# FINAL PROJECT REPORT

CENG3544, COMPUTER NETWORK SECURITY

Ahmed Ibrahim
ah1.6.1998@gmail.com
160709057

Tuesday 2nd June, 2020

## 1 Introduction

In the first part of this report, I am going to explain my thoughts about Malware, Antivirus and also the reason why Linux is preferred in SOC as SIEM. In the second part, I am going to explain the script that I wrote to collect logs of unusual Activities in the operating system.

## 2 Assignments

### 2.1 Assignment I (”Brief explanation of Malware”)

There are many types of Malware such as Virus, Trojan Horse, Worm, Ransomware, Spyware and etc. The main difference between Virus and Worm is that Most of Virus need human help to achieve their malicious goal. But, Worms are independent scripts that they replicate themselves on their own. Trojans can't replicate themselves unless they are associated with worms. Spyware is mainly used to monitor your browsing activity and take a copy of your keyboard click such as ”keylogger” program. Ransomware encrypts system files, then notify the system-user to pay some money especially by Bitcoin to decrypt the files back. The last discovered ransomware program is ”Ranger Locker” which installs ”Oracle VirtualBox” then installs a virtual OS on it. Then, it establishes a file sharing protocol between the guest(window xp) and the Host. Finally, It encrypts the files on the victim device.

### 2.2 Assignment II (”Brief explanation of Antivirus”)

Antivirus combines many of the detection scripts that discover Malware. Antivirus has the right to modify the operating system files and some of them monitor your internet connection. Free Antivirus is not recommended because it isn't frequently updated by the new malware signatures. Malware signatures is logically like the hash. The antivirus will use them to be compared with the file signature. Some Antivirus has Firewall features that will be used to track and predict malicious Access to your Computer.

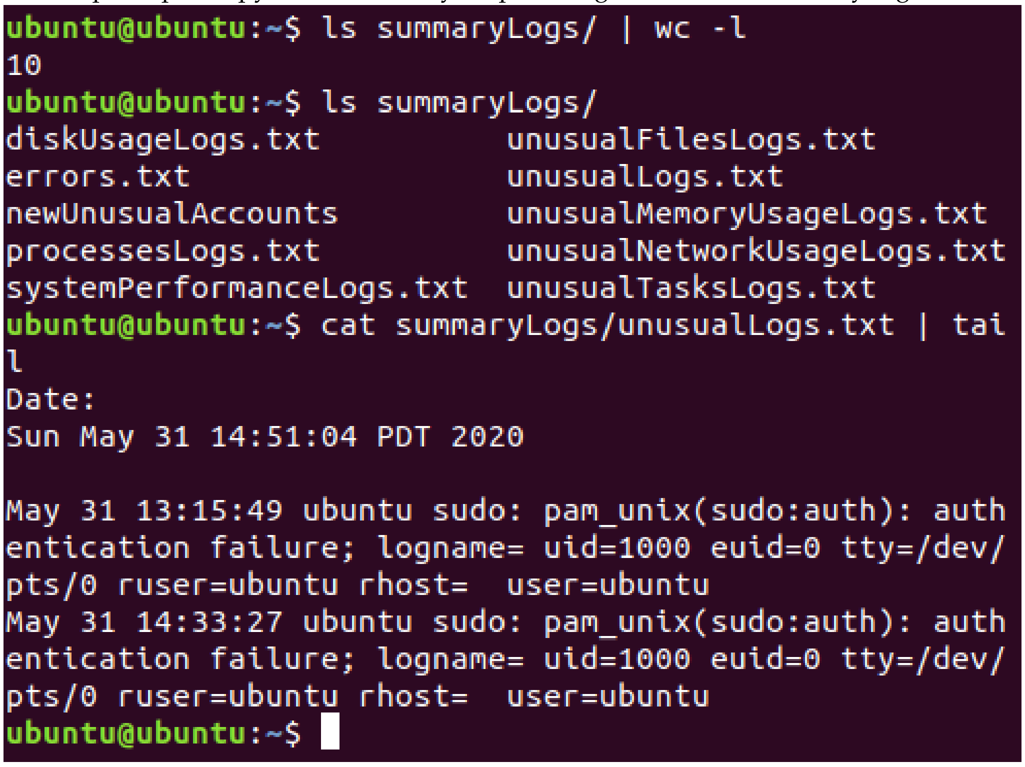
### 2.3 Assignment III ("Why is Linux preferred in SOC as SIEM?")

Linux OS is preferred in Security Operations Centers because it is an open-source program that gives the user the ability to modify it according to his needs. Linux distributions were developed mainly to be accessed by the CLI. Users can access the Linux OS remotely because CLI consumes less system resources than GUI. Unlike other operating systems, the system root user has the access right to modify the OS and its Scripts even the low-level functions.

## 3 Assignment IV ("Script")

### 3.1 Keeping a copy of Summary reports of the system to files.

The script keeps a copy of the summary output to log files under "summaryLogs" directory.



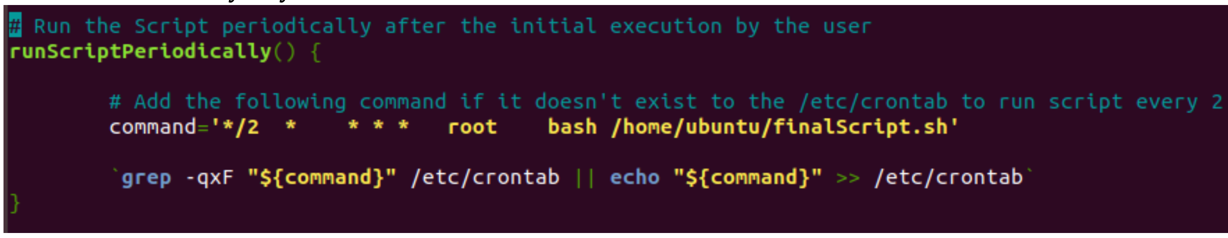
```
ubuntu@ubuntu:~$ ls summaryLogs/ | wc -l
10
ubuntu@ubuntu:~$ ls summaryLogs/
diskUsageLogs.txt          unusualFilesLogs.txt
errors.txt                 unusualLogs.txt
newUnusualAccounts         unusualMemoryUsageLogs.txt
processesLogs.txt          unusualNetworkUsageLogs.txt
systemPerformanceLogs.txt  unusualTasksLogs.txt
ubuntu@ubuntu:~$ cat summaryLogs/unusualLogs.txt | tai
l
Date:
Sun May 31 14:51:04 PDT 2020

May 31 13:15:49 ubuntu sudo: pam_unix(sudo:auth): auth
entication failure; logname= uid=1000 euid=0 tty=/dev/
pts/0 ruser=ubuntu rhost=  user=ubuntu
May 31 14:33:27 ubuntu sudo: pam_unix(sudo:auth): auth
entication failure; logname= uid=1000 euid=0 tty=/dev/
pts/0 ruser=ubuntu rhost=  user=ubuntu
ubuntu@ubuntu:~$
```

errors.txt —- keeps errors of the script
logsHistory.txt —- keeps only script-run dates

## 3.2 Running the script Periodically.

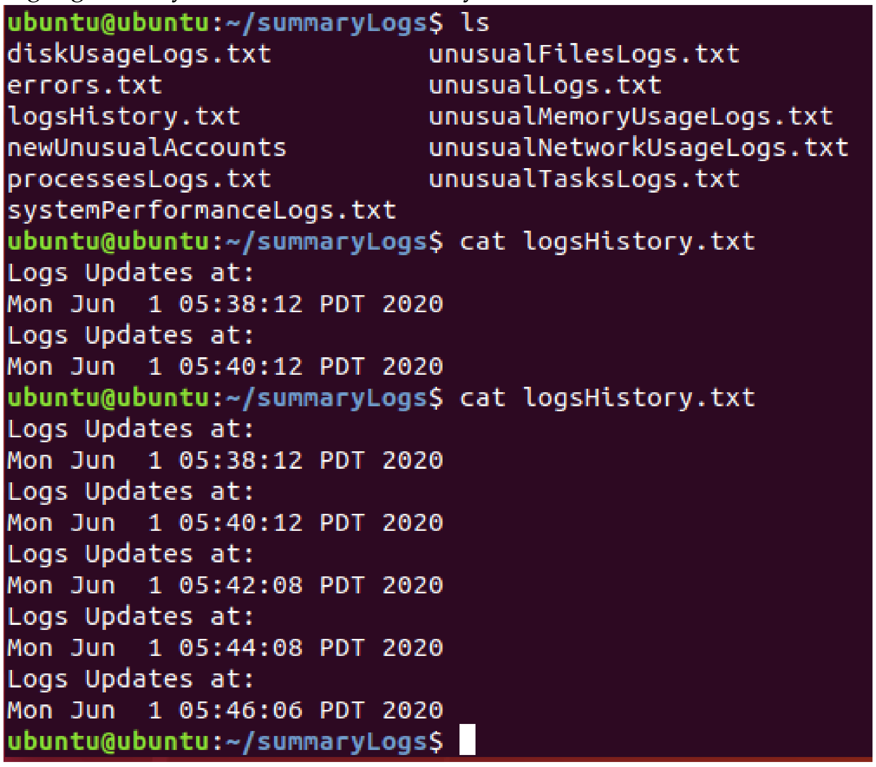
- a. Run script periodically in background after the initial execution of the code by the user

- Example: the following line runs the script every two minutes.
  ***/2 * * * * root /usr/bin/bash /home/ubuntu/finalScript.sh**
  **(*/2) means every two minutes**
  **second(*) means every hour**
  **third(*) means every day of month**
  **fourth(*) means every month**
  **fifth(*) means every day of week**



```
# Run the Script periodically after the initial execution by the user
runScriptPeriodically() {

        # Add the following command if it doesn't exist to the /etc/crontab to run script every 2
        command='*/2  *    * * *   root    bash /home/ubuntu/finalScript.sh'

        `grep -qxF "${command}" /etc/crontab || echo "${command}" >> /etc/crontab`
}
```

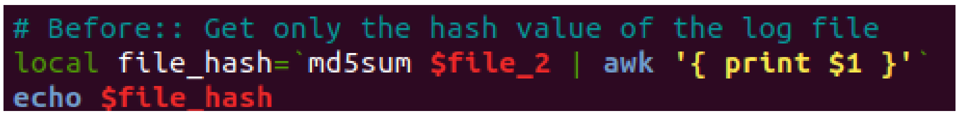
- Finally, The script is ran by the root privileges on cron jobs every two minutes. Reading logsHistory.txt to check its reliability.



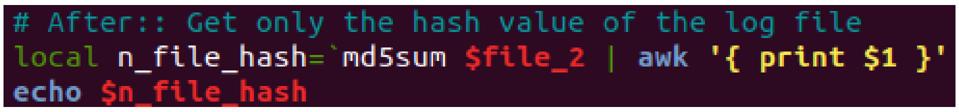
```
ubuntu@ubuntu:~/summaryLogs$ ls
diskUsageLogs.txt          unusualFilesLogs.txt
errors.txt                 unusualLogs.txt
logsHistory.txt            unusualMemoryUsageLogs.txt
newUnusualAccounts         unusualNetworkUsageLogs.txt
processesLogs.txt          unusualTasksLogs.txt
systemPerformanceLogs.txt
ubuntu@ubuntu:~/summaryLogs$ cat logsHistory.txt
Logs Updates at:
Mon Jun  1 05:38:12 PDT 2020
Logs Updates at:
Mon Jun  1 05:40:12 PDT 2020
ubuntu@ubuntu:~/summaryLogs$ cat logsHistory.txt
Logs Updates at:
Mon Jun  1 05:38:12 PDT 2020
Logs Updates at:
Mon Jun  1 05:40:12 PDT 2020
Logs Updates at:
Mon Jun  1 05:42:08 PDT 2020
Logs Updates at:
Mon Jun  1 05:44:08 PDT 2020
Logs Updates at:
Mon Jun  1 05:46:06 PDT 2020
ubuntu@ubuntu:~/summaryLogs$ 
```

## 3.3 Use of Hashing in the script

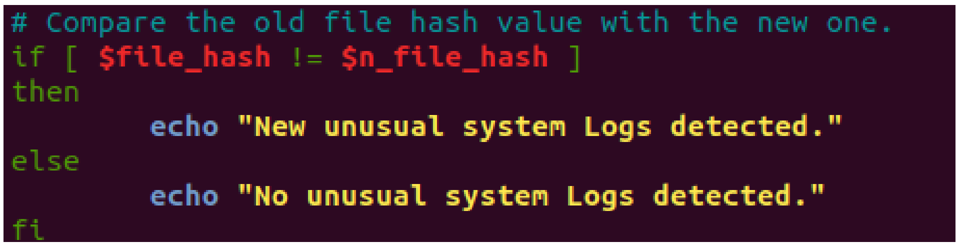
For most of log files under ”summaryLogs” directory, I get the Hash value of the file using ”md5sum” before detecting new occurred logs. Then, I get the hash value again after running the function. Finally, I check the old hash value and the new one, then print if any changes happened to the file in the command line.



```
# Before:: Get only the hash value of the log file
local file_hash=`md5sum $file_2 | awk '{ print $1 }'`
echo $file_hash
```



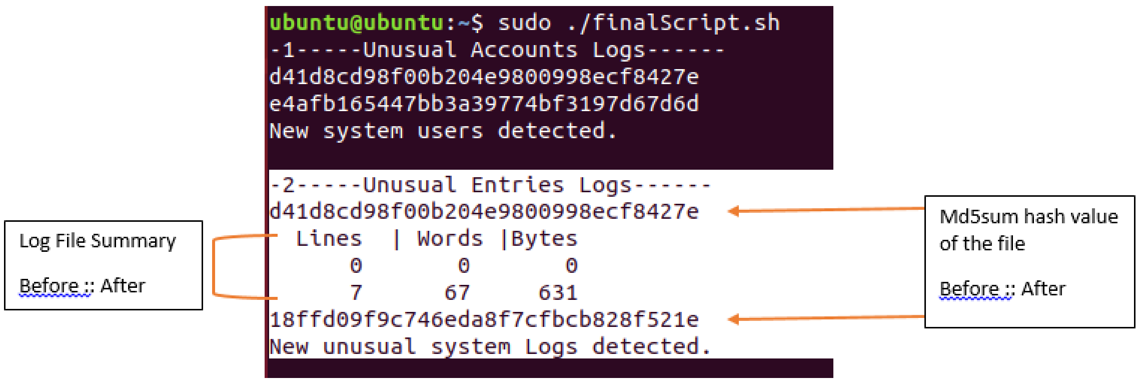
```
# After:: Get only the hash value of the log file
local n_file_hash=`md5sum $file_2 | awk '{ print $1 }'
echo $n_file_hash
```



```
# Compare the old file hash value with the new one.
if [ $file_hash != $n_file_hash ]
then
        echo "New unusual system Logs detected."
else
        echo "No unusual system Logs detected."
fi
```

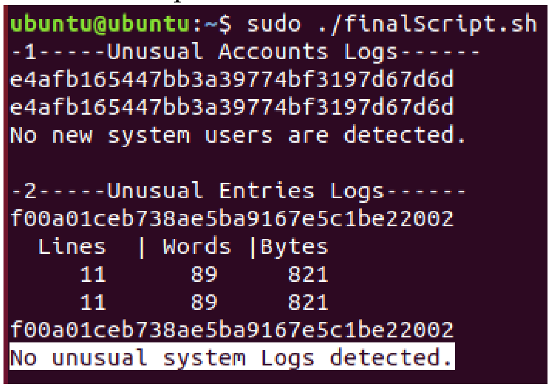
**Example:**
**Generating ”authentication failure” Error, then testing the script.**

- a. Run the script for the first time



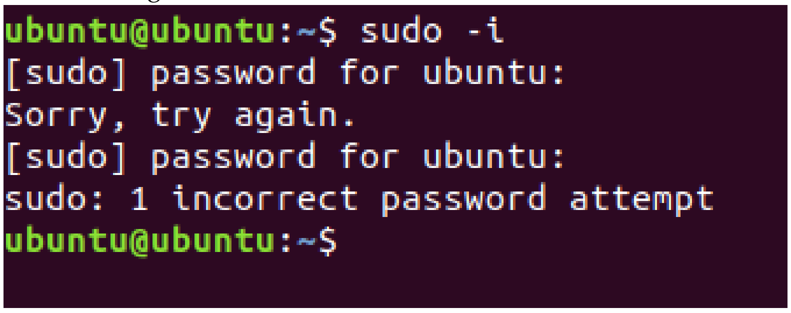
```
ubuntu@ubuntu:~$ sudo ./finalScript.sh
-1-----Unusual Accounts Logs------
d41d8cd98f00b204e9800998ecf8427e
e4afb165447bb3a39774bf3197d67d6d
New system users detected.

-2-----Unusual Entries Logs------
d41d8cd98f00b204e9800998ecf8427e
  Lines   | Words  |Bytes
      0        0       0
      7       67     631
18ffd09f9c746eda8f7cfbcb828f521e
New unusual system Logs detected.
```

- b. Run the script for the second time



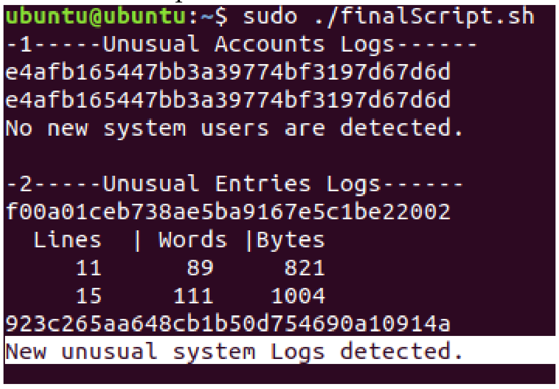
```
ubuntu@ubuntu:~$ sudo ./finalScript.sh
-1-----Unusual Accounts Logs------
e4afb165447bb3a39774bf3197d67d6d
e4afb165447bb3a39774bf3197d67d6d
No new system users are detected.

-2-----Unusual Entries Logs------
f00a01ceb738ae5ba9167e5c1be22002
  Lines   | Words |Bytes
     11       89      821
     11       89      821
f00a01ceb738ae5ba9167e5c1be22002
No unusual system Logs detected.
```

- c. Generating an Authentication failure Error



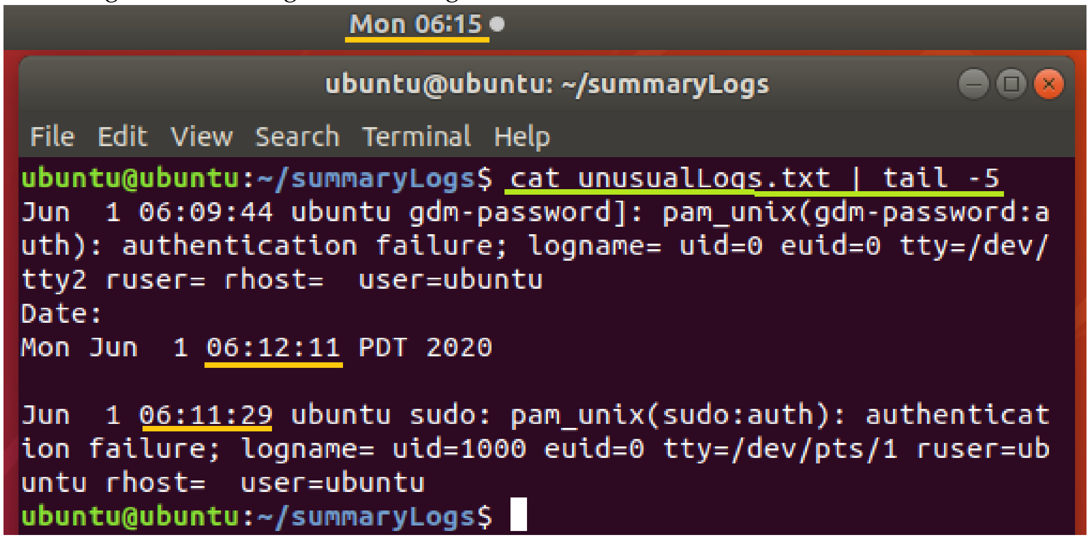
```
ubuntu@ubuntu:~$ sudo -i
[sudo] password for ubuntu:
Sorry, try again.
[sudo] password for ubuntu:
sudo: 1 incorrect password attempt
ubuntu@ubuntu:~$
```

- d. Run the script for the third time



```
ubuntu@ubuntu:~$ sudo ./finalScript.sh
-1-----Unusual Accounts Logs------
e4afb165447bb3a39774bf3197d67d6d
e4afb165447bb3a39774bf3197d67d6d
No new system users are detected.

-2-----Unusual Entries Logs------
f00a01ceb738ae5ba9167e5c1be22002
  Lines   | Words |Bytes
     11       89      821
     15      111     1004
923c265aa648cb1b50d754690a10914a
New unusual system Logs detected.
```

- e. Getting the failure Logs from the log file


Mon 06:15
ubuntu@ubuntu: ~/summaryLogs
File Edit View Search Terminal Help

```
ubuntu@ubuntu:~/summaryLogs$ cat unusualLogs.txt | tail -5
Jun  1 06:09:44 ubuntu gdm-password]: pam_unix(gdm-password:a
uth): authentication failure; logname= uid=0 euid=0 tty=/dev/
tty2 ruser= rhost=  user=ubuntu
Date:
Mon Jun  1 06:12:11 PDT 2020

Jun  1 06:11:29 ubuntu sudo: pam_unix(sudo:auth): authenticat
ion failure; logname= uid=1000 euid=0 tty=/dev/pts/1 ruser=ub
untu rhost=  user=ubuntu
ubuntu@ubuntu:~/summaryLogs$
```

**Encryption in next page**

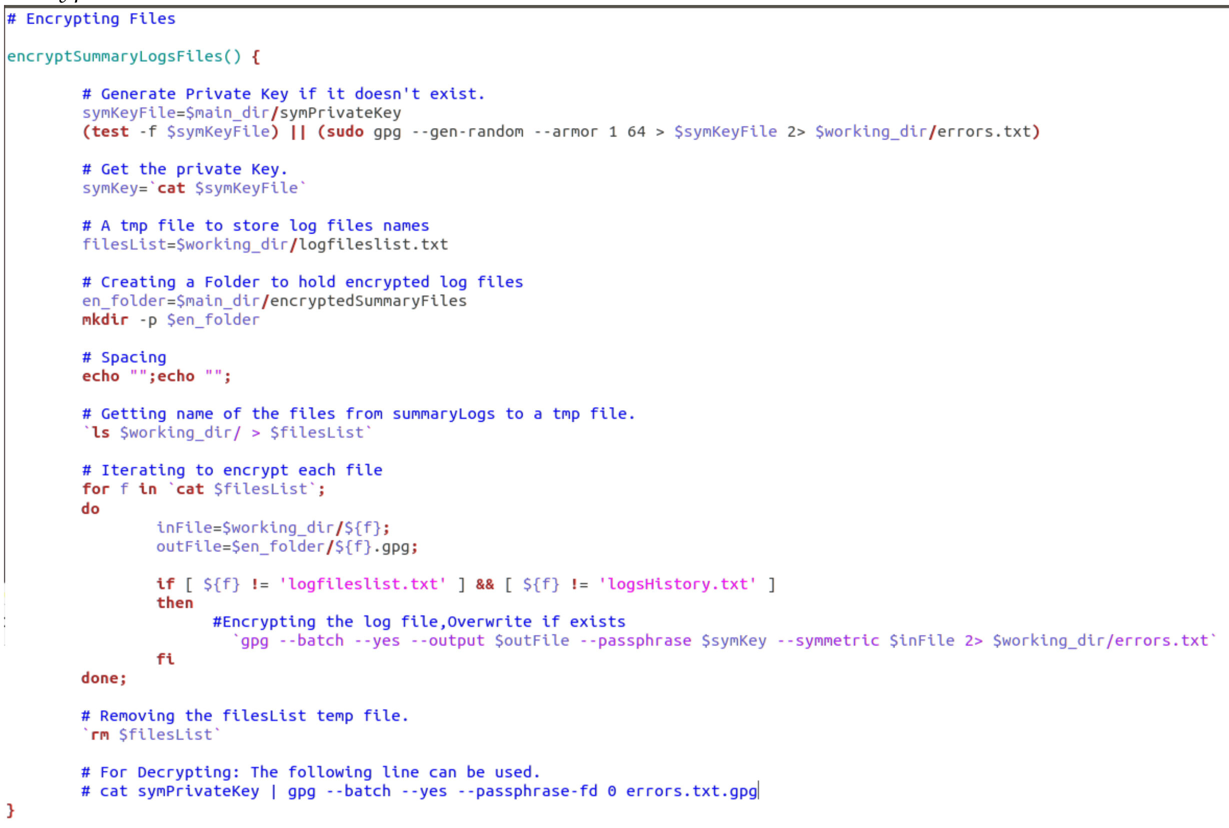
## 3.4 Encrypting the summary Log files with my Public key.

- I used gpg tool to generate a private symmetric key if it it doesn't exist. Also, gpg is used to encrypt each log file in summaryLog folder, Overwrite encrypted files if they exist and Forwarding Errors to errors.txt

- Generate a 64 character private key if doesn't exist
  **(test -f $symKeyFile)** $|||(sudogpg - -gen - random - -armor164 > \$symKeyFile$

- To encrypt the files with symmetric key
  **gpg –batch –yes –output $outFile –passphrase $symKey –symmetric $inFile**

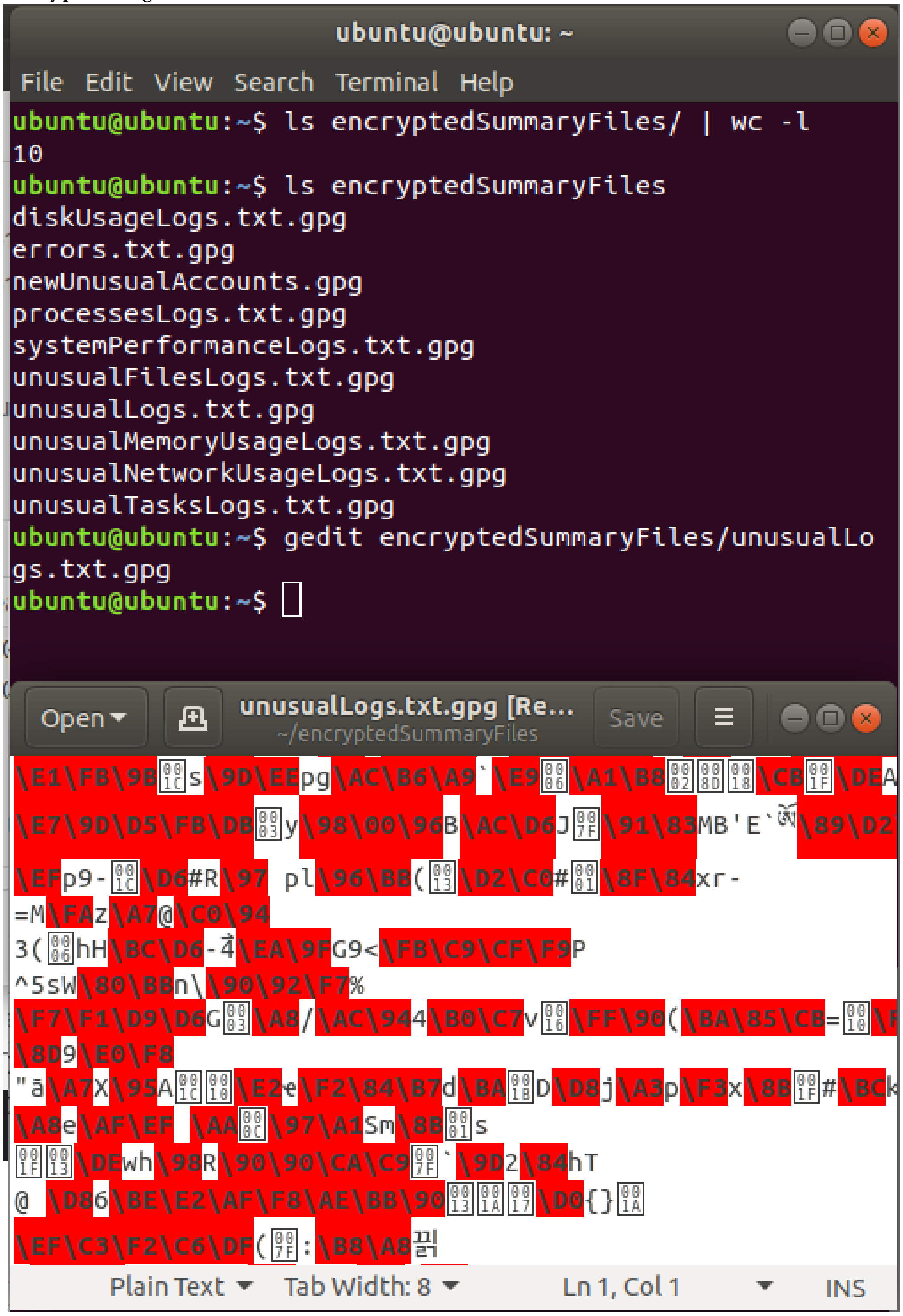
- To decrypt files.
  **cat symPrivateKey** $\|gpg - -batch - -yes - -passphrase - fd0errors.txt.gpg$
  $EncryptionFunction :: Screenshot$



```
# Encrypting Files

encryptSummaryLogsFiles() {

        # Generate Private Key if it doesn't exist.
        symKeyFile=$main_dir/symPrivateKey
        (test -f $symKeyFile) || (sudo gpg --gen-random --armor 1 64 > $symKeyFile 2> $working_dir/errors.txt)

        # Get the private Key.
        symKey=`cat $symKeyFile`

        # A tmp file to store log files names
        filesList=$working_dir/logfileslist.txt

        # Creating a Folder to hold encrypted log files
        en_folder=$main_dir/encryptedSummaryFiles
        mkdir -p $en_folder

        # Spacing
        echo "";echo "";

        # Getting name of the files from summaryLogs to a tmp file.
        `ls $working_dir/ > $filesList`

        # Iterating to encrypt each file
        for f in `cat $filesList`;
        do
                inFile=$working_dir/${f};
                outFile=$en_folder/${f}.gpg;

                if [ ${f} != 'logfileslist.txt' ] && [ ${f} != 'logsHistory.txt' ]
                then
                        #Encrypting the log file,Overwrite if exists
                          `gpg --batch --yes --output $outFile --passphrase $symKey --symmetric $inFile 2> $working_dir/errors.txt`
                fi
        done;

        # Removing the filesList temp file.
        `rm $filesList`

        # For Decrypting: The following line can be used.
        # cat symPrivateKey | gpg --batch --yes --passphrase-fd 0 errors.txt.gpg
}
```

-

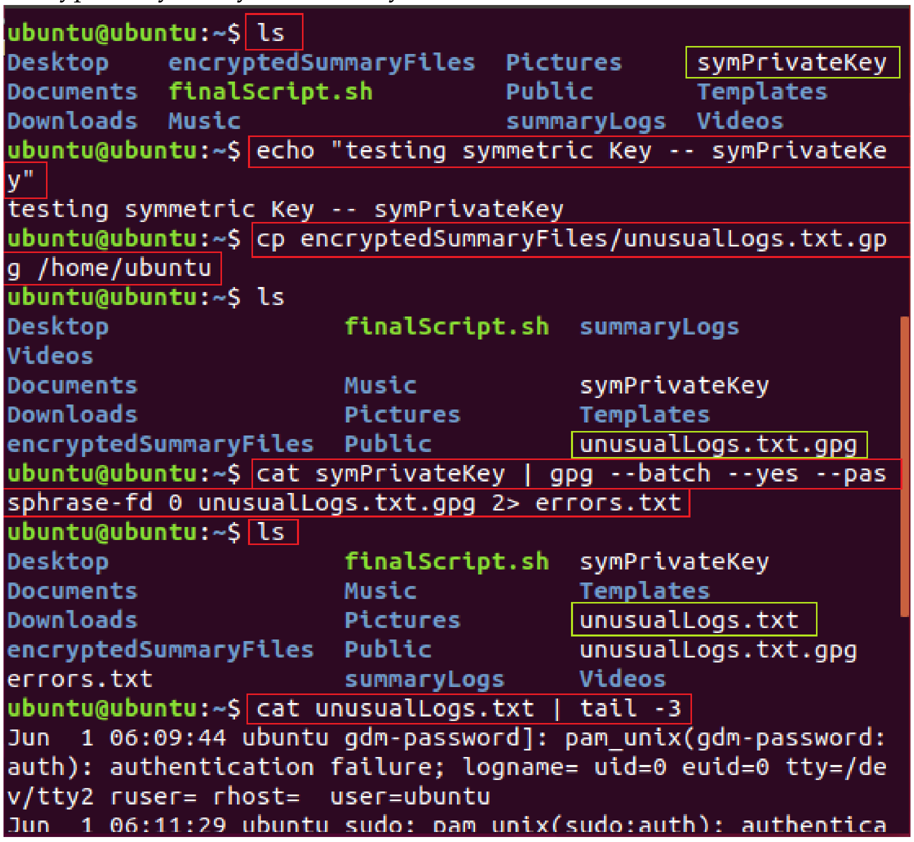
- Encrypted Log folder info ::Screenshot



```
ubuntu@ubuntu:~$ ls encryptedSummaryFiles/ | wc -l
10
ubuntu@ubuntu:~$ ls encryptedSummaryFiles
diskUsageLogs.txt.gpg
errors.txt.gpg
newUnusualAccounts.gpg
processesLogs.txt.gpg
systemPerformanceLogs.txt.gpg
unusualFilesLogs.txt.gpg
unusualLogs.txt.gpg
unusualMemoryUsageLogs.txt.gpg
unusualNetworkUsageLogs.txt.gpg
unusualTasksLogs.txt.gpg
ubuntu@ubuntu:~$ gedit encryptedSummaryFiles/unusualLo
gs.txt.gpg
ubuntu@ubuntu:~$ 
```

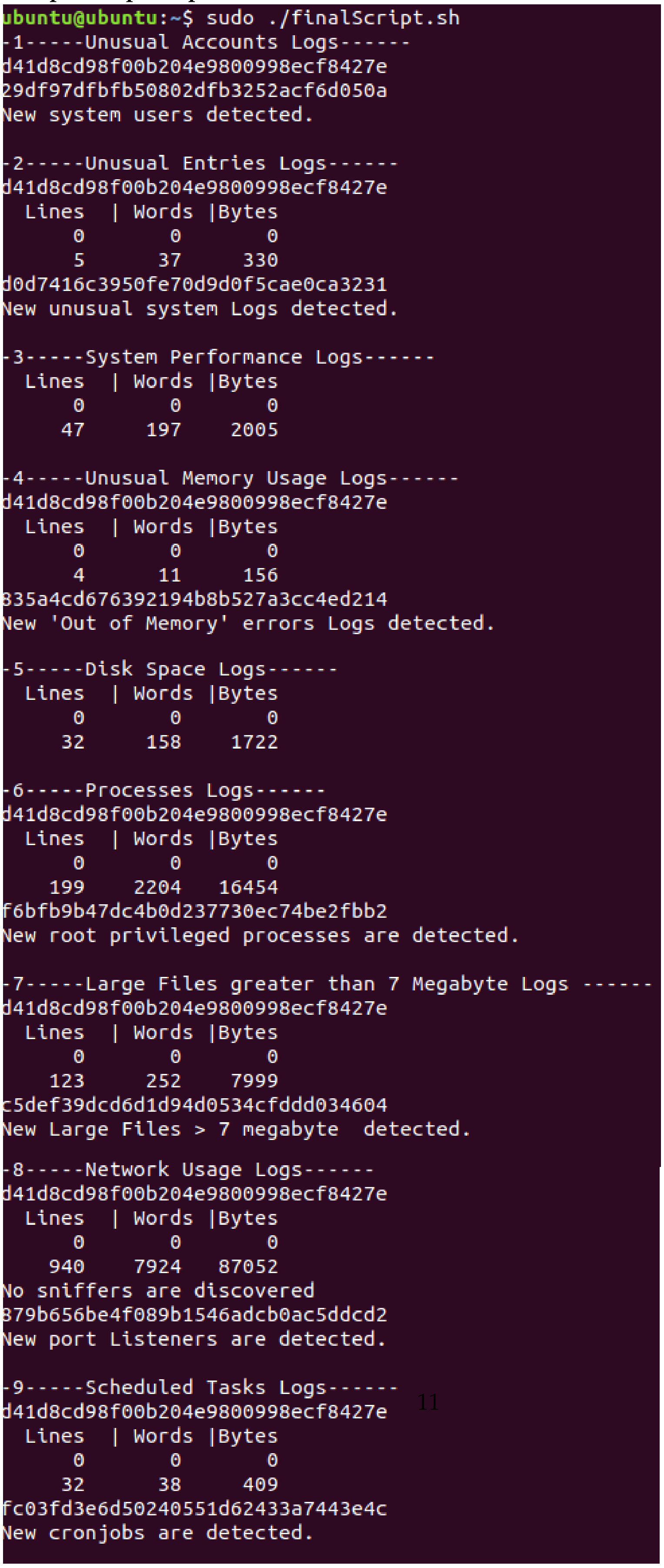
```
\E1\FB\9B[001C]s\9D\EEpg\AC\B6\A9`\E9[0006]\A1\B8[0002][008D][0018]\CB[001F]\DEA
\E7\9D\D5\FB\DB[0003]y\98\00\96B\AC\D6J[007F]\91\83MB'E`\89\D2
\EFp9-[001C]\D6#R\97 pl\96\BB([0013]\D2\C0#[0001]\8F\84xr-
=M\FAz\A7@\C0\94
3([0006]hH\BC\D6-4\EA\9FG9<\FB\C9\CF\F9P
^5sW\80\BBn\\90\92\F7%
\F7\F1\D9\D6G[0003]\A8/\AC\944\B0\C7v[0016]\FF\90(\BA\85\CB=[0010]\F
\8D9\E0\F8
"ā\A7X\95A[001C][0010]\E2e\F2\84\B7d\BA[001B]D\D8j\A3p\F3x\8B[001F]#\BCk
\A8e\AF\EF \AA[000C]\97\A1Sm\8B[0001]s
[001F][0013]\DEwh\98R\90\90\CA\C9[007F]`\9D2\84hT
@ \D86\BE\E2\AF\F8\AE\BB\90[0013][001A][0017]\D0{}[001A]
\EF\C3\F2\C6\DF([007F]:\B8\A8끫
```

- Decryption by the Symmetric Key test



```
ubuntu@ubuntu:~$ ls
Desktop     encryptedSummaryFiles  Pictures     symPrivateKey
Documents   finalScript.sh         Public       Templates
Downloads   Music                  summaryLogs  Videos
ubuntu@ubuntu:~$ echo "testing symmetric Key -- symPrivateKe
y"
testing symmetric Key -- symPrivateKey
ubuntu@ubuntu:~$ cp encryptedSummaryFiles/unusualLogs.txt.gp
g /home/ubuntu
ubuntu@ubuntu:~$ ls
Desktop                finalScript.sh  summaryLogs
Videos
Documents              Music           symPrivateKey
Downloads              Pictures        Templates
encryptedSummaryFiles  Public          unusualLogs.txt.gpg
ubuntu@ubuntu:~$ cat symPrivateKey | gpg --batch --yes --pas
sphrase-fd 0 unusualLogs.txt.gpg 2> errors.txt
ubuntu@ubuntu:~$ ls
Desktop                finalScript.sh  symPrivateKey
Documents              Music           Templates
Downloads              Pictures        unusualLogs.txt
encryptedSummaryFiles  Public          unusualLogs.txt.gpg
errors.txt             summaryLogs     Videos
ubuntu@ubuntu:~$ cat unusualLogs.txt | tail -3
Jun  1 06:09:44 ubuntu gdm-password]: pam_unix(gdm-password:
auth): authentication failure; logname= uid=0 euid=0 tty=/de
v/tty2 ruser= rhost=  user=ubuntu
Jun  1 06:11:29 ubuntu sudo: pam_unix(sudo:auth): authentica
```

**Sample script output on CLI**



```
ubuntu@ubuntu:~$ sudo ./finalScript.sh
-1-----Unusual Accounts Logs------
d41d8cd98f00b204e9800998ecf8427e
29df97dfbfb50802dfb3252acf6d050a
New system users detected.

-2-----Unusual Entries Logs------
d41d8cd98f00b204e9800998ecf8427e
  Lines   | Words |Bytes
      0        0        0
      5       37      330
d0d7416c3950fe70d9d0f5cae0ca3231
New unusual system Logs detected.

-3-----System Performance Logs------
  Lines   | Words |Bytes
      0        0        0
     47      197     2005

-4-----Unusual Memory Usage Logs------
d41d8cd98f00b204e9800998ecf8427e
  Lines   | Words |Bytes
      0        0        0
      4       11      156
835a4cd676392194b8b527a3cc4ed214
New 'Out of Memory' errors Logs detected.

-5-----Disk Space Logs------
  Lines   | Words |Bytes
      0        0        0
     32      158     1722

-6-----Processes Logs------
d41d8cd98f00b204e9800998ecf8427e
  Lines   | Words |Bytes
      0        0        0
    199     2204    16454
f6bfb9b47dc4b0d237730ec74be2fbb2
New root privileged processes are detected.

-7-----Large Files greater than 7 Megabyte Logs ------
d41d8cd98f00b204e9800998ecf8427e
  Lines   | Words |Bytes
      0        0        0
    123      252     7999
c5def39dcd6d1d94d0534cfddd034604
New Large Files > 7 megabyte  detected.

-8-----Network Usage Logs------
d41d8cd98f00b204e9800998ecf8427e
  Lines   | Words |Bytes
      0        0        0
    940     7924    87052
No sniffers are discovered
879b656be4f089b1546adcb0ac5ddcd2
New port Listeners are detected.

-9-----Scheduled Tasks Logs------
d41d8cd98f00b204e9800998ecf8427e
  Lines   | Words |Bytes
      0        0        0
     32       38      409
fc03fd3e6d50240551d62433a7443e4c
New cronjobs are detected.
```

## 4 Conclusion

In this homework, I developed 9 functions to collect unusual activities in the operation system. Also, I learned how to redirect functions output to files. Then, I used crontabs to run the script periodically for example every 2 minutes. After that, I used hashes to check files changes and notify user on CLI. Finally, I created the 10th function which encryptes the unusual log files with my public key.

## 5 references

### 5.1 *https://hostadvice.com/how-to/how-to-use-gnupg-keys-for-encrypting-messages-on-ubuntu-18-04/*

### 5.2 *https://www.youtube.com/watch?v=iEloW5QCvKI*

## 6 Script

*Listing 1:* bash version

```
#!/bin/bash


main_dir=/home/ubuntu

# Creating a folder if doesn't exist to keep log files in it.
mkdir -p $main_dir/summaryLogs

working_dir=$main_dir/summaryLogs

######### 1. Unusual\usual Accounts

copyUnusualAccountsLogs() {
        ## Creating Logs File if it doesn't exist
        file_1=$working_dir/unusualAccounts.txt
        test -f $file_1 || touch $file_1

        # Before: Get only the hash value of the log file
        local file_hash=`md5sum $file_1 | awk '{ print $1 }'`
        echo $file_hash


        # a. Finding all system Users
        all_users_accounts=`cut -d: -f1 /etc/passwd`

        # Get the account from "all_users_accounts" that doesn't exist in the file
        new_users=`grep -Fxvf $file_1 <(echo $all_users_accounts)`
        # Check if any new Users detected
        new_users_length=`echo ${#new_users}`

        # if there are new users, append new users to the file

        if [ $new_users_length -ne 0 ]
           then
                (echo "Date:" && date && echo "") >> $file_1
                (echo "## Detected new System Accounts" && echo "" ) >> $file_1
                (grep -Fxvf $file_1 <(echo $all_users_accounts)) >> $file_1
        fi

        # After: Get only the hash value of the log file
        local n_file_hash=`md5sum $file_1 | awk '{ print $1 }'`
        echo $n_file_hash

        # Compare the old file hash value with the new one.
        if [ $file_hash != $n_file_hash ]
        then
                echo "New system users detected."
        else
                echo "No new system users are detected."
        fi

}

######### 2. Unusual log entries

copyUnusualEntriesLogs() {
        ## Creating Logs File if it doesn't exist
        file_2=$working_dir/unusualLogs.txt
        test -f $file_2 || touch $file_2

        # Before:: Get only the hash value of the log file
        local file_hash=`md5sum $file_2 | awk '{ print $1 }'`
        echo $file_hash

        # a. Finding "authentication failure" Logs and storing them to tmp file
        grep 'authentication failure' /var/log/auth.log > $working_dir/tmp.txt

        new_failure_logs=`grep -Fxvf $file_2 $working_dir/tmp.txt`

        # Check if any new Users detected
        new_failure_logs_length=`echo ${#new_failure_logs}`

        # Printing current log file lines.
        echo "  Lines  | Words | Bytes"
        cat $file_2 | wc
        # O. Saving Logs to unusualLogs.txt
        if [ $new_failure_logs_length -ne 0 ]
           then
                (echo "Date:" && date && echo "") >> $file_2
                # Take the new authentication failure lines and copy them to unusualL
                grep -Fxvf $file_2 $working_dir/tmp.txt >> $file_2

        fi
        # Printing new log file lines.
        cat $file_2 | wc

        # After:: Get only the hash value of the log file
        local n_file_hash=`md5sum $file_2 | awk '{ print $1 }'`
        echo $n_file_hash

        # Compare the old file hash value with the new one.
        if [ $file_hash != $n_file_hash ]
        then
                echo "New unusual system Logs detected."
        else
                echo "No unusual system Logs detected."
        fi

        # Remove the temporary file.
        sudo rm $working_dir/tmp.txt

}

######### 3. sluggish system performance
copySystemPerformanceLogs() {

        # a. Showing Load Average and for how long the computer has been powered on
        load_avg=`uptime | grep -oP '(?=load).*'`
        up_time=`uptime -p`

        # b. Monitoring CPU, Device and Network file system utilization report
        cpu_state=`iostat`

        # O. Saving Logs to systemPerformanceLogs.txt
        file_3=$working_dir/systemPerformanceLogs.txt
        test -f $file_3 || touch $file_3

        # Get current file Lines
        echo "  Lines  | Words | Bytes"
        cat $file_3 | wc

        (echo "Date:" && date && echo "") >> $file_3
        (echo "################################" ) >> $file_3
        (echo "##␣System␣up␣time:") >> $file_3
        (echo "--------------------------------" ) >> $file_3
        (echo "$up_time" && echo "" && echo "") >> $file_3

        (echo "################################" ) >> $file_3
        (echo "##␣System␣Load␣Average:" ) >> $file_3
        (echo "--------------------------------" ) >> $file_3
        (echo "Load␣Average␣Description:" && echo "First␣cell:␣load␣average␣over␣the␣
        (echo "$load_avg" && echo "" && echo "" ) >> $file_3


        (echo "################################" ) >> $file_3
        (echo "##␣CPU,␣Device␣and␣Network␣file␣system␣utilization:") >> $file_3
        (echo "--------------------------------" ) >> $file_3
        (echo "$cpu_state" && echo "" && echo "" ) >> $file_3

        cat $file_3 | wc

}

######### 4. Excessive memory use
copyMemoryUsageLogs() {


        # a. Searching in log files for an "out of memory" error
        exc_memory_logs=`grep -i -r 'out␣of␣memory' /var/log/`


        # O. Saving Logs to unusualMemoryUsageLogs.txt

        ## Creating Logs File if it doesn't exist
        file_4=$working_dir/unusualMemoryUsageLogs.txt
        test -f $file_4 || touch $file_4

        # Before:: Get only the hash value of the log file
        local file_hash=`md5sum $file_4 | awk '{␣print␣$1␣}'`
        echo $file_hash

        # a. Finding "out of memory" Logs and storing them to tmp file
        grep -i -r 'out␣of␣memory' /var/log/ > $working_dir/tmp.txt

        new_oom_logs=`grep -Fxvf $file_4 $working_dir/tmp.txt`

        # Check if any new "out of memory" errors are detected
        new_oom_logs_length=`echo ${#new_oom_logs}`

        # Printing current log file lines.
        echo "␣␣Lines␣␣|␣Words␣|Bytes"
        cat $file_4 | wc

        # O. Saving Logs to unusualMemoryUsageLogs.txt
        if [ $new_oom_logs_length -ne 0 ]
           then
                (echo "Date:" && date && echo "") >> $file_4
                # Take the new 'out of memory' failure lines and copy them to unusual
                grep -Fxvf $file_4 $working_dir/tmp.txt >> $file_4

        fi
        # Printing new log file lines.
        cat $file_4 | wc

        # After:: Get only the hash value of the log file
        local n_file_hash=`md5sum $file_4 | awk '{ print $1 }'`
        echo $n_file_hash

        # Compare the old file hash value with the new one.
        if [ $file_hash != $n_file_hash ]
        then
                echo "New 'Out of Memory' errors Logs detected."
        else
                echo "No 'Out of Memory' errors Logs detected."
        fi

        # Remove the temporary file.
        sudo rm $working_dir/tmp.txt
}

######### 5. Decrease in Disk space
copyDiskSpaceLogs() {

        # a. Showing the amount of disk space used and available on Linux file system
        disk_state=`df`

        # O. Saving Logs to diskUsageLogs.txt
        file_5=$working_dir/diskUsageLogs.txt
        test -f $file_5 || touch $file_5


        # Printing current log file lines.
        echo "  Lines  | Words | Bytes"
        cat $file_5| wc

        (echo "Date:" && date && echo "") >> $file_5

        (echo "##############################" ) >> $file_5
        (echo "## Disk space Usage:") >> $file_5
        (echo "--------------------------------" ) >> $file_5
        (echo "$disk_state" && echo "" && echo "") >> $file_5

        cat $file_5 | wc

}


######### 6. Unusual process and services
copyProcessLogs() {
        # a. Finding  processes with root (UID 0) privileges and storing their info t
        (ps -U root -u root u) > $working_dir/tmp.txt


        # O. Saving Logs to processesLogs.txt

        ## Creating Logs File if it doesn't exist
        file_6=$working_dir/processesLogs.txt
        test -f $file_6 || touch $file_6

        # Before:: Get only the hash value of the log file
        local file_hash=`md5sum $file_6 | awk '{ print $1 }'`
        echo $file_hash

        # Grep new lines from tmp file.
        new_rpp=`grep -Fxvf $file_6 $working_dir/tmp.txt`

        # Check if any new root privileged processes are detected
        new_rpp_length=`echo ${#new_rpp}`


        # Printing current log file lines.
        echo "  Lines  | Words | Bytes"
        cat $file_6 | wc


        # O. Saving Logs to processesLogs.txt
        if [ $new_rpp_length -ne 0 ]
           then
                (echo "Date:" && date && echo "") >> $file_6
                # Take the new lines and copy them to unusualFilesLogs.txt.
                grep -Fxvf $file_6 $working_dir/tmp.txt >> $file_6

        fi
        # Printing new log file lines.
        cat $file_6 | wc

        # After:: Get only the hash value of the log file
        local n_file_hash=`md5sum $file_6 | awk '{ print $1 }'`
        echo $n_file_hash
        # Compare the old file hash value with the new one.
        if [ $file_hash != $n_file_hash ]
        then
                echo "New␣root␣privileged␣processes␣are␣detected."
        else
                echo "No␣new␣root␣privileged␣processes␣are␣detected."
        fi

        # Remove the temporary file.
        sudo rm $working_dir/tmp.txt


}
#-------------------------
######### 7. Unusual Files
copyUnusualFilesLogs() {


        # a. Finding unusual large files. Ex: files that are greater than 7 MegaBytes
        (sudo find / -size +7M -type f -exec du -Sh {} +  | sort -rh > $working_dir/t


        # O. Saving Logs to unusualFilesLogs.txt

        ## Creating Logs File if it doesn't exist
        file_7=$working_dir/unusualFilesLogs.txt
        test -f $file_7 || touch $file_7

        # Before:: Get only the hash value of the log file
        local file_hash=`md5sum $file_7 | awk '{␣print␣$1␣}'`
        echo $file_hash

        # Grep new lines from tmp file.
        new_ulf=`grep -Fxvf $file_7 $working_dir/tmp.txt`

        # Check if any new Large Files are detected
        new_ulf_length=`echo ${#new_ulf}`


        # Printing current log file lines.
        echo "␣␣Lines␣␣|␣Words␣|Bytes"
        cat $file_7 | wc

        # O. Saving Logs to unusualFilesLogs.txt
        if [ $new_ulf_length -ne 0 ]
            then
                (echo "Date:" && date && echo "") >> $file_7
                # Take the new lines and copy them to unusualFilesLogs.txt.
                grep -Fxvf $file_7 $working_dir/tmp.txt >> $file_7

        fi
        # Printing new log file lines.
        cat $file_7 | wc

        # After:: Get only the hash value of the log file
        local n_file_hash=`md5sum $file_7 | awk '{ print $1 }'`
        echo $n_file_hash

        # Compare the old file hash value with the new one.
        if [ $file_hash != $n_file_hash ]
        then
                echo "New Large Files > 7 megabyte  detected."
        else
                echo "No new Large Files > 7 megabyte detected."
        fi

        # Remove the temporary file.
        sudo rm $working_dir/tmp.txt

}

######### 8. Unusual network usage
copyNetworkUsageLogs() {

        # a. Finding port listeners and storing their info to tmp file.
        `netstat -nap > $working_dir/tmp.txt`

        # O. Saving Logs to networkUsageLogs.txt

        ## Creating Logs File if it doesn't exist
        file_8=$working_dir/unusualNetworkUsageLogs.txt
        test -f $file_8 || touch $file_8

        # Before:: Get only the hash value of the log file
        local file_hash=`md5sum $file_8 | awk '{ print $1 }'`
        echo $file_hash

        # Grep new lines from tmp file.
        new_npl=`grep -Fxvf $file_8 $working_dir/tmp.txt`

        # Check if any new Network Port Listeners are detected
        new_npl_length=`echo ${#new_npl}`


        # Printing current log file lines.
        echo "  Lines  | Words | Bytes"
        cat $file_8 | wc
        # Saving Logs to networkUsageLogs.txt
        if [ $new_npl_length -ne 0 ]
           then
                (echo "Date:" && date && echo "") >> $file_8
                # Take the new lines and copy them to unusualFilesLogs.txt.
                grep -Fxvf $file_8 $working_dir/tmp.txt >> $file_8

        fi
        # Printing new log file lines.
        cat $file_8 | wc


        # b. Looking for promiscuous mode, which might indicate a sniffe
        promiscuous_sniffer=`ip link | grep PROMISC`

        if [[ $promiscuous_sniffer -eq 0 ]] ; then
                (echo "No sniffers are discovered ")
        else
                (echo "$promiscuous_sniffer" && echo "" && echo "") >> $file_8
        fi



        # After:: Get only the hash value of the log file
        local n_file_hash=`md5sum $file_8 | awk '{ print $1 }'`
        echo $n_file_hash

        # Compare the old file hash value with the new one.
        if [ $file_hash != $n_file_hash ]
        then
                echo "New port Listeners are detected."
        else
                echo "No new port Listeners are detected."
        fi

        # Remove the temporary file.
        sudo rm $working_dir/tmp.txt

}

######## 9. Unusual scheduled tasks
copyScheduledTasksLogs() {

        # a.  Finding cron jobs that are scheduled by root
        #     Escaping the crontab file description 24 lines
        #     Then copy the cronjobs to tmp file.
        `(sudo crontab -u root -l | tail -n +24) > $working_dir/tmp.txt 2> $working_
```

```
# b. System-wide cron jobs
`(ls /etc/cron.*) > $working_dir/tmp1.txt`


# O. Saving Logs to unusualTasksLogs.txt

## Creating Logs File if it doesn't exist
file_9=$working_dir/unusualTasksLogs.txt
test -f $file_9 || touch $file_9

# Before:: Get only the hash value of the log file
local file_hash=`md5sum $file_9 | awk '{ print $1 }'`
echo $file_hash

# Grep new lines from tmp file.
new_utl_1=`grep -Fxvf $file_9 $working_dir/tmp.txt`

# Grep new lines from tmp file.
new_utl_2=`grep -Fxvf $file_9 $working_dir/tmp1.txt`

# Get length of new_utl variables
new_utl_1_length=`echo ${#new_utl_1}`
new_utl_2_length=`echo ${#new_utl_2}`

# Printing current log file lines.
echo "  Lines  | Words | Bytes"
cat $file_9 | wc

# O. Saving Logs to unusualFilesLogs.txt
if [ $new_utl_1_length -ne 0 ] || [ $new_utl_2_length -ne 0 ]
  then
      (echo "Date:" && date && echo "") >> $file_9

fi
if [ $new_utl_1_length -ne 0 ]
  then
      # Root & UID:0 Cron Jobs:.
      (echo "" && echo "###### Root Cron Jobss") >> $file_9
      grep -Fxvf $file_9 $working_dir/tmp.txt >> $file_9

fi
if [ $new_utl_2_length -ne 0 ]
  then
      #  System-wide cron jobs.
      (echo "" && echo "###### System-wide cron jobs") >> $file_9
      grep -Fxvf $file_9 $working_dir/tmp1.txt >> $file_9

fi
```

```
        # Printing new log file lines.
        cat $file_9 | wc

        # After:: Get only the hash value of the log file
        local n_file_hash=`md5sum $file_9 | awk '{␣print␣$1␣}'`
        echo $n_file_hash

        # Compare the old file hash value with the new one.
        if [ $file_hash != $n_file_hash ]
        then
                echo "New␣cronjobs␣are␣detected."
        else
                echo "No␣new␣cronjobs␣are␣detected."
        fi

        # Remove the temporary file.
        sudo rm $working_dir/tmp.txt
        sudo rm $working_dir/tmp1.txt


}

# Encrypting Files

encryptSummaryLogsFiles() {

        # Generate Private Key if it doesn't exist.
        symKeyFile=$main_dir/symPrivateKey
        (test -f $symKeyFile) || (sudo gpg --gen-random --armor 1 64 > $symKeyFile 2>

        # Get the private Key.
        symKey=`cat $symKeyFile`

        # A tmp file to store log files names
        filesList=$working_dir/logfileslist.txt

        # Creating a Folder to hold encrypted log files
        en_folder=$main_dir/encryptedSummaryFiles
        mkdir -p $en_folder

        # Spacing
        echo "";echo "";

        # Getting name of the files from summaryLogs to a tmp file.
        `ls $working_dir/ > $filesList`

        # Iterating to encrypt each file
        for f in `cat $filesList`;
        do
                inFile=$working_dir/${f};
                outFile=$en_folder/${f}.gpg;

                if [ ${f} != 'logfileslist.txt' ] && [ ${f} != 'logsHistory.txt' ]
                then
                      #Encrypting the log file ,Overwrite if exists
                        `gpg --batch --yes --output $outFile --passphrase $symKey --s
                fi
        done;

        # Removing the filesList temp file.
        `rm $filesList`

        # For Decrypting: The following line can be used.
        # cat symPrivateKey | gpg --batch --yes --passphrase-fd 0 errors.txt.gpg
}




# Keeping a code run date in a file
`echo "Logs Updates at: " >> $working_dir/logsHistory.txt`
`date >> $working_dir/logsHistory.txt`




# Run the Script periodically after the initial execution by the user
runScriptPeriodically() {

        # Add the following command if it doesn't exist to the /etc/crontab to run sc
        command='*/2  *    * * *   root    bash /home/ubuntu/finalScript.sh'

        `grep -qxF "${command}" /etc/crontab || echo "${command}" >> /etc/crontab`
}

## Calling the 11 Functions
(echo "-1------Unusual Accounts Logs------")
copyUnusualAccountsLogs
(echo "" && echo "-2------Unusual Entries Logs------")
copyUnusualEntriesLogs
(echo "" && echo "-3------System Performance Logs------")
copySystemPerformanceLogs
(echo "" && echo "-4------Unusual Memory Usage Logs------")
copyMemoryUsageLogs
(echo "" && echo "-5------Disk Space Logs------")
copyDiskSpaceLogs
```

```
(echo "" && echo "–6————Processes␣Logs————")
copyProcessLogs
(echo "" && echo "–7————Large␣Files␣greater␣than␣7␣Megabyte␣Logs␣————")
copyUnusualFilesLogs
(echo "" && echo "–8————Network␣Usage␣Logs————")
copyNetworkUsageLogs
(echo "" && echo "–9————Scheduled␣Tasks␣Logs————")
copyScheduledTasksLogs
(echo "" && echo "–10————Encrypting␣Files ————")
encryptSummaryLogsFiles
(echo "" && echo "–11————Check/Update␣crontab␣jobs ————")
runScriptPeriodically
```