CS 161 Computer Security
Spring 2010 Paxson/Wagner

Project 1

Due Februrary 18, 11:59pm

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

Getting Started

You will run the vulnerable programs and their exploits in a virtual machine (VM). VMware Player is installed on the instructional machines and is also freely available for Windows and Linux. The Mac version of VMware (VMware Fusion) is available as a free 30-day trial. The VM image that you will use is at `http://www.eecs.berkeley.edu/~csturton/classes/cs161/cs161-sp10-vm.tar.gz`. 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 just need to remember the commands, a pretty good reference can be found here: `http://www.yolinux.com/TUTORIALS/GDB-Commands.html`.

Problems

1. **(50 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, 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; we include the source only so that you can inspect it to find the vulnerability.

To get started with this problem, read Aleph One's "Smashing the Stack for Fun and Profit" (`http://insecure.org/stf/smashstack.html`). Your task is to exploit the buffer overflow vulnerability in target-q1 to launch a shell. We provide the malicious code in 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`.

Submission and Grading For this problem you will submit exploit-q1.c. The grader will log into a clean VM image as `maluser` and download your submission file to directory ~/Q1. A script will then run make, will run ./exploit-q1, and will check for the existence of the shell prompt. The `root` password on the VM used by the grader will be different from the `root` password you were given. You may also optionally submit a file, exploit-q1.txt, which includes a description of the vulnerability, how the vulnerability could be 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 and will be used to award you partial credit in the event your exploit does not work with our automated grading system. Therefore, we strongly encourage you to provide this explanation.

2. (50 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; we include the source 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.

Submission and Grading For this problem you will submit exploit-q2.c. The grading will be done as in Question 1. Success will be determined by whether the file /root/grades2.txt has been deleted. You may also optionally 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 and will be used to award you partial credit in the event your exploit does not work. We strongly encourage you to submit it.

3. (0 pts.) Feedback - Optional

Submit a text file, feedback.txt, with any feedback you may have about this project. What was the hardest

part of this project in terms of understanding? In terms of effort? Or, provide feedback on the class (e.g., what's the single thing we could do to most improve the class?). We appreciate any feedback you may have. Your answers will not affect your grade.