CS-232 SP22
Kun Kang (kk58)
Ael Lee (al87)

Program Forensics

1. Tool used: running the program.
    a. What we learned: initially the program stated that it would not delete the files, but it did
2. Tool used: strings /home/cs/232/sp2022/mystery
    a. What the tool does: The string command prints the sequences of printable characters in files (man strings)
    b. What we learned: We were able to determine that there were strings and also found the hint that we should “inspect the symbol table…” and a .symtab.
3. Tool used: file /home/cs/232/sp2022/mystery
    a. What the tool does: outputs the file type
    b. What we learned: file type is an ELF file
    c. Reference: readelf command in Linux with Examples - GeeksforGeeks
4. Tool used: readelf -h /home/cs/232/sp2022/mystery
    a. What the tool does: Displays structure of a file
    b. What we learned
        i. Displayed ELF Header containing magic number (confirmed that it is an elf file in hexadecimals prefixed with the 7f value).
        ii. Class field of 64-bit architecture was defined
    c. The 101 of ELF files on Linux: Understanding and Analysis - Linux Audit (linux-audit.com)
5. Tool used: objdump -d /home/cs/232/sp2022/mystery
    a. What the tool does: objdump -d displays the disassembled version of the executable section by section
    b. What we learned: The program includes sections labeled as: .init, .plt, .plt.got, .text, and .fini.
    c. Reference: https://en.wikipedia.org/wiki/Objdump
6. Tool used: nm /home/cs/232/sp2022/mystery
    a. What the tool does: nm lists symbols from object files
    b. What we learned: The program ran several familiar functions, including srandom(). From this, we assumed that the program had to do something with generating random numbers.
    c. Reference: https://www.ibm.com/docs/en/aix/7.2?topic=n-nm-command

7. Tool used: ls -l /home/cs/232/sp2022/mystery
    a. What the tool does: This command displays the size of the executable file.
    b. What we learned: We learned that the executable file is 16K bytes.
    c. Reference: https://stackoverflow.com/questions/11720079/linux-command-to-get-size-of-files-and-directories-present-in-a-particular-folde
8. strace
    a. strace -c /home/cs/232/sp2022/mystery
        i. What the tool does: strace -c displays the count time, calls, and errors for each system call and reports a summary on program exit.
    b. strace -t /home/cs/232/sp2022/mystery
        i. What the tool does: strace -t traces system calls and signals and prefixes each line of the trace with the wall clock time.
    c. What we learned: This tool showed us the system calls in the order they were called. The system calls in this program were: execve, mmap, pread64, arch_prctl, brk, openat, mprotect, access, fstat, close, read, write, munmap, dup, fcntl, and unlink.
9. Tool used: netstat -nat | grep LISTEN
    a. What the tool does: checks which ports are currently listening.
    b. What I learned: Even though the -h said that by default a connection to port 10234 would be established, I learned that that was not the case. Port 10234 was never established by default.
    c. Reference: https://www.cyberciti.biz/faq/unix-linux-check-if-port-is-in-use-command/
10. COMMAND-LINE OPTIONS: ./mystery -h
    a. h: show this help message
        i. Displays the command-line options
    b. n <i>: allocate <i> items
        i. Allocates <i> amount of items of memory to array of random numbers
    c. p <port>: use port instead of default (10234) and send data out that port on TCP
        i. Allows using a different port to establish a connection to
    d. s: sort
        i. Sorts the numbers generated in the order of least to greatest
    e. e: <seed>: use <seed> to seed the random number generator
        i. Generates a list of random numbers for each given <seed> value
11. What the program does:
    a. The program by default generates 100 random numbers and outputs them to the **TCP port 10234**, or is supposed to. See bug info.

    b. ./mystery -n <i>: instead of generating 100 random numbers, generates <i> random numbers
    c. ./mystery -p <port>: instead of outputting to TCP port 10234, outputs to port <port>
    d. ./mystery -s: program outputs the generated random numbers in ascending order
    e. ./mystery -e: program saves the set of generated random numbers as the <seed> number
    f. NOTE: multiple command-line options may be given

12. The bug: The executable is deleted before any system calls are made
    a. When we copied the mystery file to our local folder and entered strace -c ~/cs232/mystery/mystery, the executable was deleted, and the table showed that none of the system calls were executed.
    b. The unlink system call was executed only when no command-line options were given. We tried repetitively calling strace -c /home/cs232/sp2022/mystery and strace -c /home/cs/232/sp2022/mystery -10234, and it was only when we entered the former command when we got the "unlink failed: Permission denied" message.
        i. We can know that the unlink system call is a bug because when we run the executable with any given command-line option, the program exits with "+++ exited with 0 +++"
            1. Means exited with code 0 meaning the execution was completed with no errors
    c. The command-line option -h said that it would by default use port 10234, but that is not true. It does not use that port at all. No connection is established.