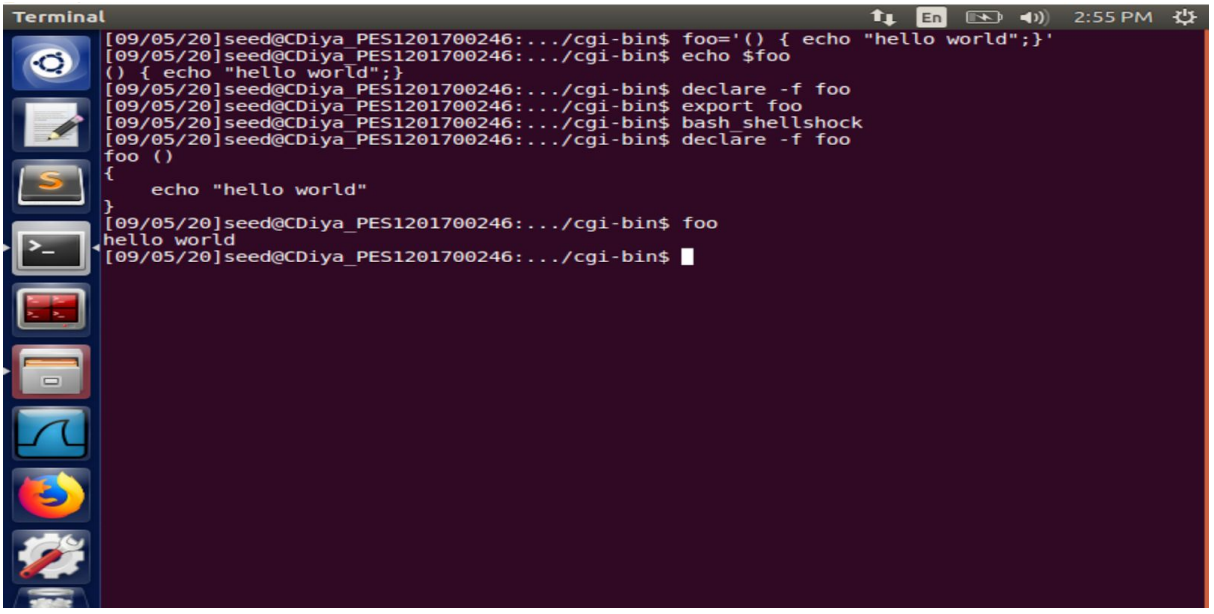
Information Security LAB 2

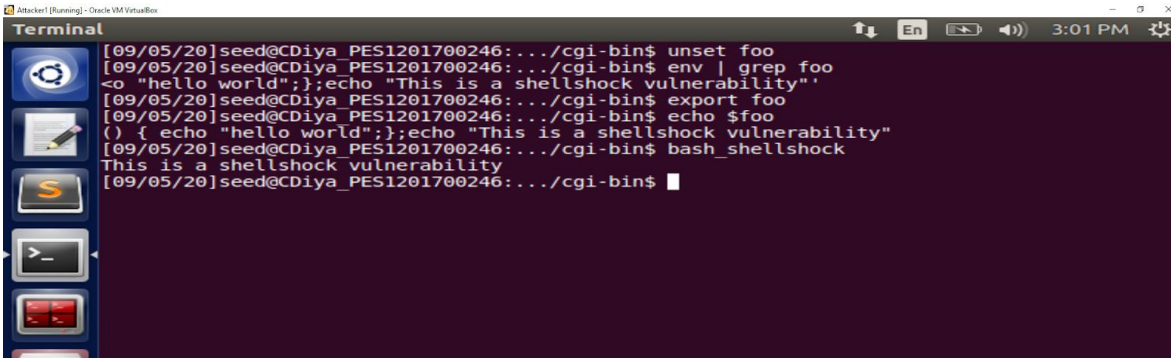
Shellshock Attack Lab

C Diya
PES1201700246

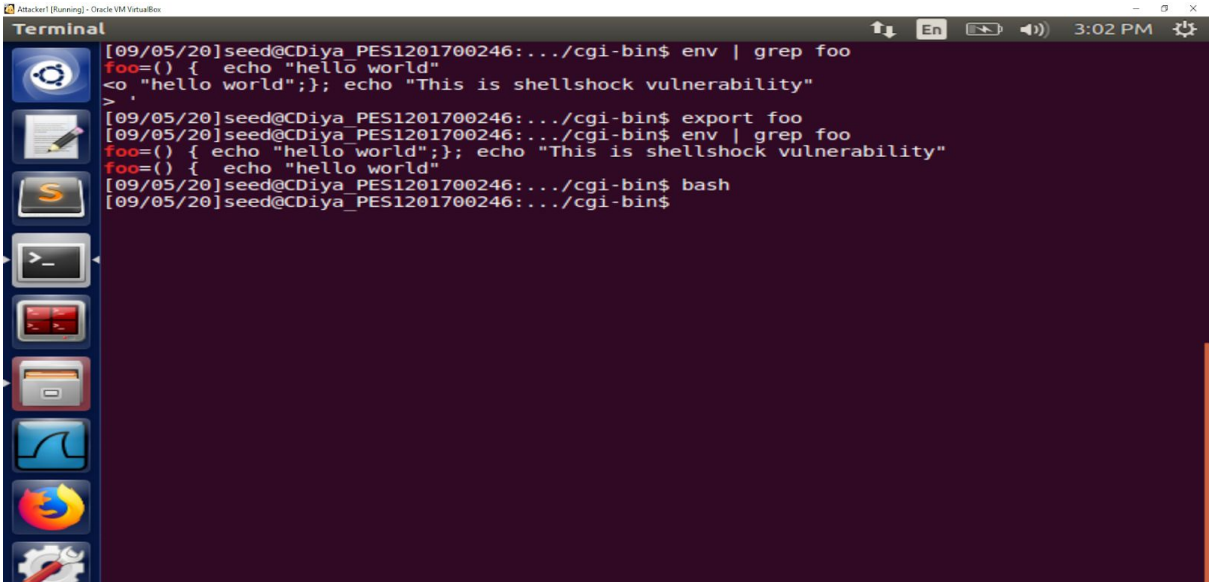
Task 1: Experimenting with Bash Function



Observation: The screenshot above shows the setting of an environment variable called “foo”. This environment variable is then declared as a function using the declare command. The foo function now contains the “Hello World” echo statement. This foo is then inherited into a bash_shellshock. On printing “foo” on the bash_shellshock, the “Hello World” program is printed from the function foo.

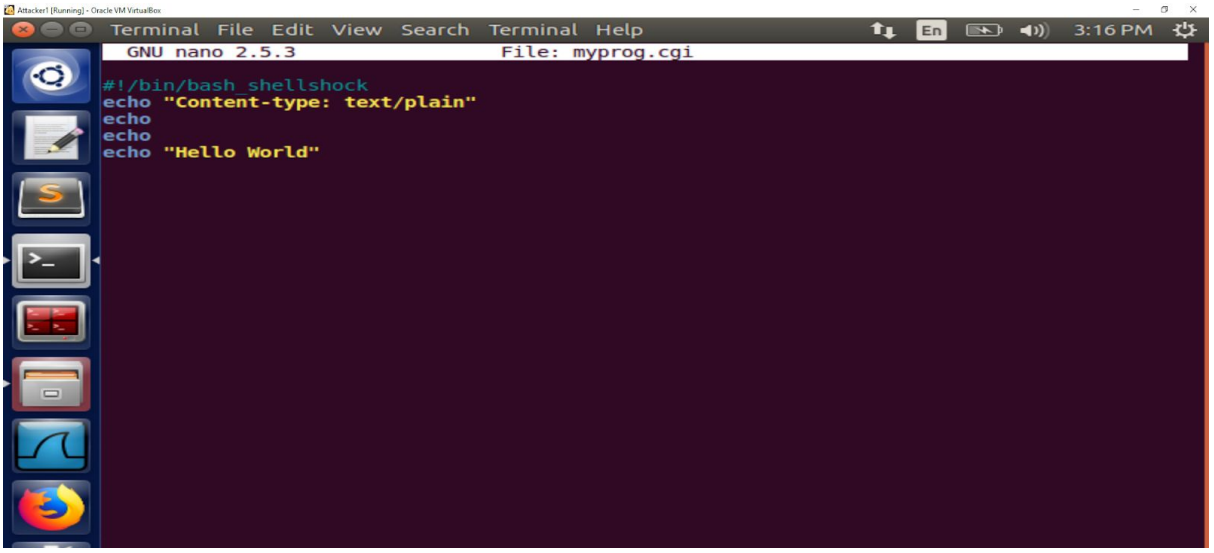


Observation: The vulnerability is further tested using the echo command. The screen shot above again shows the echo statement being printed once it enters the bash_shellshock.
When environment variables are exported and opened in another bash then this environment variable is inherited by the child bash. The child bash inherits the environment variable from the parent, parses it and now treats it as a function instead. Thus, executing foo in the child bash will echo “Hello World” as shown.

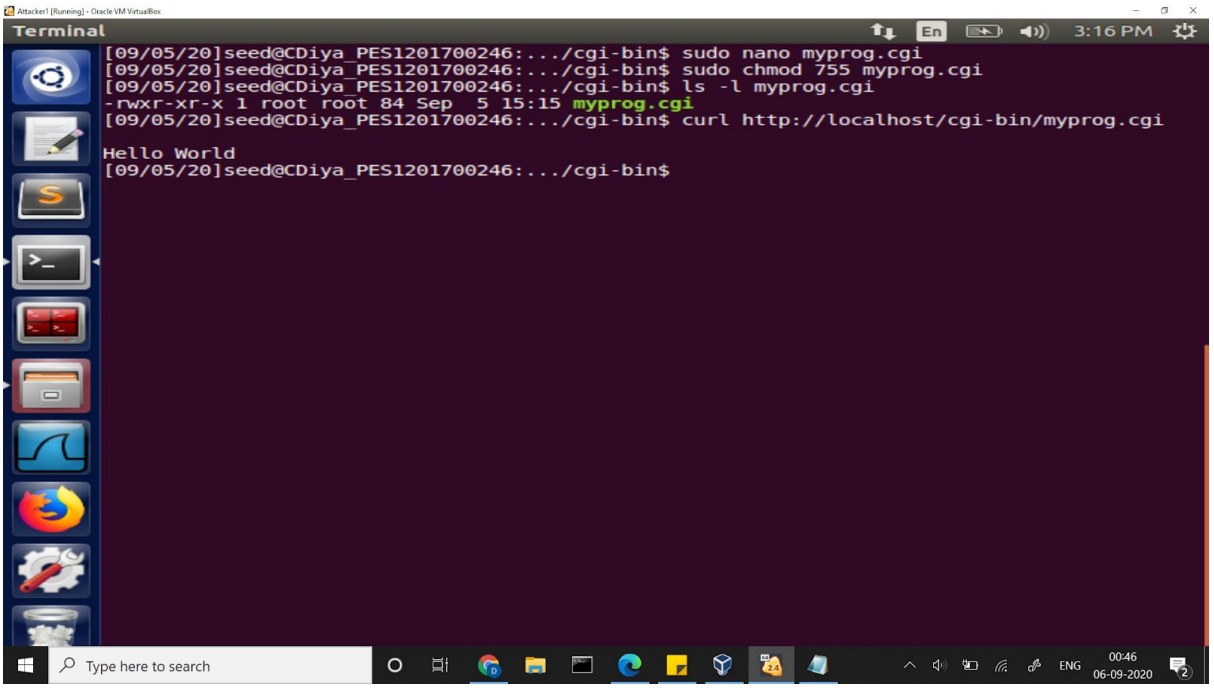


Observation: The screenshot above shows the testing on a patched version of bash. It can be observed that nothing gets printed since the echo command does not run.

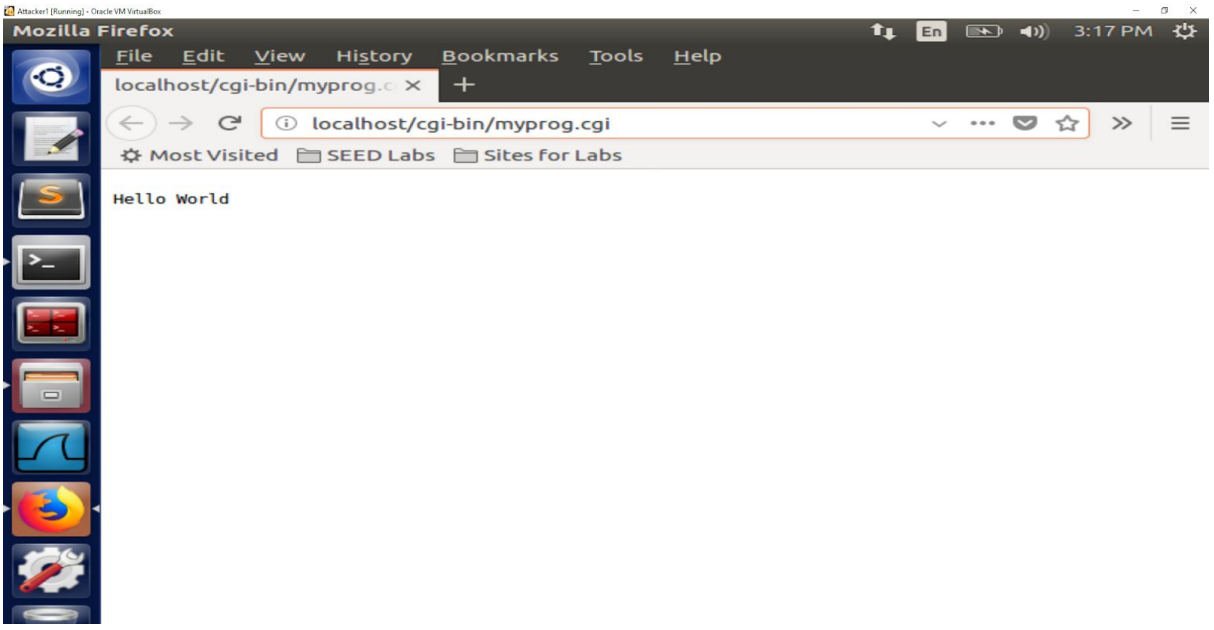
TASK 2: Setting up CGI programs



Observation: The cgi program shown above is created in the usr/lib/cgi-bin directory. The program is meant to execute “Hello World” in the bash_shellshock(unpatched version). The content type is text/plain.

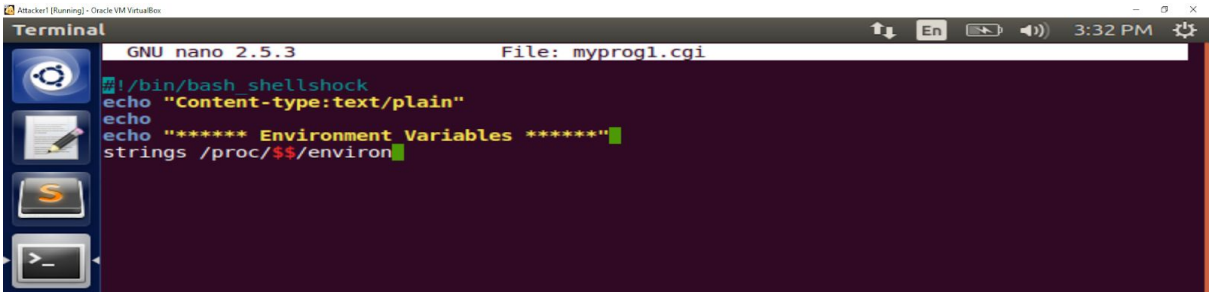


Observation: The screenshot above shows the HTTP request is generated using the curl command and this invokes the cgi script. 755 is used to make the script executable.
A cgi shell script can be used to launch a Shellshock attack on a web server and to gain privileges on the server. Web servers enable CGI, which is a standard method used to generate dynamic content on Web page.Therefore, before a CGI program is executed, a shell program will be invoked and the vulnerability of bash_shellshock will be exposed.

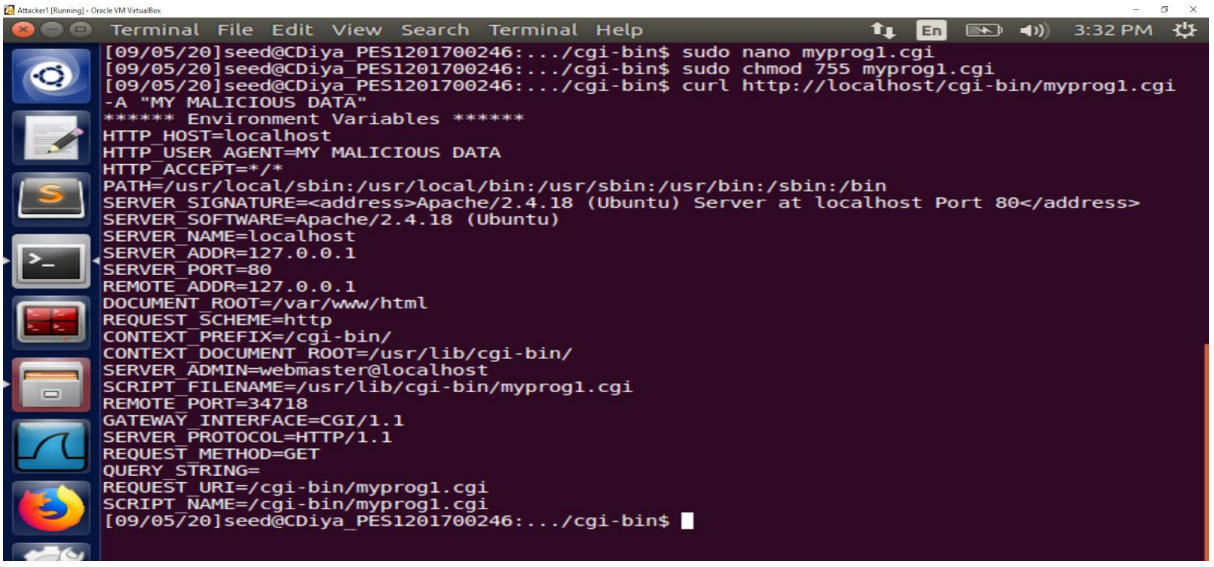


Observation: Furthermore, on viewing this from localhost, the Hello World can be seen. Thus, cgi scripts were used to print content on web servers.

TASK 3: Passing Data to Bash via Environment Variable

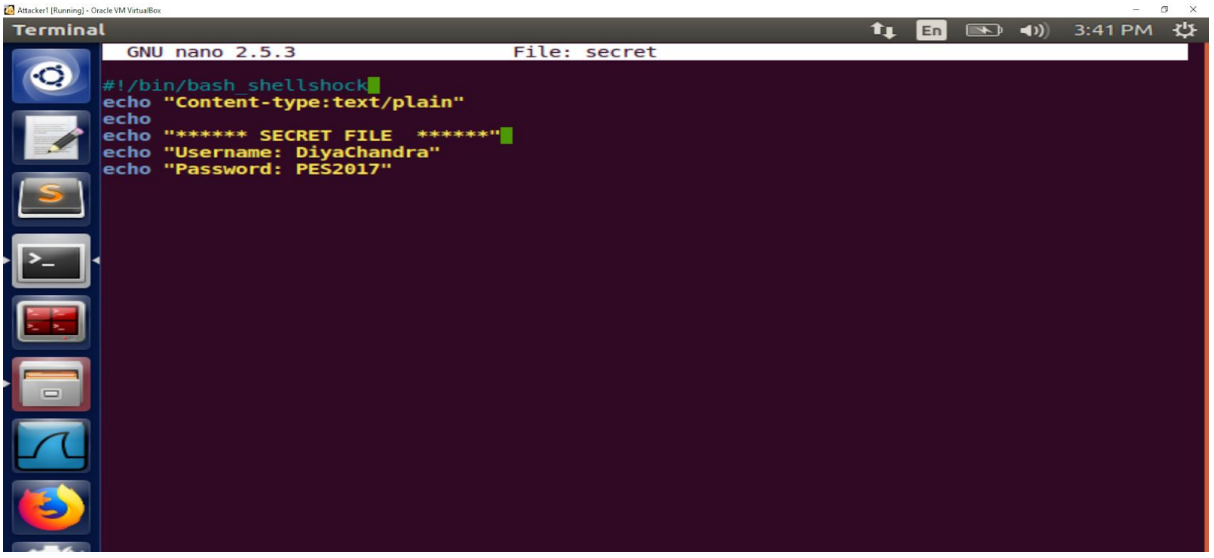


Observation: The cgi program written is used to print the environment variables.

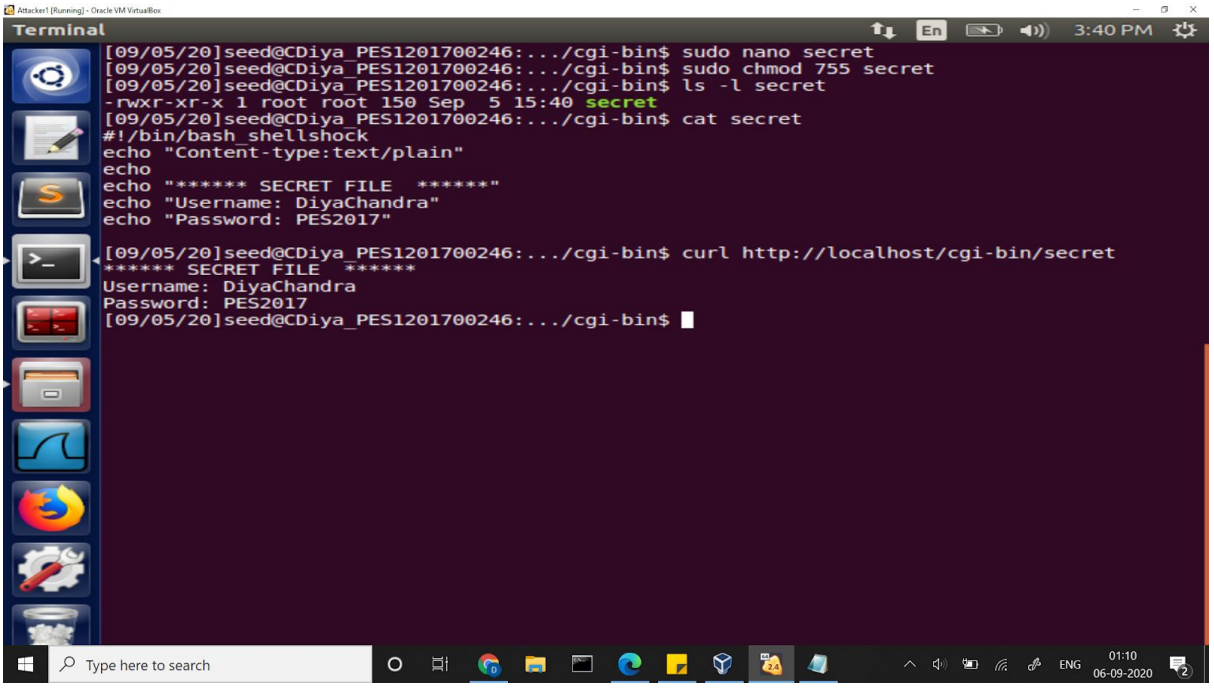


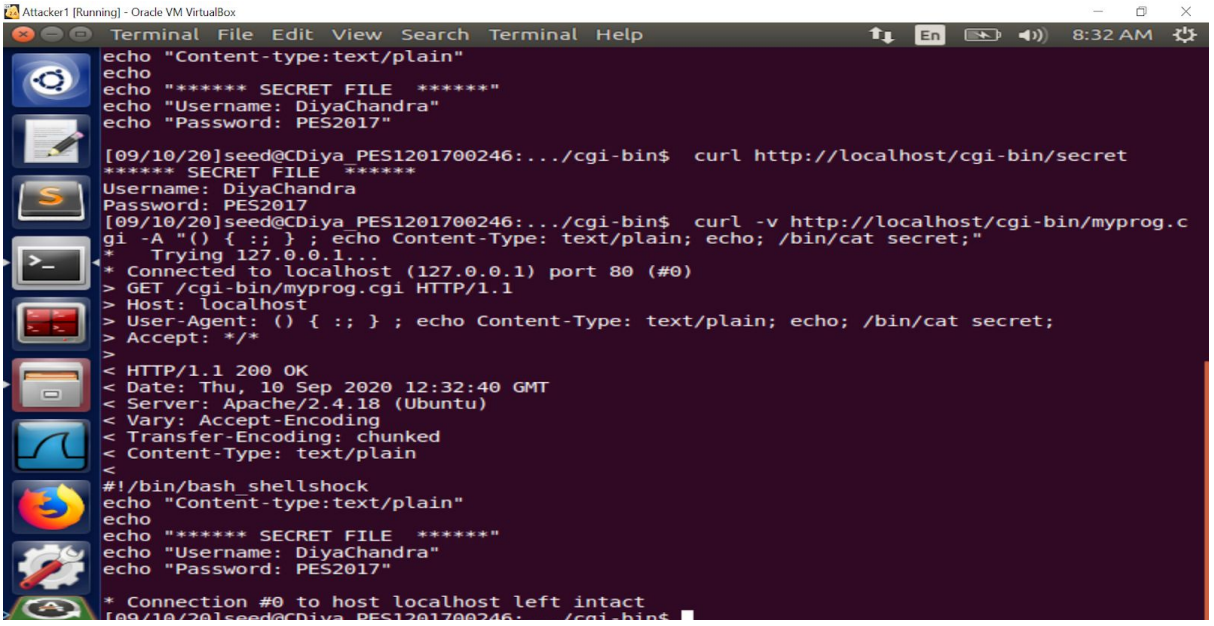
Observation: The cgi program written can be invoked using the curl command. The -A attached to the curl command can be used to add any additional data to set the HTTP_USER_AGENT header. This way any malicious data can be passed to to set headers to a HTTP request. This vulnerability can be used to inject an attacker's code into a target system. We pass "MALICIOUS DATA" to the header and it can be seen that the HTTP_USER_AGENT contains this.
Thus, any arbitrary string to the CGI program, and the string will show up in the content of one of the environment variables

TASK 4: Launching the Shellshock Attack

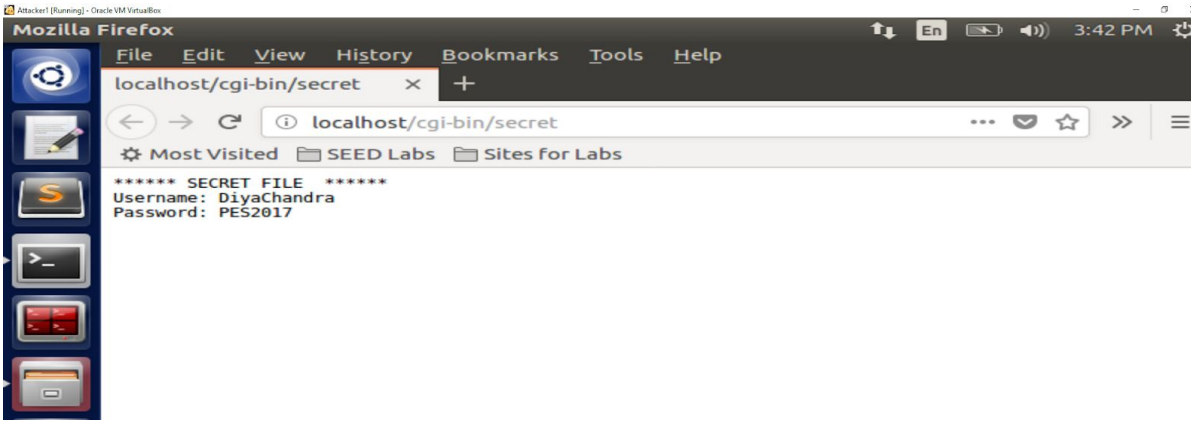


Observation: The file above is a secret file containing a username and password. This is stored in the cgi directory. The following information is attempted to be stolen

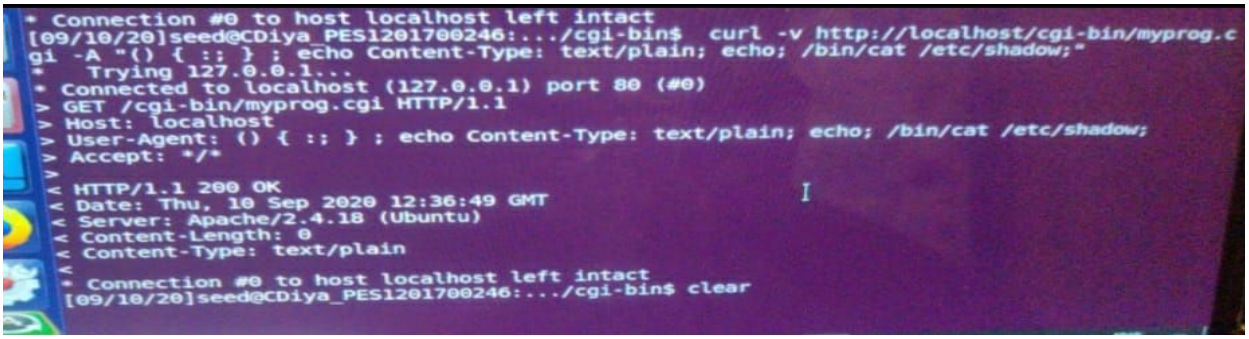




Observation: On running the curl command with the cat command on the victim machine to steal this secret data file, we send the cat command using our -A option in curl, which sets the user agent header. On running the command it can be seen that the contents of the files are stolen and printed on the console as shown



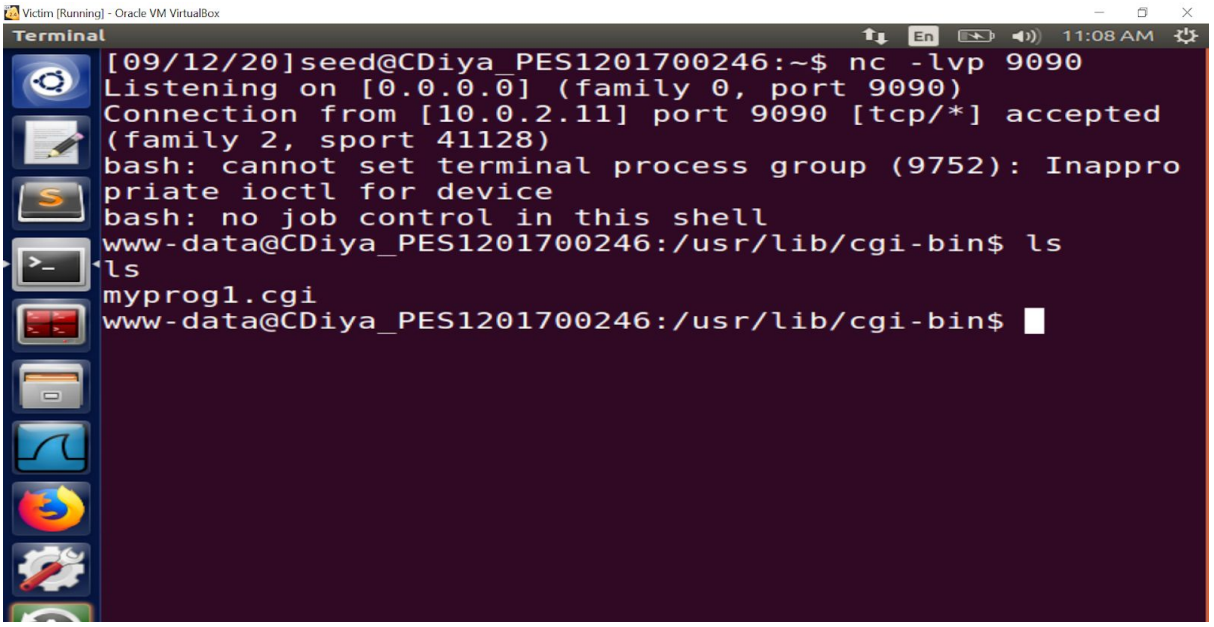
Observation: Furthermore, on viewing this from localhost,the secret content file can be seen. Thus, the shellshock attack has been launched and can steal confidential information on the victim machine.



Observation: On trying to steal the contents of the /etc/shadow file, we observe from the screenshot above that the contents are not accessible to us. This is because the file can be read only by the ROOT user and not just a seed user.

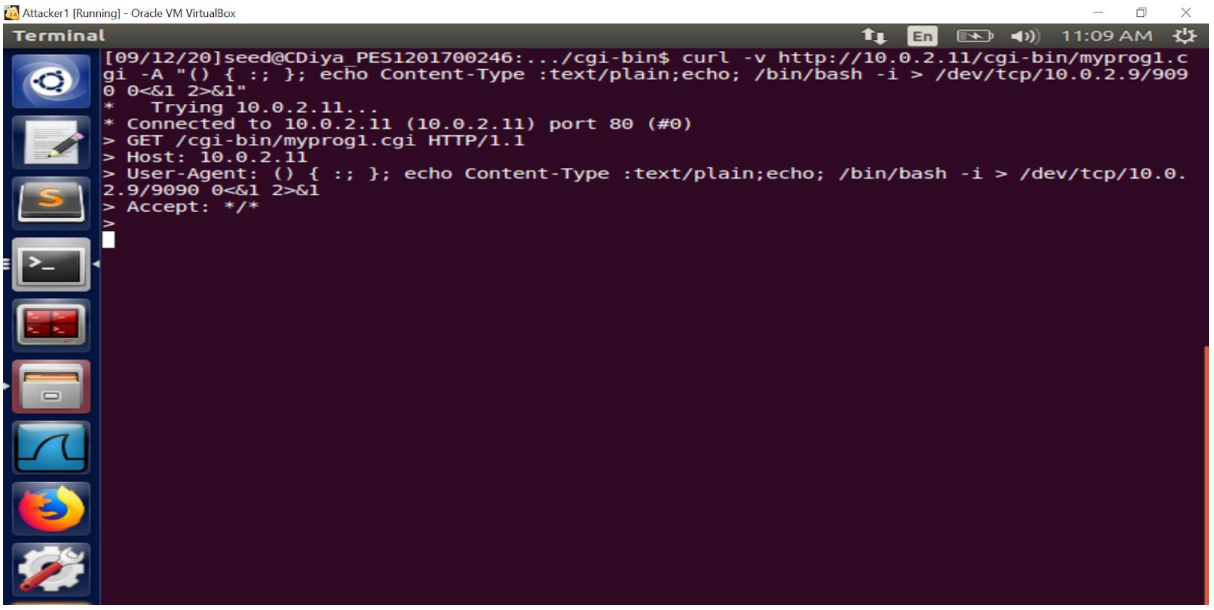
TASK 5:Getting a Reverse Shell via Shellshock Attack

10.0.2.9: Attacker machine
10.0.2.11: Victim machine



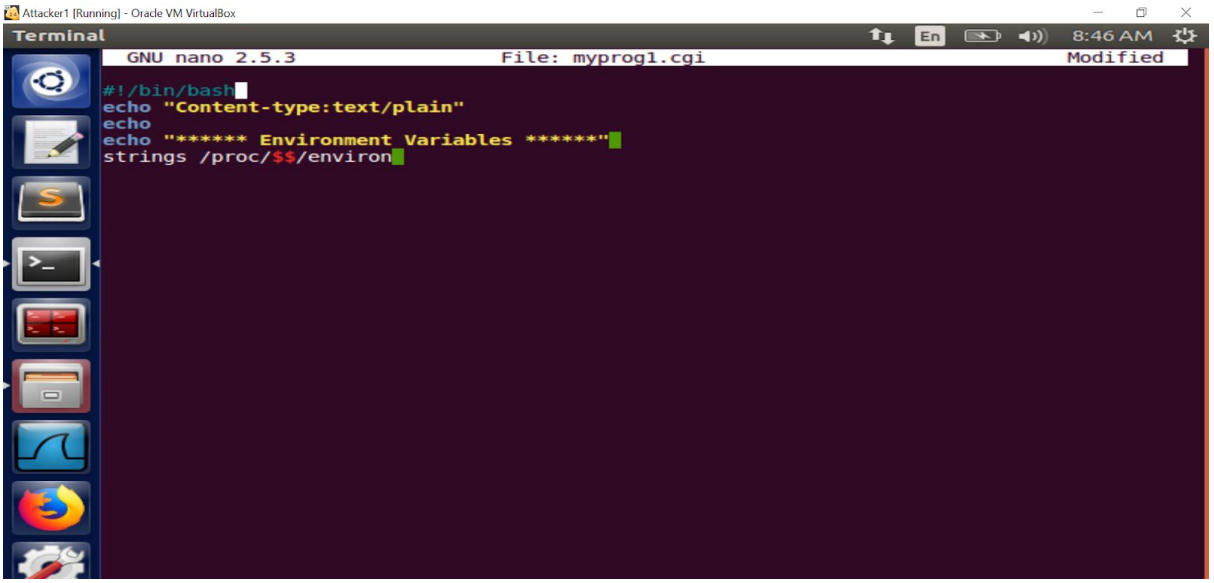
This experiment aimed towards launching an attack on a victim system and redirecting its input and output to the attacker. The attacker has access to a terminal that can type command into the victims systems and steal important data from the output.

Observation: The screenshot above shows the attacker(10.0.2.9) machine listening to the port 9090 using the netcat command. The victims(10.0.2.11) standard input and output has been redirected here.
Once the connection is made, the ls command performed on the attacker machine, leads to the victim machine and thus, outputs all the files present in the victim's current working directory. The two program names currently on the victim can be seen in the attacker terminal above.

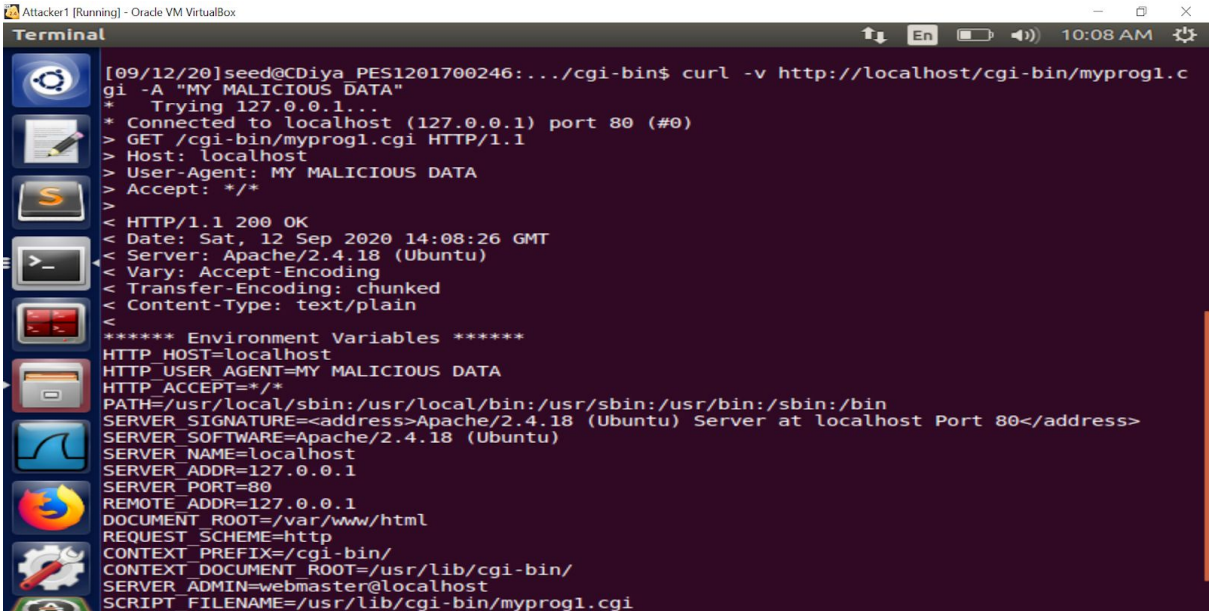


Observation: The screenshot above shows the curl command that is used to connect to the attacker machine. The port is defined as 9090 where the attacker will be listening.The connection between the victim and attacker has been established and thus, an interactive shell appears on the attacker machine that has access to the victim system.

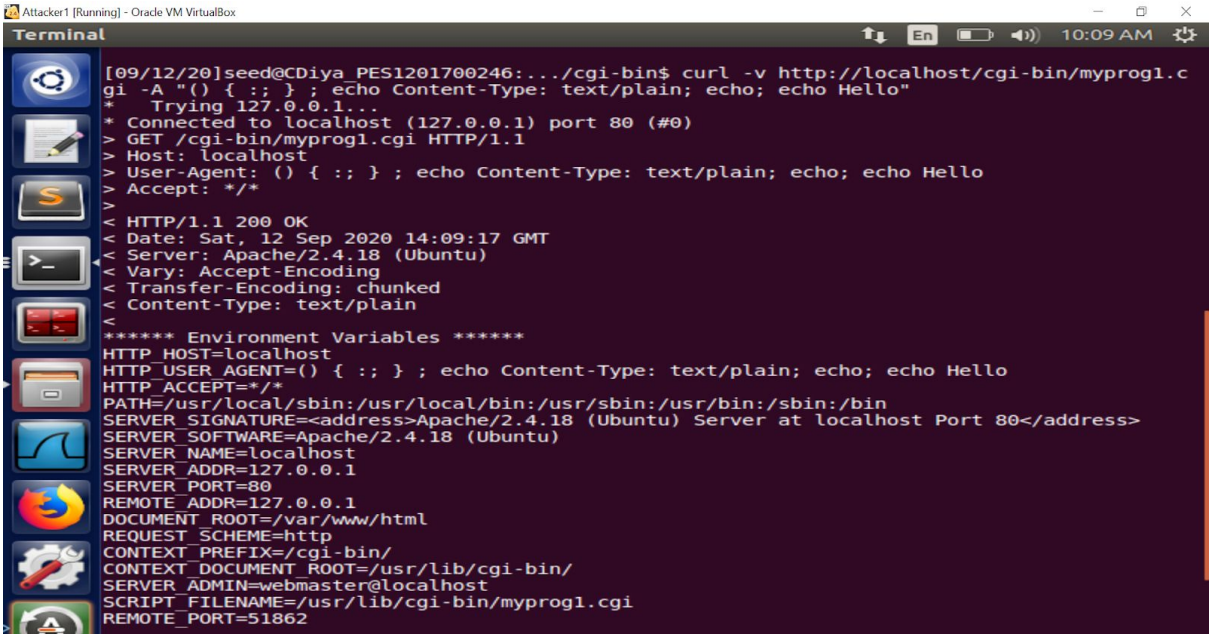
TASK 6:Using the Patched Bash



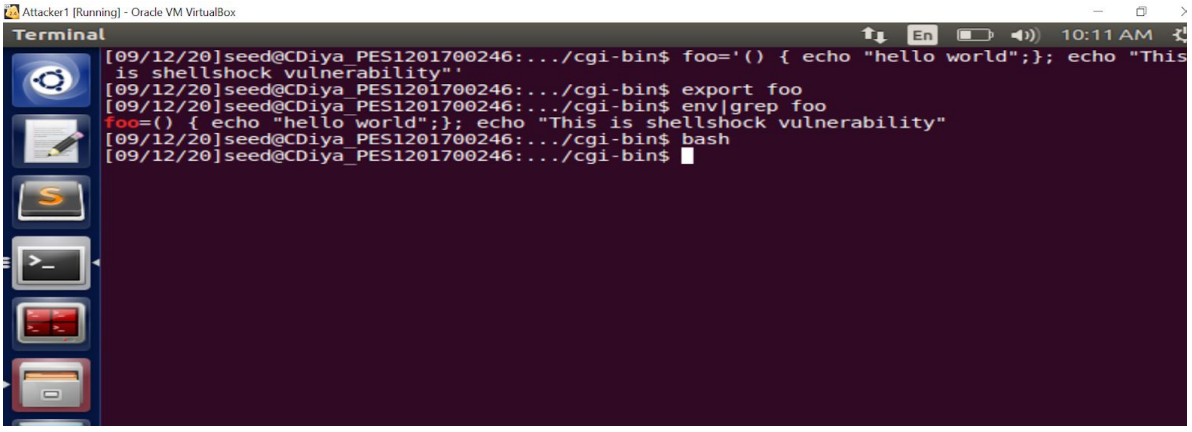
Observation: The program above uses the bash(patched version) to run 2 attacks. The first script is to print environment variables using a cgi shell script using bash



Observation: It can be observed from the screenshot above that the attack is not successful and that the header gets parsed as a string and not a command. The user_agent header only contains text and not any malicious code. Thus , this shows that using the patched version of bash prevents the attacks from happening.



Observation: The observation above shows that the curl command does not execute the attack and that the text passed would be parsed as a string and not as a command. Thus, Hello world is not printed since this is not a command but the given command is shown as a string. Thus, not malicious code can be executed or injected into a victim machine using bash(patched version)



Observation: When Task 1 is repeated, the same output is obtained using bash(patched version) Nothing gets printed on the console since it is not vulnerable to this attack