Title: OS Security Shellshock

Name: Nate Nuval

Class: CPSC 405 Operating Systems

Instructor: Andrew Marshall

Date: 11/22/2016

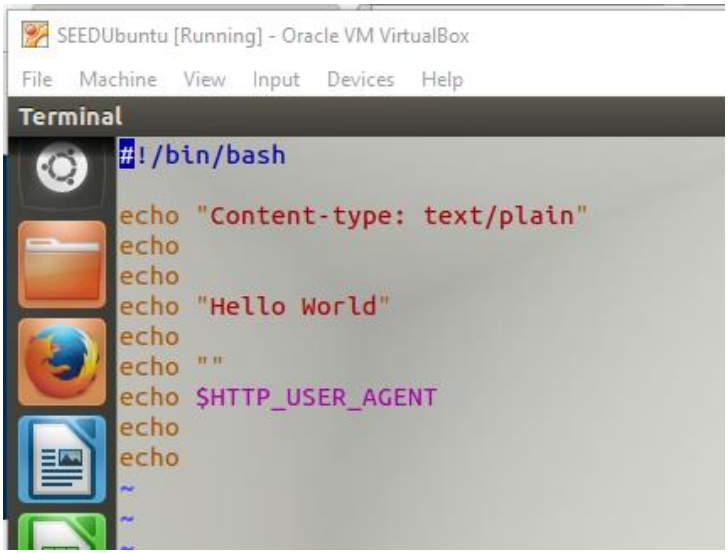
Introduction: The purpose of this lab was to explore the bash vulnerability nicknamed Shellshock. Shellshock was identified on September 24, 2014 and had gone unnoticed for years.

Methodology:

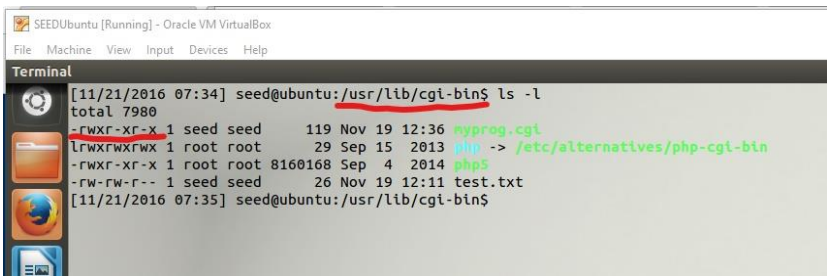
1) Attack CGI Programs

In this task, I launched a Shellshock attack on a web server using the VM, the command line, curl, and a CGI program. CGI is a standard method used by web servers to generate dynamic content.

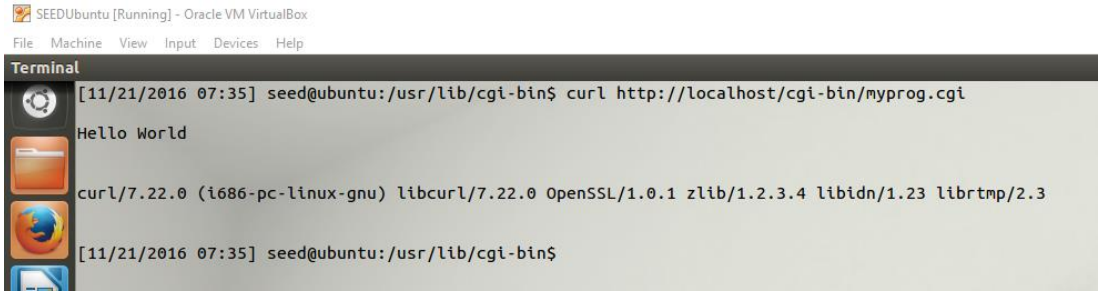
Using vim, I wrote a simple CGI program, *myprog.cgi*, based on the lab write up.



This program was then placed in the /usr/lib/cgi-bin directory. I used the command: sudo *chmod 755 myprog.cgi* to change the permissions of the file to allow it to be executed.



myprog.cgi can now be accessed through the web.

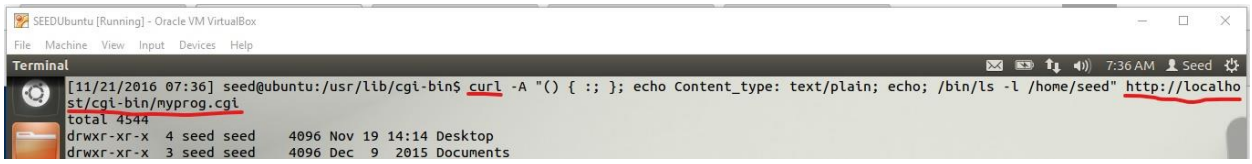


Before the CGI script is executed, Bash is invoked. This is where the vulnerability lies. Bash allows you to define functions, which is stored in Bash's environment variables.

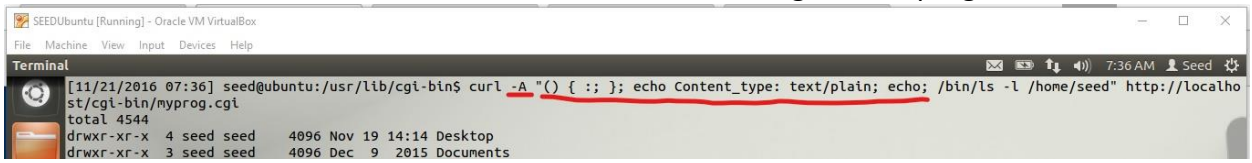
> export foo='() { echo Hello; };' *Here we have a function foo that prints out "Hello"*

> export foo='() { echo Hello; }; echo World' *Here we have additional code after the function definition. When Bash boots up it will automatically execute the additional code.*

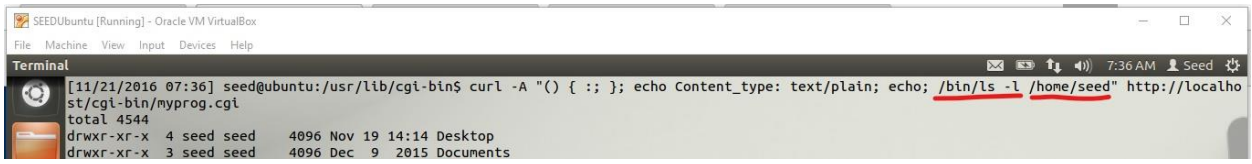
Underlined in red is the curl command calling accessing the CGI script.



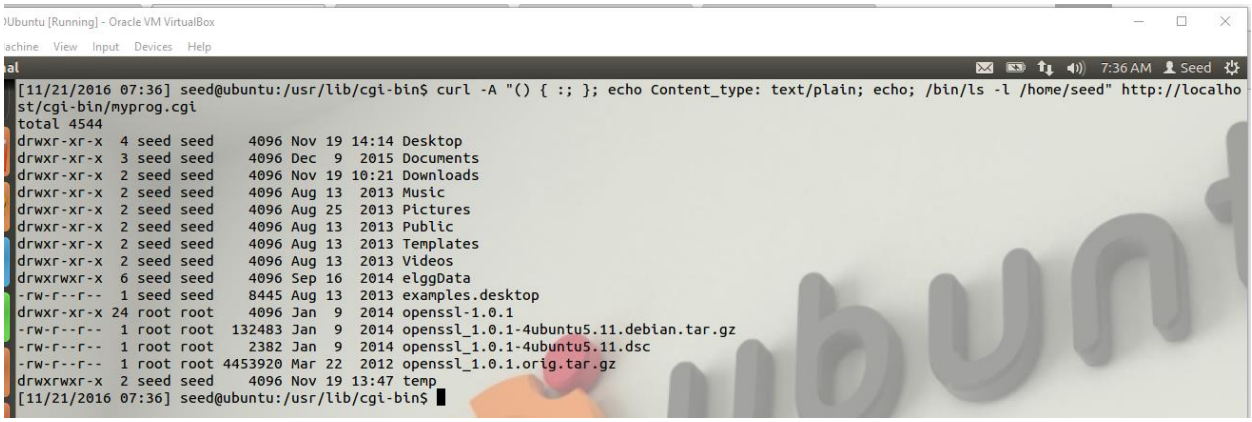
The –A flag passes the User-Agent. This is immediately followed with a function declaration as well as two other "echo" commands mimicking the CGI program.



Here is where I injected the commands that access the server.



When this curl command is executed instead of printing the cgi program, it is hijacked by the ls command to print out the contents of the /home/seed directory.



A person with malicious intent can use this bash bug to execute server compromising commands. With the correct curl command, a person add files, delete files, steal passwords, they can even commandeer control with a reverse shell.

2) Vulnerability in variables.c

The vulnerability in the variables.c file takes place within the initialization of shell variables.

variables.c

```
304  /* Initialize the shell variables from the current environment.
305     If PRIVMODE is nonzero, don't import functions from ENV or
306     parse $SHELLOPTS. */
307  void
308  initialize_shell_variables (env, privmode)
309       char **env;
310       int privmode;
311  {
```

Underlined in red is where the vulnerability is located. The if statement checks if the privilege is equal to zero and if the pattern "() {" is present.

Ex: foo = () { echo Hello; };

variables.c

```
339
340     /* If exported function, define it now.  Don't import functions from
341    the environment in privileged mode. */
342     if (privmode == 0 && read_but_dont_execute == 0 && STREQN ("() {", string, 4))
343   {
344     string_length = strlen (string);
345     temp_string = (char *)xmalloc (3 + string_length + char_index);
346
347     strcpy (temp_string, name);
348     temp_string[char_index] = ' ';
349     strcpy (temp_string + char_index + 1, string);
350
351     parse_and_execute (temp_string, name, SEVAL_NONINT|SEVAL_NOHIST);
352
```

It then passes the string to the parse_and_execute function without checking if there is anything following the "(){...}".

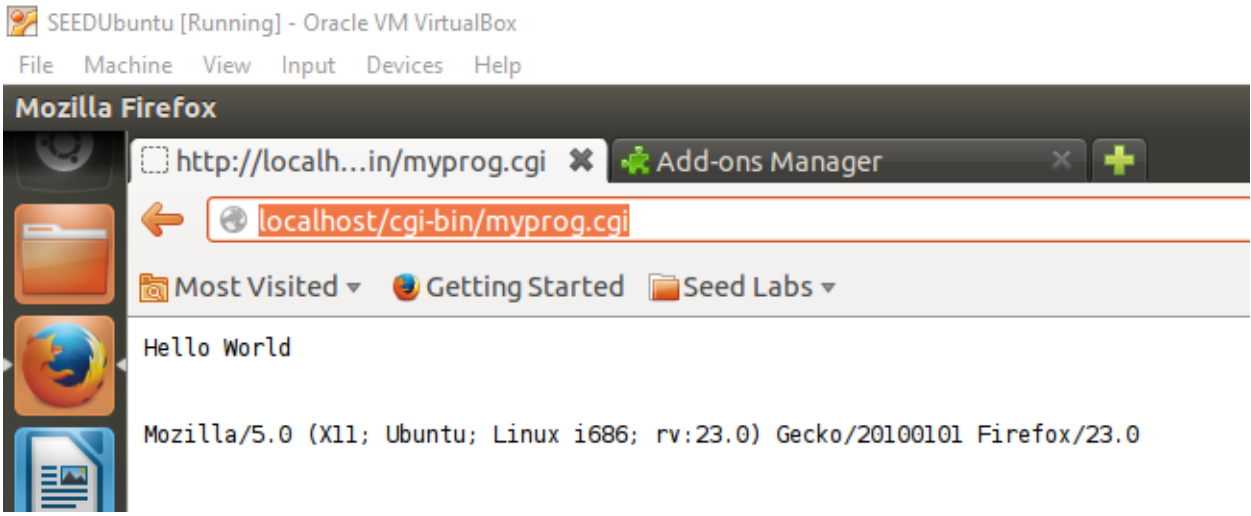
Ex: foo = () { echo Hello; }; echo "World";

It reads the code written after the function declaration as a shell script and executes it immediately.

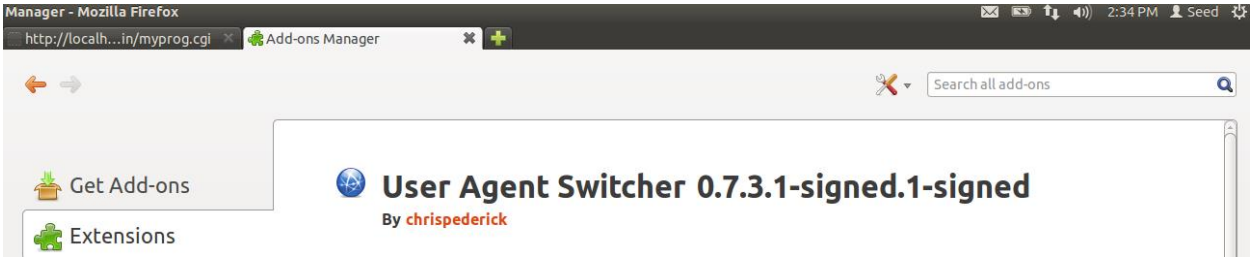
3) Attack through the Web

A Shellshock attack can be executed directly through a web browser.

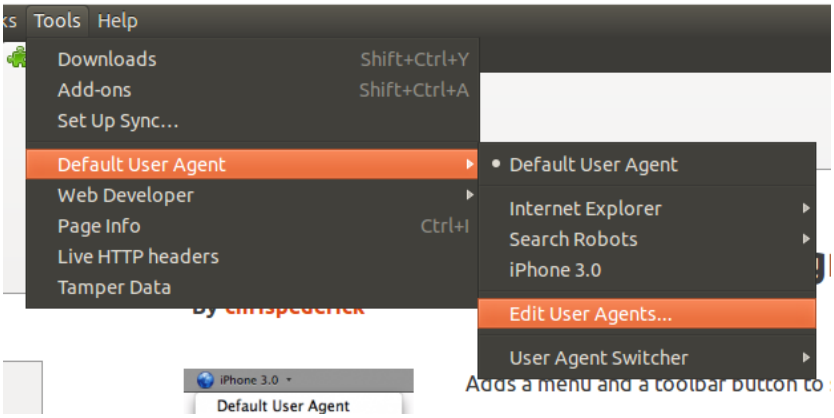
Entering the location of the *myprog.cgi* file in the firefox web browser gives me this:

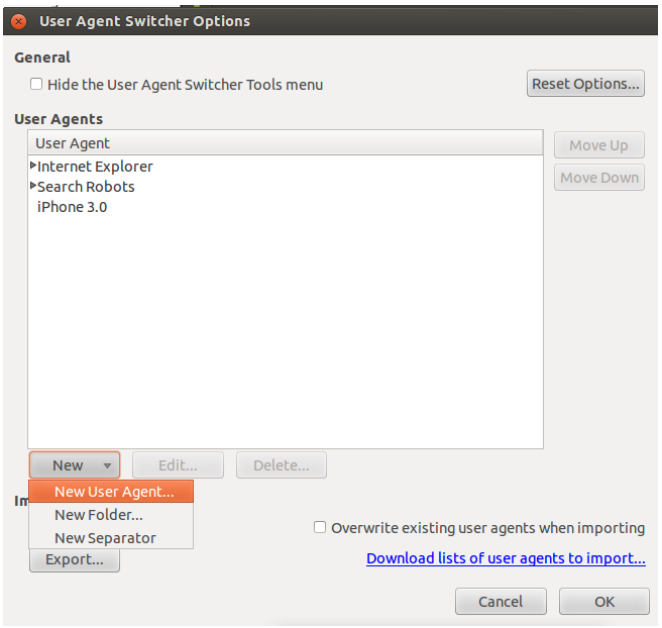


In order to launch the attack, I needed to change the user-agent from the web browser; the curl command used the –A flag. Firefox has add-ons that can complete this task.

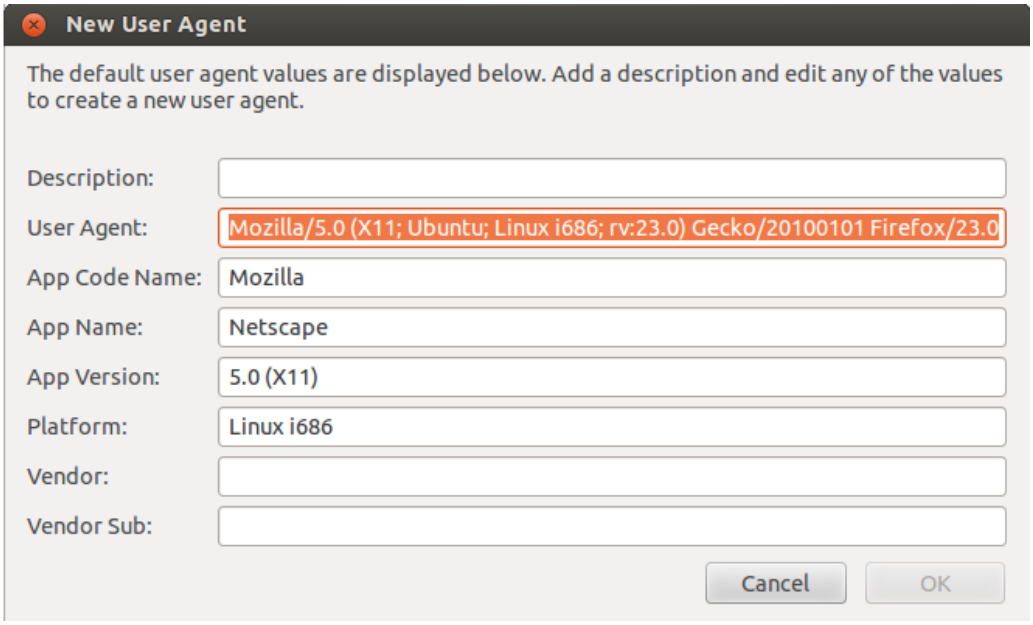


Now that there is a way to change the user-agent through the browser, the next step was to actually change the user-agent.

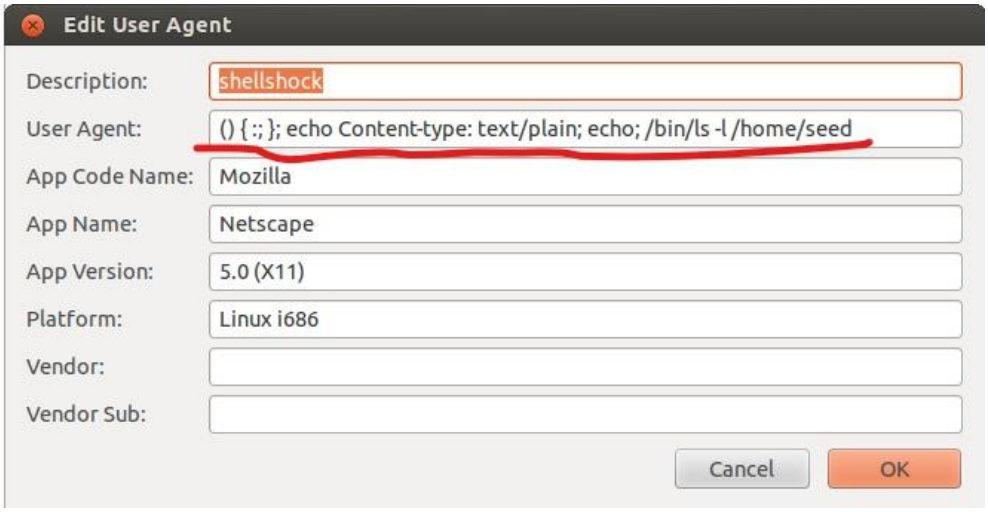




By replacing this:

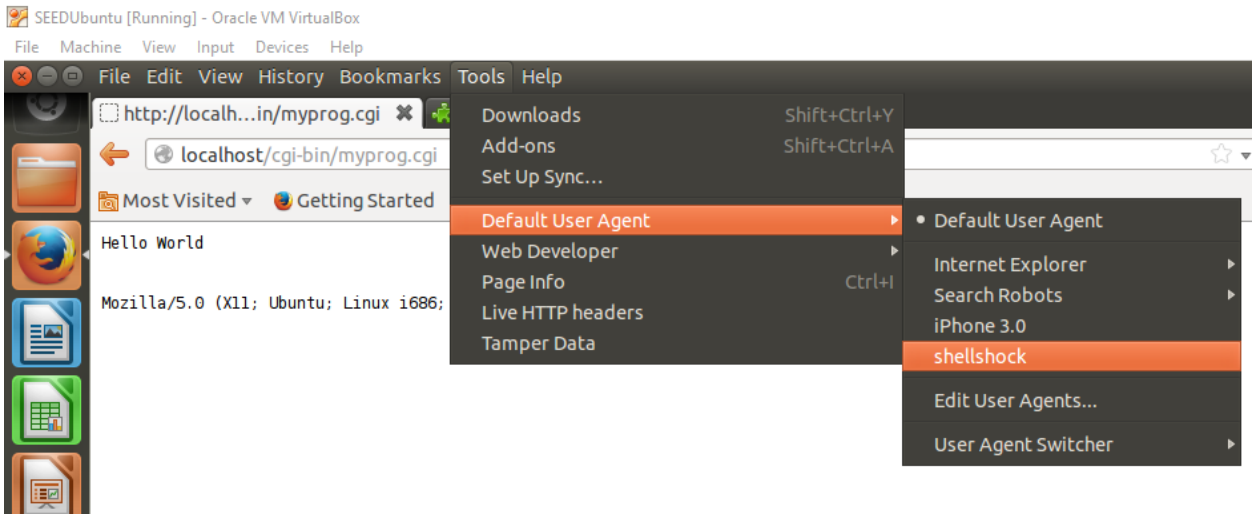


With everything that came after *curl –A* and before the address

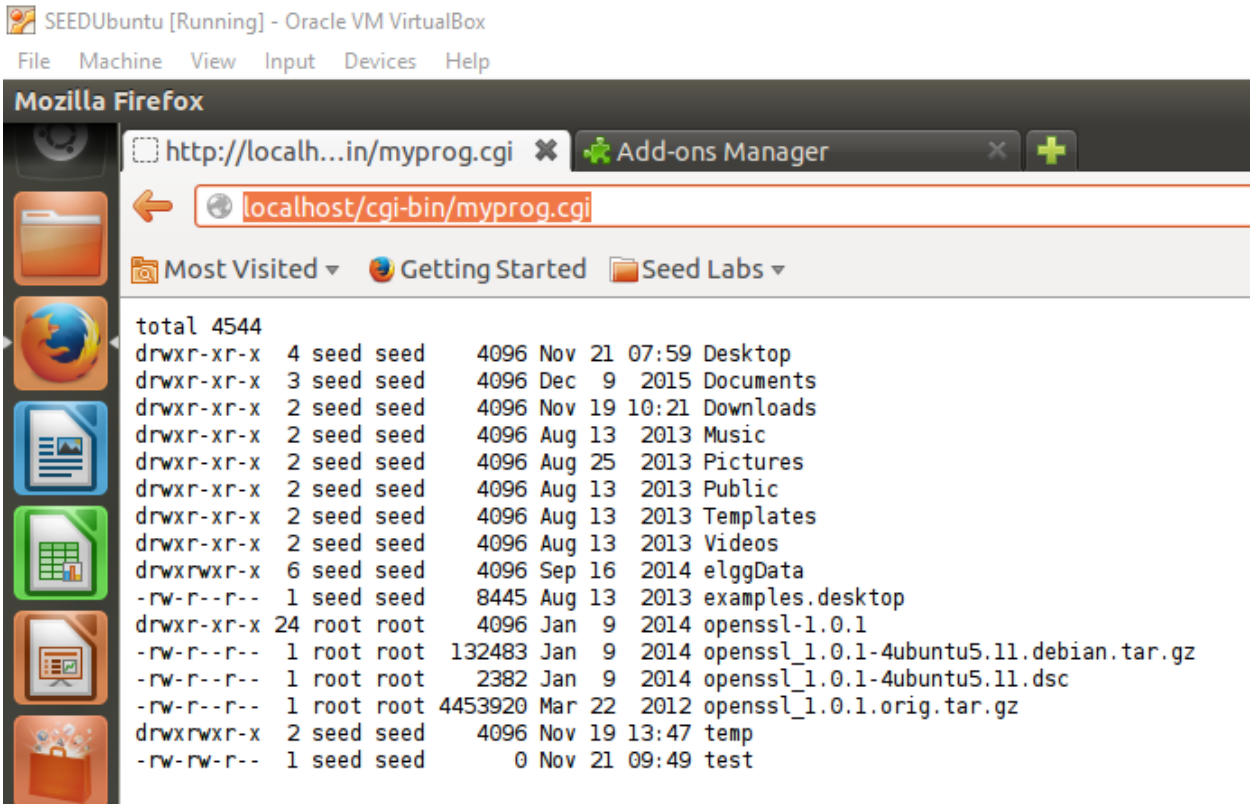


Will allow us to do the same thing as the CGI attack using curl. This will list out the contents of the /home/seed directory.

Now to execute the attack. In the browser switch the user-agent to the newly created one, I named my “shellshock”.



When you enter the address of a vulnerable server, the results are as follows:



Conclusion:

The fundamental problem of the Shellshock vulnerability is that it allows users to access a server and execute code from a remote location. Anyone with a web browser can execute this type of attack to a susceptible web server. With access to the server and certain scripts, a hacker can steal data, delete files, and even take over the shell.

This lab has shown me how dangerous a few lines of code can be. The hardest part was understanding the concepts. The actual implementation of the attack was relatively simple and quick. An operating systems programmer needs to think of all ways their code is exposed. Otherwise the smallest bug can become a huge vulnerability.