Lab 2: Shellshock

Dr. Lotfi Ben Othmane Lisa Nguyen Quang Do

06 November 2015

This lab is due on **19.11.2015**, at **23:59**.
Your dedicated repository for the labs is: `https://repository.st.informatik.tu-darmstadt.de/sse/secdev/2015/students/labs/grp-X/`. You are to submit a report named `Lab2-Submission.pdf` at this location, or upload it on the course's Moodle. We will *only* grade this document.

As always, you should detail your observations and proceedings, while keeping your answers concise. You are encouraged to join screenshots and code snippets.

The VMs and network setup will be the same as for Lab 1R.

In the second exercise, you will be examining malware traces. Be **extra-careful** when examining the files given to you. **You do not need to run them**. If you do, always run them under strace, or with Wireshark. Make sure the network is disabled.

Exercise: Shellshock (85 points)

The Shellshock attack, or Bashdoor, was discovered on September 24, 2014. This is a family of security bugs in Bash that allows attackers to make use of Internet-facing services to execute arbitrary commands in Bash. In this lab, you will get an experience of this attack, and understand how it works.

For this lab, we will need two things: (1) the Firefox web browser, and (2) an Apache web server. We will use the `Tamper Data` extension of Firefox. You are allowed to use `Wireshark` or any other tool if you feel the need for it.

1 Shellshock with CGI programs (26 points)

1.1 Apache setup

Start the Apache web server: `sudo service apache2 start`.

(a) Which folder of your system is the document root of the Apache server? In which configuration file can you change it? Which URL should you type in a web browser to access the `index.html` page located there? (3 points)

(b) Which folder of your VM is the default CGI directory for the Apache web server? In which configuration file can you change it? (2 points)

1.2 Set up the CGI program

(a) What are Common Gateway Interface (CGI) programs? What are they typically used for? (2 points)

(b) Many CGI programs are written in shell script. Save the following code snippet as `myprog.cgi`. Place your CGI program in the `/usr/lib/cgi-bin` directory and set its permission to 755 (so it is executable).

```
#!/bin/bash

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

From which URL can you access your program? (from a web browser or using the `curl` command). (1 point)

1.3 Attack

When a CGI program written in shell script is called, the shell program will be invoked first, and then, the CGI script will be executed in it. For the Shellshock attack, the CGI program itself does not matter, as it targets the Bash program.

Turn on a second VM and ensure both your machines are connected to the same NAT network. From the second machine, open Firefox and access the CGI page of the first machine.

(a) Your task is to exploit the Shellshock vulnerability of the first machine to:

- remove a file
- retrieve the `/etc/passwd` file

Describe how your attack works. You can use screenshots and code snippets to support your demonstration. (10 points)

Note: You cannot modify the bash program nor change anything on the first machine. On the other hand, you can look at the Apache logs to help craft the exploit (`/var/log/apache2/error.log`).

(b) Download the complementary files to the lab (`https://sharing.sit.fraunhofer.de/index.php/s/vdJjSBm5TQaASeS`). The file `variables.c`, contains the vulnerability exploited by the Shellshock attack. Identify the vulnerable line in the `initialize_shell_variables()` function (between lines 307 and 369). Explain how your attack makes use of this vulnerability. (8 points)

2 Shellshock with Set-UID programs (39 points)

(a) Run the following command: `sudo ln -sf /bin/bash /bin/sh`. What does it do? (1 point)

(b) What are Set-UID programs? What are they typically used for? Which commands can be used to make a regular program a Set-UID program? (3 points)

(c) `system()` and `execve()` can both be used to execute a shell command. What is the main difference between the two functions? (3 points)

(d) Compile the following program. Set its owner as `root` and make it a Set-UID program.

```
#include <stdio.h>

void main() {
        setuid(geteuid());
        system("/bin/ls -l");
}
```

Assume a regular user runs this program. What will happen to the process running the program? Will the program be run with root privileges? (4 points)

(e) Log in as a normal user. Can you use the Shellshock vulnerability to get a root console? Document your approach. (10 points)

Note: You cannot edit the Set-UID program.

(f) Remove the `setuid(geteuid())` line, and repeat your attack. Does it still work? Why? Identify the corresponding line in `variables.c`, in the `initialize_shell_variables()` function (between lines 307 and 369). (8 points)

(g) The following code snippet does the same thing as the code in question d, except it uses `execve()` instead of `system()`. Compile the code, and make it a root Set-UID program. Launch your attack on this program. What happens? Explain your observations. (10 points)

```
#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());
        execve(argv[0], argv, environ);
        return 0 ;
}
```

3 Conclusion (20 points)

Summarize what you learnt about the Shellshock vulnerability. Include the following observations:

- What is it? Why is it dangerous?
- What is the fundamental problem of this vulnerability? What, as a programmer, should one be particularly wary about?
- Other than the two scenarii seen in this lab (CGI and Set-UID programs), is there another way to exploit the Shellshock vulnerability?

Exercise 2 (15 points)

This exercise is given in the form of open challenges. There are purposely harder to solve than the guided first exercise and you will need to look for information yourself. There are several ways to solve the exercise, you are free to submit any working solution, as long as it does not invlove cheating.

All of the necessary files can be found here: `https://sharing.sit.fraunhofer.de/index.php/s/vdJjSBm5TQaASeS`

One of the machines of your company's servers has been compromised. You are tasked with the investigation.

(a) **Forensics: (5 points)** Open the file `honey.tar`. It contains the data that was found on the compromised server. Examine it and explain your observations. Include the following information:

- Which path is used to hide the malicious files?
- Locate the crackMe executable in the malicious files
- What is the nature of the program disguised as -bash?
- What is the name of the program?
- What is the main config file of the program?
- What is the name of the channel used by the attacker?
- What is the name of the server unwillingly acting as a control server?
- What is the hostname used by the attacker to identify himself to his slaves?
- What is the purpose of the "stealth" program?
- Where was the attacker when he "thought of you" today?

Note: Be **careful** when handling those files. **You do not need to run them**. If you do, try getting an idea of what they do **before** you run them. Always run them under strace, or with Wireshark. Make sure the network is disabled. We do not want to get emails from the TU asking why your laptop is attacking outside machines...

(b) **Reverse: (5 points)** You think the attacker left a message in a crackMe executable that could help you uncover his identity. Try to find the correct password. What is it? What seems to be the identity of the attacker?

Note: For those who haven't answered the first question, the executable is the `cr4ckMe` file.

(c) **Web vulnerabilities: (5 points)** You are now investigating how the attacker managed to compromise your server. You think that a particular web page of your website is vulnerable. You have copied that page on one of your local machines for further examinations (Location: on one of your VMs, at /var/www/l2e2). Can you exploit HTTP, JavaScript and PHP vulnerabilities (consecutively) to bypass the securities? Shortly describe how you did it.

Note: You can look at the source code, but remember that the attacker does not have access to it. Only answers based solely on what a remote attacker can see will be accepted.