CS 334 Computer Security **Fall 2018**

Project 1: Exploiting Code

Prof. Szajda *Due Friday, September 21, 11:59:59 pm*

In this project you will play the attacker's role. You are given two vulnerable programs and you will create exploits for them.

Getting Started

You will run the vulnerable programs and their exploits in a virtual machine (VM). VMware Fusion is installed on the Macs in the Jepson 225 lab. If you wish to work on your own machines, VMware Player is freely available for Windows and Linux, and VMware Fusion is available as a free 30-day trial. The VM image that you will use is available off our course web page. The image is a bare-bones Linux Ubuntu installation. There are two users, `root` and `maluser`. Both have the password `cs161proj`. To use the image, start VMware Player, select **Open a Virtual Machine** and browse to where you've stored the image. If it asks whether the VM was moved or copied, select **I copied it**.

You will find the debugger gdb very useful for this project; it is worth spending some time becoming comfortable with it. To start gdb with a program loaded, type

```
$ gdb <executable-name>
```

You can then start running the program with

```
$ run [arguments-to-the-executable]
```

Some useful commands are `break`, `step`, `info frame`, `info locals`, `x <address>`. If you're brand new to gdb it is worth going through a quick tutorial. A basic one is here:
`http://www.cs.cmu.edu/ gilpin/tutorial/`. If you need to remember the commands, a pretty good reference can be found here:
`http://www.yolinux.com/TUTRIALS/GDB-Commands.html`.

Problems

1. (20 pts.) Buffer Overflow Vulnerability

In your VM image is the directory `/home/maluser/Q1`. This directory contains the files `target-q1.c`, `exploit-q1.c`, `Makefile`, and `shellcode.h`. You will modify `exploit-q1.c` so that it exploits `target-q1`. `target-q1.c` has already been compiled for you. The resulting program is owned by `root` and has the setuid bit set. You should not need to recompile this file; the source is included only so that you can inspect it to find the vulnerability.

To get started with this problem, you should review our notes on basic stack smashing, and use "Smashing the Stack for Fun and Profit" by Aleph One as a reference. Your task is to exploit the buffer overflow vulnerability in `target-q1` to launch a shell. The malicious shell code is provided `shellcode.h`; you have to cause it to be executed in `target-q1`. If you are successful, you will see a root shell prompt:

```
maluser@cs161: /Q1$ ./exploit-q1
#
```

Typing `exit` at the prompt will take you back to your shell. The reason you see a root shell is because `target-q1` is owned by `root` and has the setuid bit set. Therefore, it runs with the privileges of the file owner (`root`) and any program it launches (i.e. `/bin/sh`) will also run as `root`.

Deliverables for Problem 1: For this problem you will submit `exploit-q1.c`. I will log into a clean VM image as `maluser` and download your submission file to directory `/Q1`. I will then run `make`, will run `./exploit-q1`, and will check for the existence of the shell prompt. The root password on the VM I use will be different from the root password you were given. You should also submit a file, `exploit-q1.txt`, which should include a description of the vulnerability, how the vulnerability was exploited, the stack layout showing absolute locations of the variables you had to be concerned about, and a brief description of your solution, including how you determined which address to jump to. This document should be no more than one page.

2. (20 pts.) Format String Vulnerability

The second program you need to exploit is in the directory `/home/maluser/Q2`. The files in the directory are `target-q2.c`, `exploit-q2.c`, and `Makefile`. You will modify `exploit-q2.c` so that it exploits the vulnerable program `target-q2.c`. Again, `target-q2.c` has already been compiled, is owned by root, and has the setuid bit set. You should not need to recompile this file; the source is included only so that you can inspect it to find the vulnerability.

To get started with this problem, read "Exploiting Format String Vulnerabilities" by scut/team teso (http://julianor.tripod.com/bc/formatstring-1.2.pdf). The vulnerable program, `target-q2`, takes two arguments, `username` and `userid`. The first is a string, the second is an unsigned long. The program uses the userid to determine who gets authenticated—however, you will find that the target program has been written to not view any userid as having permission to authenticate. If somehow a user gets authenticated, the program will delete a `root`-owned file, `/root/grades2.txt`. Your task is to bypass the authentication check and get the program to delete the file for you.

Deliverables for Problem 2: For this problem you will submit `exploit-q2.c`. I will then check your exploit as described above. Success will be determined by whether the file `/root/grades2.txt` has been deleted. You should also submit a file, `exploit-q2.txt` which includes a description of the vulnerability, how the vulnerability could be exploited, and a brief description of your solution, including an explanation of how the arguments to `target-q2` need to be structured. This document should be no more than one page.

Submission Instructions: All deliverables for this project should be included in a single tar file. The name of your tar file should be XXXProject1.tar, where XXX represents, in uppercase, the first three letters of your last name. (Note well: If you submit an improperly named file, or submit to the wrong box folder, I will treat it as if you have not submitted anything.)

You should submit your project by attaching the tar file to an email sent to the address

exploit.s0p9t9vh0asetjc6@u.box.com

This has the effect of dropping the tar file into the appropriate Box folders in my Box directory tree.

Moving Files Between the VM and Local Host

Files can be moved between the vm and local host using `scp`. There are, however, a few things you need to know to make this work. These are listed below.

1. You need to have turned on remote login in the VM, and enabled it for your local host username.

2. When using `scp`, you need to know that both the vm and the local host run on the same virtual local area network. So use the gateway address of the virtual OS when running scp from within the virtual OS. If running from within the local host, you need to use the username you use in the virtual machine (e.g., maluser) and need to use the full ip address, not the gateway address.

   So, to clarify, suppose the ip address in the virtual OS is listed as 192.168.202.129. You would use this full IP address if running scp from terminal in the local host. But if running it from terminal in the virtual OS, you would direct to the gateway address (typically the ip address but with 1 as the least significant quad – here 192.168.202.1).

3. If you need to find the ip address of the virtual OS or of the local host, just run `ifconfig` in a terminal in whatever location you need the ip address (vm or local host).