CS 577 Cybersecurity Lab

Lab 4 – due 10/8/15 11:59pm

Subject: Hijack control-flow and perform a code-reuse attack

You are given a toy program that includes a buffer overflow. Your goal is to write an exploit that makes use of the vulnerability to hijack control flow and inject your shellcode in the victim process.

Assumptions

For this assignment we will assume that defenses like stack protection and ASLR are not in place. However, stack and heap are non-executable. This is achieved using the following:

Disabled stack protection: the option *-fno-stack-protector* was passed to GCC when compiling the victim programs.

Disabled ASLR: the gdb debugger can disable ASLR to assist in debugging, so we are going to run the victim program through gdb to disable ASLR. gdb on *gump* exhibits this behavior

Vulnerable program

You are given the lab4_files.tar.gz archive which includes one vulnerable program. You should gain control of the instruction pointer using a stack overflow. After you gain control you should perform three return-to-libc attacks. The first attack should call the unbelievable() function included in the program. The second attack should print the string "Hello, world\n" to standard output, using one of the C library's string printing functions. The third should be the same as the second one but should also chain a second ret2libc to exit by invoking the *exit()* library call.

The program reads input from standard input. You can generate and store your exploitation strings in a file, which you can then provide to the vulnerable applications. E.g., *./stack_overflow < exploit.txt* or in gdb *run < exploit.txt* .

For simplification the stack_overflow binary is 32-bit, which means that function arguments are passed on the stack instead of registers.

Example:

func(arg1, arg2)

CALLER	CALLEE
push arg2 push arg2 call func	STACK STATE: 0xf8 arg2 0xf4 arg1 0xf0 retaddr <pushed by call to func>

Deliverables

1. Deliver the exploit input that given to the vulnerable program will exploit it implementing the **first attack**. You should be able to demonstrate how you created the exploit during your examination, if requested. Include any utilities you developed to help you create the exploit. [30%]

2. Deliver the exploit input that given to the vulnerable program will exploit it implementing the **second attack**. You should be able to demonstrate how you created the exploit during your examination, if requested. Include any utilities you developed to help you create the exploit. [30%]

3. Deliver the exploit input that given to the vulnerable program will exploit it implementing the **third attack**. You should be able to demonstrate how you created the exploit during your examination, if requested. Include any utilities you developed to help you create the exploit. [20%]

4. Include a report.txt file explaining your choices when developing the exploit, the tools you had to create to help you in developing it, and the manual investigation that you had to do. [20%]

Submission information

The code you submit for grading must build and run on the linux-lab, even if you use a different machine/environment for developing. You should use a particular host in the linux-lab for this assignment, which we have tested and will more likely result into consistently working exploits.

Submit all your files as a tar.gz archive through Canvas.