Swasti Mishra
Stella Sun
COSC 366
5 October 2022

COSC 366 Written Assignment 2

1. Overflowing Buffers
   a) If the stack is growing in the opposite direction, it would be impossible to overwrite the return addresses of the currently running function. However, if the stack is growing in the same direction (up), then whenever a new function is called, the associated function frame will be higher in memory than the function frame used to call it. This means that a hacker can use buffer overflow to change the next function frame and its return addresses.

      They can do this when strcpy() is called because it gets pushed to the top of the stack (i.e., to a higher address than func().) An attack may hijack the control flow by filling the input str with a string longer than buf, and then whatever shellcode the hacker would want to execute. The input string would then end with junk data that reaches the return address of strcpy(), and then replace that with the address of the shellcode the hacker inputted.

      If all went well (by the hacker's standards, of course), then the control flow would execute as normal until the code reaches strcpy(). At that point, strcpy() would write the input into buf until it overflowed. The overflow would reach into the next frame and overwrite the expected return address with the hacker's intended return address. Strcpy() then returns, but because it is returning the new return address, it will execute whatever the hacker inputted as the shellcode.

   b) If we assume that this system uses canaries, it is much more difficult to use the above method to overwrite return addresses. One could, however, overwrite a local variable, which could then force the program to allow more than 100 connections.

      To do this, a hacker could fill buf completely with 128 bytes of junk data using a local variable, and then use another local variable to contain a two's complement of a number less than 100 (I think the way this two's complement would look depends on the system architecture, especially with considerations for the system's endianness.) If the hacker does this properly, tmp will be overwritten and more than 100 connections can be made to the system.

2. Sharing files in Unix
   a) The security issue in this situation has less to do with the literal code and more to do with the permissions on them. Because Alice wrote alice-read and alice-write, she has read and write access to both. These programs will be executing on her behalf and can

therefore change permissions. An exploit that targets race conditions, for example, would allow other users to access her files. This exploit would take advantage of the window of time between the program checking its ruid and completing.

b) Because the security issue is with race conditions, it would make sense to close that window of time or make it as small as possible. Alice could also prevent people from running simultaneous programs. She could also set up several different conditions that have to be met within the small window of time, much like a regular household lock (with pins inside, if that makes sense.) All of the pins, or conditions, would have to align or be met without dropping any of the earlier ones to protect the file.