Australian National University

Database Security – Part 3

SQL Injection

Australian National University

A Survey

- A survey by GreenSQL (http://www.greensql.com, released on April 3, 2012) shows that **the most critical database security concerns** are:

51%	**SQL injection** attacks from internal and external users
31%	**Internal threats**, including unauthorized database access, database administrator errors, and data exposure to non-privileged internal users
18%	**Regulatory compliance**

- Surveyed more than six thousand users – IT administrators, DBAs, data security professionals and consultants.

SQL Injection

- SQL injection is mostly known as an attack for **web applications** (considered as **one of the top 10 web application vulnerabilities** of 2007 and 2010 by the Open Web Application Security Project).
- SQL injection can be used to attack any type of SQL databases.
- Web applications often access a database by
  1. Connect to the database;
  2. Send SQL statements and data to the database;$\hookleftarrow$ **SQL Injection!**
  3. Fetch the result and display data from the database;
  4. Close the connection.

What is SQL Injection?

- In **SQL injection attacks**, hackers inject a string input through the Web application which changes the SQL statement to their advantages.
- It can harm the database in various ways:
  - change the database content,
  - retrieve sensitive data in the database like credit card or passwords,
  - execute system level commands that may cause the system to deny service to the application,
  - ...

What is SQL Injection?

- In **SQL injection attacks**, hackers inject a string input through the Web application which changes the SQL statement to their advantages.
- It can harm the database in various ways:
  - change the database content,
  - retrieve sensitive data in the database like credit card or passwords,
  - execute system level commands that may cause the system to deny service to the application,
  - ...

A credit card payment processing company called CardSystems had an SQL injection attack in June 2005, in which 263,000 credit card numbers were stolen from its database. CardSystems lost large amounts of business and its assets were acquired by another company.

SQL Injection – Example

- The following query is issued by a simplistic authentication procedure:

```
SELECT * FROM users WHERE name=‘jake’ and password=‘passwd’;
```

SQL Injection – Example

- The following query is issued by a simplistic authentication procedure:

```
SELECT * FROM users WHERE name=‘jake’ and password=‘passwd’;
```

- The attacker can change the SQL statement as follows:

```
SELECT * FROM users WHERE name=‘jake’
                  AND password=‘p’ OR ‘x’=‘x’;
```

```
SELECT * FROM users WHERE name=‘jake’
                  AND password=‘p’; DROP TABLE users; --;’;
```

SQL Injection – Example

- The following query is issued by a simplistic authentication procedure:

```
SELECT * FROM users WHERE name='jake' and password='passwd';
```

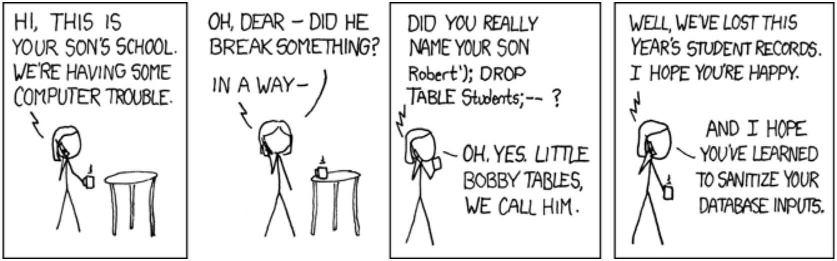
- The attacker can change the SQL statement as follows:

```
SELECT * FROM users WHERE name='jake'
                  AND password='p' OR 'x'='x';
```

```
SELECT * FROM users WHERE name='jake'
                  AND password='p'; DROP TABLE users; --;';
```

- Because the query here is constructed from strings, the use of quotes has turned the original WHERE condition into a condition **that is always true**.

SQL Injection – Another Example


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

link to this comic: http://xkcd.com/327/

SQL Injection – Protection Techniques

- Protection against SQL injection attacks can be achieved by applying certain rules to all Web-accessible procedures and functions.
  - **Parameterized queries** is used to improve security by preventing SQL injection (best solution). For example,

    ```
    PreparedStatement stmt=conn.prepareStatement(
          "SELECT * FROM users WHERE name=? and password=?");
    stmt.setString(1, user_name);
    stmt.setString(2, user_passwd);
    ```

  - **Input validation** is used to remove or escape characters from input strings. For example,

    ```
    "SELECT * FROM users WHERE name = ‘" + escape(user_name) +
          "’ and password= ‘" +escape(user_passwd) +"’"
    ```

    In this case, user_name can't be `p’ OR ‘x’=‘x`.

Summary

- Database security is critical.
  - Ensure that only authenticated users can access the system (i.e., **authentication**).
  - Ensure that authenticated users can access only data/interfaces that they are authorized to access (i.e., **authorization**).
- SQL injection attacks are an important security threat. Nevertheless, avoiding SQL injection vulnerabilities is much easier than you might think.
  - SQL Injection (`http://www.w3schools.com/sql/sql_injection.asp`)
  - Testing for SQL Injection (`https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005)`)
  - SQL Injection Prevention Cheat Sheet (`https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet`)
  - "**Foundations of Security: What Every Programmer Needs To Know**" by Neil Daswani, Christoph Kern, and Anita Kesavan