Assignment 4

System and Network Security

Deadline: 14-March-2017 (11:55 PM)

Total Marks: 100 [Implementation (Coding + correct results): 75, Viva-voice: 25]

Instructions:-

1. Submit your solution in an archive file (any format). The archive should be named **lab_assignment4.<extension>**
2. The archive should contain a directory for every question. The directory should be named **question<question-number>.**
3. Each directory should contain a file named **input.txt** and a directory named **screenshots**. Questions 6 should also contain a directory named **sourcecodes** which should contain the exploit source code.
4. The file input.txt should contain the input which you provide to the program along with the necessary explanations for questions 6.
5. The directory screenshots should contain screenshots of successful exploits for all the questions. Additionally, for questions 6 provide screenshots of failed attempts also. These screenshots should be named with the same number which is used to denote the input in the input.txt file.

	Exercise	Source	Marks
1	Provide input to the program such that it overwrites **updateMe** variable. The goal of this exercise is to demonstrate that a buffer overflow can modify the address space located **before** the vulnerable overflowed buffer. . Submit your input in the answer file. . Submit a screenshot of successful exploit.	a4_1.c	10
2	Provide input to the program such that it prints "Updated*successfully!*" The goal of this exercise is to teach you to craft your input in such a manner such that it overwrites a specific address space with a specific value. . Submit your input in the answer file. . Submit a screenshot of successful exploit.	a4_2.c	10
3	Create an environment variable **MALICIOUS,** such that when it is read into the buffer, it overflows the buffer in a manner which prints "updateMe successfully updated!" The goal of this exercise is to demonstrate that vulnerabilities can be exploited not just from interactive user input, but from other avenues of input like environment variables too. Additionally, to check if a software is vulnerable or not, data should be crammed into every type of input like environment variables, command line input, network input, GUI fields, menus etc. . Submit your input in the answer file. . Submit a screenshot of successful exploit.	a4_3.c	10
4	Provide input to the program such that the function **executeme** is executed. You will need to overwrite the **functionPointer** function pointer. The goal of this exercise is to teach you **function pointer overwrite attack**. This attack was to be covered in the next tutorial, but then I trust that all of you can do it yourself. . Submit your input in the answer file. . Submit a screenshot of successful exploit.	a4_4.c	10

5	Provide input to the program such that the function **executeme** is executed. The goal of this exercise is to demonstrate that a suitably crafted input can modify the normal execution flow of the program . Submit your input in the answer file. . Submit a screenshot of successful exploit.	a4_5.c	10
6	**Return-to-libc attack** You can follow the tutorial given below and learn more about performing a ret-to-libc exploit to spawn a shell. https://www.youtube.com/watch?v=pVBnjSQ4Fjk . https://www.youtube.com/watch?v=GwtPmwa2PLg . Submit the exploit source code along with a screenshot of successful exploit. . Compile the program without “–z execstack” flag.	a4_6.c	25