

Lab Exercise 3

Due: November 8, 2016 (**note extension!**)

Points: 100

For this laboratory exercise, you are to work in teams of 2–3 people. When you turn in your results, be sure to list the team members.

We will use interactive grading for the labs. After the due date, we will schedule 10 minute periods where a team can meet with the instructor or TA. At that meeting, we'll examine your solution, talk to you about it and ask questions, and may ask that you demonstrate it to us. All members of the team will receive the same grade. So when you come to the meeting, make sure all team members understand how the lab was done, and can answer questions about the lab and the results.

This laboratory exercise has you implement two types of buffer overflows. The first is a simple overflow that causes a parameterless routine to execute. The second is a return-to-libc (or arc) attack.

You will need a virtual machine that is available via the web at <http://nob.cs.ucdavis.edu/private/sec-ex3>. The user name is "secexer" and the password is "Intel!" (without the quotes, of course). When you start the virtual machine, you will find two programs, *bad.c* and *realbad.c* and two executables, *bad* and *realbad*, in your directory. Note that the last executable is setid-to-root.

Buffer Overflow I

In the the Homework area of the Canvas class web site is a program *bad.c* (also see below). This program contains a buffer overflow vulnerability; see the call to *gets(3)* at line 13. Your job is to exploit the overflow by providing input to the running process that will cause the program to invoke the function *trap* (which, you may notice, is not called anywhere else). You will know you've succeeded when you run the program, give it your input, and it prints "Gotcha!"

The following questions will help guide you. Please turn in your answers to them, a hex dump of the input you use to call trap, and a typescript or screen shot of you running the program *bad*, giving it your input, and showing its output.

1. What is the address of the function *trap*? How did you determine this?
2. What is the address on the stack that your input must overwrite (please give both the address of the memory location(s), and their contents)? How did you locate this address?
3. What is the address of *buf*?
4. The sled is the input you give to alter the return address stored on the stack. What is the minimum length your sled must be?

Buffer Overflow II

Now you will extend the overflow attack. In the the Homework area of the Canvas class web site is a program *realbad.c* (also see below). As before, this program contains a buffer overflow vulnerability. Your job is to exploit the overflow by providing input to the running process that will cause the program to invoke the function *runcom* and cause the *system(3)* function to be executed with a command embedded in the input you have given. You must pass in a parameter that is a Linux command, which the program will then execute. (I recommend the command *id(1)*.)

Please turn in the following:

1. A hex dump of the input you use. Please also show where the parameter to *trap* is in your input.
2. A screenshot of the program's output for that input.

Recovery

If you accidentally delete or change the executables, you can recreate them yourself. First, compile the source using *gcc* **with the option *-fno-stack-protector***; if you omit the flag, the attempt to overflow the stack will be blocked and so the lab will not work. That's it for *bad*. For *realbad*, once you compile it, do the following:

```
sudo chown root realbad
sudo chmod 4755 realbad
```

The Programs

bad.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int trap(void)
5 {
6     printf("Gotcha!\n");
7     exit(0);
8 }
9
10 int getstr(void)
11 {
12     char buf[12];
13     gets(buf);
14     return(1);
15 }
16
17 int main(void)
18 {
19     getstr();
20     printf("Overflow failed\n");
21     return(1);
22 }
```

realbad.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void runcom(char *cmd)
5 {
6     system(cmd);
7     exit(0);
8 }
9
10 int getstr(void)
11 {
12     char buf[12];
13     gets(buf);
14     return(1);
15 }
16
17 int main(void)
18 {
19     getstr();
20     runcom("echo Overflow failed");
21     return(1);
22 }
```