Chritopher LaJon Morgan

Nov 8, 2013

Homework 11 CS 465

**Buffer Overflow Defenses; Answer the following:**

* **List at least 5 defenses against buffer overflow attacks and provide a sentence or two describing what they are or how they work.**

1. Write Correct Code

The issue exploited by a buffer overflow attack is simply one that exists because a programmer didn’t enforce his assumptions. In essence it is a programmer’s faulty or irresponsible code that creates the overflow issue. If the programmer simply wrote code that checked and sterilized all input to insure that it matched expectations this attack would be useless. So the write correct code defense is to write code that enforces expectations onto its input and that checks bounds, etc.

1. Non-executable Buffers

One possible aspect of a buffer overflow attack can be to write code onto the stack, jump to it, and then execute it. This allows the attacker to inject code directly into the system and then execute it. However, the non-executable stack defense prevents the execute part of this attack. The defense works by parking parts of memory as non-executable. This means that if the CPU is ever asked to execute code in a non-executable area it will throw an exception. This prevents an attacker from writing his code onto the stack and then executing it.

1. Array bounds checking

Every flavor of buffer overflow attack is possible because come array pointer is allowed to be incremented outside of its bounds and then have data written there. Therefore, the array bounds checking defense simply enforces a bounds check on all array pointers. Therefore, if an array pointer is ever incremented outside of its bounds an exception is thrown. This prevents all flavors of the buffer overflow attack.

1. Code pointer integrity checking

This defense focuses on the idea that most buffer overflow attacks target control code, therefore, this defense places a known values around code that the system wants to ensure will not be altered. If these values are ever found altered this signals corruption of the control code and throws an exception. These known values are often referred to as canaries and there are several different types of canaries (e.g., random, static, random XOR).

1. Address space randomization(ASLR)

A common practice for buffer overflow attacks is to direct control of the program to an address in the stack containing injected exploitation code. This

* **We learned in class about a null terminator canary and a random canary. What is a limitation of the null terminator canary?**

The predominant assumption of this paper’s recommendations on finding a buffer overflow vulnerability is that one can find out something about the source code. They’re

* **What limitation of the random canary led to the development of the XOR canary??**