Assignment 3 – Intro to Format String Exploits
Due Friday 05/24/19

Setup: From the Sakai page for this assignment download the binary file “assign3” along with “inetd.c” and “inetd”. You will need to get these files into your Ubuntu virtual machine. For part I you will probably want to disable randomize_va_space. A working solution for part II must work with randomize_va_space enabled. You may notice that assign3 contains no networking code. In order to facilitate network interaction with assign3, the target will instead be running the inetd program which, upon receipt of an incoming network connection, forks and execs assign3. The following command will be used on the target:

```
./inetd –p 4444 ./assign3
```

If you want to make the stack move around a little bit for practice, without going to full blown randomization, you may append an arbitrary number of command line arguments to the above command. Some examples:

```
./inetd –p 4444 ./assign3 aaaaa bbbbbb cccccc dddddd
./inetd –p 4444 ./assign3 `perl –e 'print "A"x256'`
```

Part I (80 points)**:** Answer the following questions and develop the exploits described below.

1. How does this program accept input from a user?
2. Describe the vulnerability present in this program.
3. Discuss any restrictions on the user's input that must be taken into account when attempting to trigger the vulnerability.
4. Discuss the structure of a user input that will successfully avoid any such restrictions and allow you to successfully take control of this program. Be as detailed as possible (size, format, content…). Make sure you detail exactly what you are overwriting in order to take control of the vulnerable application.
5. Develop an exploit that injects reverse/callback shellcode as part of the user supplied input and transfers control to that shell code. You must utilize shellcode that results in an interactive shell. Your shellcode must not depend on the presence of any executable other than /bin/sh. In particular, do not rely on the presence of netcat. As with assignment two, you will be given one chance to utilize information from gdb on the target to update any addresses in your payload in the event that your exploit demonstration fails on the first attempt. Updating your payload may be somewhat more involved than it was on assignment 2, so practicing such an update is highly recommended before you schedule a demonstration time.
6. Discuss a target independent means of determining the location of your shellcode. In other words, if you knew a target was running this service, but you had no idea where your shellcode would end up in memory, how would you go about determining the address at which your shellcode is likely to land so that you could successfully exploit the target (remember you don't get to run a debugger on the target in this case, all you get to do is connect to and interact with the service. Think information leak here).

Part II (10 pts):

1. Create an exploit which in a ***single*** connection to a target, determines a good address for your shellcode, then without disconnecting, or establishing a second connection, generates and sends the additional data to exploit and obtain a shell on the target which you use without any additional network connections taking place (no more than one three way handshake may take place between your computer and the target computer). In other words, automate the steps to determine a shellcode address, and then exploit the target. Such an exploit must work against a target that is using ASLR. You are not limited to a callback payload here. In fact such a payload would violate the single connection requirement. You are not allowed to disconnect and reconnect as part of your exploit.

Part III (10 pts):

2. Modify your exploit for part I or part II by changing the payload to one you have written yourself. Your payload may not use the execve system call and may not invoke a shell. Instead, your payload should make use of system calls to open a file named "key", read the content of that file and send that content back to your attacking computer over a network socket. In order to test your payload you will need to create a file named key on your Ubuntu virtual machine in the same directory from which you launch inetd. You must not make any assumptions about the size of the key file.

Deliverables

1. Upload answers to the questions above, including listings for all exploits developed and payloads used, to Sakai (or the Submit share) NLT 1700 on the due date.
2. Demonstrate your exploits to your instructor NLT 1700 May 31 2018.