Duncan Van Keulen
Program Forensics Assignment
CS232 Spring 2021

1. What the program does
    a. Based on everything below, this mystery program generates *n* random numbers based on the -n <i> command line argument. It then outputs these numbers to a TCP port if provided (see BUG section below).
2. BUG - 3 options
    a. Not sure if it's a bug or a feature that the program deletes itself (I'm guessing it's a feature considering the output of the program with no arguments)
    b. I believe the actual bug in the program is that it doesn't close the connection to the TCP ports it establishes. I can't run the program multiple times one the same port, or listen to the same port again.
    c. The other option for the bug that may stem from the previous is that it does not output on the default port it says it does. When no port is provided, it simply writes out the numbers to the terminal.
3. Tool used: running the program.
    a. What we learned: When provided with zero arguments, the program deletes itself after printing it will delete all of your files. When run with command line arguments specifying the -n it generates random numbers.
4. Tool used: strings
    a. What the tool does: according to the linux manual, the strings command prints out the printable character sequences that are at least 4 chars long, mainly useful for determining the contents of non-text files.
    b. What we learned: the command line arguments and what they do. This can also be discovered through opening the binary file in vscode.
        i. -h shows the help message,
        ii. -n lets you specify how many numbers to allocate
        iii. -p allows specification of the port to send data via TCP to
        iv. -s allows you to sort the numbers,
        v. -e allows you to seed the random number generator
5. Tool used: du - disk usage
    a. What the tool does: displays how many counts of 1024 bytes are in a file/directory.
    b. What we learned: The size of the mystery binary is 16*1024 bytes = 16384 bytes (~14kb)
6. Tool used: file command
    a. What the tool does: According to the linux manual, this command tests each argument in an attempt to classify it with three sets of tests.
    b. What we learned: mystery is a ELF 64-bit Least Significant Byte (LSB) executable for x86-64 systems, dynamically linked on linux with the GNU/Linux compiler.

7. Tool used: strace (got idea from here)
    a. What the tool does: records all of the system calls used when running a program, using command line arguments -c and -t it can count how many times each was called and how much time each took to run.
    b. What we learned:
        i. The system calls used were *write* which takes the most time (scales linearly with the count of numbers because this is what's writing them out to the stderr), *munmap*, *brk, fstat, read, close, mmap, mprotect, access* (with 3 errors always), *execve, arch_prctl, sysinfo,* and *openat*.
        ii. Also, with 10,000 numbers and sorting enabled, the program runs in 0.006371 seconds, with sorting disabled, the program runs in 0.006175 seconds. This would seem like sorting is adding a bunch of time, however, with 100,000 numbers, it runs in 0.683343 seconds with sorting, and without sorting it runs in 0.826916 seconds. This is very interesting because since it spends basically all of its time writing, it seems like writing with sorting enabled or only ascending numbers is much faster than writing with sorting disabled. This is probably because of cache?
        iii. With a tcp port provided, the program takes much longer as well. The same run of the program run with sorting and 100,000 numbers took 1.11981 seconds to run with the TCP port as opposed to the .683343 without the TCP port specified.
8. Tool used: nm
    a. What the tool does: Lists the symbols from the object files (symbol tables)
    b. What we learned: I can see that in the main function, there is a *malloc* command (most likely allocating something for the random numbers), *perror, printf, puts* are used, and it appears there are *qsort* and *random* functions as well.
9. Tool used: nc - (got idea from here)
    a. What the tool does: netcat allows listening on net ports
    b. What I learned: when a port is provided, the program publishes the random numbers on a TCP port that can then be listened to with the nc command to view them.
10. Tool used: readelf (idea from here)
    a. What the tool does: Displays info about ELF files
    b. What I learned:
        i. The data of the file is stored in 2's complement, little endian format.
        ii. The address of the entry point is 0x400b80
        iii. I can also see the symbol table again with the qsort, write, puts, srandom functions… and whether they're global or local.
        iv. I can see the section headers and the size of them (64 bytes), the number of program headers (9) and the size of them (56)...