CSE 5/7359 – Software Security

Lab 2/3 – Buffer Overflows

Lab 2

Parts 1 & 2 Due March 18

Lab 3

Part 3 Exploit Period: Sep 22 – April 1

Part 4 Exploit Write-ups Due: April 1

Objective: We will write and exploit buffer overflows for this lab.

1. Develop a useful program with (1) one intentional buffer overflow. By a “useful” program, I mean that the program should simulate some real world process or program (e.g. ATM, movie rental, classified system authentication, etc.). You will not be graded on the complexity of the program, but on the exploitability, ingenuity and creativity of the buffer overflow. The following requirements are mandatory:

* The buffer overflow vulnerability shall be exploitable to give access to a value or resource that is otherwise protected by the program or operating system (e.g. free money, unauthorized login/capability). Crashing the program, does not count as an exploit.
* The program shall execute on 32bit Kali Linux which you should have already downloaded and setup (see tasks in Blackboard)
* The program shall have no more than 10 inputs from the user (this is to prevent coders from trying to hide the overflow in excessive inputs)

### Submission:

* All Lab 2 files must be submitted by March 18 – This is so that we can begin exploiting them the next day, and everyone has the same number of days to attempt exploits.
* Each student shall submit the source code and any other required files as well as a README that explains how to compile the app, what it does and the goal of a successful exploit (NOT how to exploit)

2. Use Blackboard to turn in a simple “how-to” for me on EXACTLY how to exploit your buffer overflow… screen shots are appreciated. This is also due April 1. Do not share this how-to with any classmate.

3. On March 18, I will be placing all of the source code files in Blackboard for you to download, compile and exploit. You may not modify the source code in any way, but may need to replace or tamper with input files to exploit the program unless the README specifically instructs not to. Remember that you are exploiting other students’ programs, so some may not work properly. If a program does not work or you believe it is not exploitable (after you’ve studied the source code), simply explain that in your write-up (see part 4). You must exploit 10 of the programs to receive full credit for this portion. Partial credit will be given for fewer than 10. Please work alone. Questions can be submitted to the “Assignments” discussion board, but do not give hints or answers.

## 4. Use Blackboard to turn in a document that explains your overall approach to exploiting buffer overflows and the specific methods for each successful exploit from part 3. The explanations should be brief but complete (i.e. “at the Enter Name prompt input 15 A’s). Also explain the output/value gained by the buffer overflow.

L33t pointz\* (5 max):

* I will place several extra binaries/source in Blackboard along with the students programs. Successfully exploiting these programs will be rewarded with a l33t point for every one properly exploited. These may or may not have source code available.
* I will give up to two (2) 133t points for creativity, ingenuity, and difficulty level in your program. This means that if you find non-standard ways to create the buffer overflow condition, make it more difficult to exploit (remember it must still be exploitable), or come up with some other ways to add l33tness to your program, you will be rewarded.
* I will reward one (1) l33t point for those who exploit over 80% of the binaries (or 80% of those that are truly exploitable).
* If you can use an exploit to get a command shell from one of the programs (don’t know if this will be possible), you will be rewarded the max 5 points.

\*L33t pointz are “extra credit” and will apply to this assignment only

***NOTE: Parts 1 & 2 of this assignment will be graded as Lab 2. Parts 3 & 4 will be graded at Lab 3***

***Hint: You must disable ASLR on Kali Linux if you wish to attempt any exploits that involve memory addresses. To do this try this command “***echo 0 > /proc/sys/kernel/randomize\_va\_space***”***

***Also when you compile you will need to disable stack protections by compiling with this flag “***-fno-stack-protector***”***