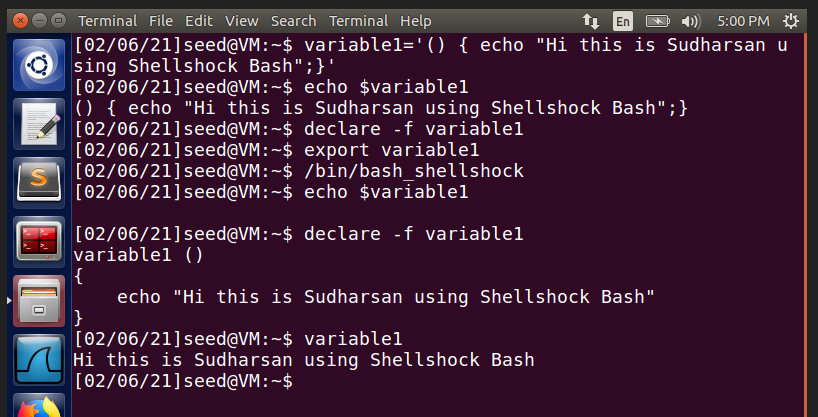
**ASSIGNMENT – 2**

Name: Sudharsan Srinivasan

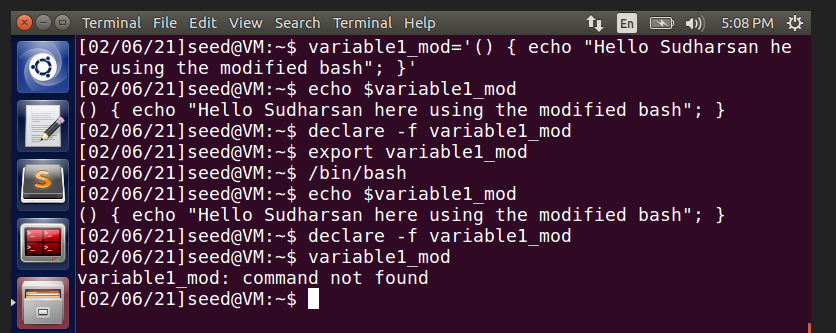
ID: 1001755919

**Task 1 – Experimenting with Bash Function:**

* For Task 1, we check to see if the Bash function is vulnerable to Shellshock or not. A variable (variable1) is declared with a value/string message stored in it as shown below.



* After setting the value for the variable, we declare and export the variable. **declare** is done to set the shell variables and **export** is used to make these shell variables, environment variables.
* Now, we use the **/bin/bash\_shellshock** command. This command makes the shell environment variables into a shell function. In the above image, it can be seen that variable variable1 has been converted into **variable1()** function. Thus, showing that the environment is vulnerable to Shellshock attack.
* Now, we create a new variable to check for modified bash activity as shown in the below image. Variable1\_mod is created.

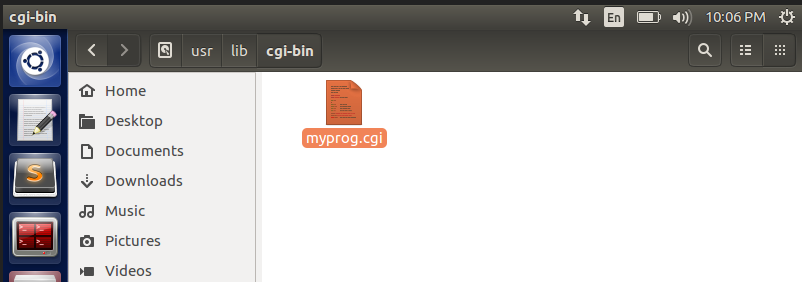


* After setting the value for the variable, we declare and export the variable. **declare** is done to set the shell variables and **export** is used to make these shell variables, environment variables.
* Next, we use **/bin/bash** command to check if the bash shell is vulnerable to Shellshock attack. Then, we use the echo command to display the contents of the variable and we see that the variable1\_mod displays the contents, showing that the attack does not affect bash shell.

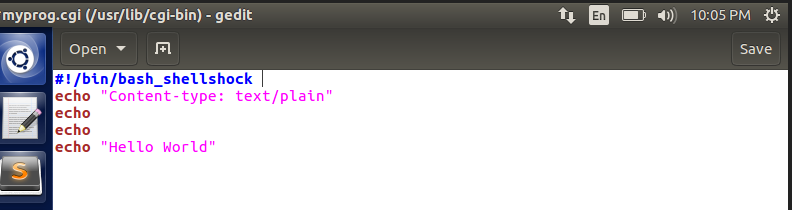
This shows that the bash shell program does not convert the variable into a function and displays the contents as it is, showing that it is not vulnerable to Shellshock attack, whereas the Shellshock program converts it into a function, showing that it is vulnerable to attack.

**Task 2 – Setting up CGI Programs:**

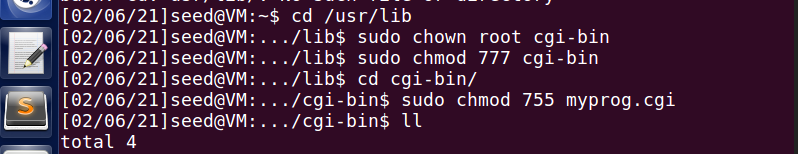
* For this task, we copy the code given in the pdf and store it in the below directory.



* The program is named **myprog.cgi** and its contents are shown as follows:

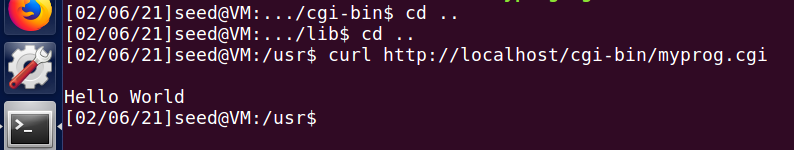


* Now, we give root permissions to both the folder and the cgi program, as shown below. It can be noted that the permission is set to 755, so that the program is executable.



* Next, we navigate to the usr directory and execute the following command as shown in the image below:

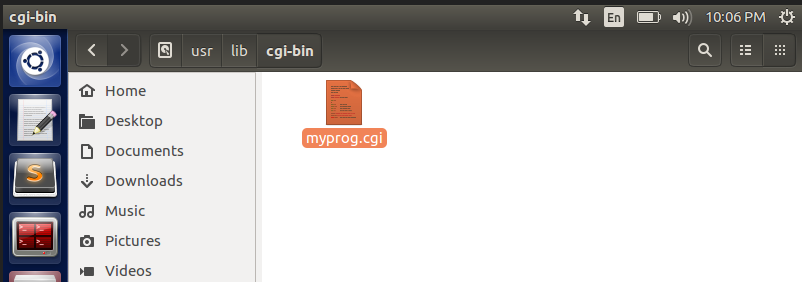
**curl http://localhost/cgi-bin/myprog.cgi**



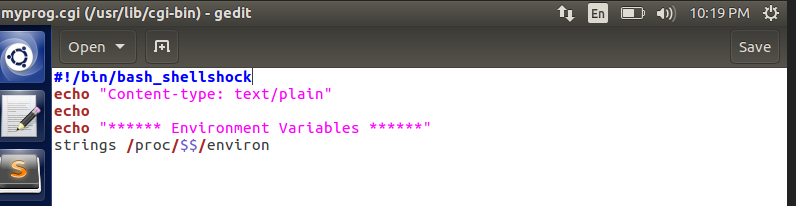
* We can see that the contents of the file “Hello world” has been displayed in the command line. This shows the **curl command is used to pull the data from the http url specified by the user**, in our case, we use the curl command to pull the cintents of the file myprog.cgi and display it.

**Task 3 – Passing Data to Bash via Environment Variable:**

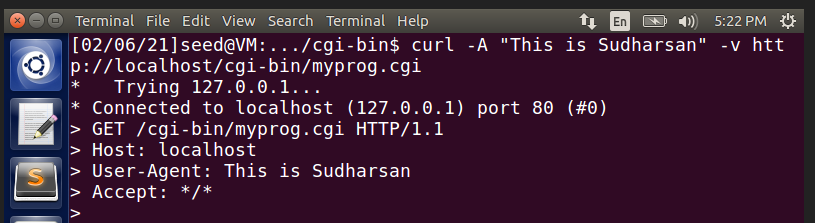
* For this task, we copy the code given in the pdf and store it in the below directory.



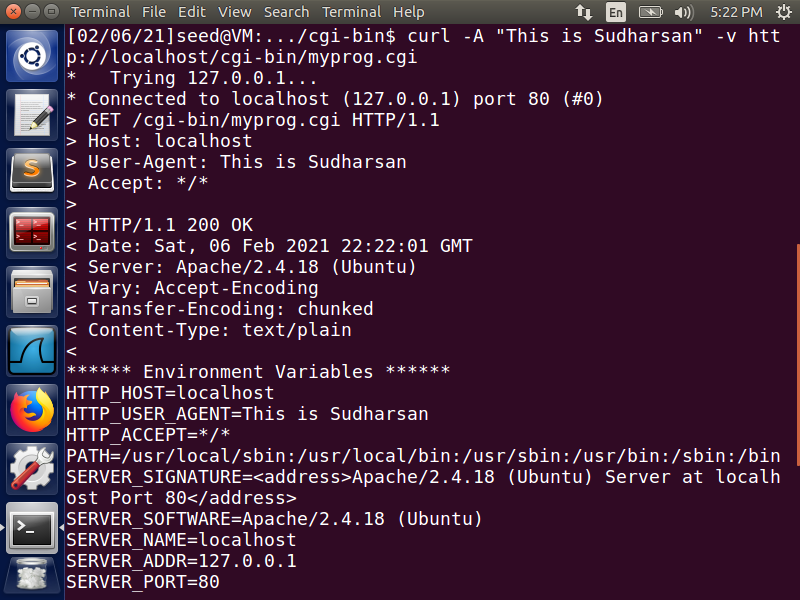
* The program is named **myprog.cgi** and its contents are shown as follows:

****

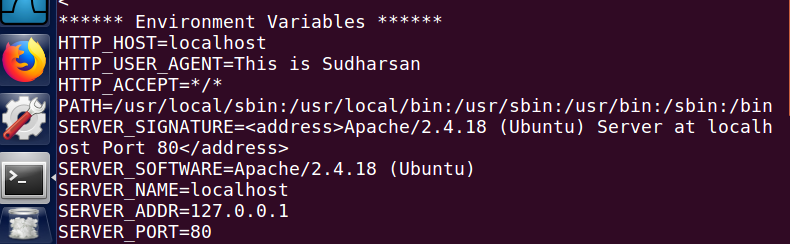
* In this task, we try to pass data to bash using command line through **curl** command. Curl command is used to transfer data to (or) from the server. In this case, we transfer data using it, by using **curl -A** command as shown below, where -A represents ‘USER AGENT’ field.

****

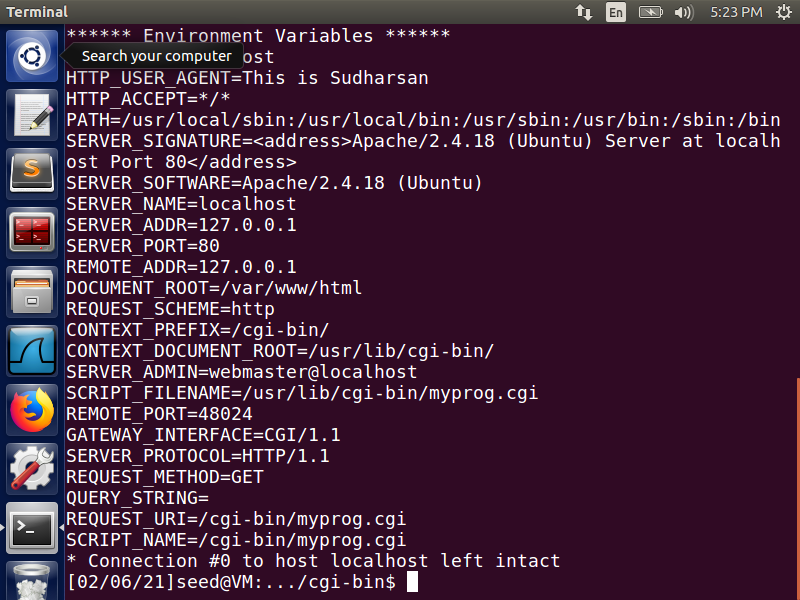
* As seen above, the string “This is Sudharsan” is being passed through the server, onto the environment variable HTTP\_USER\_AGENT field.

****

* The field HTTP\_USER\_AGENT is populated with the data that we sent as seen below.

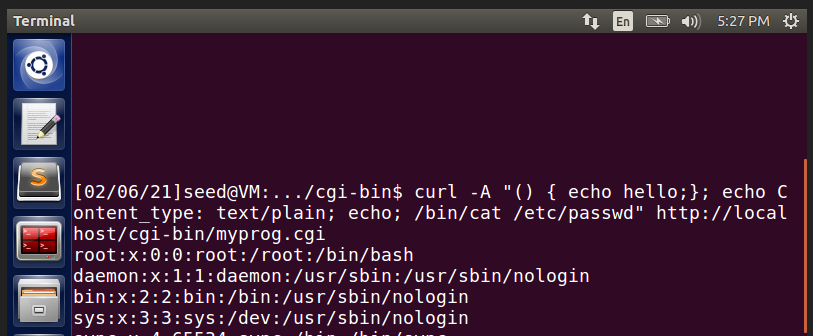
****

* Thus, we are successful in passing data to Bash through the environment variables, using curl command through the http server.

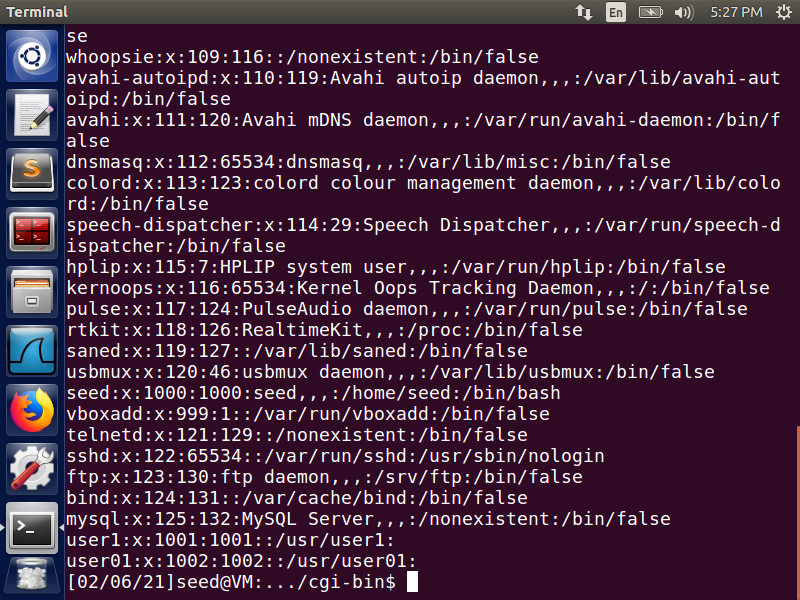
****

**Task 4 – Launching the Shellshock Attack:**

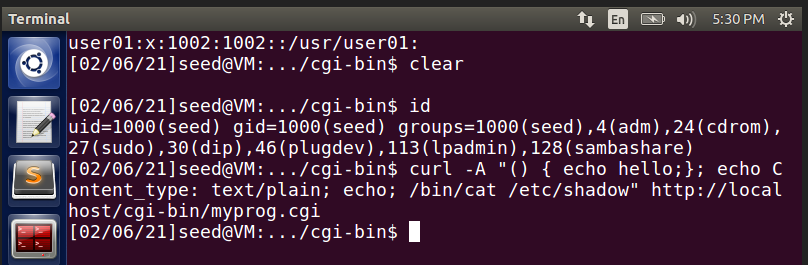
* For this task, we try to launch shellshock attack to try to steal the contents of password file that is present in the server.



* An environment variable is passed using **curl -A** command and we try to use the vulnerability of **bin\_shellshock** present inside the program myprog.cgi to get access to the contents of the passwd file.
* The command above shows that we try to access **/bin/cat /etc/passwd** file on the server, which normally should not be accessible.



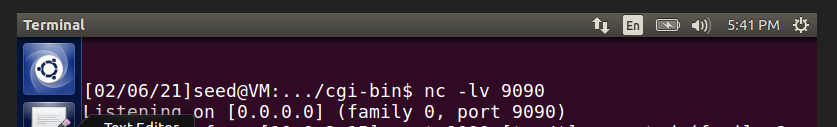
* The contents of the passwd file are displayed by fetching it from the server, thus proving that the program is vulnerable, and it can be exploited by attackers to fetch contents from any file in the server using the **curl** command.
* Now, we then further try to access the shadow file present in the server, **/bin/cat /etc/shadow**



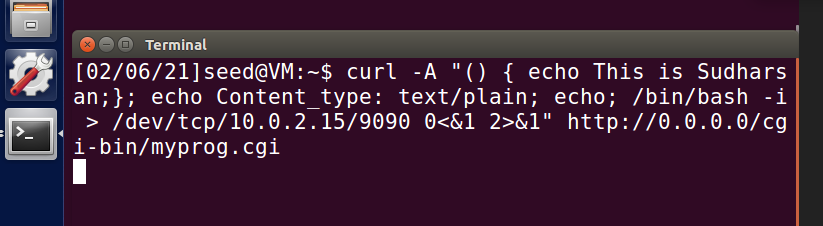
* As we see in the above image, the command to try to access shadow file is given. But on executing the command, we can see that nothing has been displayed. This proves that the file is not accessible and not vulnerable to the attack. Since it is a root file, a normal user cannot access the file and therefore no output is displayed.

**Task 5 – Getting a Reverse Shell via a Shellshock Attack:**

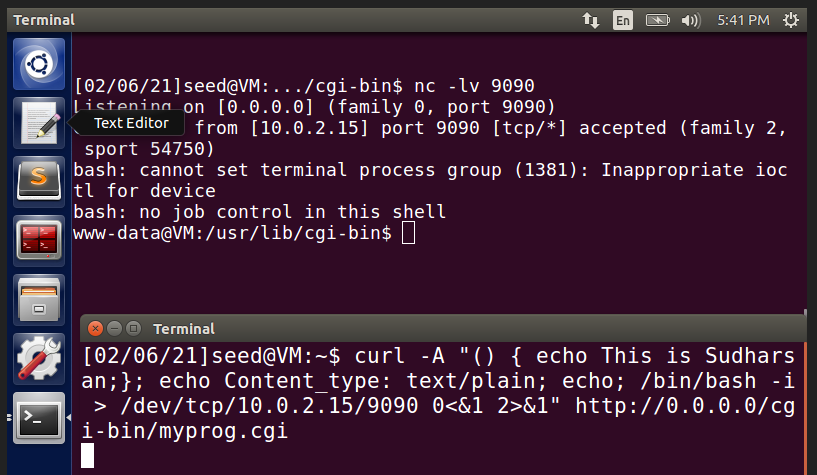
* Reverse Shell is a shell process started on a machine, with its input and output being controlled by someone from a remote computer.
* For this Task, we open up two different terminals, considering one as the victim terminal and the other as the attacker terminal.
* On the victim’s terminal, we initiate a connection using the **nc** command. This nc (netcat) command is used to read/write data across network connections using HTTP/TCP/UDP protocols, with port 9090 and shell identity is 0.0.0.0



* From the attacker’s terminal, we try to pass a value to one of the fields of the environment variables, in this case, we pass value to HTTP\_USER\_AGENT field.



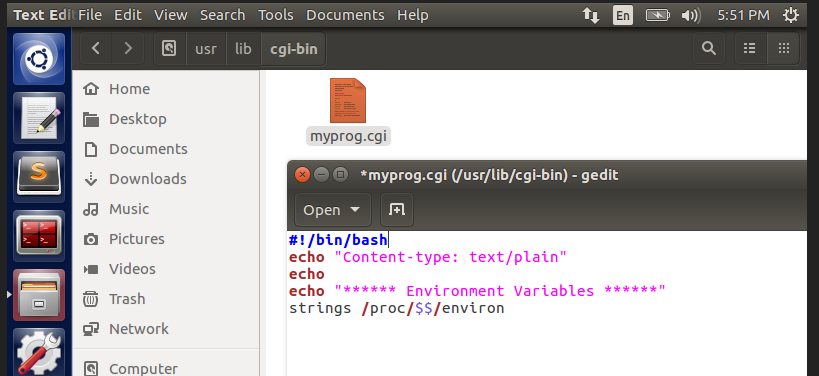
* The above command denotes that an interactive bin/bash shell is created, dev/tcp/10.0.2.15/9090 denotes that the tcp connection is referring to the victim’s port 9090 and we try to access the myprog.cgi program on the server through the shell identity 0.0.0.0 since that is the shell identity of the victim.



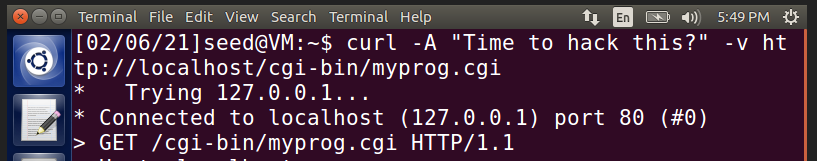
* As seen from the above result, a connection has been established on the victim’s terminal from the attacker’s side. Since we passed “**() {**” from the attacker’s terminal, the bash converts the variable -A (HTTP\_USER\_AGENT) field into a function and writes into the CGI program through bash. Thus, showing that we are able to establish a connection to another remote terminal through bash thereby achieving the reverse shell attack.

**Task 6 – Using the Patched Bash:**

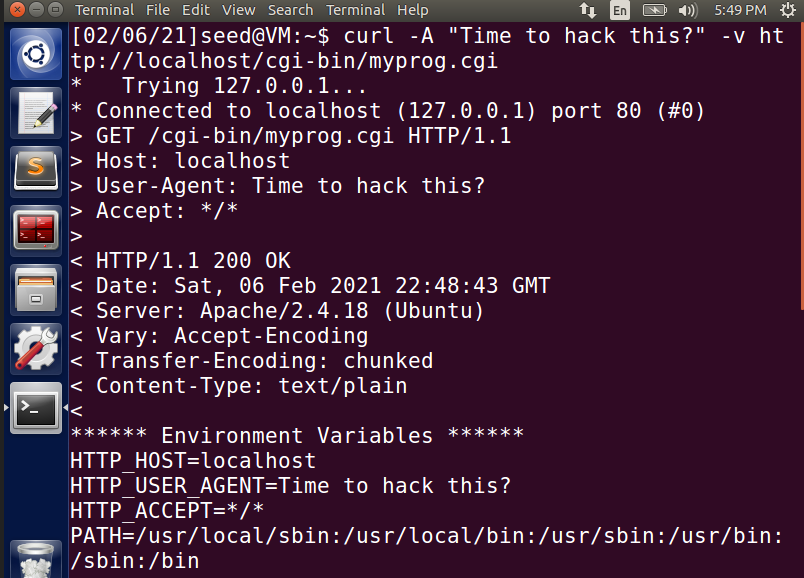
* For this task, we modify the CGI program and run it using /**bin/bash**. The CGI program is modified accordingly as seen below:



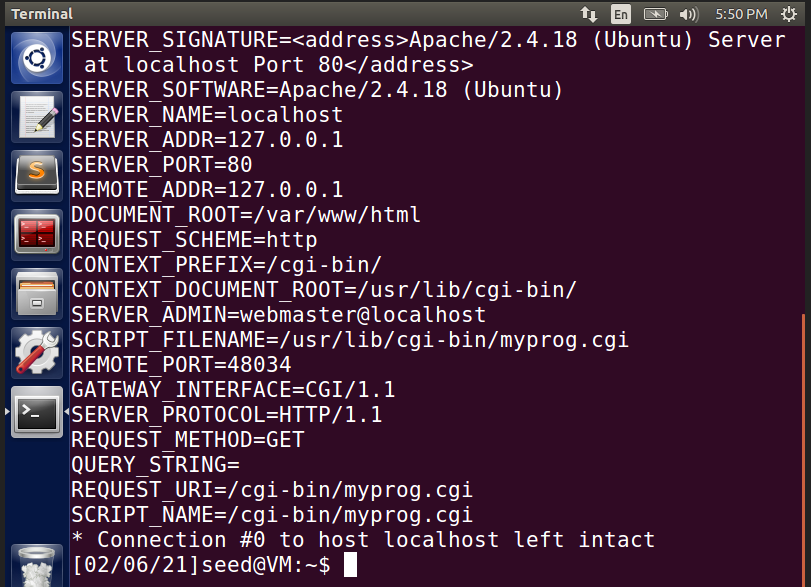
* Now, we try to perform Task 3 and observe the outcome to see how is **/bin/bash** different from **/bin/bash\_shellshock** which we did earlier.
* We try to pass value to the environment variable field **HTTP\_USER\_AGENT** using curl -A command onto the CGI program present in the server.



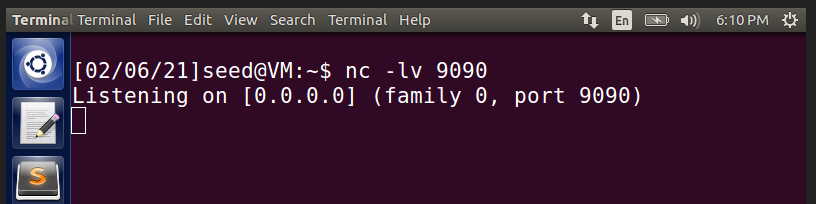
* As we can see from the below image, the value has been passed ono the environment variable **HTTP\_USER\_AGENT**



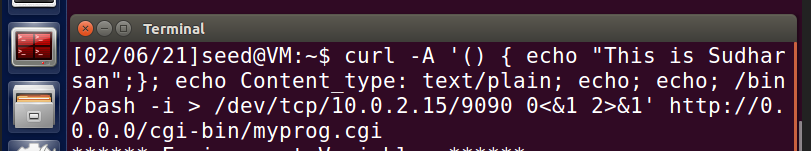
* From the attacker’s terminal, we try to pass a value to one of the fields of the environment variables, in this case, we pass value to HTTP\_USER\_AGENT field.



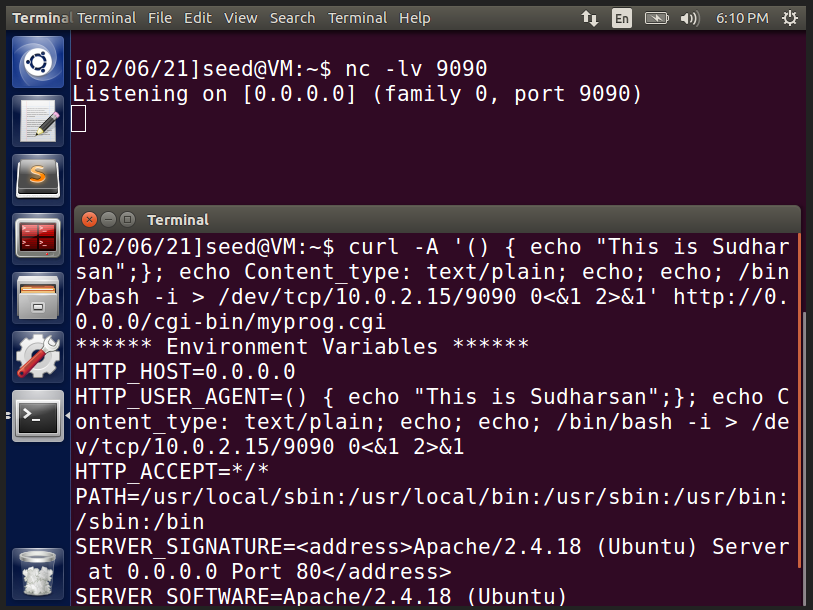
* As seen from the above image, if we try to pass data to environment variables using Bash, we can see that we are able to send data even in case of patched /bin/bash
* Now, we try to perform Task 5 and observe the outcome to see how is **/bin/bash** different from **/bin/bash\_shellshock** which we did earlier, using the modified CGI program.
* We open up two different terminals, considering one as the victim terminal and the other as the attacker terminal.



* On the victim’s terminal, we initiate a connection using the **nc** command. This nc (netcat) command is used to read/write data across network connections using HTTP/TCP/UDP protocols, with port 9090 and shell identity is 0.0.0.0



* The above command denotes that an interactive bin/bash shell is created, dev/tcp/10.0.2.15/9090 denotes that the tcp connection is referring to the victim’s port 9090 and we try to access the myprog.cgi program on the server through the shell identity 0.0.0.0 since that is the shell identity of the victim.



* Here, we can see that there is no connection established on the victim’s terminal, clearly showing that it is not vulnerable to the shellshock attack.
* This shows that even if we try to pass value to environment variables through the server, it does not get attacked because of the modified /bin/bash program, which prevents reverse shell attack from happening since there is no shellshock to exploit and launch an attack.