**Secure Software Development WS 2015/2016**

Team Member : Jason Christian – Xiaozhou Ji

Exercise Topic : Lab 2 – Shellshock Attack

Group Nr. : 18

# Shellshock with CGI Program

## Apache Setup

1. Document root of the Apache server is located in **var/www**

You can change the document root of the server from the entry in **etc/apache2/sites-available/default**

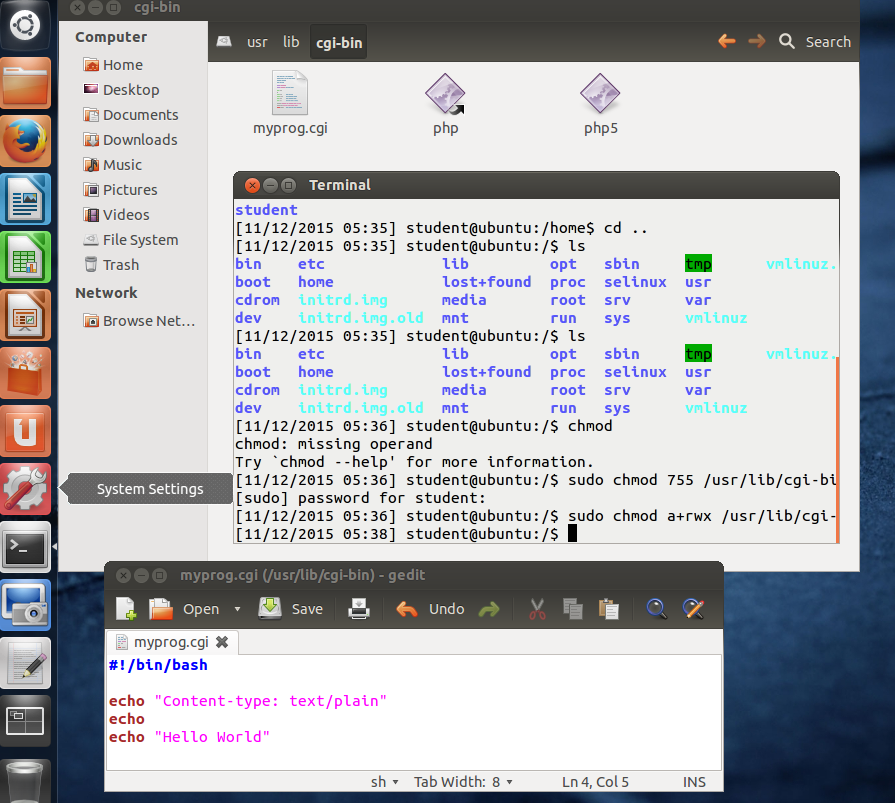
To access the index.html page on the document root, you should type **Localhost** or **127.0.0.1**

1. The default directory for CGI scripts in Apache2 server is in **/usr/lib/cgi-bin**

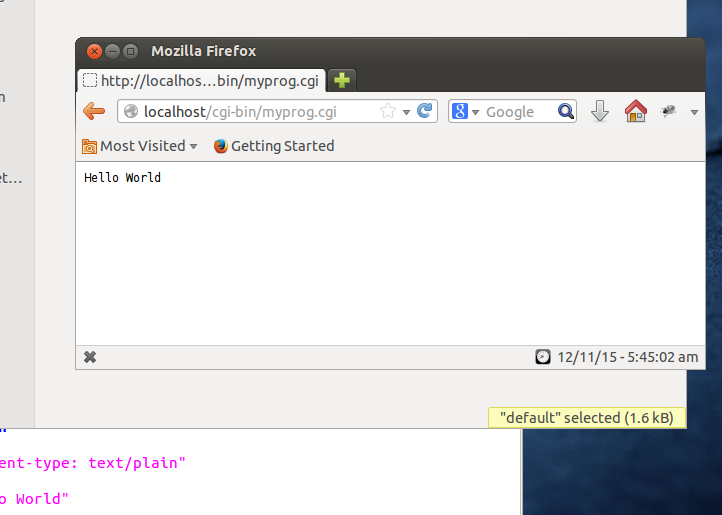
You can change the CGI directory of the server by **etc/apache2/sites-available/default**

## Set up CGI Program

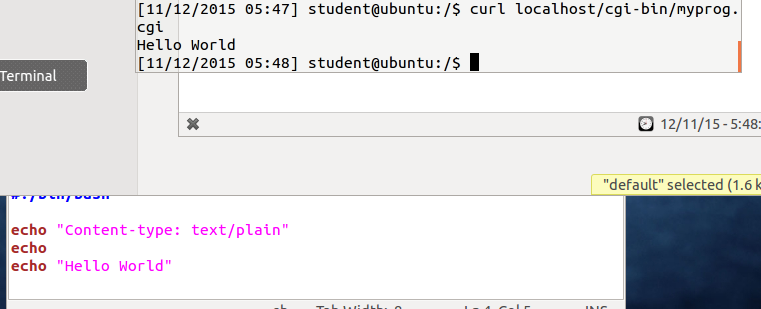
1. CGI is a protocol of a web server that enables the request from the client to be processed by an application program (typically written in Perl or C++) in the server and the result is sent back to the client. It’s typically used for processing or doing calculation from the user’s request, such as filling out / validating a form, dynamic web document generation, monitoring web traffic, and other things.
2. **Myprog.cgi** in cgi-bin folder



You can access the script from the web browser by typing the URL **localhost/cgi-bin/myprog.cgi**

****

Or from *curl* by executing the command **curl localhost/cgi-bin/myprog.cgi** from your terminal

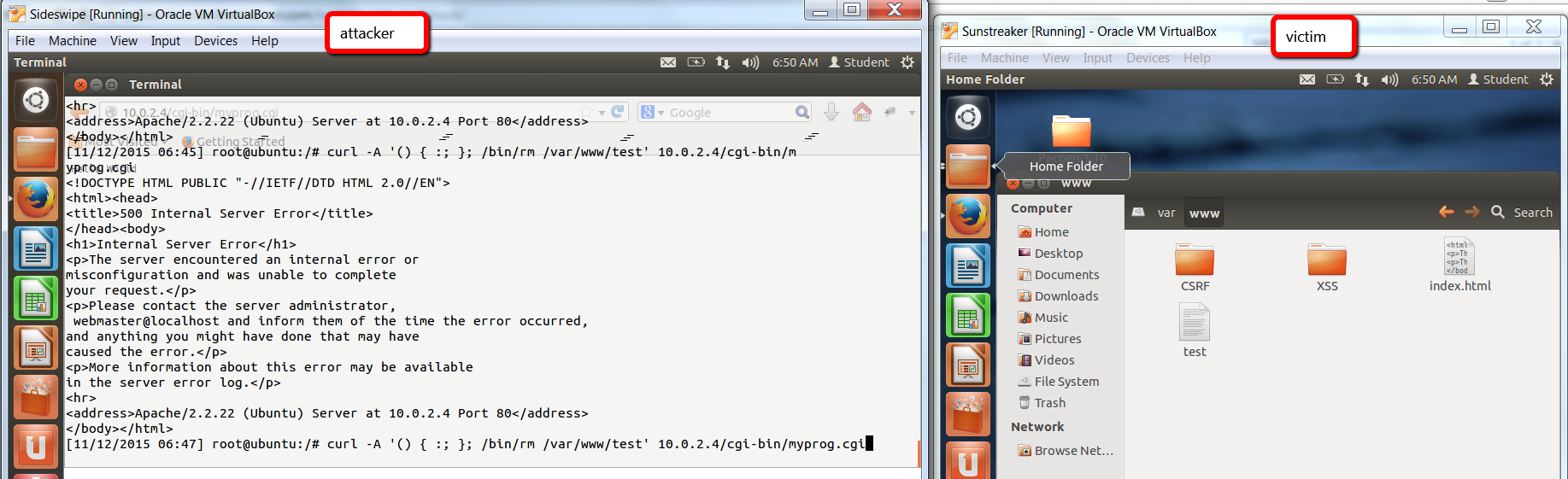


## Attack

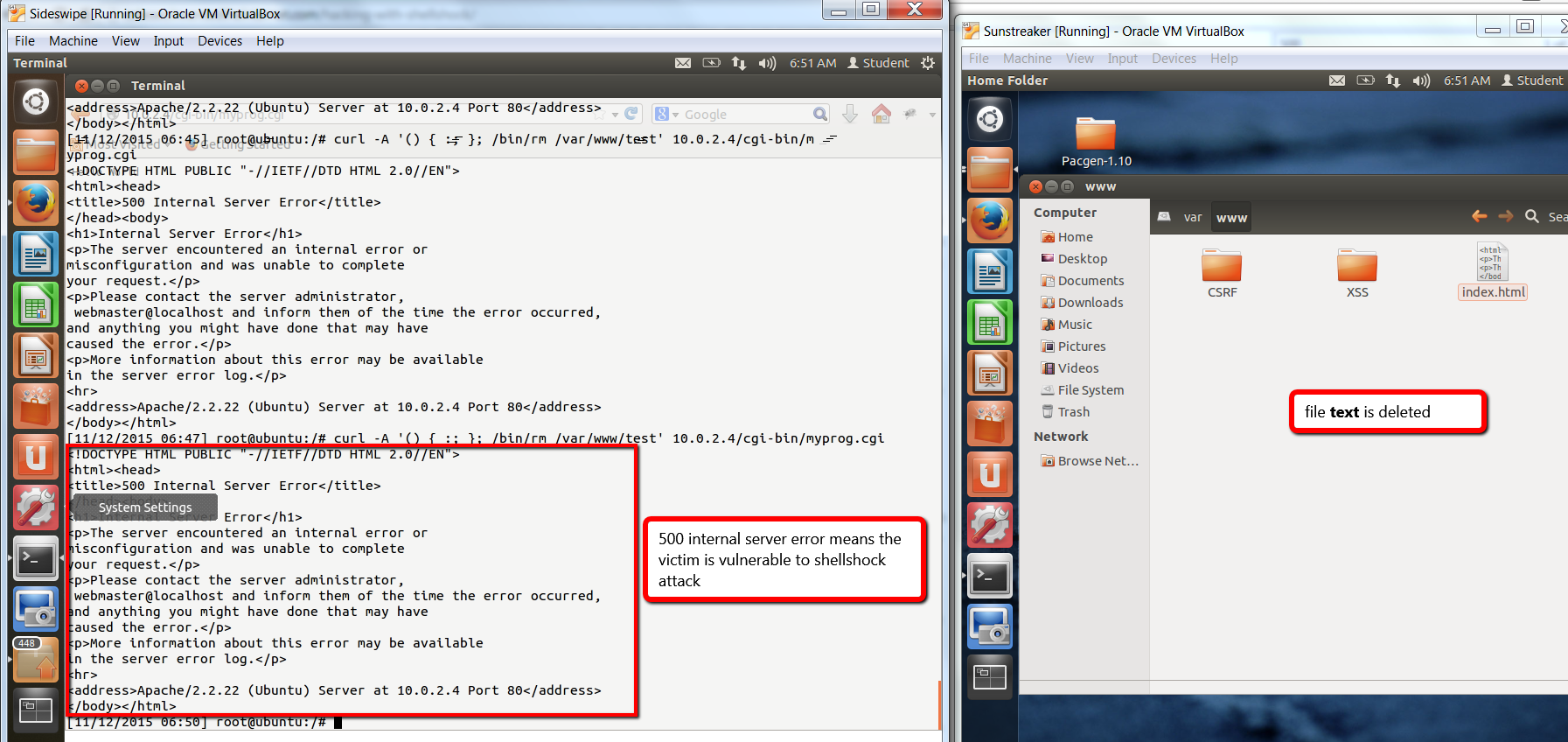
1. The Shellshock attack works by inserting a magic string **() { :; };** in the HTTP user-agent header along with a bash command after that. You can use *curl* or *env* to launch the attack. The vulnerability is that the bash program will process the bash command after the magic string.

Example on removing a file using curl:

Curl –A ‘() { :; }; /bin/rm /var/www/test’ 10.0.2.4/cgi-bin/myprog.cgi

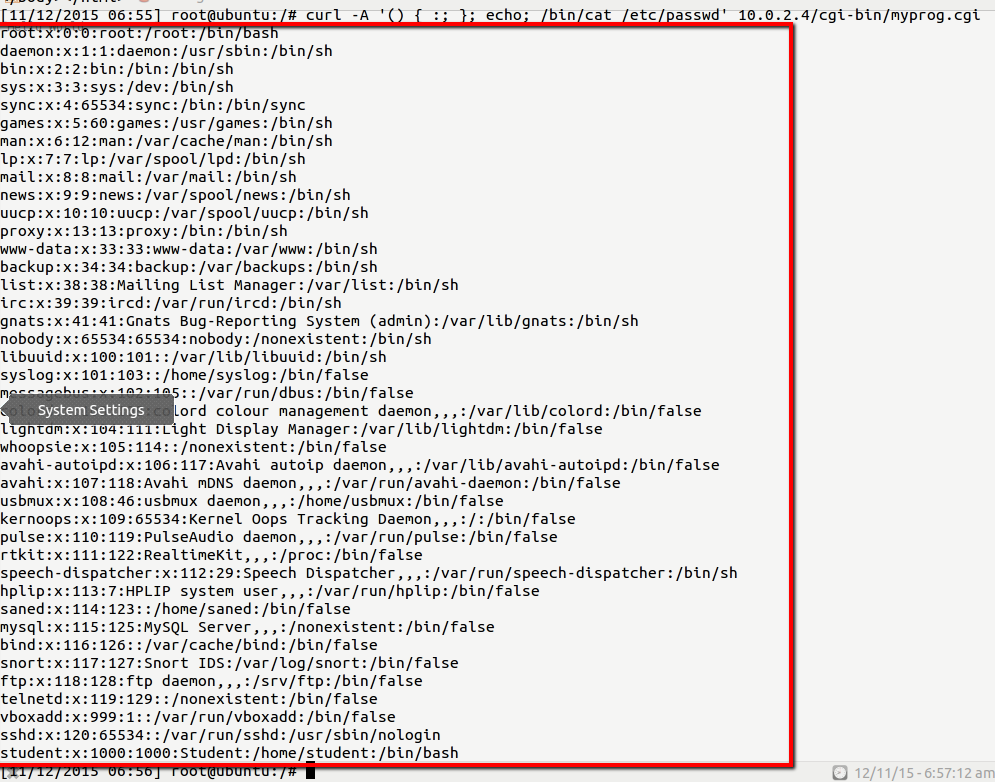


After execution:

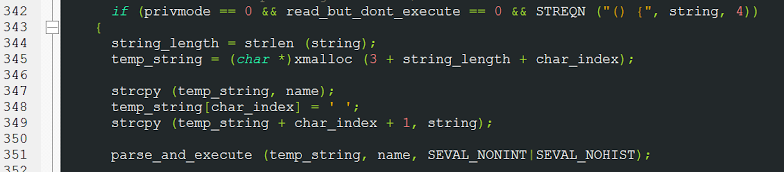


Example on retrieving **etc/passwd** file from the victim’s machine:

curl –A ‘() { :; }; echo; /bin/cat /etc/passwd’ 10.0.2.4/cgi-bin/myprog.cgi



1. The vulnerability of the code lies in the *if* statement on line 342-352. The *If* command executes anything that passes the string () { as it’s a regular bash command (the parse\_and\_execute function with parameter SEVAL\_NONINT | SEVAL\_NOHIST).



# Shellshock with Set-UID programs

1. Ln –sf /bin/bash /bin/sh will create a soft link to the file **/bin/bash** through **/bin/sh**. That means we could access bash from the generated sh file.
2. SetUID program is a program that can be granted more access than what they are already assigned for.

They are used when a program needs to access a certain resource which requires more privileges than normal programs (e.g. password checking, web server for accessing server-side resource/root, etc.). In C, a program can be set into SetUID program by using the function seteuid

1. **System()** spawns a shell and execute the command as a separate process in it

Example:

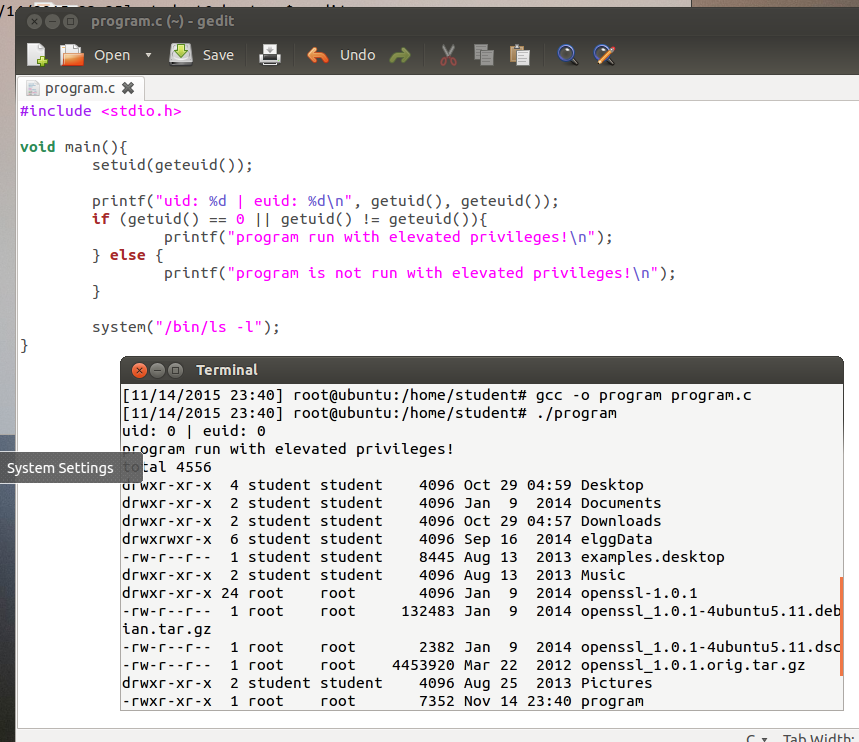
System(“cat file.txt”) equals to /bin/sh –c cat file.txt

**Execve()** will remove your current program with the command executed by execve()

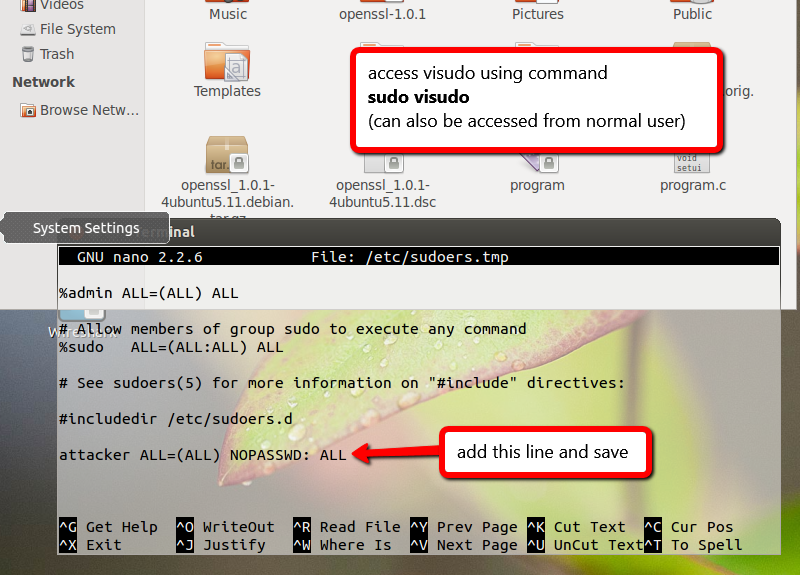
Execve is similar to closing the first program and starting the new one based on the parameter

1. Because the program uses system(), the first process is replaced by the ls program inside system function.

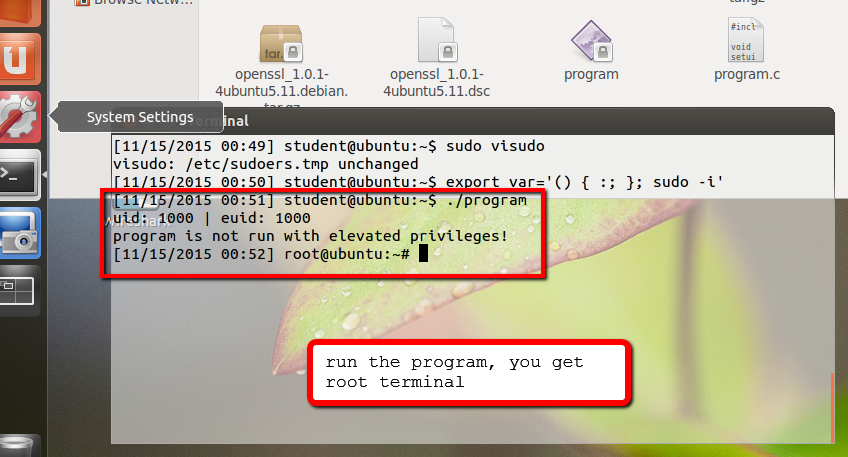
The program runs with root privileges.



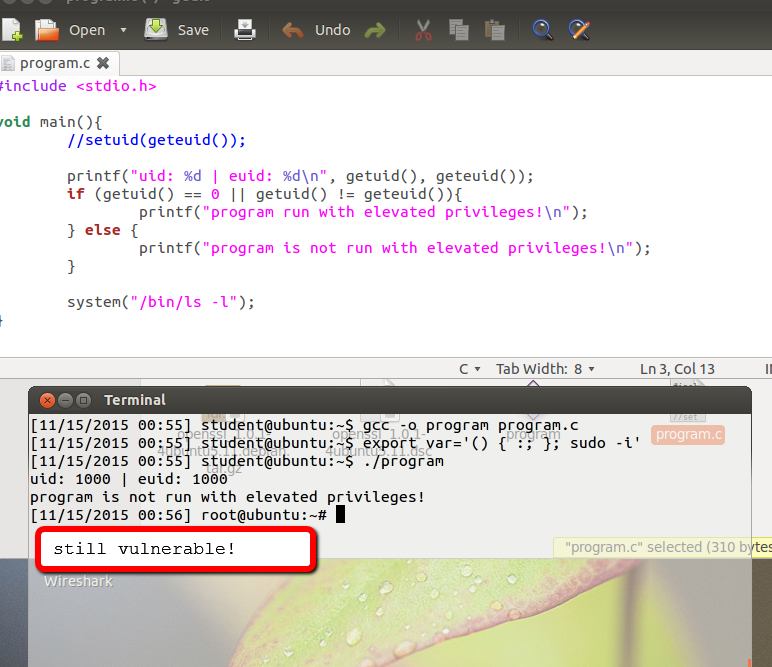
1. We can gain root terminal by exporting an environment variable which utilizes the magic string () { :; }; that exploits shellshock vulnerability. Before that, we should modify the sudoers file (**etc/sudoers.tmp**) to enable root access without password. After exporting the environment, we can casually executes the program and it will show you the root terminal



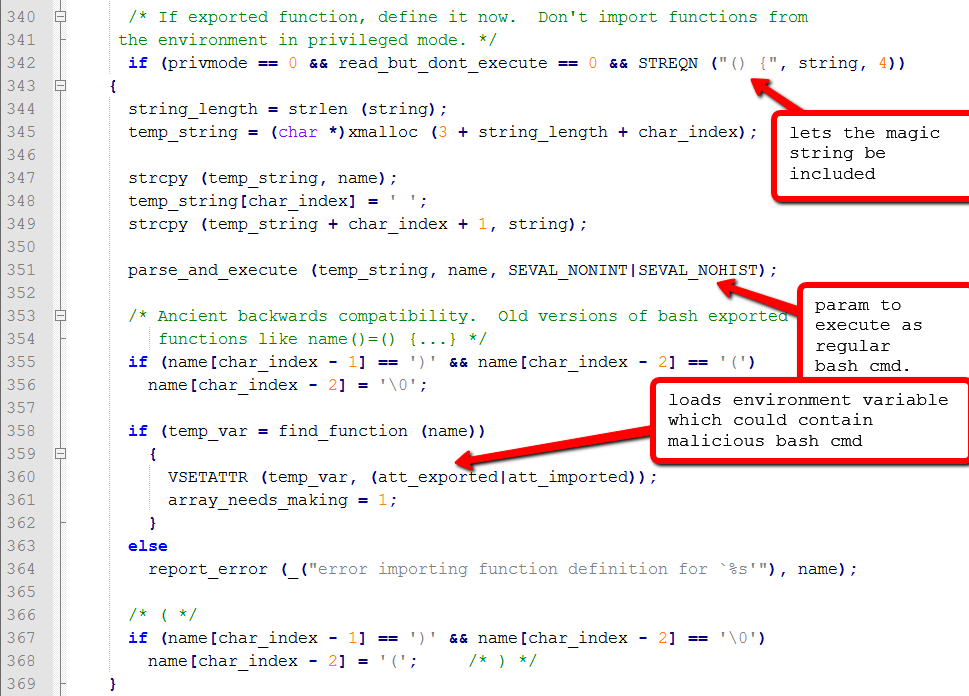




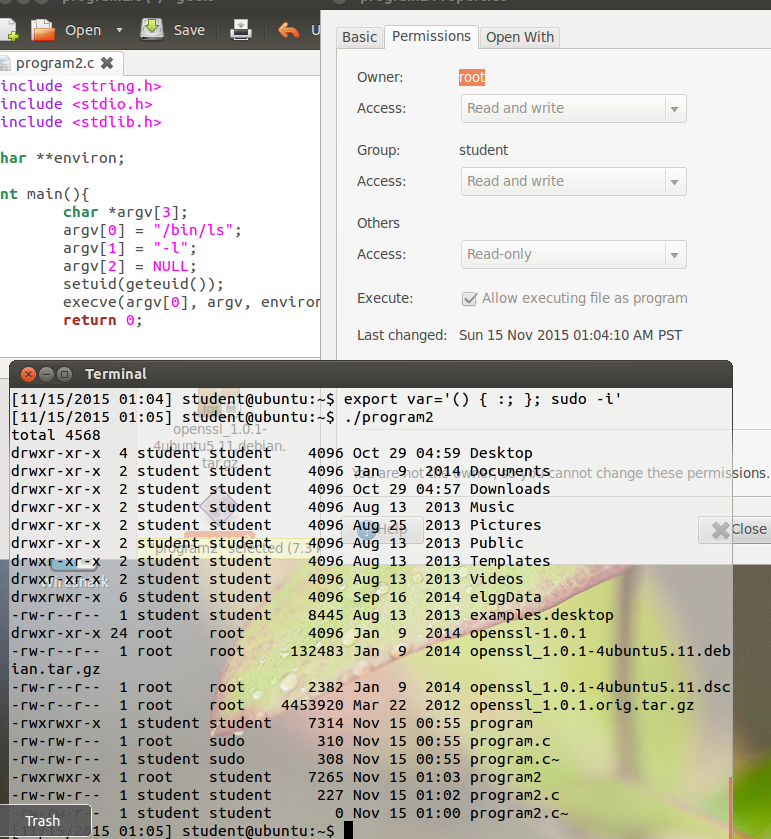
1. It can still return a root terminal



This is caused by the vulnerability caused by variables.c line 342-369 that lets bash to load environment variables and executes it as a regular bash command (the environment variable loading is between line 358-362)



1. The 2nd code executed does not return a root terminal



This is due to **execve** function removes the first program, which is loaded with malicious environment variable, and spawn a new program (in this case, the ls command). Because of that, the malicious environment variable does not get executed.

# Conclusion

1. Shellshock attack is a bug in linux system that utilizes the vulnerability of the bash command processing that allows attacker to run his bash own bash command in the victim’s machine

This attack is dangerous because the attacker can access victim’s machine’s confidential data or do anything that requires root privileges without even entering the password for root (e.g. accessing /etc/passwd, removing programs in bin folder, auto-download malware, etc.)

1. The problem lies in the vulnerability of the code shown in **variables.c** file which allows a string of () { :; }; to be executed as a regular bash command.

Programmer should wary about:

* System updates. Make sure to update the system to the latest patch with eliminates the shellshock vulnerability
* Programs that can access elevated privileges.
* SSH from unknown machine

1. Other ways:

* Using SSH to gain access to the victim’s machine, and elevate privileges using either Set-UID programs or CGI programs
* Using HTTP request with the magic string () { :; }; and bash command
* Using trojan that executes malicious bash command