HY-457
Assignment 2

Assigned: 13 / 3 / 2019
Due: 3 / 4 / 2019

Introduction

In this assignment you are going to develop in C, an **access control logging system**. This system will keep track of all file accesses and modifications. Every such operation will generate an entry in a log file, stored for further investigation by a separate high privileged process. For this assignment you will need to use LD_PRELOAD which gives the ability to instruct the linker to bind symbols provided by a shared library before other libraries. This way, you are going to override the C standard library functions that handle file accesses and modifications (fopen, fwrite) with your own versions in order to extend their functionality. You will also test your logging system against a ransomware, so your access control logging needs to be able to detect ransomware behavior.

Objectives:

1) Event logging

As a first task, you need develop a shared library (logger.so) which overrides (using LD_PRELOAD) the standard I/O library of C in order to get and log the needed information for each file access before continuing with the standard I/O operation. In this way:

1. Every time a user creates a file, the log file must be updated with information about the creation of the file. You need to modify the fopen() function in such way so the creation of a file can be distinguished from the opening of an existing file.[1]
2. Every time a user tries to open a file, the log file will be updated with information regarding the file access attempt. For this case fopen() functions need to be intercepted and information about the user and the file access has to be collected.
3. Similarly, every time a user tries to modify a file, the log file will be updated with information regarding the file modification attempt. For this case of fwrite() functions need to be intercepted and information about the user and the file access has to be collected. Every

[1] A file is created when it opens for writing, and the filename does not already exist in the path where it will be saved.

fopen()/fwrite() function should create a new entry in a log file. The log must be stored somewhere where it can be accessible by all users. Each entry (row) has to follow the following format (columns):

a) **UID**: unique ID (integer) assigned to the user by the system[2]
b) **file name**: the path and name of the accessed file
c) **date**: the date this action occurred
d) **time**: the time this action occurred
e) **type**: this field describes whether the corresponding file was opened for read or write. It prints 0 for the creation of a file, 1 if the action performed to this file was an open and 2 if it was a write
f) **action_denied**: this field reports if the action was denied to the user with no access privileges. It prints 1 if the action was denied to the user, or 0 otherwise.
g) **fingerprint**: this field reports the digital fingerprint of the file the time the event occurred. This digital fingerprint is the hash value[3] of the file contents.

2) Log Monitoring

Develop a separate monitoring application (monitor.c), responsible for monitoring the logs created by Event logger (Task 1). This Log monitor:

a) Parses the log generated from Event logger in Task1 and extracts all incidents where malicious users tried to access multiple files without permissions. In particular, as an output, it prints all users that tried to access more than 10 different files.
b) For a given file input, Log monitor tracks all users that have accessed the specific file. By comparing the digital fingerprints, Log monitor checks how many times the file was indeed modified. As an output, it prints a table with the number of times each user has modified it.
c) Ransomware detection:
Ransomware is a type of malicious software that is used to block access to files, by encrypting them. Its main goal is to extort money from users in order to decrypt their files.

i) In many cases ransomware tries to hide malicious files in directories populated by huge amounts of files. In this scenario a ransom will create a big volume of files. You need to find if **x**[4] files were created in the last 20 minutes.

---

[2]Note: check getuid() function

[3] Note: You can use the md5 hash function. See openSSL md5 for more information. You are allowed to use whichever hash function you prefer.

[4] **x** is an integer specified by the user as input (e.g. find if more than 40 files where created the last 20 minutes).

ii) Ransomware will also try to encrypt files and discard the unencrypted version. You need to find and report all the events in the log where a ransomware opened an unencrypted file and created an encrypted one. Encrypted files end with the suffix ".encrypt"

3) Test your system

Develop a simple testing application (tester.c) to appropriately demonstrate the above tasks. Your testing application has to create files which afterwards will be used as input to perform your extended fopen()/fwrite() calls.

4) Bonus (20%)

The bonus of this assignment includes the implementation of a simplistic ransomware. The main functionality of the ransomware is to "create-encrypt a number of files and then, delete them". Of course, a proper ransomware would be able to detect the access control logging system and bypass it. However, for the purposes of this assignment, you are required to provide only the functionality that we ask for. More specifically, you should develop a C program that demonstrates the creation, encryption & deletion of a number of files inside a directory in a certain amount of time X, as follows:

1. Read the contents of a file
2. Create a new file
3. Write the encrypted data to the new file in the same location, with the same name and an additional ".encrypt" extension. You are free to use whatever encryption function you prefer (take hints from assignment 1).
4. Delete the original file

Your ransom should also be able to create a huge volume of files, so given a directory as an argument, it should create **Y**[5] files to that directory. **Y** is also defined as an argument.

Note: We assume that the ransomware will only have the basic functionality that we ask for, so it won't bypass the logging system or do anything equivalent.

[5] **Y** is an integer that specifies the number of the files your ransomware needs to create.

Test your ransomware:

You can use openssl to ensure the correct encryption of a file. Two aes-ecb[6] examples follows, make sure you use the correct encryption function as parameter and the correct key.
The encryption password is "1234".

Encrypt: `$ openssl enc -aes-256-ecb -in test.txt -out test.txt.encrypt -k 1234`
Delete: `$ rm test.txt`
Decrypt: `$ openssl aes-256-ecb -in test.txt.encrypt -out test.txt -d -k 1234`

Tool Specifications:

Log monitor will receive the required arguments from the command line upon execution as such:
Options:

-m, Print malicious users.

-i <filename>, Print table of users that modified the file <filename> and the number of modifications.

-v <number of files>, If more than <number of files> files were created the last 20 minutes, it prints the total number, otherwise it prints a notification message that the logfile parsing was successfully completed with no suspicious results.

-e, Prints all the files that were encrypted by the ransomware.

-h, Help message.

Notes:

1) If no appropriate option was given, Log Monitor has to print the appropriate error message followed by the above help message.
2) You can store your log file on the /tmp directory (e.g. /tmp/mylog_hy457.log).
3) You need to create a Makefile to compile your library and programs.
4) You need to create a Readme with your **name**, your **login**, and a short description (1-2 lines) of your implementation.

Submit

You should submit a folder named assign_2_yourAM with your source code (shared library, log monitor, tester, ransomware), a Readme and a Makefile.

[6] aes-ecb is used only for demonstration purposes you can use whichever cryptographic algorithm you prefer.