Assignment 5

Assigned: 23/11/2020

Due: 4/12/2020 (23:55)

Implement a basic ransomware

For this assignment, you will develop a basic ransomware. Ransomware is a type of malicious software that is used to block access to the files of a user by encrypting them. Its main goal is to extort money from users in order to decrypt their files back. The main functionality of the ransomware is to encrypt a number of files and delete the original unencrypted files. 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 basic functionality that we ask for.

Step 1: Implement the ransomware

More specifically, you should develop a bash script (named "ransomware.sh") that demonstrates the creation, encryption & deletion of a number of files inside a directory in a certain amount of time. More specifically, you should (1) create/select the files to be encrypted, (2) produce the new, encrypted files, and finally (3) delete the original files. **We strongly suggest creating new text files that will be then encrypted. Do not risk encrypting and deleting files that you wouldn't want to lose or corrupt.**

Your ransomware should also be able to create a big volume of files, so *given a directory as an argument*, it should create X[1] files in that directory. *X is also given as an argument*.

You can use the OpenSSL binaries for the encryption of the files. Two *aes-ecb* examples follow, make sure you use the correct **encryption function** as parameter and the correct **key**. In the following example, the encryption password is "1234".

Encryption: $ openssl enc -aes-256-ecb -in test.txt -out test.txt.encrypt -k 1234
File deletion: $ rm test.txt
Decryption: $ openssl aes-256-ecb -in test.txt.encrypt -out test.txt -d -k 1234

For more information on the OpenSSL command line tool:

- https://linux.die.net/man/1/openssl
- https://wiki.openssl.org/index.php/Command_Line_Utilities

[1] X is an integer number that specifies the number of the files your ransomware must create.

Step 2: Use the Access Control Logging tool from Assignment 4[2]

Your ransomware will make use of the *shared library* you implemented for Assignment 4 "Access Control Logging", named "logger.so". This means that every access type (create, open or write) will be logged with an entry in the log file named "file_logging.log". Therefore, the entries must have the same format as the previous assignment:

1. **UID**: The unique user ID assigned by the system to a user (hint: see *getuid()* function).
2. **File name**: The path and name of the accessed file.
3. **Date**: The date that the action occurred.
4. **Timestamp**: The time that the action occurred.
5. **Access type**: For *file creation*, the access type is 0. For *file open*, the access type is 1. For *file write*, the access type is 2.
6. **Is-action-denied flag**: This field reports if the action was denied to the user with no access privileges. It is 1 if the action was denied to the user, or 0 otherwise.
7. **File fingerprint**: The digital fingerprint of the file the time the event occurred. This digital fingerprint is the hash value of the file contents (hint: You can use the md5 hash functions: https://www.openssl.org/docs/man1.1.0/man3/MD5_Init.html).

Events that must be logged:

1. **File creation**: Every time a user creates a file, the log file must be updated with information about the creation of the file. Make sure to modify the fopen() function in a way that the *creation* of a file can be distinguished from the *opening* of an existing file.
2. **File opening**: Every time a user tries to open a file, the log file must be updated with the corresponding file access attempt information. For this case, fopen() functions need to be intercepted and information about the user and the file access has to be collected.
3. **File modification (write)**: Every time a user tries to modify a file, the log file will be updated with the corresponding file modification attempt information. This means that fwrite() functions need to be intercepted and information about the user and the file access has to be collected. Every fopen()/fwrite() function should create a new entry in a log file.

Step 3: Detect the ransomware by enriching the Access Control Log Monitoring tool from Assignment 4

Enrich your Access Control Monitoring tool, named "acmonitor.c", from Assignment 4 to have the following extra functionality. The Access Control Monitoring tool will detect the existence of your ransomware. More specifically:

[2] This step only requires to use the logger.so shared library from your previous assignment (Assignment 4).

1. In many cases, a 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 files (at minimum) were created in the last 20 minutes.
2. A ransomware will also try to encrypt files and then discard the unencrypted version of those files. You need to find and report all the events in the log where a ransomware opened an unencrypted file and created an encrypted one. Remember that encrypted files end with the suffix ".encrypt".

Tool Specification

The enriched Access Control Log Monitoring tool (Step 3) will receive the required arguments from the command line upon execution. Specifically, you should add the (1) *-v <number of files>* and the (2) *-e* options, and keep the *-m, -i, -h* options as-is from Assignment 4.

Options

-m	Prints malicious users
-i <filename>	Prints table of users that modified the file given and the number of modifications
-v <number of files>	Prints the total number of files created in the last 20 minutes
-e	Prints all the files that were encrypted by the ransomware
-h	Help message

Notes

1. If no appropriate option was given, the Access Control Monitoring tool has to print the appropriate error message.
2. You need to create a Makefile to compile your library and programs (you must submit it with your source code) or you can use the one provided to you in the previous assignment (Assignment 4).

3. We do not provide you with a source code corpus, since you have to enrich the source code of the previous assignment (Assignment 4).
4. You need to create a README with your name, your AM and a short description (1-2 lines) of your implementation.
5. You must submit the following files: README, Makefile, logger.c, logger.so, acmonitor.c, ransomware.sh
6. You should place all these files in a folder named <AM>_assign5 and then compress it as a .zip file. For example, if your login is 2020123456 the folder should be named 2020123456_assign5 you should commit 2020123456_assign5.zip.
7. Use the tab "Σ υ ζ ή τ η σ η" in courses for questions.