CS 232 Homework 3
Chan Kim

Tools

1. stat - check the size of binary
   stat -c %s mystery
   Result: 14008
   What I learned: The size of binary is 14008.

2. file - Check the file type
   file mystery
   Result: mystery: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=24122f9cfec5c11a9f4038cf9845e906b5ca9e32, not stripped
   What I learned: the executable file format is ELF 64-bit. From the part ‘dynamically linked’, I can know that it is able to link to dynamic library. I also know that is symbol info from ‘not stripped’.

3. ldd - Identify shared library dependencies of the file
   ldd mystery
   Result:
   ```
   linux-vdso.so.1 =>  (0x00007ffff3b52000)
   libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f32e7211000)
   /lib64/ld-linux-x86-64.so.2 (0x00007f32e75db000)
   ```
   What I learned: the file includes 3 libraries.

4. strings - Extract string from the file
   strings mystery
   What I learned:

   1. There is 6 commands which are
   -h: show this help message
   -n <i>: allocate <i> items
   -p <port>: use port instead of default (10234) and send data out that port on TCP
   -s: sort
   -e <seed>: use <seed> to seed the random number generator
   --default

   2. Several functions from glibc2.2.5 and glibc2.4 versions libraries are used..
   ex) srandom, unlink, stdout, puts, qsort, strlen, htons, close, __stack_chk_fail, htons, close, __libc_start_main, strcmp, malloc, fflush, listen, bind, perror, bzero, accept, atoi, sprintf, exit, fwrite, sleep, stderr, and socket

   3. I also know that there is output strings.
   ex)
   ```
   Unknown option.
   I just deleted all your files...
    not.
   unlink failed
   ERROR opening socket
   ERROR on binding
   ERROR on accept
   ERROR writing to socket
   ```

4. From some other information, I can assume section headers.

5. Running the program
   What I learned: when I run the program, it gives a message that ‘I just deleted all your files...not.**’,** and it deletes the file. When I run the program with -h, it shows the program’s usage page as it is described at the strings command in previous. When I run the program with -n, -p, and -e commands without each commands’ required value, the commands return “Segmentation fault (core dumped)” error message. -s command returns 100 different numbers. ./mystery -e 1 also returns similar value with -s. ./mystery -n 1 returns the same value as the first value of ./mystery -e 1, and the second value of ./mystery -n 2 is also same as the second value of ./mystery -e 1. However, ./mystery -e 2 returns different values. -p 1 returns a message ERROR on binding: Permission denied. When the value is 1 - 1023, it return the same error message, but when the number is 1024 like ./mystery -p 1024, the program is stopped without any message and ending.

6. nm - To check the symbol type in the object file
   nm mystery
   The symbol type. At least the following types are used; others are, as well, depending on the object file format. If lowercase, the symbol is usually local; if uppercase, the symbol is global (external). There are however a few lowercase symbols that are shown for special global symbols ("u", "v" and "w").
   "B"
   "b" The symbol is in the uninitialized data section (known as BSS).
   "D"
   "d" The symbol is in the initialized data section.
   "R"
   "r" The symbol is in a read only data section.
   "T"
   "t" The symbol is in the text (code) section.
   "U" The symbol is undefined.
   "W"
   "w" The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the symbol is determined in a system-specific manner without error. On some systems, uppercase indicates that a default value has been specified.
   * The above description is from man page.
   What I learned: I found that most functions, which are shown at strings command, are the undefined symbol.

7. readelf - To show the information of ELF file
   Readelf -a mystery
   What I learned: according to strings command, I assumed section headers, and the result of readelf described same result as strings command showed. I also learned that the main is located in 0x400b80 from entry point address.

8. objdump - disassemble the object file
   What I learned: objdump returns the file’s assembly code which looks readable. Based on the address of main from readelf command, I looked at the assembly code, and I found that strcmp check commands. Strcmp is also shown by strace and ltrace.
   Simply, the assembly code works like below.

Main ...
Jump 400ed1 ...

```
Jump if less 400cf9
    Less 400d1b     <stcmp@plt> <= This part seems to check -h based on the next information.
        Jump if not equal 400d38
            equal
                400d29 <puts@plt> ⇐ I can assume that this part is from -h that shows Usage
                exit
            not equal
                400d5a<stcmp@plt> <= this part check another command just like previous. .
                Jump if not equal 400d94
                    equal
                        jump 400ecd
                            jump if less 400cf9
                    not equal
                        jump if not equal 400df0
    not less
        jump if equal 400ef3
            equal
            jump if not equal 400f65
                not equal
                    exit
                equal
                    400f27 <fwrite@plt> <= I can assume this is for "I just deleted all
                                          your files" based on ltrace command
                                          from next part
                    400f5e <fwrite@plt> <= for same reason, this part is for "not." string
                    Jump 400f6f
                        exit
            not equal
            jump if equal 400f79
                equal
                jump 400fc4
                    jump if less 400fa4
                    …
```

Based on this information, I can assume every commands is determined by if statement.

9. strace and ltrace - tracking system call and library call dynamically

   What I learned:

   When I run the program, it automatically deletes the file. Strace and ltrace show that the system call unlink delete it.

   From strace: unlink("/home/ck45/school/232/homework03/mystery") = 0

   From ltrace: unlink("/home/ck45/school/232/homework03"...)

   -n command uses fstat system call and output values according to it by the system call write.

   -s and -e commands work similarly.

   -p command uses socket, bind, and fcntl system call, so it works differently to compare other commands.

   Ltrace shows that the function strcmp is used for checking which command is called.

10. gdb - The GNU debugger. It allows user to check inside a program while the program executes, or it was crashed.

    gdb -q mystery

What I learned: From gdb command, I am able to look at the bug which the program has.
In particular situation, the program shows segmentation fault (core dumped) error.
When I look at the inside by using gdb it shows

```
Program received signal SIGSEGV, Segmentation fault.
0x00007ffff7a5a1c0 in ____strtoll_l_internal () from /usr/lib/libc.so.6
(gdb) bt
#0  0x00007ffff7a5a1c0 in ____strtoll_l_internal () from /usr/lib/libc.so.6
#1  0x00007ffff7a56cd0 in atoi () from /usr/lib/libc.so.6
#2  0x0000000000400e78 in ?? ()
#3  0x00007ffff7a41f4a in __libc_start_main () from /usr/lib/libc.so.6
#4  0x0000000000400ba9 in ?? ()
```

The result shows that the bug is from atoi() which may have no value since it returns 0 when it receives an invalid value.
When I run the program, it shows another bug when I run with -p 1024 (more specifically the number >= 1024.

```
(gdb) r -p 1024
Starting program: /home/chan/Sources/school/232/homework03/mystery -p 1024
^C
Program received signal SIGINT, Interrupt.
0x00007ffff7b17b74 in accept () from /usr/lib/libc.so.6
(gdb) bt
#0  0x00007ffff7b17b74 in accept () from /usr/lib/libc.so.6
#1  0x0000000000401103 in ?? ()
#2  0x00007ffff7a41f4a in __libc_start_main () from /usr/lib/libc.so.6
#3  0x0000000000400ba9 in ?? ()
```

I found that the bug is from accept().