Network and System Security (ET2595)

Assignment 2: System Integrity Verifier (SIV)

Ripan Kumar Dhar
BTH acronym: ridh19
P.no. 8800727-8398
Email: ripankumardhar@gmail.com

Introduction

The main goal of this assignment is to build a System Integrity Verifier (SIV) program. This SIV program will perform its verification in two steps. Firstly, an important directory needs to be declared which you want to monitor. This directory is called monitored directory. Then the SIV program will perform **initialization mode** operations. During this initialization mode, it will save all the directories and files' information into a database. Here, the information includes full_path, size, owner user and group, access right, last modification time and date and a message digest of its contents. Secondly, the SIV program will perform **verification mode** operations. During this step, the program will compare the same monitored directory's information with the previously created database file. This is how the SIV program will verify the monitored directory.

Usage

Help mode

```
python3 siv.py -h
```

Initialization mode:

```
python3 siv.py -i -D <monitored_directory> -V <verification_fileDB> -R <report_file.txt> -H <hash_function (MD-5 or SHA-1)>
```

Verification mode:

```
python3 siv.py -v -D <monitored_directory> -V <verification_fileDB> -R <report_file.txt>
```

Design and Implementation

Requirements

- The SIV program was developed and tested on Ubuntu 20.04 OS.
- Programming language used: Python version- 3.8
- List of libraries used in this program: sys, os, argparse, hashlib, pwd, json, datetime, grp

Usage of the libraries

- sys: this library was used to call some system function. For example: *sys.exit()* was used to terminate the program.
- os: this library was used to call some operating system functions. For example: *os.path.walk()* was used to traverse inside the monitored directory. *os.path.isdir()* was used to check if a particular directory exists. *os.path.isfile()* was used to check if a particular file exists.
- argparse: this library was used to take and sort out the system arguments.
- hashlib: this library was used for message digest. This library supports both MD-5 and SHA-1.
- pwd: this library was used to get the current working directory in so many places.

- json: this library was used to work with json data format.
- datetime: this library was used mainly to calculate how much time was taken for initialization or verification mode.
- grp: this library was used to get access to the group database of the unix group database.

As it was mentioned earlier in the usage section that there are 3 modes of this program. Those are: help mode, initialization mode and verification mode.

Help mode
At the beginning of this program, it will check if the help mode was called. This mode, the program will just show the usage and then it will terminate. Help mode was implemented from line 12 to line 17.

System arguments
System arguments will be taken using argparse library. From line 19 to line 37 was used for taking the arguments in the *args* variable. Then the parameters were simplified later from line 40 to line 43.

Initialization mode
Initialization mode is from line 59 to line 225. Steps of this mode are as following:

- Check if the monitored directory exists. If the monitored directory does not exist, then the program will terminate by calling *sys.exit()*.
- It will check if the given hash is supported or not. The supported hash functions are “MD-5” and “SHA-1”. If the hash is not supported, then the program will terminate by calling *sys.exit()*.
- It will check if the verification firle and/or report file are inside the monitored directory. If any of those files are inside the monitored directory, then the program will terminate by calling *sys.exit()*.
- It will check if the verification file already exists or not. If it already exists, then it will ask if you want to overwrite the file. You will have two options: “yes” or “no”. If you choose “yes”, then the file will be overwritten. If you choose “no” or invalid input, then the program will terminate by calling *sys.exit()*.
- It will check if the report file already exists or not. If it already exists, then it will ask if you want to overwrite the file. You will have two options: “yes” or “no”. If you choose “yes”, then the file will be overwritten. If you choose “no” or invalid input, then the program will terminate by calling *sys.exit()*.
- Start counting the time.
- It will start traversing inside the monitored directory by calling the function *os.path.walk()* by which it will traverse through all the directories.

- For each file of those directories, it will save its full path, size, owner user, owner group, access permission status, last modification time and message digest (MD-5 or SHA-1) of its contents.
- It will save these data in json format. Then write those information in the verification file. At the end of the verification file, it will mention which hash function was used.
- Stop counting time. Then calculates how much time was taken for this operation.
- Write a summary in the report file which includes monitored directory path, verification file path, number of directories parsed, number of files parsed and total time taken for initialization.

Verification mode

Verification mode is from line 242 to line 436. Steps of this mode are as following:

- Check if the verification file already exists or not. If the verification file does not exist, then the program will terminate by calling *sys.exit()*.
- It will check if the verification firle and/or report file are inside the monitored directory. If any of those files are inside the monitored directory, then the program will terminate by calling *sys.exit()*.
- It will check if the report file already exists or not. If it already exists, then it will ask if you want to overwrite the file. You will have two options: “yes” or “no”. If you choose “yes”, then the file will be overwritten. If you choose “no” or invalid input, then the program will terminate by calling *sys.exit()*.
- Start counting the time.
- It will load all the information from the specified verification file in json format.
- It will get which hash function was used during the initialization mode.
- It will start traversing inside the monitored directory by calling the function *os.path.walk()* by which it will traverse through all the directories.
- It will check if any (size, owner user, owner group, access permission status, last modification time, hash value) of these information were changed or not. If any value was changed then it will count it as 1 warning.
- It will also check if any file or directories were created or deleted. If so, then it will count 1 warning for each of those changes.
- Stop counting time and calculate how much time was taken for this operation.
- Write a summary of this operation in the report file which includes: monitored directory path, verification file path, number of directories parsed, number of files parsed, number of warnings and total time taken for initialization.

Limitation

The SIV program will not work properly if some files and/or directories inside the monitored directory require root permission to access.