Scott Christensen

11/10/16

C.S. 465

H.W. #12: Buffer Overflow Defenses

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

-One can use canaries, or known values in your code, usually placed between a buffer and control data on the stack. Should a buffer overflow attack occur, the canary value would be the first to be changed. If that happens, we can know that there has in fact been a buffer overflow attack.

-One can use Bounds checking in their code. What one does is to detect if a variable is within a certain bounds before it's used at runtime. This can be set to be within a certain range of numbers of if it is being used as an array index that could be within a certain array size.

-Tagging is another way of prevention buffer overflow attacks. What one does is mark certain areas of memory as non-executable or even non-allocated so that no random commands or addresses can be accessed during a buffer overflow attack.

-Use strongly typed programming languages. By using these, your code as no direct memory access so that if your code is confiscated, commands to shells or random addresses will not be possible. Java and C# are among these languages in which you can be relatively safe.

-Validate input as it comes in. One should check input that is too long or uses that wrong data types. Especially watch out for any commands injected into parameters, so be sure to properly sanitize your inputs.

2. Describe the attack that defeated a random canary. Explain how the XOR canary defeats the attack.

Random canaries are randomly generated values, usually based off of the time and system entropy randomness. What an attacker can do though is simply change addresses and go around the random canary.

A Random XOR canary solves this problem by storing an XOR result of a random canary and the return address If the attacker changed the return address, the xor'd random canary will not match and we will know we're being attacked.