SQL Injections

By

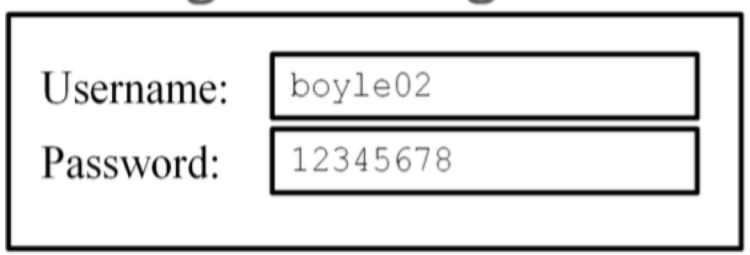
Luigi Penaloza

SQL Injection allows attackers to manipulate SQL statements to return results that were not intended. Most often used to attack websites.

Attackers can use the retrieved information for malicious purposes including modifying databases and utilizing information you shouldn't have access to.

Legitimate

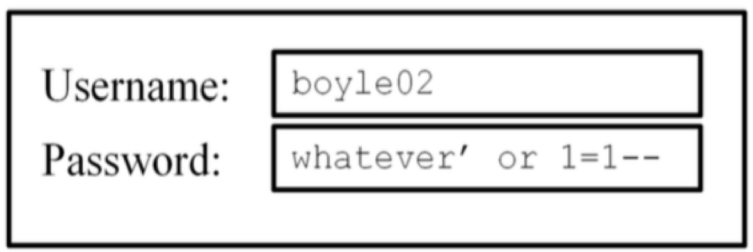
SQL statement below shows parameters passed to a database for a *legitimate* login:



SELECT FROM Users WHERE username='**boyle02**' AND password='**12345678**';

Attack

SQL statement below shows SQL injection by passing *unexpected* parameters and will *always* return a true value:



SELECT FROM Users WHERE username='**boyle02**' AND password='**whatever' or 1=1--**';

SQL Injections can:

Retrieve sensitive information

- Usernames/ Passwords
- Credit Card information
- SSN

Manipulate Data

- Delete records
- Truncate tables
- Insert records

Manipulate Database Objects

- Drop tables
- Drop databases

Retrieve System Information

- Identify software and version information
- Determine server hardware
- Get a list of databases
- Get a list of tables
- Get a list of column names within tables

Manipulate User Accounts

- Create new admin accounts
- Insert admin level accounts into the web-app
- Delete existing accounts

Sony Data Breaches:

- A series of 3 different data breaches: April, May and June 2011
- A combined **102.6 million** accounts stolen from:
  - Sony PlayStation Network (PSN)
  - Sony Online Entertainment
  - Sony pictures.com

- <u>All of this was done with SQL injections.</u>

- Believed that anonymous group was to blame.

Types of SQL Injections:

Blind SQL Injection

- Hardest type

1. Boolean-based SQL injection- by asking a question
2. Time-based SQL injection- by inducing a time delay

Error-Based SQL Injection

- An error-based SQL injection
- Simplest type; but only runs with MS-SQL Server
- Causes an application to show an error to extract the database
- ask a question to the database, and responds with an error including the data you asked for.

Union-Based SQL Injection

- Most popular type of SQL injection
- Uses the UNION statement, or the integration of two select statements, to obtain data from the database.

Injection Queries:

Blind SQL Injection

- http://localhost/htm/product-list.php?StatusFilter=**' drop table DimUser --**
- SELECT * FROM DimUser WHERE UserName='jprom' and Password=**'' drop table DimUser --**'

Conditional Response

- http://localhost/htm/product-details.php?ID=603 **and substring(@@VERSION,1,20) = 'Microsoft SQL Server'**
- SELECT ProductKey FROM DimProduct WHERE ProductKey=603 **and substring(@@VERSION,1,20) = 'Microsoft SQL Server'**

Return a List of Data ***(Such as User Accounts)***

- http://localhost/htm/product-list.php?StatusFilter=**' or 1=0 union select x=null, x=UserName, x=Password, x=null from DimUser --**
- SELECT ProductKey FROM DimProduct WHERE status=**'' or 1=0 union select x=null, x=UserName, x=Password, x=null from DimUser --**' ORDER BY ProductAlternateKey

Prevention:

Use bound variables with prepared statements.

Developers can prevent SQL Injection vulnerabilities in web applications by utilizing **parameterized database queries with bound, typed parameters and careful use of parameterized stored procedures in the database**.

This can be accomplished in a variety of programming languages:

- Java
- .NET
- PHP

Injected SQL will not be executed because it is treated as a value and not as a statement.

PHP Parameterized Queries

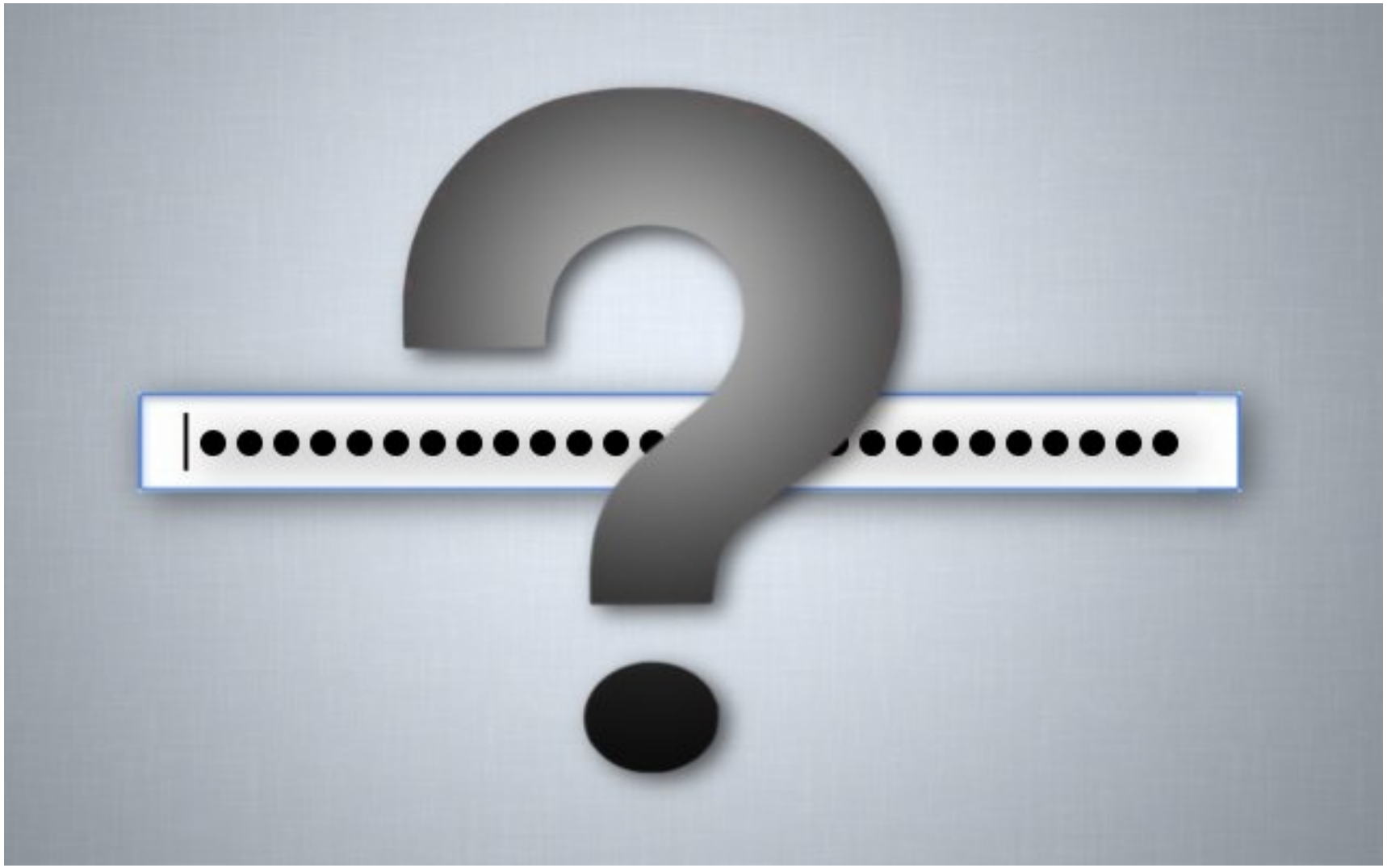
Not Safe (Non-Parameterized)

```
$tsql_States = sprintf("SELECT * FROM vw_DimState WHERE stateCode='%s' AND
countryCode='%s'", $_GET['State'], $_GET['Country']);
$stmt_States = sqlsrv_query($conn, $tsql_States);
$row_States = sqlsrv_fetch_array($stmt_States, SQLSRV_FETCH_ASSOC);
```

Safe (Parameterized)

```
$tsql_States = "SELECT * FROM vw_DimState WHERE stateCode=? AND countryCode=
$params_States = array($_GET['State'], $_GET['Country']);
$stmt_States = sqlsrv_query($conn, $tsql_States, $params_States);
$row_States = sqlsrv_fetch_array( $stmt_States, SQLSRV_FETCH_ASSOC);
```

Thank you.



LUIGI PENALOZA.

Sources:

Corporate Computer Security by Randall J Boyle and Raymond R Panko

https://security.berkeley.edu

https://www.esecurityplanet.com

https://latesthackingnews.com