CSUS, College of Engineering and Computer Science
Department of Computer Science
CSC 154 – Computer System Attacks and Countermeasures

Lab (Extra Credit) – Shellshock Attack

Goal: To fully demonstrate the shellshock attack.

Instructions: Please refer to attached lab instructions with this document.

Deliverable: A lab report, an **electronic submission** to **Canvas**, is expected to **document** and **explain** all the **commands/code** that you use, and **include the screen shots** when you achieve the major milestones in the lab. A demo may be requested when necessary.

Requirement: The report will all be evaluated based on the following grading criteria.

Correctness	25%
Completeness	25%
Clarity	25%
Quality of English writing	25%

Shellshock Attack Lab

1 Overview

On September 24, 2014, a severe vulnerability in Bash was identified. Nicknamed Shellshock, this vulnerability can exploit many systems and be launched either remotely or from a local machine. In this lab, students need to work on this attack, so they can understand the Shellshock vulnerability. The learning objective of this lab is for students to get a first-hand experience on this interesting attack, understand how it works, and think about the lessons that we can get out of this attack.[1]

2 Lab Tasks

2.1 Task 1: Attack CGI programs

In this task, we will launch the Shellshock attack on a remote web server. Many web servers enable CGI, which is a standard method used to generate dynamic content on Web pages and Web applications. Many CGI programs are written using shell script. Therefore, before a CGI program is executed, the shell program will be invoked first, and such an invocation is triggered by a user from a remote computer.

Step 1: Set up the CGI Program. You can write a very simple CGI program (called `myprog.cgi`) like the following. It simply prints out `"Hello World"` using shell script.

```
#!/bin/bash

echo "Content-type: text/plain"
echo
echo
echo "Hello World"
```

Please place the above CGI program in the `/usr/lib/cgi-bin` directory and set its permission to 755 (so it is executable). You need to use the root privilege to do these (using `sudo`), as the folder is only writable by the root. This folder is the default CGI directory for the Apache web server. If you want to change this setting, you can modify `/etc/apache2/sites-available/default`, which is the Apache configuration file.

To access this CGI program from the Web, you can either use a browser by typing the following URL: `http://localhost/cgi-bin/myprog.cgi`, or use the following command line program `curl` to do the same thing:

[1]The first version of this lab was developed on September 29, 2014, just five days after the attack was reported. It was assigned to the students in our Computer Security class on September 30, 2014. This is to demonstrate how quickly we can turn a real attack into educational materials.

```
$ curl http://localhost/cgi-bin/myprog.cgi
```

In our setup, we run the Web server and the attack from the same computer, and that is why we use `localhost`. In real attacks, the server is running on a remote machine, and instead of using `localhost`, we use the hostname or the IP address of the server.

Step 2: Launch the Attack. After the above CGI program is set up, you can launch the Shellshock attack. The attack does not depend on what is in the CGI program, as it targets the Bash program, which is invoked first, before the CGI script is executed. Your goal is to launch the attack through the URL `http://localhost/cgi-bin/myprog.cgi`, such that you can achieve something that you cannot do as a remote user. For example, you can delete some file on the server, or fetch some file (that is not accessible to the attacker) from the server.

Please describe how your attack works. Please pinpoint from the Bash source code `variables.c` where the vulnerability is. You just need to identify the line in the `initialize_shell_variables()` function (between Lines 308 and 369).

2.2 Task 2: Attack `Set-UID` programs

In this task, we use Shellshock to attack `Set-UID` programs, with a goal to gain the root privilege. Before the attack, we need to first let `/bin/sh` to point to `/bin/bash` (by default, it points to `/bin/dash` in our SEED Ubuntu 12.04 VM). You can do it using the following command:

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

Task 2A. The following program is a `Set-UID` program, which simply runs the `"/bin/ls -l"` command. Please compile this code, make it a `Set-UID` program, and make `root` be its owner. As we know, the `system()` function will invoke `"/bin/sh -c"` to run the given command, which means `/bin/bash` will be invoked. Can you use the Shellshock vulnerability to gain the root privilege?

```
#include <stdio.h>

void main()
{
  setuid(geteuid()); // make real uid = effective uid.
  system("/bin/ls -l");
}
```

It should be noted that using `setuid(geteuid())` to turn the real uid into the effective uid is not a common practice in `Set-UID` programs, but it does happen.

Task 2B. Now, remove the `setuid(geteuid())` statement from the above program, and repeat your attack. Can you gain the root privilege? Please show us your experiment results.

In our experiment, when that line is removed, the attack fails (with that line, the attack is successful). In other words, if the real user id and the effective user id are the same, the function defined in the environment variable is evaluated, and thus the Shellshock vulnerability will be exploited. However, if the real user id

and the effective user id are not the same, the function defined in the environment variable is not evaluated at all. This is verified from the bash source code (`variables.c`, between Lines 308 to 369). You can get the source code from the lab web site. Please pinpoint exactly which line causes the difference, and explain why Bash does that.

Task 2C. Another way to invoke a program in C is to use `execve()`, instead of `system()`. The following program does exactly what the program in Task 2A does. Please compile the code, and make it a `Set-UID` program that is owned by `root`. Launch your Shellshock attack on this new program, and describe and explain your observation.

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

char **environ;

int main()
{
  char *argv[3];

  argv[0] = "/bin/ls";
  argv[1] = "-l";
  argv[2] = NULL;

  setuid(geteuid()); // make real uid = effective uid.
  execve(argv[0], argv, environ);

  return 0 ;
}
```

2.3 Task 3: Questions

This is a writing task, please answer the following questions in your report:

1. Other than the two scenarios described above (CGI and `Set-UID` program), is there any other scenario that could be affected by the Shellshock attack? We will give you bonus points if you can identify a significantly different scenario and you have verified the attack using your own experiment.
2. What is the fundamental problem of the Shellshock vulnerability? What can we learn from this vulnerability?

3 Submission

You need to submit a detailed lab report to describe what you have done and what you have observed, including screenshots and code snippets. You also need to provide explanation to the observations that are interesting or surprising. You are encouraged to pursue further investigation, beyond what is required by the lab description. Your can earn bonus points for extra efforts (at the discretion of your instructor).