A1 Milestone

Jiaming Zhang

September 2018

1 sploit1

The first vulnerability is a buffer overflow bug.

In the function 'copyFile'(line 37 - line 42), it is trying to copy the content from source file to destination file until the EOF is reached. However, the stack variable 'buffer' at line 26 only has length 3000, and the while loop did not check this constrain.

By making a file with content length longer than 3000, it will overwrite other data on the stack, include i, len, stack frame pointer, and eventually the return address. We can put the shellcode at the beginning of the file, and it will be copied to buffer, then we change the return address to point to the beginning of our buffer, when function copyFile returns, it will run our shellcode, and start a root privilege shell.

To fix this vulnerability, we can add another condition to the while loop, when i(counter) reached 2999(the end of buffer), we need to stop looping as well. Also, change the last byte in the buffer to null character.

2 sploit2

The second vulnerability is a string formatting bug.

In the function 'usage'(line 186), the usage of printf is risky. It print variable output without using the string literals. Which means, attackers are able to control the string format by changing the argv[0] passed to the main function.

To use this vulnerability, we first pass in a string like "AAAA%x%x%x...." and by looking at the output, we are able to find four consecutive byte with value 0x41. Then we align them to a word, and remove extra %x, to make the stack frame pointer point to the beginning of the formatting string. Then we change the beginning of the formatting string to be the return address of function 'usage', modify the last %x to %nx where n is the number that you

want to write to the address minus the original length of the output, lastly add %n to write to the target address. In my case, the number are too large to write for the printf, so I break it to two parts, and use %n twice to write the higher two bytes and lower two bytes separately.

To fix this bug, we should use the printf properly. Change the printf to 'printf("%s", output).