

# Unix System Administration

Chris Schenk

Lecture 19 – Tuesday Mar 18

CSCI 4113, Spring 2008

# SQL Meta Queries

- Stuff for creating DBs/tables, altering tables
- Create a database;
  - `CREATE DATABASE <db>;`
- Create a table
  - `CREATE TABLE <table>;`
- Alter an existing table
  - `ALTER TABLE <table> ADD <column> <type>;`
- Drop a table or database
  - `DROP TABLE <table>;`
  - `DROP DATABASE <db>;`

# Creating Users

- Easiest way to create users is to first create a database and add users to it
  - `CREATE DATABASE mydb;`
- Users added with the 'GRANT' command
- `GRANT ALL PRIVILEGES ON <db>.<tables> TO '<user>'@'<host>';`
  - `GRANT ALL PRIVILEGES ON mydb.* to 'chris'@'localhost'`
  - Host is where the user can connect FROM!

# Sample Database

- **CREATE DATABASE login\_monitor;**
- **USE login\_monitor;**
  - Tell MySQL that we're modifying this database
- **Create a table for tracking user logins**
  - ```
CREATE TABLE logins (  
  id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  username VARCHAR(20),  
  ip VARCHAR(15),  
  time TIMESTAMP DEFAULT 0);
```
- **Insert some data into it:**
  - ```
INSERT INTO logins (username, ip, time) VALUES  
  ('chris', '128.138.242.249', NOW());
```

# Viewing Sample Data

- Select the information out of the 'logins' table

```
- mysql> SELECT * FROM logins;
```

```
+-----+-----+-----+-----+
| id | username | ip | time |
+-----+-----+-----+-----+
| 1 | chris | 128.138.242.249 | 2008-03-12 22:36:09 |
+-----+-----+-----+-----+
```

```
1 row in set (0.00 sec)
```

- Select a specific column

```
- mysql> SELECT username FROM logins;
```

```
+-----+
| username |
+-----+
| chris |
+-----+
```

```
1 row in set (0.00 sec)
```

# Selecting Specific Data

- Let's add one more row:

```
- INSERT INTO logins (username, ip, time) VALUES ('root',  
  '128.138.242.249', NOW());
```

- Select only root:

```
- mysql> SELECT * FROM logins WHERE username='root';  
+----+-----+-----+-----+  
| id | username | ip | time |  
+----+-----+-----+-----+  
| 2 | root | 128.138.242.249 | 2008-03-12 22:41:44 |  
+----+-----+-----+-----+  
1 row in set (0.00 sec)
```

- Select IP 128.138.242.249:

```
- mysql> SELECT * FROM logins WHERE ip='128.138.242.249';  
+----+-----+-----+-----+  
| id | username | ip | time |  
+----+-----+-----+-----+  
| 1 | chris | 128.138.242.249 | 2008-03-12 22:36:09 |  
| 2 | root | 128.138.242.249 | 2008-03-12 22:41:44 |  
+----+-----+-----+-----+  
2 rows in set (0.00 sec)
```

# PHP + MySQL

- Don't know how to connect to MySQL in PHP?
  - Google it!
    - Except there's lots of bad code out there like I said
- Pretty easy to connect, query, and close
  - ```
$resource = mysql_connect($host, $user, $pass);  
mysql_select_db($db);  
$results = mysql_query("select * from logins");  
while($row = mysql_fetch_assoc($results)) {  
    print "User: ".$row['username'];  
    print ", IP: ".$row['ip']."\n";  
}  
mysql_close();
```
  - All of this requires packages php5-cli and php5-mysql
- Web version simply uses HTML to format

# Selecting Data from User Input

- Let's say we want to let users login to a page
  - Simple username/password combo
- We have a 'users' table with following columns:
  - Username, password, emailaddress
    - Ignore the fact that the password is plain-text
- To perform a login, we simply create a query:
  - `$query = "SELECT * FROM users WHERE username='$username' AND password='$password'";`
- We run the query with `mysql_query($query);`

# SQL Injection

- Part of the SANS Top 20 Vulnerabilities list
  - <http://www.sans.org/top20/#s1>
- SQL Injection attack by white-hat hackers
  - <http://www.unixwiz.net/techtips/sql-injection.html>
- SQL Injection cheat-sheet
  - <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- Thousands of web servers are vulnerable
  - Poorly written applications will allow access to data within the database because of lack of input sanitization on web pages

# Preventing SQL Injection

- Very, very simple ways to mitigate injection
- Do any of the following:
  - Check input fields for valid input
    - Characters and length (maybe with regex)
  - Use prepared statements
    - Inputs cannot modify the query itself
  - Use stored procedures
    - Even better than prepared statements, eliminates writing actual SQL queries!
  - Notify admins on query syntax errors
    - Error reporting on the application itself

# PHP Prepared Statements

- We have to convert our mysql code to mysqli
  - Newer, better library in php5
  - Object oriented, must keep track of our connection within a variable (\$conn in the example)
- Three extra function calls required
  - ```
$stmt = mysqli_prepare($conn, $query);  
mysqli_stmt_bind_param($stmt, ...);  
mysqli_stmt_execute($stmt);
```
- Prepared statements effectively prevent your code from being susceptible to SQL injection