Assignment 3

System and Network Security
Deadline: 13th March (11:59 PM)

Instructions:

- Submit your solution in an zip file. The archive should be named **assignment3.zip**
- The archive should contain a directory for every question. The directory should be named **question<question-number>.**
- Each directory should contain a file named **input.txt** and a directory named **screenshots**. Questions 5 and 6 should also contain a directory named **sourcecodes** which should contain the exploit source code.
- The file input.txt should contain the input which you provide to the program along with the necessary explanations for questions 5 and 6.
- The directory screenshots should contain screenshots of successful exploits for all the questions. Additionally, for questions 5 and 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 prints "modifyMe successfully modified!" • 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.	q1.c	10
2	Create an environment variable **MALICIOUS**, such that when it is read into the buffer, it overflows the buffer in a manner which prints "modifyMe successfully modified!" • 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.	q2.c	10
3	Provide input to the program such that the function **tryToExecute** is executed. You will need to overwrite the **functionPointer** function pointer. • The goal of this exercise is to teach you function pointer overwrite attack. • Submit your input in the answer file. • Submit a screenshot of successful exploit.	q3.c	10
4	Provide input to the program such that the function tryToExecute 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.	q4.c	10
5	Use **exploit2.c** provided in the article "smashing the stack for fun and profit' by Aleph One and try to spawn a shell. Use the same shellcode as provided in the article. • Link: http://insecure.org/stf/smashstack.html • q5.c is the vulnerable.c program used by Aleph One in this article.	q5.c	30

	• Submit all the inputs along with the screenshots that you tried but didn't work for you. (Marks will be cut if we don't see diligent effort from your side.) • For each failed input explain why it didn't work for you. • If one of the input works for you, submit that input along with the screenshot. • Submit **exploit2.c** along with detailed comments in the source code explaining what the program is doing at each step.		
6	Use **exploit3.c** provided in the article "smashing the stack for fun and profit' by Aleph One and spawn a shell. Use the same shellcode as provided in the article. • Submit all the inputs along with the screenshots that you tried and did or didn't work for you. • For each failed input explain why it didn't work for you. You need to spawn a shell. • Marks will not be given if a screenshot of successful exploit is not submitted. • Submit **exploit3.c** along with detailed comments in the source code explaining what the program is doing at each step	q5.c	30