



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

CORSO DI PENETRATION TESTING  
AND ETHICAL HACKING

# Metodologie di Penetration Testing

## *Momentum: 1*

STUDENTE

**Vincenzo Emanuele Martone**

DOCENTE

**Prof. Arcangelo Castiglione**

Anno Accademico 2022-2023

<b>Indice</b>	<b>i</b>
<b>Elenco delle figure</b>	<b>iii</b>
<b>1 Introduzione</b>	<b>1</b>
1.1 Processo di <i>Penetration Testing</i> . . . . .	1
1.2 Strumenti utilizzati . . . . .	2
1.3 Infrastruttura di rete . . . . .	2
1.4 Struttura del documento . . . . .	3
<b>2 Pre-Exploitation</b>	<b>4</b>
2.1 Target Scoping . . . . .	4
2.2 Information Gathering . . . . .	4
2.3 Target Discovery . . . . .	5
2.3.1 Informazioni preliminari . . . . .	5
2.3.2 Scansione con <i>nmap</i> . . . . .	6
2.3.3 Determinazione <i>MAC Address</i> con <i>arping</i> . . . . .	6
2.3.4 Scansione con <i>arp-scan</i> . . . . .	7
2.3.5 OS Fingerprinting attivo con <i>nmap</i> . . . . .	7
2.3.6 OS Fingerprinting passivo con <i>p0f</i> . . . . .	8
2.4 Target Enumeration . . . . .	9
2.4.1 <i>TCP</i> port scanning con <i>nmap</i> . . . . .	9
2.4.2 <i>UDP</i> port scanning con <i>unicornscan</i> . . . . .	11

---

2.5	Vulnerability Mapping . . . . .	12
2.5.1	OpenVAS . . . . .	12
2.5.2	Nessus . . . . .	13
2.5.3	DIRB . . . . .	14
2.5.4	Nikto2 . . . . .	15
2.5.5	OWASP ZAP . . . . .	15
2.5.6	Paros Proxy . . . . .	15
2.5.7	WhatWeb . . . . .	15
2.5.8	WafW00f . . . . .	15
2.5.9	sqlmap . . . . .	15
2.5.10	Osservazioni sulle vulnerabilità rilevate . . . . .	15

---

## Elenco delle figure

---

1.1	Infrastruttura di rete virtuale . . . . .	3
2.1	Output del comando <i>ifconfig</i> . . . . .	5
2.2	Output del comando <i>nmap</i> ( <i>ping scan</i> ) . . . . .	6
2.3	Output del comando <i>arping</i> . . . . .	6
2.4	Output del comando <i>arp-scan</i> . . . . .	7
2.5	Output del comando <i>nmap</i> ( <i>SO Fingerprinting</i> ) . . . . .	8
2.6	Output del comando <i>p0f</i> (risposta <i>HTTP</i> ) . . . . .	9
2.7	Output del comando <i>p0f</i> (risposta <i>SSH</i> ) . . . . .	10
2.8	Output del comando <i>nmap</i> ( <i>aggressive scan</i> ) . . . . .	10
2.9	Output del comando <i>unicornscan</i> . . . . .	11
2.10	Aerogramma dei rilevamenti di <i>OpenVAS</i> . . . . .	13
2.11	Ortogramma dei rilevamenti di <i>OpenVAS</i> . . . . .	13
2.12	Aerogramma dei rilevamenti di <i>Nessus</i> . . . . .	14

Il presente documento ha come obiettivo quello di illustrare le metodologie utilizzate nell'ambito dell'attività progettuale svolta nel contesto del corso di *Penetration Testing and Ethical Hacking*, tenuto dal prof. Arcangelo Castiglione presso l'Università degli studi di Salerno durante l'anno accademico 2022/2023. L'attività progettuale in questione consiste nello svolgimento del processo di *Penetration Testing* su un asset vulnerabile *by-design*; nello specifico, è stata scelta la macchina virtuale *Momentum: 1* messa a disposizione sulla piattaforma *VulnHub* dall'utente *ALIENUM*.

### 1.1 Processo di *Penetration Testing*

Il processo di *Penetration Testing* è stato svolto in maniera conforme alle modalità illustrate durante il corso, pertanto sono state previste (in ordine) le seguenti fasi:

1. **Target Scoping:** definizione degli accordi tra le parti coinvolte nel processo di *Penetration Testing*;
2. **Information Gathering:** raccolta di informazioni relative all'asset sia dal punto di vista della parte umana che dal punto di vista tecnologico;
3. **Target Discovery:** individuazione della macchina target all'interno della rete;
4. **Vulnerability Mapping:** individuazione delle vulnerabilità presenti sulla macchina target;

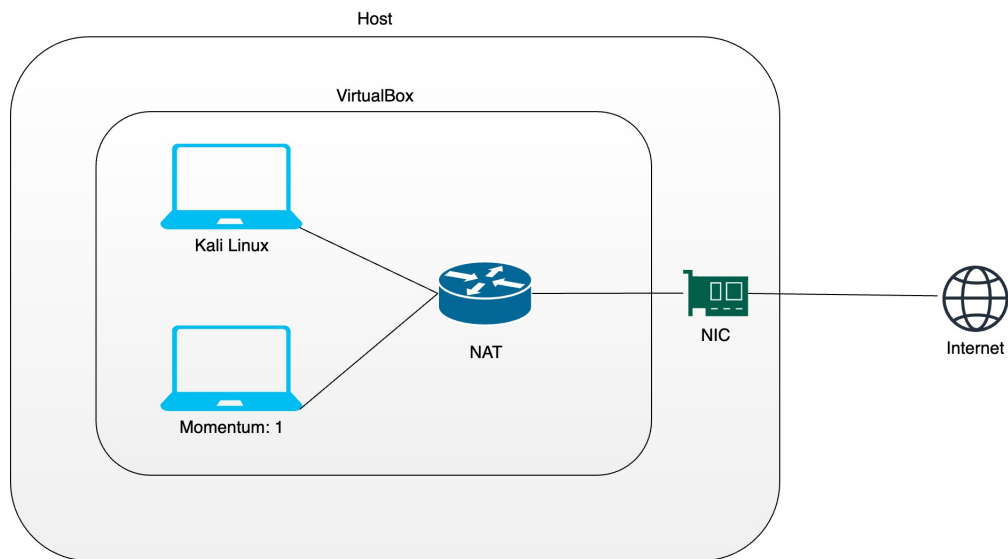
5. **Target Exploitation:** sfruttamento delle vulnerabilità individuate nel corso della fase precedente, finalizzato all'ottenimento dell'accesso alla macchina target;
6. **Privilege Escalation:** ottenimento dei massimi privilegi sulla macchina target al fine di acquisire ulteriori informazioni;
7. **Maintaining Access:** realizzazione di opportuni software, chiamati *backdoor*, volti al mantenimento dell'accesso sulla macchina target, in modo tale da evitare di dover rieseguire le precedenti fasi per accedervi nuovamente.

## 1.2 Strumenti utilizzati

Al fine di svolgere l'attività di *Penetration Testing* è risultato necessario l'impiego di un ambiente di virtualizzazione che coinvolgesse la macchina target ed un'eventuale macchina attaccante dotata di opportuni strumenti utili all'analisi dell'asset vulnerabile. L'ambiente di virtualizzazione utilizzato è *VirtualBox 7.0.8 r156879* mediante il quale è stata configurata una rete virtuale la cui infrastruttura verrà trattata nell'ambito del successivo paragrafo. La scelta del sistema operativo della macchina attaccante è ricaduta su *Kali Linux* (di cui è stata installata la release 2023.2) in quanto risulta essere una delle distribuzioni *Linux* più utilizzate nell'ambito di contesti come *Cybersecurity*, *Penetration Testing* e *Digital Forensics*. *Kali Linux* fornisce diversi strumenti preinstallati, utili per l'analisi da svolgere; alcuni di questi strumenti sono stati ampiamente utilizzati nell'ambito del processo di *Penetration Testing*, per cui verranno elencati e descritti nell'ambito della trattazione delle diverse fasi del processo svolto.

## 1.3 Infrastruttura di rete

La configurazione dell'infrastruttura di rete risulta cruciale in quanto permette la comunicazione tra la macchina target e quella attaccante, oltre a consentire a quest'ultima di collegarsi alla rete per aggiornare i tool di cui dispone ed accedere ai database di vulnerabilità, *exploit* e *payload*. Mediante l'apposito strumento messo a disposizione da *VirtualBox* è stata realizzata una rete virtuale con *NAT* avente come indirizzo *IPv4 10.0.2.0/24*, alla quale sono state collegate le macchine virtuali *Kali* e *Momentum*. L'infrastruttura di rete è illustrata nella figura 1.1 in una versione semplificata che non tiene conto degli host virtuali di *VirtualBox* utilizzati per la gestione del *NAT* e del *DHCP*.



**Figura 1.1:** Infrastruttura di rete virtuale

## 1.4 Struttura del documento

La presente trattazione verrà suddivisa in quattro capitoli:

- Il primo capitolo fornisce una panoramica sulle fasi del lavoro svolto, sugli strumenti utilizzati e sull'infrastruttura della rete virtuale;
- Il secondo capitolo tratta la fase di *Pre-Exploitation* che copre le fasi di *Target Scoping*, *Information Gathering*, *Target Discovery* e *Vulnerability Mapping*;
- Il terzo capitolo tratta la fase di *Target Exploitation*;
- Il quarto capitolo tratta la fase di *Post-Exploitation* che copre le fasi di *Privilege Escalation* e *Maintaining Access*.

## 2.1 Target Scoping

Il processo di *Penetration Testing*, come evidenziato nella fase introduttiva, ha uno scopo puramente didattico, per cui non è prevista una fase di accordo tra le parti coinvolte in quanto l'asset da analizzare è una macchina virtuale vulnerabile *by design*. Non vi è, infatti, un cliente dal quale raccogliere requisiti e con il quale definire obiettivi di business e modelli dei costi. Il processo verrà svolto senza particolari vincoli formali relativi all'asset.

## 2.2 Information Gathering

La caratterizzazione dell'asset da analizzare può generalmente avvenire mediante molteplici *tool* e coinvolgere diversi aspetti dell'asset stesso. Dal momento che si sta trattando una macchina virtuale vulnerabile *by-design* contestualizzata in un'attività progettuale avente uno scopo didattico non risulta utile ricorrere a particolari tecniche *OSINT* (*Open Source INTelligence*), né a tecniche volte all'ottenimento di informazioni di routing e record DNS. Sono state, tuttavia, consultate le informazioni di base dell'asset disponibili sulla piattaforma *VulnHub* che mette a disposizione la macchina virtuale. Le informazioni fornite sono le seguenti:

- **Nome della macchina:** *Momentum: 1*;
- **Sistema Operativo:** *Linux*;



- **DHCP Server:** abilitato;
- **Indirizzo IP:** assegnato in automatico.

Non risultano, dunque, note le informazioni relative all'indirizzo *IP* della macchina né le credenziali di accesso alla stessa.

## 2.3 Target Discovery

L'individuazione della macchina *Momentum: 1* all'interno della rete è stata effettuata, in accordo con quanto descritto nel capitolo introduttivo, utilizzando una macchina virtuale con *Kali Linux* connessa alla medesima rete.

### 2.3.1 Informazioni preliminari

Prima di procedere alla trattazione delle metodologie di individuazione della macchina target è necessario considerare alcuni aspetti dell'architettura di rete virtuale nell'ambito della quale è stata svolta l'attività di *Penetration Testing*. La gestione del *NAT* e del *DHCP* da parte di *VirtualBox* fa sì che risultino connessi alla rete degli host aventi *IP* 10.0.2.1, 10.0.2.2 e 10.0.2.3. Alla rete risulterà altresì connessa la macchina virtuale con *Kali Linux* della quale è stato rilevato l'indirizzo *IP* (10.0.2.15) mediante il comando *ifconfig* il cui output è illustrato nella figura 2.1.

```
(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::7525:725e:c670:3d88 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:c7:e1:36 txqueuelen 1000 (Ethernet)
    RX packets 3719 bytes 238023 (232.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6349 bytes 385837 (376.7 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 2856 bytes 159140 (155.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2856 bytes 159140 (155.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

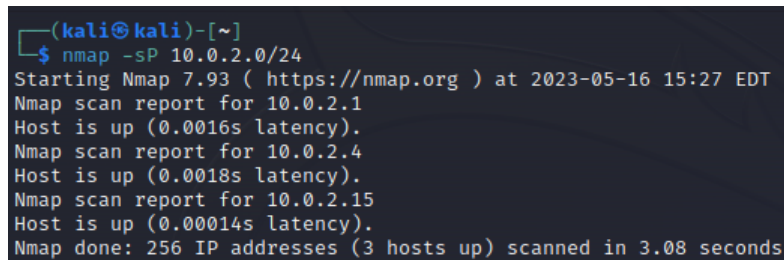
Figura 2.1: Output del comando *ifconfig*

### 2.3.2 Scansione con *nmap*

Come evidenziato in precedenza, l'indirizzo IP di *Momentum: 1* non è noto in quanto viene assegnato mediante il servizio di *DHCP* di *VirtualBox*. Per tale ragione è stata eseguita una scansione volta alla rilevazione della macchina target sulla rete *10.0.2.0/24* mediante il comando:

```
$ nmap -sP 10.0.2.0/24
```

Questo comando effettua un *ping scan* di tutti gli host della rete specificata in input [1], nell'ambito della quale vengono rilevati 3 host attivi, come mostrato nella figura 2.2. Dal momento che, come specificato in precedenza, l'indirizzo *10.0.2.1* fa riferimento ad un host di *VirtualBox* e l'indirizzo *10.0.2.15* è relativo alla macchina virtuale con *Kali*, risulta immediato stabilire che l'indirizzo IP di *Momentum: 1* è *10.0.2.4*.



```
(kali㉿kali)-[~]  
$ nmap -sP 10.0.2.0/24  
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-16 15:27 EDT  
Nmap scan report for 10.0.2.1  
Host is up (0.0016s latency).  
Nmap scan report for 10.0.2.4  
Host is up (0.0018s latency).  
Nmap scan report for 10.0.2.15  
Host is up (0.00014s latency).  
Nmap done: 256 IP addresses (3 hosts up) scanned in 3.08 seconds
```

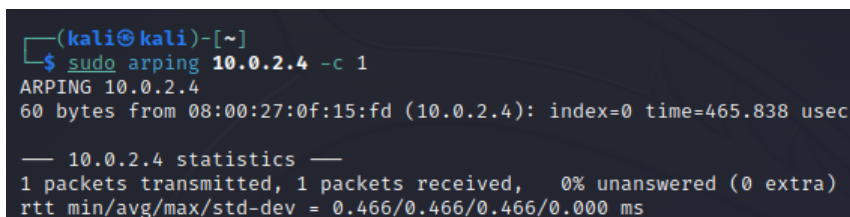
Figura 2.2: Output del comando *nmap* (*ping scan*)

### 2.3.3 Determinazione MAC Address con *arping*

A partire dall'indirizzo IP di *Momentum: 1*, risulta possibile arricchire la conoscenza della macchina target individuandone il MAC Address. Ciò è possibile mediante il comando:

```
$ sudo arping 10.0.2.4 -c 1
```

Tale comando invia un'ARP Request al dispositivo indicato in input [2]. Mediante l'output, illustrato nella figura 2.3, si stabilisce che il MAC Address di *Momentum: 1* è *08:00:27:0f:15:fd*.



```
(kali㉿kali)-[~]  
$ sudo arping 10.0.2.4 -c 1  
ARPING 10.0.2.4  
60 bytes from 08:00:27:0f:15:fd (10.0.2.4): index=0 time=465.838 usec  
  
— 10.0.2.4 statistics —  
1 packets transmitted, 1 packets received, 0% unanswered (0 extra)  
rtt min/avg/max/std-dev = 0.466/0.466/0.466/0.000 ms
```

Figura 2.3: Output del comando *arping*

```
(kali㉿kali)-[~]  
$ sudo arp-scan 10.0.2.0/24  
Interface: eth0, type: EN10MB, MAC: 08:00:27:c7:e1:36, 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      08:00:27:5b:8c:04      (Unknown)  
10.0.2.4      08:00:27:0f:15:fd      (Unknown)
```

Figura 2.4: Output del comando *arp-scan*

### 2.3.4 Scansione con *arp-scan*

Durante il processo di *Penetration Testing* è stato utilizzato un approccio volto all’ottenimento delle medesime informazioni mediante molteplici tool al fine di confrontarne i risultati per massimizzare il quantitativo di informazioni ottenute nell’ambito di una determinata fase. A tale scopo ci si è serviti del tool *arp-scan* per effettuare una scansione sulla rete *10.0.2.0/24* mediante il comando:

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

L’output del comando, illustrato nella figura 2.4, mostra che la scansione ha rilevato 4 host, per ciascuno dei quali ha fornito il relativo indirizzo *IP* ed il relativo *MAC Address*. La precedente scansione (effettuata con il tool *nmap*) non ha rilevato gli host *192.168.1.2* e *192.168.1.3*, tuttavia, come evidenziato in precedenza, questi host sono gestiti da *VirtualBox* per cui non hanno rilevanza nel processo di *Penetration Testing* effettuato. L’host avente indirizzo *IP 10.0.2.4* e *MAC Address 08:00:27:0f:15:fd* è relativo alla macchina target.

### 2.3.5 OS Fingerprinting attivo con *nmap*

Al fine di arricchire la conoscenza relativa alla macchina target è stata effettuata un’operazione di *OS detection* mediante il comando:

```
$ sudo nmap -O 10.0.2.4
```

L’output ottenuto (figura 2.5) fornisce diverse informazioni relative al sistema operativo in esecuzione sulla macchina target. È possibile stabilire che si tratta di un sistema *Linux* presumibilmente ad una versione 4.15 o 5.6 (quando *nmap* non riesce a stabilirlo con precisione mostra tutte i possibili match [3]); il tool fornisce, infine, la *CPE*<sup>1</sup> di riferimento (*cpe:/o:linux:linux\_kernel:4 cpe:/o:linux\_kernel:5*). Sono, altresì, presenti informazioni relative

<sup>1</sup>CPE (*Common Platform Enumeration*) è un sistema di naming strutturato per sistemi operativi, software e packages [4]

```
(kali㉿kali)-[~]
$ sudo nmap -O 10.0.2.4
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-16 15:38 EDT
Nmap scan report for 10.0.2.4
Host is up (0.00051s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:0F:15:FD (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
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 2.88 seconds
```

**Figura 2.5:** Output del comando *nmap* (SO Fingerprinting)

alle porte aperte ed ai servizi attivi sulla macchina target, che nell’ambito della fase di *Target Discovery*, sono state sfruttate unicamente per effettuare OS Fingerprinting passivo; sulla macchina target risultano aperte la porta 80 (servizio *HTTP*) e la porta 22 (servizio *SSH*).

### 2.3.6 OS Fingerprinting passivo con *p0f*

La fase di OS Fingerprinting attivo non ha portato all’individuazione dell’esatta versione del sistema operativo in esecuzione sulla macchina target: è stata, dunque, svolta una fase di OS Fingerprinting passivo, finalizzata all’ottenimento di ulteriori informazioni in merito, utilizzando il tool *p0f*. La tecnica di fingerprinting passivo adottata consiste nel porsi in ascolto su una specifica interfaccia di rete ed ispezionare i pacchetti *TCP/IP* intercettati al fine di individuare le informazioni desiderate. Tale operazione è stata svolta mediante il comando:

```
$ sudo p0f -i eth0
```

Il passo successivo consiste nel fare in modo che la macchina target invii pacchetti sull’interfaccia di rete *eth0*; a tale scopo sono state effettuate delle richieste ai servizi esposti dalla macchina, ossia *HTTP* e *SSH*, mediante i comandi:

```
$ curl -X GET http://10.0.2.4/
$ ssh user@10.0.2.4
```

A tali richieste corrispondono delle risposte, intercettate dal tool *p0f* e dalle quali sono state ottenute le informazioni riportate nelle figure 2.6 e 2.7. Dalle sezioni relative alla macchina target (che riportano il parametro *server* uguale a *10.0.2.4/80* o a *10.0.2.4/22*) si evince che il tool *p0f* non è riuscito a stabilire la versione del sistema operativo in esecuzione in quanto al

```
.-[ 10.0.2.15/48760 → 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*45,7:mss,sok,ts,nop,ws:df:0  
|  
|_____  
|  
.-[ 10.0.2.15/48760 → 10.0.2.4/80 (mtu) ]-  
|  
| server    = 10.0.2.4/80  
| link      = Ethernet or modem  
| raw_mtu   = 1500  
|  
|_____  
|  
.-[ 10.0.2.15/48760 → 10.0.2.4/80 (http request) ]-  
|  
| client    = 10.0.2.15/48760  
| app       = ???  
| lang      = none  
| params    = none  
| raw_sig   = 1:Host,User-Agent,Accept=[*/*]:Connection,Accept-Encoding,Accept-  
| Language,Accept-Charset,Keep-Alive:curl/7.88.1  
|  
|_____  
|  
.-[ 10.0.2.15/48760 → 10.0.2.4/80 (http response) ]-  
|  
| server    = 10.0.2.4/80  
| app       = Apache 2.x  
| lang      = none  
| params    = none  
| raw_sig   = 1:Date,Server,?Last-Modified,?ETag,Accept-Range=[bytes],?Content-  
| Length,?Vary,Content-Type:Connection,Keep-Alive:Apache/2.4.38 (Debian)  
|  
|_____  
|
```

**Figura 2.6:** Output del comando *p0f* (risposta HTTP)

parametro *os* è associata la stringa '???'. L'operazione di OS Fingerprinting passivo non ha, dunque, condotto ai risultati sperati in quanto non è stato possibile arricchire ulteriormente la conoscenza relativa alla versione del sistema operativo in esecuzione.

## 2.4 Target Enumeration

Nel corso delle precedenti fasi sono state svolte diverse scansioni volte al rilevamento della macchina target sulla rete virtuale. Tali scansioni hanno portato alla scoperta di alcuni servizi in esecuzione sulla macchina target, che nell'ambito della fase di *Target Enumeration* sono stati ulteriormente caratterizzati.

### 2.4.1 TCP port scanning con *nmap*

L'utilizzo del tool *nmap* risulta utile anche durante la fase di *TCP port scanning* in quanto, mediante le opzioni messe a disposizione, è possibile eseguire diversi tipologie di scansioni volte al rilevamento delle porte *TCP* aperte. Consultando la pagina del manuale di *Linux*

```

.-[ 10.0.2.15/46086 → 10.0.2.4/22 (syn+ack) ]-
|
| server   = 10.0.2.4/22
| os       = ???
| dist     = 0
| params   = none
| raw_sig  = 4:64+0:0:1460:mss*45,7:mss,sok,ts,nop,ws:df:0
|
|_____

.-[ 10.0.2.15/46086 → 10.0.2.4/22 (mtu) ]-
|
| server   = 10.0.2.4/22
| link     = Ethernet or modem
| raw_mtu  = 1500
|
|_____

```

**Figura 2.7:** Output del comando *p0f* (risposta SSH)

relativa ad *nmap* [1] è stato scoperto che l'opzione *-A* consente di effettuare un'*aggressive scan* che consiste in operazioni di *OS detection*, *version scanning*, *script scanning* e *traceroute*. Tale tipo di scansione risulta essere particolarmente efficace in quanto in grado di rilevare un maggior numero di informazioni rispetto alle altre tipologie di scansioni disponibili. È stato, dunque, eseguito il seguente comando:

```
$ nmap -A 10.0.2.4 -p- -oX aggressive_scan.xml
```

L'opzione *-p-* indica che il range di porte da scansionare va da 1 a 65535, mentre l'opzione *-oX* serve per salvare l'output del tool in formato *XML* sul file *aggressive\_scan*. Al fine di agevolarne la consultazione, il file è stato opportunamente convertito in *HTML* mediante il comando:

```
$ xsltproc aggressive_scan.xml -o aggressive_scan.html
```

#### Address

- 10.0.2.4 (ipv4)

#### Ports

The 65533 ports scanned but not shown below are in state: **closed**

- 65533 ports replied with: **conn-refused**

Port	State (toggle closed [0]   filtered [0])	Service	Reason	Product	Version	Extra info
22	tcp	open	ssh	syn-ack	OpenSSH	7.9p1 Debian 10+deb10u2
	ssh-hostkey	2048 5c8e2cccc1b03e7c0e2234d860314e62 (RSA) 256 81fdc64c5a500a27ea833864b98bbdc1 (ECDSA) 256 c18f87c1520927605f2e2de0080372c8 (ED25519)				
80	tcp	open	http	syn-ack	Apache httpd	2.4.38
	http-title	Momentum   Index				
	http-server-header	Apache/2.4.38 (Debian)				

**Figura 2.8:** Output del comando *nmap* (aggressive scan)

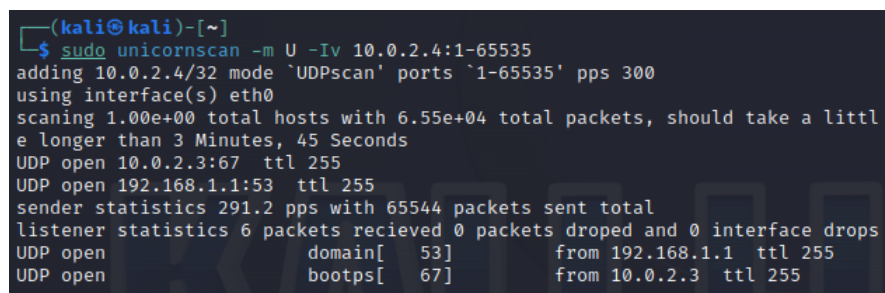
Il risultato dell'aggressive scan (illustrato nella figura 2.8) è reperibile al link: e fornisce svariate informazioni relative allo stato delle porte *TCP* della macchina target. Come già rilevato dalla scansione eseguita nell'ambito della fase di *SO Fingerprinting*, le porte 22 e 80 risultano aperte e sono utilizzate rispettivamente da un server *SSH* e da un server *HTTP*. Sono, altresì, riportati i valori delle chiavi pubbliche impiegate dagli algoritmi di autenticazione di *SSH* (*RSA*, *ECDSA*, *ED25519*), il nome della pagina di default del server *HTTP* (*Momentum* | *Index*) e le informazioni sulla versione del server *HTTP* (*Apache/2.4.38 (Debian)*). Un'ulteriore informazione di estrema importanza riguarda lo stato delle restanti 65533 porte che, avendo risposto con un messaggio di *conn-refused*, risultano senza dubbio chiuse; non sono, dunque, necessarie ulteriori scansioni per le porte *TCP*.

### 2.4.2 UDP port scanning con *unicornscan*

Dal momento che il tool *nmap* risulta inefficiente nell'ambito delle scansioni *UDP*, è stato utilizzato il tool *unicornscan*:

```
$ sudo unicornscan -m U -Iv 10.0.2.4:1-65535
```

L'opzione *-m U* indica la tipologia di scansione da eseguire (*UDP*), mentre l'opzione *-Iv* specifica che l'output deve essere stampato a schermo in tempo reale e che deve essere di tipo *verbose*. È stato, infine, specificato l'indirizzo *IP* dell'host da scansionare con il relativo range di porte (da 1 a 65535). I risultati, illustrati nella figura 2.9, mostrano che le porte *UDP* rilevate



```
(kali@kali)-[~]
└─$ sudo unicornscan -m U -Iv 10.0.2.4:1-65535
adding 10.0.2.4/32 mode 'UDPscan' ports '1-65535' pps 300
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a littl
e longer than 3 Minutes, 45 Seconds
UDP open 10.0.2.3:67 ttl 255
UDP open 192.168.1.1:53 ttl 255
sender statistics 291.2 pps with 65544 packets sent total
listener statistics 6 packets recieved 0 packets dropped and 0 interface drops
UDP open domain[ 53] from 192.168.1.1 ttl 255
UDP open bootps[ 67] from 10.0.2.3 ttl 255
```

Figura 2.9: Output del comando *unicornscan*

non sono relative alla macchina target, bensì al router e ad un host virtuale di *VirtualBox* e fanno riferimento rispettivamente al servizio *Domain Name System* (*DNS*) e al servizio *Bootstrap Protocol Server* (*BPS*) (l'elenco delle porte ben note è regolato dallo standard *RFC 1340* [5]). Tali porte sono state rilevate in quanto *unicornscan* segnala come aperte le porte relative agli host da cui riceve dei pacchetti sulla rete [6]. Si può, dunque, concludere che non vi sono porte *UDP* aperte sulla macchina target.



## 2.5 Vulnerability Mapping

Individuati i servizi erogati dalla macchina target, risulta opportuno svolgere una fase volta all'identificazione delle vulnerabilità della stessa. Nell'ambito di tale analisi ci si avvarrà sia di strumenti preinstallati in *Kali Linux* che di strumenti appositamente installati e configurati. L'utilizzo di molteplici tool risulta cruciale in tale fase in quanto le modalità di rilevazione delle vulnerabilità si basano su euristiche estremamente variabili. Le successive sezioni sono dedicate alla trattazione dei tool impiegati e delle relative vulnerabilità rilevate. In particolare sono stati sia utilizzati tool *general purpose* che tool specializzati nell'individuazione di vulnerabilità di *Web Application*, in quanto durante le analisi svolte nelle precedenti fasi è stato rilevato il servizio *HTTP* attivo sulla porta 80.

### 2.5.1 OpenVAS

*OpenVAS* (Open Vulnerability Assessment System) è un security scanner open source facilmente installabile su *Kali Linux*. Tale tool, di cui è stata installata la versione 22.4.1, mette a disposizione un'interfaccia *Web-based* mediante la quale sono stati configurati i parametri necessari allo svolgimento della scansione sulla macchina target; in particolare:

- Sono state scansionate tutte le 65535 porte;
- È stato utilizzato un *Minimum Quality of Detection* del 70% in modo da avere un soddisfacente compromesso tra rilevamento delle vulnerabilità e rischio di incorrere in falsi positivi;
- È stato utilizzato lo scanner di default di *OpenVAS*, che fa uso delle informazioni fornite dai servizi di feed *NVT*, *SCAP*, *CERT* e *GVMD DATA*;
- La configurazione impiegata per le scansioni è '*Full and fast*'.

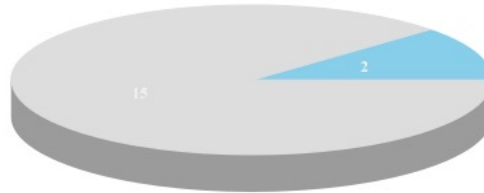
La scansione ha richiesto circa 14 minuti generando un report ('*openvas-report.pdf*') reperibile nella directory *tools-output* (o al seguente link: ). Sono state rilevate 15 informazioni di log e 2 vulnerabilità aventi una severity bassa (figg. 2.10 e 2.11); le vulnerabilità in questione sono le seguenti:

- **[Severity: 2.6] TCP Timestamps Information Disclosure**: la macchina target implementa la rilevazione dei timestamp via *TCP* dando la possibilità di computarne l'uptime;
- **[Severity: 2.1] ICMP Timestamp Reply Information Disclosure (CVE-1999-0524)**: la macchina target ha risposto ad una richiesta *ICMP* di timestamp. Tale informazione

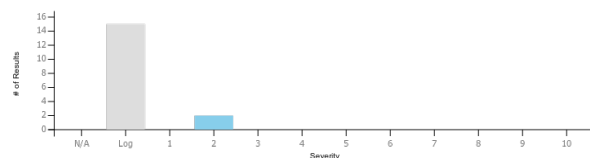


potrebbe essere sfruttata per violare generatori di numeri casuali *time-based* deboli presenti in servizi eventualmente installati sulla macchina target.

La scansione effettuata con *OpenVAS* non ha rilevato vulnerabilità gravi, bensì comportamenti dannosi sfruttabili da attaccanti sotto determinate condizioni.



**Figura 2.10:** Aerogramma dei rilevamenti di *OpenVAS*



**Figura 2.11:** Ortogramma dei rilevamenti di *OpenVAS*

### 2.5.2 Nessus

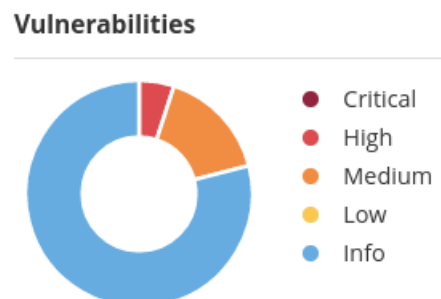
*Nessus* è un software proprietario che mette a disposizione diversi piani di utilizzo, tra cui quello gratuito *Essentials*, provvisto di funzionalità limitate ed utilizzato nell'ambito del presente processo di *Penetration Testing*, nella sua versione 10.5.2. Questo tool consente di effettuare numerose tipologie di scansioni, alcune delle quali sono disponibili con il piano *Essentials*. In totale sono state effettuate due differenti scansioni, ciascuna configurata con opportuni parametri:

- La prima scansione effettuata è di tipo '*Basic Network Scan*' ed è stata eseguita lasciando invariati i parametri predefiniti ad eccezione di quello relativo alle porte da scansionare, impostato a tutte le 65535 porte. Tale scansione non ha rilevato alcuna vulnerabilità ma solo informazioni di log, riportate nel report '*nessus-basic-report.pdf*' reperibile nella cartella '*tools-output*' (o al link: );
- La seconda scansione effettuata è di tipo '*Web Application Tests*' ed è specifica per il rilevamento delle vulnerabilità delle *Web App*. I parametri specificati per la scansione sono quelli di *default*, ad eccezione di quello relativo alle porte da scansionare, impostato

a tutte le 65535 porte, e di quello relativo alla tipologia di scansione, impostata come scansione complessa. Tale scansione ha rilevato 16 informazioni di log, 3 vulnerabilità aventi severity media ed una vulnerabilità avente severity alta (fig. 2.12). Il report *'nessus-web-all-ports-complex'* è reperibile nella cartella *'tools-output'* (o al link: ).

Complessivamente *Nessus* ha rilevato le seguenti vulnerabilità:

- **[Severity: 7.5] CGI Generic SSI Injection (HTTP headers):** il web server contiene degli script CGI che non effettuano un'opportuna sanificazione dell'input, rendendosi vulnerabile ad un *SSI Injection* che consente l'esecuzione di comandi arbitrari;
- **[Severity: 5.3] Browsable Web Directories:** il web server consente di navigare le directory;
- **[Severity: 5.0] Web Application Information Disclosure:** alla ricezione di una richiesta malformata, il Web Server fornisce il path fisico alle directory;
- **[Severity: 4.3] Web Application Potentially Vulnerable to Clickjacking:** il server non imposta un *X-Frame-Options* nell'header della risposta esponendo il sito ad attacchi di tipo *clickjacking*.



**Figura 2.12:** Aerogramma dei rilevamenti di *Nessus*

### 2.5.3 DIRB

Al fine di approfondire la conoscenza relativa al *Web Server*, ci si è avvalsi dell'utilizzo di *DIRB* (2.22), un *Web Content Scanner* preinstallato in *Kali Linux*, mediante il comando:

```
$ dirb http://10.0.2.4 -o /home/kali/Desktop/dirb-report.log
```

Il report (*'openvas-report.log'*), disponibile nella cartella *'tools-output'* (o al seguente link: ), riporta diversi percorsi relativi al *Web Server*, tra cui *'html'*, *'css'*, *'img'* e *'manual'* con le relative sottocartelle. La conoscenza di tali percorsi sarà utile nel corso delle successive fasi.

### 2.5.4 Nikto2

*Nikto2* è un tool *open source* facilmente installabile su *Kali Linux*, che consente di effettuare scansioni di vulnerabilità di *Web Server*. Nell'ambito del presente lavoro tale tool è stato utilizzato nella sua versione 2.5.0, eseguendo una scansione mediante il seguente comando:

```
$ nikto -h http://10.0.2.4 -C all -Format html -o nikto2_report.html
```

Il tool ha enumerato le seguenti vulnerabilità, di cui è reperibile un report (*'nikto2-report.log'*) nella directory *'tools-output'* (o al link: ):

- Assenza dell'opzione *X-Frame-Options* nel header come protezione dal clickjacking;
- Assenza dell'opzione *X-Content-Type-Options* nell'header, esponendo il browser al rischio di *MIME-sniffing*;
- Gli *ETag* della risposta espongono informazioni relative al numero dell'*inode*;
- La versione di *Apache* (2.4.38) risulta essere obsoleta;
- Il Server consente i metodi *GET*, *POST*, *OPTIONS* e *HEAD*;
- Il Server contiene un file di default di *Apache*, ossia */icons/README*;
- È possibile listare diverse directory del server, tra cui *'css'*, *'img'*, *'manual'* e *'manual/images'*.

### 2.5.5 OWASP ZAP

*OWASP ZAP* è un vulnerability scanner facente parte del progetto *Open Web Application Security Project*. Tale tool mette a disposizione una semplice *GUI* ambito del presente lavoro è stata utilizzata la versione 2.12.0

### 2.5.6 Paros Proxy

### 2.5.7 WhatWeb

### 2.5.8 WafW00f

### 2.5.9 sqlmap

### 2.5.10 Osservazioni sulle vulnerabilità rilevate

---

## Bibliografia

---

- [1] *nmap(1) - Linux man page*, Maggio 2023. (Citato alle pagine 6 e 10)
- [2] *arping(8) - Linux man page*, Maggio 2023. (Citato a pagina 6)
- [3] "Nmap manual." (Citato a pagina 7)
- [4] "Cpe dictionary - nist." (Citato a pagina 7)
- [5] "Assigned Numbers." RFC 1340, July 1992. (Citato a pagina 11)
- [6] *Unicornscan Documentation Getting Started*. (Citato a pagina 11)