



UNIVERSITY OF LIVERPOOL

CYBER SECURITY SOCIETY

SQLi + Command Injection

Disclaimer

Anything you learn in these sessions is FOR EDUCATIONAL PURPOSES ONLY and we are NOT RESPONSIBLE FOR YOUR ACTIONS! The tools we will show you aren't illegal but using them against a network you don't own or where you don't have the explicit written permission to use them is HIGHLY ILLEGAL and almost always against the terms of service.

DO NOT UNDER ANY CIRCUMSTANCES USE THE TOOLS AND TECHNIQUES SHOWN AGAINST ANY UNIVERSITY OWNED PRODUCT, WEBSITE OR NETWORK, YOU WILL BE PUNISHED BY THE DEPARTMENT/UNIVERSITY AND COULD BE PROSECUTED IN SOME CASES.

There are hundreds of websites where you can practice these techniques in a safe, legal environment without the risk of causing real damage or facing prosecution.

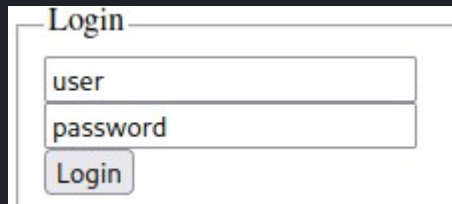
SQL Injection

- Structured Query Language is used to interact with most databases
- SQL Injection, similar to command injection, is when you can escape some user input in order to run your own SQL commands.
- SQLi is pretty uncommon in the wild but is good to know as a low hanging fruit and seen often in CTFs



How is SQL used

```
1 <fieldset><legend>Login</legend>
2 <form method="POST">
3   <input type="text" name="username" placeholder="username"><br>
4   <input type="text" name="password" placeholder="password"><br>
5   <input type="submit" value="Login">
6 </form>
7 <pre>
8 <?php
9 // Check whether a username and password have been submitted
10 if (isset($_POST["username"]) && isset($_POST["password"])) {
11   // Connect to the database
12   $database = new mysqli("127.0.0.1", "root", "password", "web");
13
14   // Make a database query using the submitted username and password
15   $result = $database->query(
16     "SELECT id, username, password FROM users WHERE username='" . $_POST["username"] . "' AND password='" . $_POST["password"] . "'";
17   );
18
19   // Quit if query errored
20   if ($result == false) {
21     die($database->error);
22   }
23
24   // Check whether any users were found
25   if ($result->num_rows > 0) {
26     // Output ID of found user
27     $row = $result->fetch_assoc();
28     echo "Welcome " . $row["id"];
29   } else {
30     echo "Invalid login";
31   }
32 }
33 ?>
34 </pre>
35 </fieldset>
```



A screenshot of a web login form. The form has a title "Login" and contains two input fields: "user" and "password". Below the input fields is a "Login" button. The form is styled with a light background and a simple border.

SELECT id, username, password FROM users WHERE username='user' AND password='password'

How do we exploit this

- Use ' to escape string and insert our own SQL

<pre>' OR '1'='1 ' OR 1=1 -- ' OR 1=1 #</pre>	Always pass WHERE check (login as different account)
<pre>' UNION ALL SELECT 1, 'admin', 'admin'</pre>	Insert arbitrary data into response (Need to provide same number of rows as the original query)
<pre>' UNION ALL SELECT 0, flag, 0 FROM flags</pre>	Select data from another table This can also be used to enumerate the database using information_schema



```

1 <fieldset><legend>Login</legend>
2 <form method="POST">
3   <input type="text" name="username" placeholder="username"><br>
4   <input type="text" name="password" placeholder="password"><br>
5   <input type="submit" value="Login">
6 </form>
7 <pre>
8 <?php
9 // Check whether a username and password have been submitted
10 if (isset($_POST["username"]) && isset($_POST["password"])) {
11   // Connect to the database
12   $database = new mysqli("127.0.0.1", "root", "password", "web");
13
14   // Make a database query using the submitted username and password
15   $result = $database->query(
16     "SELECT id, username, password FROM users WHERE username=' . $_POST["username"] . ' AND password=' . $_POST["password"] . '"
17   );
18
19   // Quit if query errored
20   if ($result == false) {
21     die($database->error);
22   }
23
24   // Check whether any users were found
25   if ($result->num_rows > 0) {
26     // Output ID of found user
27     $row = $result->fetch_assoc();
28     echo "Welcome " . $row["id"];
29   } else {
30     echo "Invalid login";
31   }
32 }
33 ?>
34 </pre>
35 </fieldset>

```

➔ SELECT id, username, password FROM users WHERE username='user' AND password="" OR '1'='1'



Variants of SQLi

- **Filtered**
The code removes and/or escapes some characters to try to prevent injection
- **Blind**
You cannot see the raw result of the query



Avoiding filtering

- Encode your string as hex
 - In SQL 'admin' is the same as 0x61646d696e
- If spaces are blocked use comments
 - SELECT/**/flag/**/FROM/**/example
- Multi-character Unicode characters



Time-based blind

```
/*Resulting query - Time-based attack to verify database version. */  
SELECT * FROM card WHERE id=1-IF(MID(VERSION(),1,1) = '5', SLEEP(15), 0)
```

- If the version starts with '5' the server will wait for 15 seconds before sending the response
- This then needs to be repeated for every character in the thing you are trying to ex-filtrate
- In general don't do this without an automatic script (takes a long time)



How to avoid being SQLi

- Use your libraries built in parametrization

```
<?php

mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
$mysqli = new mysqli("example.com", "user", "password", "database");

/* Non-prepared statement */
$mysqli->query("DROP TABLE IF EXISTS test");
$mysqli->query("CREATE TABLE test(id INT, label TEXT)");

/* Prepared statement, stage 1: prepare */
$stmt = $mysqli->prepare("INSERT INTO test(id, label) VALUES (?, ?)");

/* Prepared statement, stage 2: bind and execute */
$id = 1;
$label = 'PHP';
$stmt->bind_param("is", $id, $label); // "is" means that $id is bound as an integer and $label as a string

$stmt->execute();
```



Sqlmap

- This is a tool designed to automate SQLi
- It is very useful for doing blind injections
- Sometimes it is blocked by WAF, proxy, or rate-limiting

```
-$ sqlmap --wizard
```

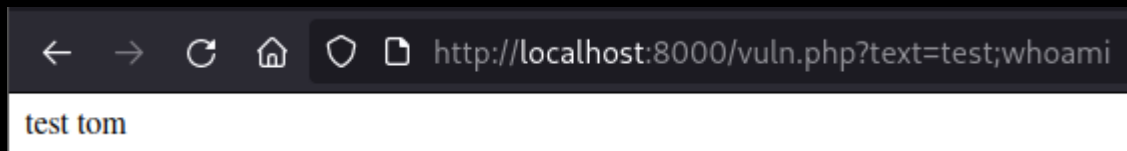
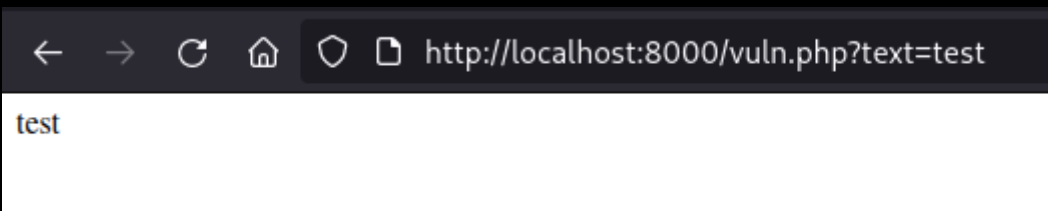


Command injection

- Command injection is when you exploit a website or service to run system level commands.
- The most basic form of this is when user input is passed directly to a command as an argument.



```
1 <?php
2 echo shell_exec("echo " . $_GET["text"]);
3 ?>
```



Command separators

<code>command1;command2</code>	Runs command1 then command2
<code>command1 && command2</code>	Runs command1 then command2 if command1 didn't error
<code>comamnd1 command2</code>	Runs command1 then command2 if command1 did error

Arguments don't matter

```
> echo foo bar baz; pwd && cd /  
foo bar baz  
/home/tom
```



Filtering

- When blocking command injection user input may be filtered.
- Regularly blocked characters include:
; & | ` \$
- Only way to get around this is keep trying a different method if you get blocked



Command substitution

- You can use `$(command)` or ``command`` to insert the output of the command as an argument to another.
- This is useful where other methods are filtered or you are in an escaped string.




```
1 <?php
2 echo shell_exec("echo \"\" . addslashes($_GET["text"]) . "\"");
3 ?>
```

← → ↻ 🏠 🛡️ 📄 http://localhost:8000/vuln2.php?text=test

test

← → ↻ 🏠 🛡️ 📄 http://localhost:8000/vuln2.php?text=test;whoami

test;whoami

← → ↻ 🏠 🛡️ 📄 http://localhost:8000/vuln2.php?text=\$(whoami)

tom

Blind

- For blind injection you cannot see the output of the command you run.
- You may still be able to see whether command succeeded: this can tell you if your injection is working.
- You will need to find another way to exfil data though. Reverse shells are especially useful when doing blind attacks.



Challenges

SQLi

tryhackme.com/room/sqlinjectionlm - more guidance

tryhackme.com/room/sqlilab

Command injection

tryhackme.com/room/injection

