Scott Christensen

11/03/16

C.S. 465

H.W. #11: Stack Smashing

- What are two different ways to succeed at a stack smashing attack described in the paper?

  The first method described is a way of overflowing the stack by writing information, and allocate data past the end of an array to interrupt the flow of execution. This is done so as to force it to jump to a random address. At that address, we will want the program to open a shell from which we can execute any other commands desired or have that address point back to a location of the buffer where we have arbitrary commands stored.

  The second method described is just a variation of the above attack, but for when the buffer you are trying to overflow is so small that a shellcode will ruin addresses and you wouldn't have enough NOPs to guess the address. You can only obtain a proper shell if you know the program's environment variables and place the shellcode in one.

- How does the paper recommend you find a buffer overflow vulnerability? Do you know of any other ways to find this vulnerability?

  Since C doesn't have any built-in bounds checking, overflows are usually found when writing past the end of a character array. You can use a number of C functions to help you read the characters of the buffer to help you find inconsistencies. Functions like strcat(), strcpy(), sprintf(), vsprint(), and especially grep(1).

  Apparently you want to use these functions to look for functions that are of a static buffer size and their other argument is derived from user input. You also want to use these function to help you find is the use of a while loop to read one character at a time into a buffer from stdin or some file until the end of line, end of file, or some other delimiter is reached.

- What two questions do you have about stack smashing attacks that you can bring to class?

My first question about this attack is if this attack can only happen if all of the preconditions defined at the beginning of the paper are met; on an Intel x86 CPU a Linux operating system. Must these conditions be true for the computer we are trying to execute the attack on or does the attacker only need to use this hardware? The paper was unclear about these details.

My second question is about just how many commands one would be able to fit into a given shell, buffer, or environment variable used in these attacks? The more commands that could be put into these respective containers the more damage that can be done per attack, so it seems like something that an attacker would like to have an in depth knowledge of and most certainly something someone trying to find such an attack would wish to be aware of.