

Security from the Database Perspective

Morgan Tocker <morgan@mysql.com>

Some of the Problems

- SQL Injection
- Denial of Service
- Running with higher privileges than required
 - Both in Operating System, and in MySQL.
- Exposing the database server on 3306

SQL Injection

- The application does not check input data before passing it on to the database.

SQL Injection (cont.)

- An Example:

```
SELECT * FROM users WHERE username = '$1'  
AND password = '$2';
```

- Typically this ends up looking like:

```
SELECT * FROM users WHERE username = 'morgo'  
AND password = 'mypassword';
```

SQL Injection (cont.)

- The risk:
If I specify username of `morgo` and password of `' or '='`
- `SELECT * FROM users WHERE username = 'morgo'
AND password = ' or '=';`

SQL Injection (cont.)

- PHP has a helpful (but ultimately useless) feature called `magic_quotes_gpc`
- The previous example becomes `' or ''=''`, which is correctly escaped.

Why magic quotes and addslashes won't cure all

- Integer based SQL injection
 - `mysql_real_escape_string()` is also vulnerable.
- Multibyte character set attacks.

Integer based attacks

- `SELECT * FROM users WHERE id = $1;`
Can become:
- `SELECT * FROM users WHERE id = 0 OR username=CONCAT(char(109), char(111), char(114), char(103), char(111));`
 - `-> SELECT * FROM users WHERE username='morgo';`
- No single quotes required to build strings!

Multibyte Characterset Attacks

- “In GBK, 0xbf27 is not a valid multi-byte character, but 0xbf5c is. Interpreted as single-byte characters, 0xbf27 is 0xbf (i) followed by 0x27 ('), and 0xbf5c is 0xbf (i) followed by 0x5c (\).” [1]
- \$username = chr(0xbf) . chr(0x27) . ' OR username = /*'; [2]
- See:
http://ilia.ws/archives/103-mysql_real_escape_string-versus-Prepared-Statements.html
<http://shiflett.org/blog/2006/jan/addslashes-versus-mysql-real-escape-string>

What are the limits of SQL Injection

- Can I bypass log in systems?
- Can I change data?
- Can I send email or run system commands?
 - Not in MySQL.

Injection Examples (cont.)

- `SELECT access_level FROM my_privileges_table WHERE user_id = 1 AND 1=0 UNION SELECT 99 FROM DUAL;`
- `SELECT 99 FROM DUAL;`
- `SELECT session_data FROM sessions WHERE sessid='eab71244afb687f16d8c4f5ee9d6ef0e' AND 1=0 UNION SELECT 's:14:\\"this is a test\";' FROM DUAL;`
- Inject custom data

Use information_schema to gain more information.

- `SELECT session_data FROM sessions WHERE sessid='eab71244afb687f16d8c4f5ee9d6ef0e' AND 1=0 UNION SELECT DATABASE();`
- `SELECT session_data FROM sessions WHERE sessid='eab71244afb687f16d8c4f5ee9d6ef0e' AND 1=0 UNION SELECT group_concat(table_name) FROM information_schema.tables where table_schema='test' group by table_schema;`
- Note: The number of columns must match in both queries in the UNION, but datatypes do not need to.

Denial Of Service

- Be careful that no part of your application consumes too many resources. For example:
- `SELECT * FROM users WHERE username = 'morgo' ..;`
Returns 1 row.
- `SELECT * FROM users WHERE username = 'morgo' OR "="`
Returns all rows.
- If MyISAM uses table lock to ensure consistency for duration of read (expensive).

Enough of SQL Injection

- Now for other attacks.

Intentionally corrupting masters or slaves (proof of concept)

- `DELETE FROM users WHERE user_id = 1 AND @@server_id = 1;`

Denial Of Service when not injecting.

- Deep Search results
 - Some sites choose to limit the number of results to first N hundred
- Inherently expensive pages:
 - `shell> ab -c 10 -n 100 http://www.couchsurfing.com/people/guaka`
When I volunteered, friend of a friend calculations were expensive.
- Think of these like “pressure points”. Low cost to hit, high cost to return.

Exposing the Database on 3306

- Potential protocol vulnerabilities
- Risk of Denial of Service
 - Watch out for potentially expensive reverse lookups

Running Mysqld with higher privileges

- Risk that MySQL will be able to READ/WRITE to other Operating System files.
- Most obvious use is with LOAD DATA LOCAL INFILE feature.
 - Potential to load in /etc/passwd
- Can be disabled with --local-infile=0

<http://dev.mysql.com/doc/refman/5.0/en/load-data-local.html>

The MySQL Privilege System

- MySQL combines the username and the host to create a login.
- You can find what privileges you have with:
 - `mysql> SHOW GRANTS;`
 - `mysql> SHOW GRANTS FOR 'morgo'@'hostname';`

Setting Privileges

- You can use the `mysql.user` table manually then issue `FLUSH PRIVILEGES` (this method is not recommended).
- You can use the `GRANT` and `REVOKE` statements:
 - `GRANT privilege_name ON database.table TO 'username'@'host' IDENTIFIED BY 'password' [WITH GRANT OPTION];`

Example Web Application

- GRANT DELETE, INSERT, SELECT, UPDATE, CREATE TEMPORARY TABLES ON database.* TO 'application_user'@'application I' IDENTIFIED BY 'mypassword';

Fun problems associated with using MySQL's root user.

- An attacker with the SUPER privilege can disable the binary log, execute a query, turn the binary log back on!
- ```
SET SQL_LOG_BIN=0;
DELETE FROM user_action_log WHERE user_id = 10;
SET SQL_LOG_BIN=1;
```

# Defenses

- Log queries, and not using indexes:
  - `log-slow`    `long-query-time=1`    `log-queries-not-using-indexes`
- Snipe queries that take too long to execute (assuming transactional design), or email yourself alerts by monitoring the `SHOW PROCESSLIST`.

# Defenses (cont.)

- Use low session-based buffer values (tmp\_table\_size, max\_heap\_table\_size etc).
- Set user limits within users to parts of the application?
  - Max user connections, Max connections per hour, max queries per hour. See: <http://dev.mysql.com/doc/refman/5.0/en/grant.html>
- Limit the max\_join\_size to prevent cartesian products.

# The End.

- Other things I can talk about:
  - Storing Hierarchal Data (Adjacency List, Nested Set, “Custom”)
  - Effective data caching
  - Data migration strategies
  - Debugging MySQL Crashes (gdb), building testcases, filing bugs.
  - Building HA Applications
  - What makes a good backup