Project Phase 5: Code Review

Group:

* Ben Mikailenko
* Roman

(Both group members participated in both reviews)

# Review 1 - Program 1 – Program Author – Ben Mikailenko

1. **Configuration Data** 
   1. Protections are user configurable without recompiling the program

Yes, protections are read in the .acl file which is found through the name of the input file. The program reads the input file name, adds “.acl” to the end of the name, and attempts to open the .acl file.

* 1. Configuration data can be readily updated

Yes, configuration data can be opened in the .acl file and updated with a text editor.

* 1. It’s easy to see the current configuration for a file

Yes, the configuration can be opened with a text editor and read.

* 1. There are protections against invalid configuration data

The program does have protections against invalid data. It checks to see if there is a whole username per acl file. If there is any more or less than that, the is rejection. This is found in the isValid() function.

* 1. The program enforces the configuration correctly, and won’t share things it shouldn’t

The program does not have any r w or b access configuration in the acl file. So the program will share things it shouldn’t, which is not good.

1. **File Checks**
   1. Program checks that configuration can only be accessed by file owner

The program checks to see if the file owner is the same as the .acl file owner by reading the st\_uid of both files and comparing for likeness. This is found in the isOwner function.

* 1. Program will only share files, and process configuration, stored in ordinary files

The program checks both files to see if they are links or not, aka real files, using the is\_link function. This is found in the islink() function.

* 1. Checks are made so as to avoid TOCTTOU issues (e.g. by open() and then fstat())

No checks are made to avoid time issues.

1. **SetUID**
   1. Program calls seteuid()

The program does not call seteuid()

* 1. Program does not require root access

The program does not check for root access.

* 1. Program runs in EUID mode to open configuration / source files

The program does not run EUID mode.

* 1. Program runs in RUID mode to open destination file

The program does not run RUID mode.

* 1. Program spends as little time as possible in EUID mode

The program does not run EUID mode so it doesn’t spend any time.

1. **Memory Safety**
   1. Program is free of (obvious) memory issues memory leaks, double-frees, use of uninitialized memory, use after free, etc

Yes, the program frees all memory and allocates memory to the appropriate locations. There is free(fp) for closing the file. There is minimal use of the file between opening and closing making it good for memory and speed.

* 1. Program does not use unsafe string operations – e.g. strcat() or strcpy()

The program does use unsafe string operations, strcat() and strcpy(). I’d recommend using a safer function like strlcpy().

1. **Copy**
   1. Program (should) successfully copy file

Yes, the program successfully copied the file.

* 1. Copy is managed sensibly, without the use of system() or exec()

Copy is done by reading each char from the first file and “copying” the char to the target file. This doesn’t use system() or exec(). I’m not sure if there are better ways in doing this, but this does not look like the best way to copy a file.

1. **General Style**
   1. Code is generally consistently styled and readable

The code is very readable, indented well, and has good commenting.

* 1. Code is sufficiently commented to explain its own functioning

Yes, there’s a comment for almost every line of code. If anything, the code is over commented. Can be commented less but with more generalization.

* 1. Program checks return codes from functions that can return errors (e.g. malloc())

The program does not check any return codes for return errors. Not in the fp open or in any of the functions. Malloc is used but not checked for an error.

# Review 2 - Program 2 – Program Author – Roman

1. Configuration Data
   1. Protections are user configurable without recompiling the program
   2. Configuration data can be readily updated
   3. It’s easy to see the current configuration for a file
   4. There are protections against invalid configuration data
   5. The program enforces the configuration correctly, and won’t share things it shouldn’t
2. File Checks
   1. Program checks that configuration can only be accessed by file owner
   2. Program will only share files, and process configuration, stored in ordinary files
   3. Checks are made so as to avoid TOCTTOU issues (e.g. by open() and then fstat())
3. SetUID
   1. Program calls seteuid()
   2. Program does not require root access
   3. Program runs in EUID mode to open configuration / source files
   4. Program runs in RUID mode to open destination file
   5. Program spends as little time as possible in EUID mode
4. Memory Safety
   1. Program is free of (obvious) memory issues memory leaks, double-frees, use of uninitialized memory, use after free, etc
   2. Program does not use unsafe string operations – e.g. strcat() or strcpy()
5. Copy
   1. Program (should) successfully copy file
   2. Copy is managed sensibly, without the use of system() or exec()
6. General Style
   1. Code is generally consistently styled and readable
   2. Code is sufficiently commented to explain its own functioning
   3. Program checks return codes from functions that can return errors (e.g. malloc())