Assignment: System Integrity Verifier (SIV)

Lokesh Kola
200007175490
lokl20@student.bth.se

1 INTRODUCTION

1.1 System Integrity Verifier (SIV)

The System integrity verifier is a tool that monitor and analysis the network or system. Alerting when vulnerable activities performed. The SIV detects the automated scripts if exists in folder and files. It also detects all kind of activities like renaming, file adding, file deleting, file size increase or decrease, and so on activities in the file directories.

1.2 Goal of this Assignment

The goal of the assignment is to implement a simple system integrity verifier (SIV) for Linux systems. This SIV need to monitor given directory and detects the file system modifications like file size change, folder added or removed, permissions, last modified date, and some other things (will discuss in further sections). The SIV need to make a report of all the modifications with the old values and new values. The SIV should be executed in two modes. They are Initialization and Verification modes.

In this assignment the goal of SIV is to mainly focus on modifications of

1. New or Removed files/directories.
2. Files/Directories with different sizes than recorded.
3. Files with different digest message than computed before while initialization.
4. Files/Directories with different user/group.
5. Files/Directories with modified access rights (permissions).
6. Files/Directories with different last modification dates.

2 DESIGN AND IMPLEMENTATION

2.1 Design

Initially the implementation design was implementing verification and initialization code in separate modules because it looks very clear and readable. The required functions were designed out of both code as the required functions were reused by both initialization and verification. The required functions are

- Directory Exists -> checks the given directory exists or not.
- Creating Files -> Checking the verification and report files if not exist creates for initialization mode and for verification mode checks both file and creates if no report exists and quit if no verification file exists.
- Get Information -> Provides the file/Directory information by using stat.
- Hash Message -> Provides the Hash digest message by taking the files as parameter.
- Verification -> Implementation of verifying the information of file/directory and comparing with the verification file and modification were wrote to report.
- Initialization -> Implementation of monitoring the given directory and making all the data in verification file.
- Main SIV -> The main code, which need to execute on terminal by giving parameters. This code is responsible for taking correct parameters for initialization and verification modes.

The design of the SIV is shown below.

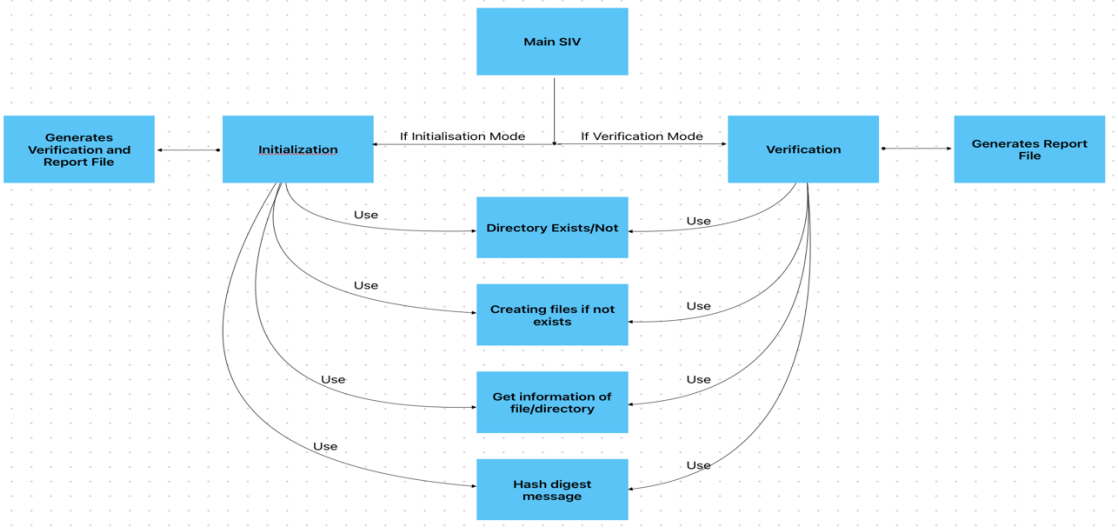

Figure 1: Design of SIV

2.2 Implementation

The selected programming language is Python as python has dictionaries, and easy to store and load to json file for each file and directory and then load them to verification file at once.

As discussed in design the implementation part is divided into Initialization, Verification and Required Functions. Initially the main siv module is executed with parameters and by depending on parameters the verification or initialization modules got executed. The brief explanation is as follows.

2.2.1 Initialization Implementation

Initially the variables are declared for count of file/directories. For Initialization the parameters were looks as follow

./siv -i –D <monitored_directory> -V <verification_file> -R <report_file> -H<hash_function>

The function gets all the parameters namely monitored directory, verification file, report file and hash function (md5 or sha1). Then the monitored directory is initially checked if exists or not. If not exists, the program will exit by showing relevant message.

Then the verification file and report file were checked if exists or not. If not exists creates new files and wrote to newly created file. And checked that both files were out of monitored directory and if inside monitored directory the program will exit by giving a relevant message that file should be outside of monitored directory. By now all the environment is established and good to go for checking the file and directories inside the monitored directory and add information to verification file.

The start time is recorded and stored in a variable and implemented a nested for loop for iterating in between the files and directories and its files as well. The pseudo code is as follows:

```
for subdirs, dirs, files in os.walk(Monitered_Directory):
        for filename in files:
        //getting information of files by using get_information function and adding the information to python dictionary and the dictonary is dumped to json file.
        for dirname in dirs:
        //getting information of directories by using get_information function and adding the information to python dictionary and the dictonary is dumped to json file.
        Finally the verification file is written with the json file.
```

The first "for loop" is doing the operations on all files. The second "for loop" will be working on all the directories. Initially If we consider one file then the file path is stored in a variable and then the size, owner, group, permissions, last modified date and hash message of the file is stored to variables. Then dumping the data of file to json (data structure).

Similarly, if we consider one directory then the file path is stored in a variable and then the size, owner, group, permissions, and last modified date of the file is stored to variables. Then dumping the data of file to json file (data structure).

After writing the details of all files and directories to json file then those details are transferred to Verification File.

In the middle of loop the count of both files and directories are stored and then used for Report File. The path of directory and verification file, count of directories and files, time for Initialization will be stored in Report File

2.2.2 Verification Implementation

Initially the variables are declared for count of file/directories. For Initialization the parameters were looks as follow

siv -v –D <monitored_directory> -V <verification_file> -R <report_file>

The function gets all the parameters namely monitored directory, verification file, report file. Then the monitored directory is initially checked if exists or not. If not exists, the program will exit by showing relevant message.

Then the verification file and report file were checked if exists or not. If not exists creates new files and wrote to newly created file. And checked that both files were out of monitored directory and if inside monitored directory the program will exit by giving a relevant message that file should be outside of monitored directory. By now all the environment is established and good to go for checking the file and directories inside the monitored directory and verify information with verification file.

The pseudo code is as follows:

Reading the verification file and storing into a json file for comparison. Then check the current information of the files/directories by using the below nested loop.

```
for subdirs, dirs, files in os.walk(Monitered_Directory):
        for filename in files:
        //getting information of current files by using get_information function and comparing the information with json file and if modifications detected write to file (both old and new values).
        for dirname in dirs:
        //getting information of current directories by using get_information function and comparing the information with json file and if modifications detected write to file (both old and new values).
        Finally the all changes made were reported to Report file
```

After Initialization, the details of files and subdirectories were stored to the verification file. The created verification file is used to verify if any changes detected. Like initialization we use "nested for loop" for iteration and gather the same details of the files and directories again and comparing the details with verification file. If any changes detected, then write the change detected and what change to the report file.

For comparison json is used. Initially verification file is loaded in to json file and then for every file comparing the key values like file size, file owner, ...

and checking the current details, if doesn't match write to report file. The same procedure is repeated for the directories and updated in report file if changes

detected. The detailed information will be added to report file for changes. Finally, the path of directory and verification file, count of directories and files,

time for Verification will be stored in Report File.

The verification file looks as follow


```
{
    "lokl/vDB": {
        "File Path": "lokl/vDB",
        "File Size": 5118,
        "File Owner": "student",
        "File Group": "student",
        "File Permissions": "-rwxrwxr-x",
        "File Last Modified Date": "Wed Jan  4 17:57:10 2023",
        "Message": "ef912fc65b673ce686e929127e629d7a8bdb2d3d",
        "Hash Function": "sha1"
    },
    "lokl/verify.txt": {
        "File Path": "lokl/verify.txt",
        "File Size": 757,
        "File Owner": "student",
        "File Group": "student",
        "File Permissions": "-rwxrwxrwx",
        "File Last Modified Date": "Wed Jan  4 16:16:42 2023",
        "Message": "fa7d333e0afd0b3c204f23a4f275569d2a249294",
        "Hash Function": "sha1"
    },
    "lokl/data": {
        "Directory Path": "lokl/data",
        "Directory Size": 4096,
        "Directory Owner": "student",
        "Directory Group": "student",
        "Directory Permissions": "drwxrwxr-x",
        "Directory Last Modification Date": "Wed Jan  4 16:16:42 2023"
    },
    "lokl/royal": {
        "Directory Path": "lokl/royal",
        "Directory Size": 4096,
        "Directory Owner": "student",
        "Directory Group": "student",
        "Directory Permissions": "drwxrwxr-x",
        "Directory Last Modification Date": "Wed Jan  4 17:54:32 2023"
    },
    "lokl/lokesh": {
        "Directory Path": "lokl/lokesh",
        "Directory Size": 4096,
```


The report file looks as follow:

```
Change Detected :lokl/lokesh has different last modified date
 Old Last Modified : Wed Jan  4 17:33:04 2023
 New Last Modified : Wed Jan  4 18:01:27 2023

Change Detected : The file lokl/lokesh/vDB size has been changed
 Old Size : 1705
 New Size : 6822

Change Detected : The File lokl/lokesh/vDB has different accesss rights
 Old Permissions : -rwxrwxr-x
 New Permissions : -rwxrwxrwx

Change Detected :The File lokl/lokesh/vDB has different last modified Date or Time
 Old Last Modified : Wed Jan  4 16:16:42 2023
 New Last Modified : Wed Jan  4 18:01:27 2023

Change Detected :lokl/lokesh/vDB has different message digest
 Old Message Digest : e9396053487c97ce9e88e3e94bb348fb106c0722
 New Digest Message : 692e73b48ba263173d99248dc0925b054d161a8a

{
    "Directory Path": "lokl",
    "Verification File Path": "/home/student/SIV/verification",
    "Number of Directories": 4,
    "Number of Files": 7,
    "Time to finish the verification mode": 3.0012130737304688,
    "Number of changes detected": 5
}
```

2.2.3 Required Functions Implementation

In this Module the required functions were defined, the functions were isdirexists(checking if directory exists or not), Creating Files(creates verification and report files if not exists),hash_message(getting hexdigest of given file and only md5 or sha1 by using sha1 and md5 modules from python.) and get_informantion(gives the information about the given file by using stat in python)

3 USAGE

siv <-i|-v|-h> –D <monitored_directory> -V <verification_file> -R <report_file> -H <hash_function>
for Initialization:
siv <-i> –D <monitored_directory> -V <verification_file> -R <report_file> -H <hash_function>
For Verification:
siv <-v> –D <monitored_directory> -V <verification_file> -R <report_file>