<u>Excersize 5 – Return Oriented Programming</u>

In this exercise you will manipulate the stack memory in order to run code:

0. README.1ST
    a. You can base your answers on the provider overflower.py
    b. Make sure exercise_mode.sh has been turned on from ex03_04
    c. Make sure to document your work and thought process thoroughly in your own words!
1. Return to system() shellcode: [document your work in q1.txt and q1.py runs the exploit]
    a. Refamiliarize yourself with the binary we used to exploit in exercise #2. “stacko1”
    b. Stacko1 is almost the same binary, however this time as you can by running “execstack stacko1” – The stack is not executable! (The difference is in the ELF header).
    c. Use gdb to find where the return pointer is stored iern the stack.
    d. Locate how many bytes you need in order to over-run it. [Remember, it is not immediately after the buffer]
    e. Now that we are in-sync with the stack return pointer, use it to return to system():
        i. Run gdb, load the binary and stop, now find the real address of system() function in the Libc library (not the address in the Procedure Linkage Table (aka PLT)).
        ii. Use “./tools/memmap.py ./stacko1” to see that the address of the function is in the range of the appropriate table.
        iii. Use “./tools/memmap.py ./stacko1 string” too look for the address for “/bin/sh”
        iv. Restructure the stack to contain the two pointer so that /bin/sh is executed!
    f. Run everything and document all the addresses and strings you’ve found, and how you’ve done it.
2. Return to system(“/bin/cat /etc/shadow”) and then exit() shellcode: [document your work in q2.txt and q2.py runs the exploit]
    a. Repeat steps of answer to #1 but this time, do not run “/bin/sh” instead call /bin/cat /etc/shadow.
    b. To do that you will need to load the string “/bin/cat /etc/shadow” on stack(Remember, our original overflows only limit is the edge of the memory).
    c. Restructure the stack so it runs system(“/bin/cat /etc/shadow”)
    d. Restructure the stack so that it returns from system and immediately follows onto exit()
    e. Restructure the stack that the process exits with an error code of 77
    f. Please document all that you’ve done.
3. Locating gadgets: [document your work in q3.txt]
    a. Run ./rop_ptrace.py to get a brief explanation of what it does.
    b. Use ./rop_ptrace.py to locate a gadget that (“pop esp”)
    c. Find a gadget that loads a value into eax, ebx, ecx, and edx.
    d. Find a gadget that loads 2 or more of these registers one after the other.
    e. Find a gadget that moves something into [eax]
4. Information Leak: [document your work in q4.txt and q4.py runs the exploit]
    a. Using return to printf(), print the value of the unsigned long located at [ecx]
5. BONUS Exercise [10pts]:
    a. Create an infinite loop of puts(“Hello, World!”). [5pts] [q5a.txt,q5a.py]
    b. Create something that loops 15 times [5pts] (it has to be a loop). [q5a.txt, q5a.py]