Due Monday 05/03/21 1700

For this assignment, you must develop a working exploit that utilizes a bind shell payload. If you choose not to create your own payload, you might want to investigate metasploit which contains a payload generator called msfvenom. A free, community version of Metasploit is available online from Rapid 7, or you can find metasploit pre-installed in Kali Linux.

The target binary (assign2) is built to run on the class target Ubuntu 18.04 virtual machine (available here: https://nps.box.com/s/3k4ue8uoudkzrod8luaggeliluhwfefj, user: ubuntu, password: password). Download the assignment binary and copy it into your virtual machine.

For this assignment, address space layout randomization (ASLR) will be disabled. You may disable ASLR in your VM using the following command:

```
$ sudo sysctl –w kernel.randomize_va_space=0
```

Please not that this setting is not persistent and you will need to reissue the command following any reboot.

The assignment is meant to be launched by the provided inetd utility (already in the VM) which will allow you to access the assignment over a tcp connection: For example, to listen on port 5555 you would issue the following command:

```
# ./inetd –p 5555 ./assign2
```

You can also run the assignment from the command line and do initial debugging with gdb at the command line. Your final demonstration however will take place across a network.

You may want to disassemble the binary. You can do it with objdump (under cygwin or within the appropriate vm):

```
# objdump -M intel -d assign2
```

which will dump to the screen, so you may want to redirect to a file for easier reading. Alternatively you may want to open the file using a more capable disassembler such as Ghidra (which also has a decompiler and is available here: https://ghidra-sre.org/) or IDA Pro, a freeware version of which is available here: https://www.hex-rays.com/products/ida/support/download_freeware.shtml)

The binary is not malicious in any way. For starters you might try running it to see if the binary generates any output then go from there. The `strace` and `ltrace` programs can help give you a sense of the system calls and library calls that are being made by the program. Get creative with your input to see if you can get the program to crash. You may want to observe things in gdb. If you are communicating with assign2 via a network connection, you will need to first connect to the target inetd so that it will fork off an assign2 process, then attach gdb to the

assign2 child process and set any desired breakpoints before you begin sending input to assign2 process.

For testing purposes you may assume that the assign2 binary will be launched in the following manner:

1. Boot the target Ubuntu image in VMware
2. ssh into the target as user ubuntu
3. Issue the following command: `./inetd -p xxxx ./assign2`
   where xxxx is the port number that inetd will listen on.

Deliverables

1. Answer the following questions and upload your answers via the Sakai assignment submission page:
   a) Describe the manner in which user input is accepted by the vulnerable program.
   b) Which function's stack frame contains the buffer that is being overflowed?
   c) What is the minimum amount of input that must be supplied to cause the program to crash in any way? What causes the program to crash at that point?
   d) What is the minimum amount of user input that must be supplied to completely corrupt the saved return address you are attempting to control? Describe the technique you used to determine this number.
   e) Are there any restrictions on the content and format of the user supplied input? In other words, must you satisfy any input validation checks and are there any byte values that you cannot send inside your buffer.
   f) What programming problem allows this program to be exploited? How could the programmer have prevented the buffer overflow?
2. Perl, C, or Python source code or command line (upload via the Sakai assignment submission page) for the attack that you have implemented. Using comments in your source file, you must also detail the exact commands used to execute the attack and the order in which they must be executed in order for the instructor to recreate your attack.
3. Assembly language listings (upload via the Sakai assignment submission page) of the payload that you use in your attack. A hex dump is not sufficient here. You must also indicate where you obtained your payload. If you used metasploit to create your payload, you must supply the exact `msfvenom` command line used to generate your payload. DO NOT encode your payloads.
4. A step by step description, (upload via the Sakai assignment submission page) of how an attacker executes a bind port style attack along with a description of a networking environment in which a bind port attack will fail.
5. Arrange to demonstrate your attack to your instructor prior to COB on 07 May using the code that you submitted to Sakai.
   a) You must be prepared to carry out your attack against the Ubuntu 18.04 target provided by your instructor.
   b) You will be expected to understand how to modify your attack (if necessary) to work successfully on the target subnet designated by your instructor.