System Integrity Verifier

NASIRALI KOVVURU, Blekinge Institute of Technology

1 INTRODUCTION

The following assignment System Integrity Verifier can be used by network analysts and Managers to know the vulnerable activities, if any happened in their systems. It is an automated script which detects the files, folders, subfolder changes, if any happens. There are many applications in big Multinational Companies like apple, google etc.

In my solution, I have implemented SIV in **Python (3.5.2)** language, the reason for choosing this language is, I am familiar with it and there are many known modules to me those can help me to work on.

2 Design and Implementation

There are mainly six type of changes that can occur to a record or file, those are 1) content changing, 2) file type change, 3) file access permission change, 4) file group change, 5) file size change, 6) file owner change.

To detect all above change, in my solution I make use of Hashing algorithms to calculate the checksum of the files based on the contents as well as permissions and all the above types of information, this checksum is unique, that can sense very small change in the file even though with completely different checksum value. Such all the above changes of the file can be detected in my solution.

In my solution, I make use of SHA-1 AND MD-5 algorithms to calculate the checksums of each file.

There are two phases in this implementation those are 1) Integrity 2) verification

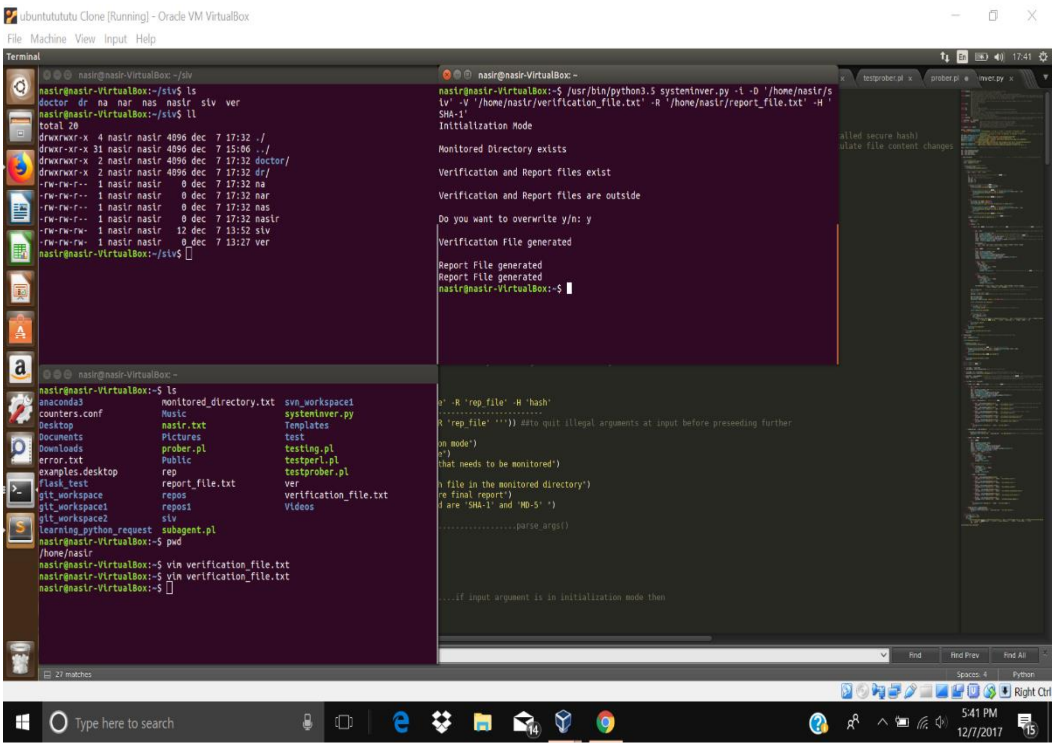
2.1 Integrity

In this phase, we will walk through all directories and files in our specified directory and according to the contents of those files, each file is given by a hash value (in my case I have created all empty files resulted in same hash value to all files) so, the **hash value is solely dependent on the content and permissions of the file**.

To implement the above logic, I make use of argparse, hashlib, json, os, textwrap, datetime modules of the python language. This solution first searches for all the directories and sub-directories in the Folder that we have specified and add their entry in a file that we have specified from the command prompt in Json object notation format. Then it will search for all the files in the folder that we have specified from the command line and depending on the content in the files, each file is given by one hash value, and all those values are taken to a file (in my case I took all those results to verification.txt file). Such our solution prepares a verification file which has all the details (hash value, access permission, group file belongs, size of file, recently open time, user) of each file and folder of our directory.

Then based on the above result, a report file is generated that consists of information about how many directories it has parsed, how many files it has parsed and how much time does it takes to parse. This two files information can be seen by the manager or network analyst to know all the activities in their file system of interest.

Such in this integrity phase we can know how many files, folders, sub folders are there in our interested directory, and for integrity purpose we can know the hash value of each file, and can know if any intrusion activity happened or not by basing the result our script returns.


ubuntututu Clone [Running] - Oracle VM VirtualBox
File Machine View Input Help
Terminal
nasir@nasir-VirtualBox: ~/siv
nasir@nasir-VirtualBox: ~
27 matches
Find
Find Prev
Find All
Spaces: 4
Python
Type here to search
5:41 PM
12/7/2017

```
nasir@nasir-VirtualBox:~/siv$ ls
doctor  dr  na  nar  nas  nasir  siv  ver
nasir@nasir-VirtualBox:~/siv$ ll
total 20
drwxrwxr-x  4 nasir nasir 4096 dec  7 17:32 ./
drwxr-xr-x 31 nasir nasir 4096 dec  7 15:06 ../
drwxrwxr-x  2 nasir nasir 4096 dec  7 17:32 doctor/
drwxrwxr-x  2 nasir nasir 4096 dec  7 17:32 dr/
-rw-rw-r--  1 nasir nasir    0 dec  7 17:32 na
-rw-rw-r--  1 nasir nasir    0 dec  7 17:32 nar
-rw-rw-r--  1 nasir nasir    0 dec  7 17:32 nas
-rw-rw-r--  1 nasir nasir    0 dec  7 17:32 nasir
-rw-rw-rw-  1 nasir nasir   12 dec  7 13:52 siv
-rw-rw-rw-  1 nasir nasir    0 dec  7 13:27 ver
nasir@nasir-VirtualBox:~/siv$
```

```
nasir@nasir-VirtualBox:~$ /usr/bin/python3.5 systeminver.py -i -D '/home/nasir/s
iv' -V '/home/nasir/verification_file.txt' -R '/home/nasir/report_file.txt' -H '
SHA-1'
Initialization Mode

Monitored Directory exists

Verification and Report files exist

Verification and Report files are outside

Do you want to overwrite y/n: y

Verification File generated

Report File generated
Report File generated
nasir@nasir-VirtualBox:~$
```

```
nasir@nasir-VirtualBox:~$ ls
anaconda3                monitored_directory.txt  svn_workspace1
counters.conf            Music                    systeminver.py
Desktop                  nasir.txt                Templates
Documents                Pictures                 test
Downloads                prober.pl                testing.pl
error.txt                Public                   testperl.pl
examples.desktop         rep                      testprober.pl
flask_test               report_file.txt          ver
git_workspace            repos                    verification_file.txt
git_workspace1           repos1                   Videos
git_workspace2           siv
learning_python_request  subagent.pl
nasir@nasir-VirtualBox:~$ pwd
/home/nasir
nasir@nasir-VirtualBox:~$ vim verification_file.txt
nasir@nasir-VirtualBox:~$ vim verification_file.txt
nasir@nasir-VirtualBox:~$
```

```
e' -R 'rep_file' -H 'hash'
------------------------
R 'rep_file' '''))  ##to quit illegal arguments at input before preseeding further

on mode")
e")
that needs to be monitored")

h file in the monitored directory")
re final report")
d are 'SHA-1' and 'MD-5' ")

..................parse_args()

....if input argument is in initialization mode then
```

Screenshot1: Execution of solution script in integrity mode by using SHA-1 hashing algorithm.

In the Screenshot1, the following command is used

$/usr/bin/python3.5 systeminver.py -I -D ‘/home/nasir/siv’ -V ‘/home/nasir/verification_file.txt’ -R ‘/home/nasir/report_file.txt’ -H ‘SHA-1’

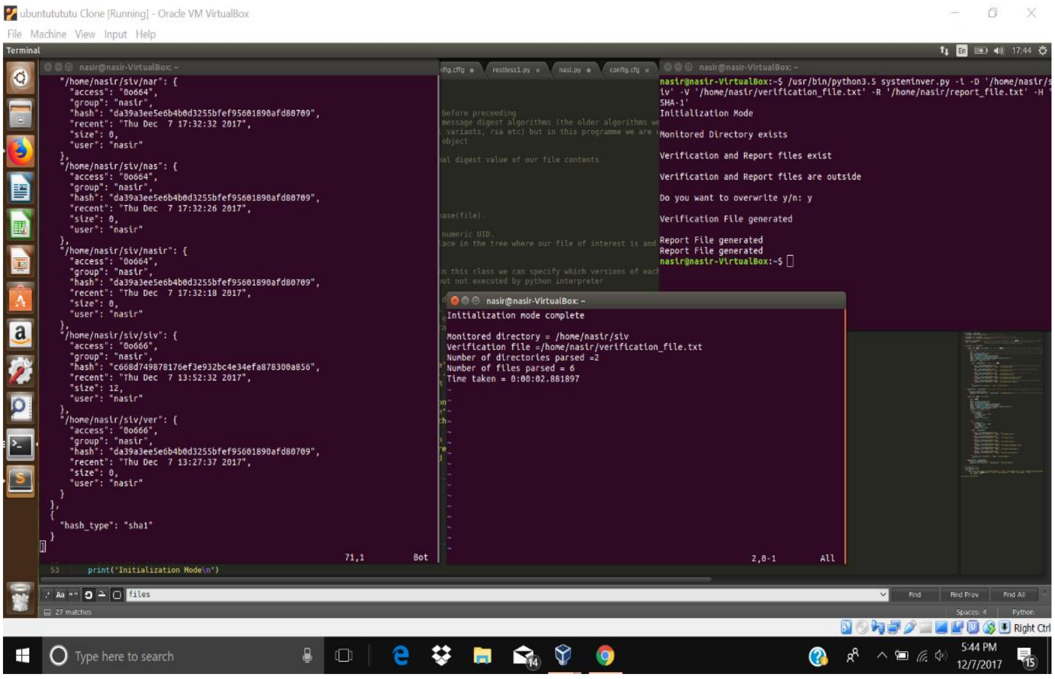
The above command tells firstly the interpreter which we need to use to execute the script (in my case that is python 3.5 interpreter), secondly the name of the script which we want to execute (in my case that is systeminver.py), thirdly -I represents that it is in Integrity mode, fourthly -D represents the directory where our important files are presented (in my case, I have given the path to that directory), fifthly -V represents where we have our verification file (Note: it should be out from the directory of our interest), sixthly -R represents where we have our report file.

Note: all the above options (-I, -D, -V, -R should be included in our programme such that those arguments can take as input and process them).

After hitting enter after that command, it asks for the permission to overwrite the previous observations in that file, if you press y then our programme executes and corresponding verification and report files are generated, if you press n the programme will terminates. Screenshot1 show that things clearly.

Then two files are created and are filled with the readings of the script as shown in the Screenshot2.

Similarly, in screenshots 3 and 4, we can observe the same behaviour in MD-5 hashing algorithm. Note: Both hash results are different.

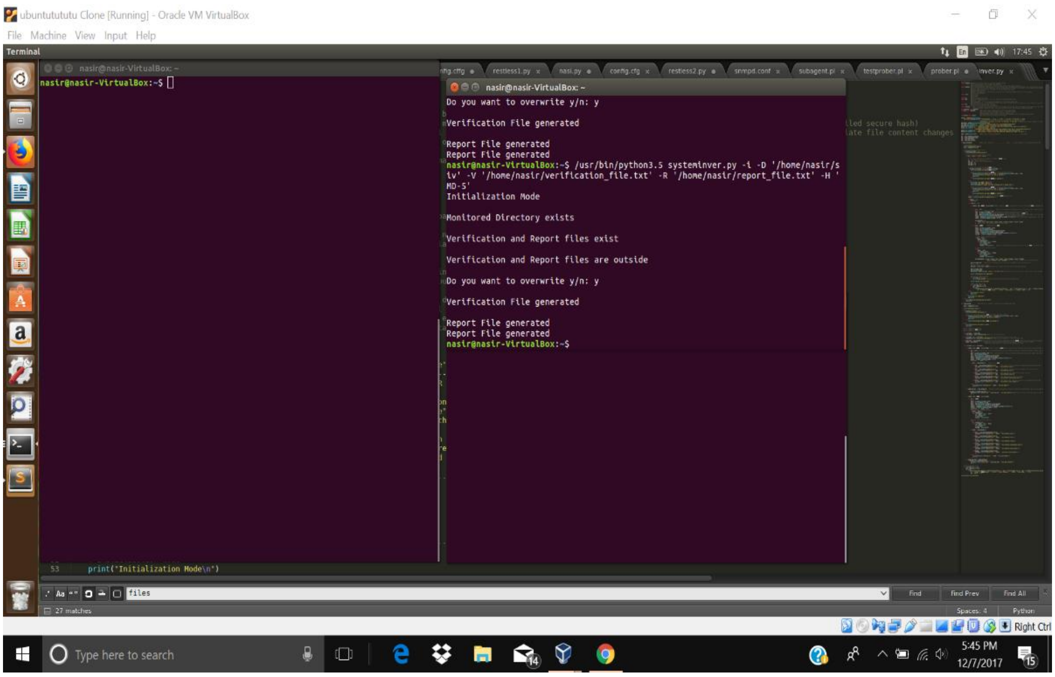


```
    "/home/nasir/siv/nar": {
      "access": "0o664",
      "group": "nasir",
      "hash": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
      "recent": "Thu Dec  7 17:32:32 2017",
      "size": 0,
      "user": "nasir"
    },
    "/home/nasir/siv/nas": {
      "access": "0o664",
      "group": "nasir",
      "hash": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
      "recent": "Thu Dec  7 17:32:26 2017",
      "size": 0,
      "user": "nasir"
    },
    "/home/nasir/siv/nasir": {
      "access": "0o664",
      "group": "nasir",
      "hash": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
      "recent": "Thu Dec  7 17:32:18 2017",
      "size": 0,
      "user": "nasir"
    },
    "/home/nasir/siv/siv": {
      "access": "0o666",
      "group": "nasir",
      "hash": "c668d749878176ef3e932bc4e34efa878300a856",
      "recent": "Thu Dec  7 13:52:32 2017",
      "size": 12,
      "user": "nasir"
    },
    "/home/nasir/siv/ver": {
      "access": "0o666",
      "group": "nasir",
      "hash": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
      "recent": "Thu Dec  7 13:27:37 2017",
      "size": 0,
      "user": "nasir"
    }
  },
  {
    "hash_type": "sha1"
  }
]
                                                          71,1          Bot
```

```
nasir@nasir-VirtualBox:~$ /usr/bin/python3.5 systeminver.py -i -D '/home/nasir/s
iv' -V '/home/nasir/verification_file.txt' -R '/home/nasir/report_file.txt' -H '
SHA-1'
Initialization Mode

Monitored Directory exists

Verification and Report files exist

Verification and Report files are outside

Do you want to overwrite y/n: y

Verification File generated

Report File generated
Report File generated
nasir@nasir-VirtualBox:~$
```

```
Initialization mode complete

Monitored directory = /home/nasir/siv
Verification file =/home/nasir/verification_file.txt
Number of directories parsed =2
Number of files parsed = 6
Time taken = 0:00:02.881897
                                                    2,0-1         All
```

Screenshot2: The output files generated by the script in Integrity mode by using SHA-1 hash.

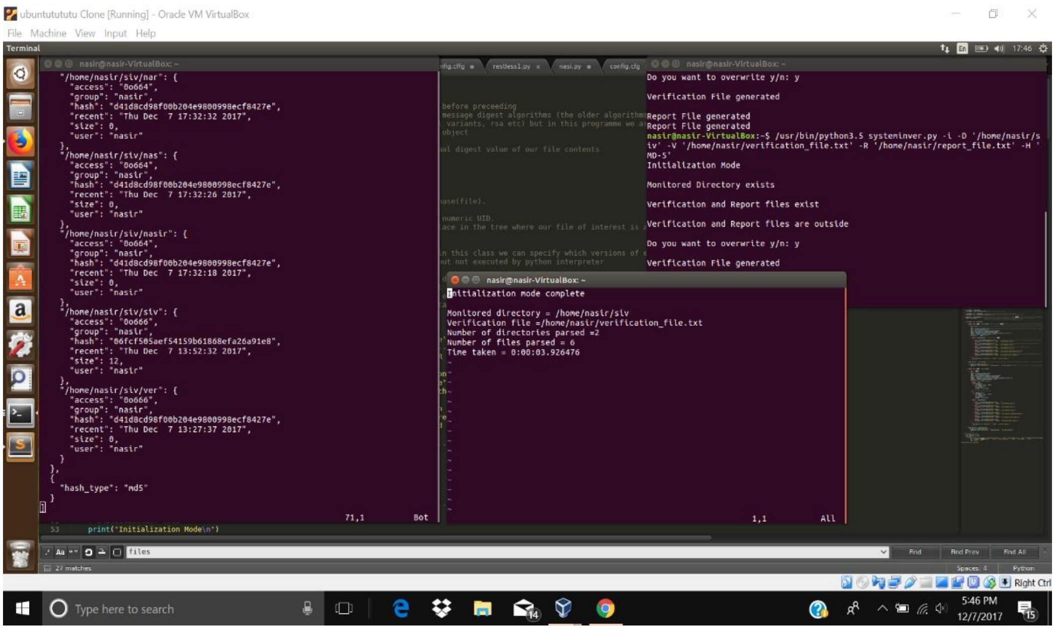


```
nasir@nasir-VirtualBox:~$
```

```
Do you want to overwrite y/n: y

Verification File generated

Report File generated
Report File generated
nasir@nasir-VirtualBox:~$ /usr/bin/python3.5 systeminver.py -i -D '/home/nasir/s
iv' -V '/home/nasir/verification_file.txt' -R '/home/nasir/report_file.txt' -H '
MD-5'
Initialization Mode

Monitored Directory exists

Verification and Report files exist

Verification and Report files are outside

Do you want to overwrite y/n: y

Verification File generated

Report File generated
Report File generated
nasir@nasir-VirtualBox:~$
```

```
    print("Initialization Mode\n")
```

Screenshot3: Execution of solution script in integrity mode by using MD-5 hashing algorithm.



```
    "/home/nasir/siv/nar": {
      "access": "0o664",
      "group": "nasir",
      "hash": "d41d8cd98f00b204e9800998ecf8427e",
      "recent": "Thu Dec  7 17:32:32 2017",
      "size": 0,
      "user": "nasir"
    },
    "/home/nasir/siv/nas": {
      "access": "0o664",
      "group": "nasir",
      "hash": "d41d8cd98f00b204e9800998ecf8427e",
      "recent": "Thu Dec  7 17:32:26 2017",
      "size": 0,
      "user": "nasir"
    },
    "/home/nasir/siv/nasir": {
      "access": "0o664",
      "group": "nasir",
      "hash": "d41d8cd98f00b204e9800998ecf8427e",
      "recent": "Thu Dec  7 17:32:18 2017",
      "size": 0,
      "user": "nasir"
    },
    "/home/nasir/siv/siv": {
      "access": "0o666",
      "group": "nasir",
      "hash": "06fcf505aef54159b61868efa26a91e8",
      "recent": "Thu Dec  7 13:52:32 2017",
      "size": 12,
      "user": "nasir"
    },
    "/home/nasir/siv/ver": {
      "access": "0o666",
      "group": "nasir",
      "hash": "d41d8cd98f00b204e9800998ecf8427e",
      "recent": "Thu Dec  7 13:27:37 2017",
      "size": 0,
      "user": "nasir"
    }
  },
  {
    "hash_type": "md5"
  }
]
                                                  71,1          Bot
```

```
Do you want to overwrite y/n: y

Verification File generated

Report File generated
Report File generated
nasir@nasir-VirtualBox:~$ /usr/bin/python3.5 systeminver.py -i -D '/home/nasir/s
iv' -V '/home/nasir/verification_file.txt' -R '/home/nasir/report_file.txt' -H '
MD-5'
Initialization Mode

Monitored Directory exists

Verification and Report files exist

Verification and Report files are outside

Do you want to overwrite y/n: y

Verification File generated
```

```
Initialization mode complete

Monitored directory = /home/nasir/siv
Verification file =/home/nasir/verification_file.txt
Number of directories parsed =2
Number of files parsed = 6
Time taken = 0:00:03.926476
```

Screenshot4: The output files generated by the script in Integrity mode by using MD-5 hash.

Logic behind the implementation:

1) The very first step is to take the command line arguments to our script, and to specify each flag and its corresponding mode. Such script can know what are all the folders and files, hashing algorithm it should make use of.

2) Next, we need to check whether the files our script write are outside our interested Directory or not. If they are inside print a notification on the terminal.

3) Next, our solution pop's up a choice in our terminal (do you want to overwrite?), if you press y to accept then all the previously saved things in the verification file will be overwrites with the new results which are obtained from this script. If you press n then programme should terminate.

4) Next, if your choice is y then we should write a looping statement to walk through all the directories and files of the target Directory; mean while we should initiate some other variables which can uniquely count directory count, file count, subdirectory count.(we know that our operating system first checks the subdirectory count, then checks the directory count, then checks the file count) in the similar way we should name our variables in the loop to initialize. Those variables maynot be filled in json object notation.

5) Now, from the above directory, subdirectory, file count variables we will write sub loops, here each subloop will do walk on their corresponding files and folders and know their corresponding parameters (path, size, user, group, recent, access, hash (only for files)) and the obtained values should be output to verification.txt file which we specify at -V flag.

6) And all the outputs obtained from the above steps should be send to report file(report.txt) which is specified at -R flag.

2.2 Verification

In the command line if you give the command -v instead of -i, then the script runs the verification mode logic.

$/usr/bin/python3.5 systeminver.py -v -D '/home/nasir/siv' -V '/home/nasir/verification_file.txt' -R '/home/nasir/report_file.txt'

Here, hash function is not needed to specify, since hash values of each file are calculated in the earlier step(integrity mode) itself. That means, here we are making use of those hash values which are previously calculated in the past mode (integrity mode) i.e.. verification_file.txt, and we will verify the files are changed or not. And the result is written into report_file.txt file.

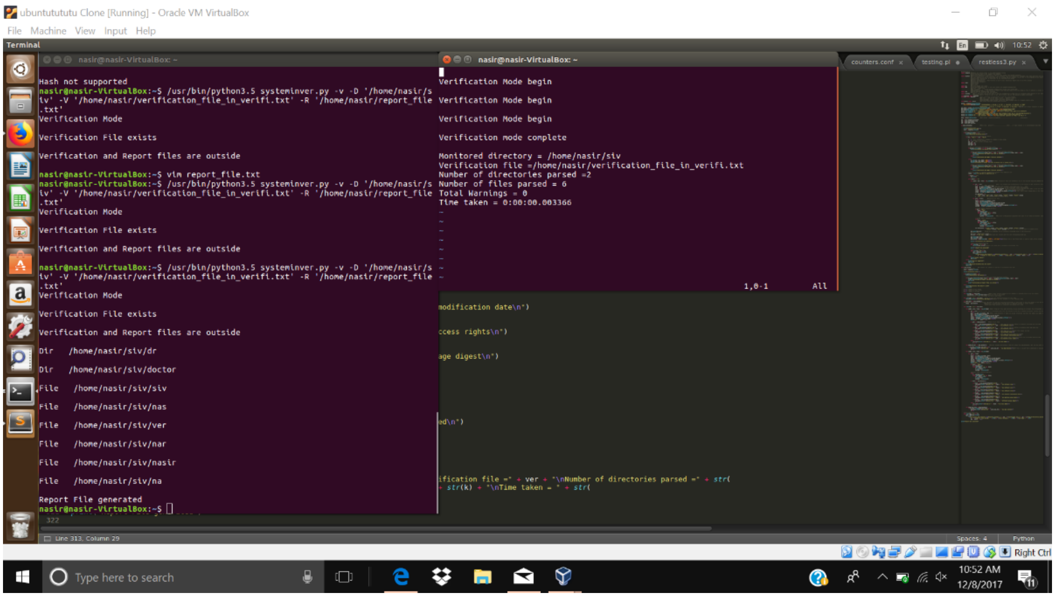
Verification mode verifies whether the files are modified or not. If so, it will tell how many changes made to the files and how many files are affected in our specified directory.

The following Screenshots5 depicts its behaviour.

help mode: The flag -h can be used to initiate this mode. In our code, we need to specify the corresponding logic at the input arguments section.

Help mode logic is very useful to the new users who don’t know about our solution, these mode gives a brief overview of each command line argument (-V, -i, -H, -D, -v) and in which preceding we need to use them.

The following Screenshot6 dipicts its behaviour.

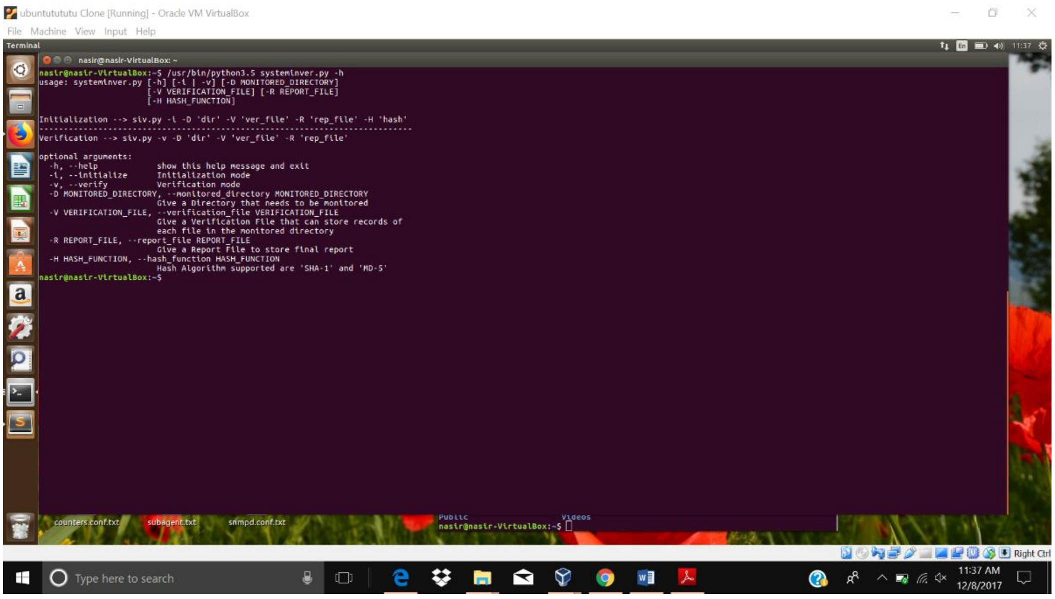


```
Hash not supported
nasir@nasir-VirtualBox:~$ /usr/bin/python3.5 systeminver.py -v -D '/home/nasir/s
iv' -V '/home/nasir/verification_file_in_verifi.txt' -R '/home/nasir/report_file
.txt'
Verification Mode

Verification File exists

Verification and Report files are outside

nasir@nasir-VirtualBox:~$ vim report_file.txt
nasir@nasir-VirtualBox:~$ /usr/bin/python3.5 systeminver.py -v -D '/home/nasir/s
iv' -V '/home/nasir/verification_file_in_verifi.txt' -R '/home/nasir/report_file
.txt'
Verification Mode

Verification File exists

Verification and Report files are outside

nasir@nasir-VirtualBox:~$ /usr/bin/python3.5 systeminver.py -v -D '/home/nasir/s
iv' -V '/home/nasir/verification_file_in_verifi.txt' -R '/home/nasir/report_file
.txt'
Verification Mode

Verification File exists

Verification and Report files are outside

Dir   /home/nasir/siv/dr

Dir   /home/nasir/siv/doctor

File   /home/nasir/siv/siv

File   /home/nasir/siv/nas

File   /home/nasir/siv/ver

File   /home/nasir/siv/nar

File   /home/nasir/siv/nasir

File   /home/nasir/siv/na

Report File generated
nasir@nasir-VirtualBox:~$ 
```

```
Verification Mode begin

Verification Mode begin

Verification Mode begin

Verification mode complete

Monitored directory = /home/nasir/siv
Verification file =/home/nasir/verification_file_in_verifi.txt
Number of directories parsed =2
Number of files parsed = 6
Total Warnings = 0
Time taken = 0:00:00.003366
~
~
~
~
~
~
~
~
                                                      1,0-1         All
```

```
modification date\n")

ccess rights\n")

age digest\n")

ed\n")

ification file =" + ver + "\nNumber of directories parsed =" + str(
+ str(k) + "\nTime taken = " + str(
```

Screenshot5: The output file generated by the solution script in verification mode.



```
nasir@nasir-VirtualBox:~$ /usr/bin/python3.5 systeminver.py -h
usage: systeminver.py [-h] [-i | -v] [-D MONITORED_DIRECTORY]
                      [-V VERIFICATION_FILE] [-R REPORT_FILE]
                      [-H HASH_FUNCTION]

Initialization --> siv.py -i -D 'dir' -V 'ver_file' -R 'rep_file' -H 'hash'
----------------------------------------------------------------------------
Verification --> siv.py -v -D 'dir' -V 'ver_file' -R 'rep_file'

optional arguments:
  -h, --help            show this help message and exit
  -i, --initialize      Initialization mode
  -v, --verify          Verification mode
  -D MONITORED_DIRECTORY, --monitored_directory MONITORED_DIRECTORY
                        Give a Directory that needs to be monitored
  -V VERIFICATION_FILE, --verification_file VERIFICATION_FILE
                        Give a Verification File that can store records of
                        each file in the monitored directory
  -R REPORT_FILE, --report_file REPORT_FILE
                        Give a Report File to store final report
  -H HASH_FUNCTION, --hash_function HASH_FUNCTION
                        Hash Algorithm supported are 'SHA-1' and 'MD-5'
nasir@nasir-VirtualBox:~$
```

Screenshot6: The help option working in script.

Logic behind the implementation:

1) It is very important to consider the output file that is take over from the Integrity mode at -V flag, which is going to validate by our script.

2) Firstly, we need to take the Json object that is present in the verification file which is initialized at -V flag into a variable. Now we will use that variable (here variable is a list i.e., we have saved that file in a list form meanwhile list of dict objects, it is very important to know)

3) Next, we are going to compare size, user, group, date modified of each file with respect to the values that are in the verification file. If they does not matches then our solution should need to send notification to the report file. Such in our solution we need to write different looping statements to different things (files and folders). And the corresponding notifications (different size, different user, different group, different modification date, different access right) should be send to report file. Similarly to directories and subdirectories (use in your loop say nasir[0], nasir[1], nasir[2] for knowing json contents of subfolders, folders, files).

4) And such generate a report file.

To implement the above operations, I have used python programming language 3.5 version 3.5.2 interpreter. The reason for using python is, it is user friendly language, more over I know some important modules in it and those are enough to develop this programme.

3 Usage

The use of this tool is to detect the file system modifications occurring within a directory tree for a Linux system. This tool outputs the statistics and warnings about the changes in files that are modified to report file. The tool verifies the integrity of a monitored directory.

Initialization Mode:/usr/bin/python3.5 systeminveri.py -i -D ‘path of the directory want to monitor’ -V ‘path the verification_file’ -R ‘path of the report_file’ -H ‘hash’

Verification Mode:/usr/bin/python3.5 systeminveri.py -v -D ‘path of the directory want to monitor’ -V ‘path of the verification_file’ -R ‘path of the report_file’

Help mode: /usr/bin/python3.5 systeminveri.py -h

4. Limitations

This solution only detects the changes, but it cannot restrict the intruder, if it can tell who changed this file or folder like that. In integrity mode our solution doesn’t give hash value of the total folder, like if we know the hash value change of a particular directory then we can go to the corresponding files only to see the changes. In verification mode, the report file gives how many files are changed but it not gives the clear description or path of those files. If such intelligence is also added to this solution then it may be compatible enough to handle Realtime desktop user files monitoring activities.

The files with (,) symbol will not be accepted by this solution. Remaining all symbols are taken by the compiler as file names.