Assignment 1

sploit2.c:

Vulnerability: use **Formatted String Sploit**

How to exploits the vulnerability:

1. The code in submit.c line 372, function print_usage passed a string with size 640 which is larger than the buffer with size 281, the wrong use of string length for snprintf gives me chance to overflow the buffer; and traced the argument cmd I found it is saved in the argv[0] and there is not any check for argv[0].
2. To exploits the vulnerability, first create a very big string that combined with NOP, address and shellcode, then put this string into the argv[0], after run the code, point would jump to the address we set and then run the shellcode(similar with sploit1).

How to fixed: just change line 372: snprintf(txt, 641, "Syntax:\n\t%s <path> [log message]\n" to snprintf(txt, 281, "Syntax:\n\t%s <path> [log message]\n".

sploit3.c:

Vulnerability: use **TOCTOU**

How to exploits the vulnerability:

1. The code would write the message to the submit.log file, and the password for the root user would save in the /etc/passwd file, but we can not write on this file, so we can use the symbolic to link the submit.log path to passed's path
2. To exploits the vulnerability, we should use fork to have 2 threading, one thread remove the submit.log, and then call symlink("/etc/passwd", "/home/user/submit.log"); to link the path, another thread run the submit file, with message which is the new password for each user.
3. While the race start, it have chance that second thread do the submit programs, but at this time the path for submit.log is actually for passwd, so the program would modify the file to the message which is the password we want.

How to fixed: Accessing a file name or path name multiple times is permitted if the program can verify that every operation operates on the same file.