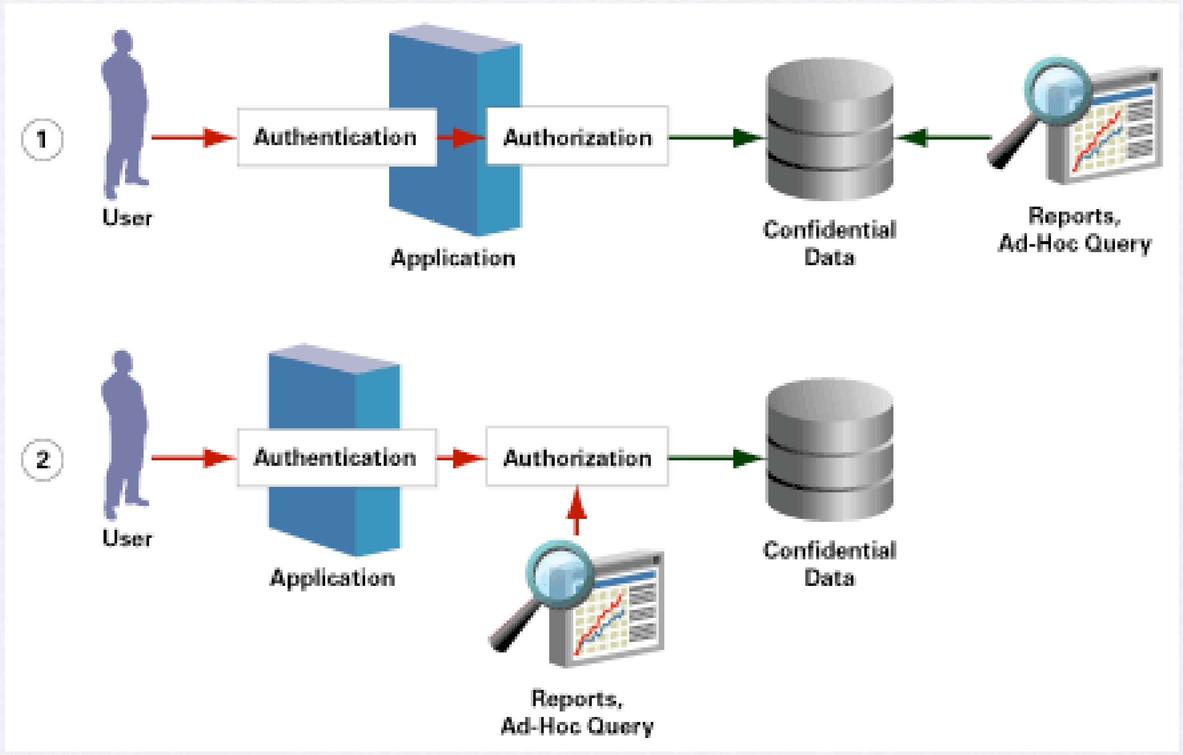
More than one valid approach to mitigate risk.

- Network architecture
- Policy-level
- Application configuration
- Experience and training

Architectural Approach

With thanks to Oracle

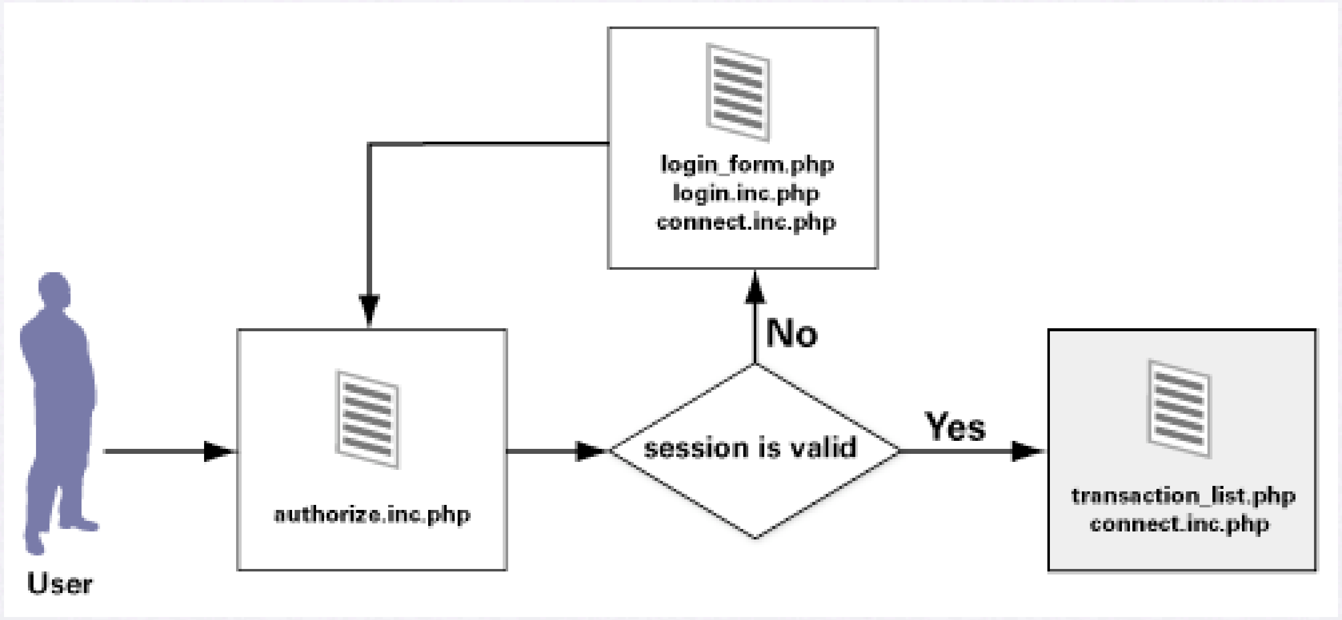
- Best practice architecture, keep the access control policies and enforcement close to the data:



http://www.oracle.com/technology/pub/articles/deployphp/seliverstov_deployphp.html

Reinventing the wheel

Common scenario for web app access control:



http://www.oracle.com/technology/pub/articles/deployphp/seliverstov_deployphp.html

Lock-down goals

- Least privilege: MySQL processes best run under a unique UID/GID that is not used by any other system process
- Only local access to MySQL to be allowed. Let your application provide an isolated layer for defense in depth.
- MySQL root's account is either removed or protected by a hard to guess password.
- The administrator's account will be renamed.
- Anonymous access to the database (by using the *nobody* account) must be disabled.
- All default/sample databases and tables removed.

Pre-Installation (on FC4)

Anticipate least privileges for the database environment.

```
pw groupadd mysql
```

```
pw useradd mysql -c "MySQL Server" -d /dev/null -g mysql -s /sbin/
nologin
```

Consider chroot.

Downside to chroot is likely to be availability.

Disk / partition demands are different when chrooted.

Harder to tune, as separate spindles all have to be mounted relative to the chrooted path.

Compilation Directives

```
./configure --prefix=/usr/local/mysql --
with-mysqld-user=mysql --with-unix-socket-
path=/tmp/mysql.sock --with-vio  --with-
openssl=/usr/local/openssl --with-mysqld-
ldflags=-all-static

make

su

make install
```

Stage 1 Access Control

By default, the MySQL root user has no password. Ouch! You can check this with mysql -u root; if you get a mysql prompt, no root password is set. The first thing you should do is drop this user or at least set a strong password.

Never give the system root password to the MySQL root user.

To set the initial root password, open a mysql prompt -- mysql -u root mysql -- and enter the following:

```
mysql> DELETE from db where HOST=”%”;

mysql> UPDATE user SET Password=PASSWORD('new_password')
    ->                 WHERE user='root';
mysql> FLUSH PRIVILEGES;
```

Limitations on Users

New since the last time I gave this talk, i.e., in MySQL 4.0.1:

The number of queries that an account can issue per hour.

The number of updates that an account can issue per hour

The number of times an account can connect to the server per hour

Local Account Setup

- As well as setting the root password, you should remove anonymous accounts:

```
mysql> DELETE FROM user WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

- Create a new user with specific privileges using the `GRANT` statement. For example:

```
GRANT USAGE ON mydb.* TO 'someuser'@'localhost'
IDENTIFIED BY 'some_pass';
FLUSH PRIVILEGES;
```

Avoid Remote Accounts

But if you must, give netmasked address for remote users.

```
GRANT USAGE ON mydb.* TO 'someuser'@'localhost'
IDENTIFIED BY '12.34.56.78/255.255.255.240';
```

```
Compile --with-libwrap=/usr/local/tcp_wrappers
```

Once built, add /etc/hosts entries for allowed subnets..

Stage 2 Access Control

Once a user is in, each query is checked against a privilege table which can be summarized with:

```
global privileges

OR (database privileges AND host privileges)
OR table privileges
OR column privileges
```

The following two tables clarify both the privileges granted or revoked and in what order these are matched.

Grant Matching

	Password	User	Host	Db	Table	Column
user	x	x	x			
host			x	x		
db		x	x	x		
tables		x	x	x	x	
columns		x	x	x	x	x

Stuck? try this:

```
mysql> SHOW GRANTS FOR 'user'@'127.0.0.1';
```

Privilege Types

- Alter
- Delete
- Create
- Drop
- Execute
- Select
- Update

User Table Only
Reload
Shutdown
Process (!)
File
Show
Super
etc.

What a Grant Statement Should Look Like

-
  ```
  mysql> GRANT ALL ON mydb.* TO
  'joe'@'localhost'
      ->      IDENTIFIED BY 'joepass'
      ->      WITH MAX_QUERIES_PER_HOUR 20
      ->           MAX_UPDATES_PER_HOUR 10
      ->           MAX_CONNECTIONS_PER_HOUR 5
      ->           MAX_USER_CONNECTIONS 2;
  ```

Reset MySQL Password?

- obtain mysql user account access on the system, or root.
- run with --skip-grant-tables
- log in with any user and manipulate the users table directly. All access control is gone.

Happy Trails?

- No talk about this subject would be complete without a warning about history files.
- MySQL has a history.
- Your shell has a history.
- Those who do not learn about history are destined to have it repeated.

Database touches the Application-layer

robust session handling typically demands solid hashing
SQL injection mitigation can involve both application and database techniques

Never allow unhashed password storage! See RFC 2195!!!

```
$query = "INSERT INTO user VALUES
('DummyUser',md5('DummyPassword'))";
```

AES and DES encrypt and decrypt are available in MySQL for row-level crypto

Practical Summary

- Keep 3306 closed to the outside world.
- Monitor logs and activity.
- Set up the gate, know the limitations of it, and then watch it closely.
- No such thing as *Web Applications for Dummies*. Retain good people.
- Keep informed: there **is** strength in numbers.
- Stay patched.

Crypto Enabled MySQL

To Grid-enable MySQL is to allow client authentication using X509 certificates. Using the X509 certificates issued by a Certificate Authority (CA) allows for robust authentication. To do this in MySQL, one needs to connect over an SSL encrypted channel. Perl DBI can be compiled to support this.

http://www.star.bnl.gov/STAR/comp/Grid/MySQL/GSI/testing.html

Basic Setup

- Build and install OpenSSL 0.9.6
- Configure MySQL --with-vio and --with-openssl as mentioned above
- SHOW VARIABLES LIKE ‘%openssl%’ should indicate that have_openssl=YES
- Next, generate key pairs.

OpenSSL Setup

```
DIR=~/openssl
PRIV=$DIR/private
mkdir $DIR $PRIV $DIR/newcerts
cp /usr/share/openssl.cnf $DIR/openssl.cnf
replace .demoCA $DIR -- $DIR/openssl.cnf
```

PKI From Scratch Dance

1. Certificate Authority

```
•   > openssl req -new -keyout cakey.pem -out $PRIV/cacert.pem -config $DIR/openssl.cnf
```

2. Server Certificate Signing Request and Key

```
•   > openssl req -new -keyout $DIR/server-key.pem -out $DIR/server-req.pem \
•   > -days 3600 -config $DIR/openssl.cnf
•   > openssl rsa -in $DIR/server-key.pem -out $DIR/server-key.pem
•   > openssl ca -policy policy_anything -out $DIR/server-cert.pem \
•   > -config $DIR/openssl.cnf -infiles $DIR/server-req.pem
```

3. Client Certificate Signing Request and Key

```
•   > openssl req -new -keyout $DIR/client-key.pem -out $DIR/client-req.pem \
•   > -days 3600 -config $DIR/openssl.cnf
•   > openssl rsa -in $DIR/client-key.pem -out $DIR/client-key.pem
•   > openssl ca -policy policy_anything -out $DIR/client-cert.pem \
•   > -config $DIR/openssl.cnf -infiles $DIR/client-req.pem
```

my.cnf for Client and Server

Add the following lines to my.cnf.
be sure to replace $DIR with the actual location of the pem files.

```
[server]
ssl-ca=$DIR/cacert.pem
ssl-cert=$DIR/server-cert.pem
ssl-key=$DIR/server-key.pem


[client]
ssl-ca=$DIR/cacert.pem
ssl-cert=$DIR/client-cert.pem
ssl-key=$DIR/client-key.pem
```

Crypto-Style Grants

`REQUIRE SSL` limits the server to allow only SSL connections

`REQUIRE X509 "issuer"` means that the client should have a valid certificate, but we do not care about the exact certificate, issuer or subject

`REQUIRE ISSUER` means the client must present a valid X509 certificate issued by issuer "`issuer`".

`REQUIRE SUBJECT "subject"` requires clients to have a valid X509 certificate with the subject "`subject`" on it.

`REQUIRE CIPHER "cipher"` is needed to ensure strong ciphers and keylengths will be used. (ie. `REQUIRE CIPHER "EDH-RSA-DES-CBC3-SHA"`)

Coming in Version 5.0

MySQL 5.0 is in beta, and has been feature-frozen since April.

Back in 4.1, its abstracted table-type has been put to advantage with odd engines like Archive (only insert, no update);

Blackhole for fast replication, part of the grid efforts.

Improvement to MyISAM for logging (allowing concurrent selects with inserts-at-table-end).

In 5.0 we're seeing stored procedures per the SQL:2003 standard, triggers, updatable views, XA (distribution transaction), SAP R/3 compatible server side cursors, fast precision math, a federated storage engine, new optimizer for better handling of many-table joins, and an optional "strict mode" to turn some of MySQL's friendly nonstandard warnings into compliant errors.

MySQL Cluster: non-indexed columns to be stored on disk.

the move to yaSSL

- seems to be motivated by license terms
- MySQL AB appears to want all major pieces to be “dual license-able”
- protocol guidance by RFC 2246, TaoCrypt
  - TaoCrypt Provides RSA, DES, 3DES, ARC4, MD2, MD5, SHA-1, RIPEMD-160, DSS, Diffie-Hellman, Random Number Generation, Large Integer support, and base 16/64 encoding/decoding.

Better Defaults Needed

- mysql_install_db that includes grant statements to limit account logins
- more sane defaults (root/ ‘’ !?@#$!?!?)
- access-control-oriented stored procedures
- require local socket connection (not listening on 3306)?

Needed Installation Improvements

- alternative, improved installation script
- would not insert test databases
- would not insert default root user acct.
- prompts for or provides robust password
- current script is written in bash, so this would be relatively easy to write

Thanks for coming.

Next session:
GPG on Mac OS X

Mat Caughron, CISSP
PHP Consulting
(402) 968-1332
mat@phpconsulting.com gpg: 35FC 7ECC 1F67 4525 FE96 A0C3 FB3B B58A 9E3C D47E