Isai Mercado Oliveros

misaie

November 11, 2015

Buffer Overflow - Option 3

Buffer overflow is one of the most dangerous vulnerabilities that a program can have. It consists of asking the user for input without checking bounds of the input. If the input exceeds the length of the memory space reserved for the input then the user can override every single value of the stack. For a normal user overwriting the stack might just result in a exception that terminates the program but for an attacker it means that they have control over the flow of the program.

For this project, I first injected shell code in the stack to execute, then when I did it successfully I when to create another small program to make a environment variable with my shell code, and then I modified my first program to execute the environment variable instead of executing the shell code from a stack address. However, I kept reading and with modern stack flags such as non executable stack portions, it is impossible to write shell code on variable and then execute it, so attackers started to use portions of the existing code to call the functions they needed.

This kind of attack is called return oriented programming. In this attack, the attacker does not write machine language code to execute because the stack, heap, and other portions of memory will not be executable, but instead, the attacker uses the existing program code and linked libraries to execute the code they want. The most simple example of return oriented programming is called “return to lib c”. In this attack the return address is overwritten by the address of the c system call “execvp” or similar to execute a program where the parameter is a pointer to the string “/bin/sh ”. This pointer is directly stored in the stack before the return address because once the program is executing the libc function the libc function will look for the parameters on positive indexes from the return address.Therefore, the attacker overflow the stack until it reaches the return address. In the return address he puts the address of the “execvp” function and then it stores the pointer to the string “/bin/sh” which can be stored in the stack or in a environment variable.

The attack return to lib c is the most basic form of return oriented programming and it is very powerful. More complex programs write addresses in the stack to existing functions that will take the attackers desire parameters because the attacker will provide the parameters in the stack. In this manner, the attacker calls a function, gives parameters, the stack keeps popping values just to find a new return address with parameters, that it will execute, and when the function returns to be given the next address to execute, the attackers flow will keep giving return addresses and parameters until he executes what ever he wants to.

Return oriented programming is Turing complete; or in other words, it can execute whatever the attacker wants to. In addition, as to 2015 there are not known ways to stop this. Some researchers have implemented program flow checking which checks if a function is allowed to call another function by checking its graph flow. However, this graph flow checking and other solutions come with heavy performance impacts so it is not a feasible solution.