Host-based Intrusion Detection System in Shell Script

3413ICT: Network Security

Assignment 2

22/05/16

Aidan McMillan

S2946192

CONTENTS

NOTE BEFORE USING PROGRAM

The script.sh file contains the program and MUST be located within the /home/seed/Assignment/ directory.
The folder that will be tested MUST be called testFolder and MUST be located at /home/seed/Assignment/testFolder directory.
There is a bug where FILENAMES CANNOT CONTAIN WHITESPACE.

1 INTRODUCTION

This report details the creation and implementation of an Intrusion Detection System. This is meant to work by generating a validation file of all the data within a given folder and comparing that validation file with the current settings in order to see if any data has been tampered with. This system faced two primary objectives:

i) The ability to create a verification file from a given directory.
ii) The ability to compare the current attributes of these files with the attributes previously stored within the verification file.

The learning outcomes of this task revolve around learning about file modification and integrity and focuses on detecting possible intrusions on the given data.

2 PLATFORM AND TOOLS

This program was created using Bourne shell script on a Unix system (Ubuntu 32-bit) and makes use of default Unix commands in addition to shell commands. These tools were used to create an "internal" program (an all in one application reacting to user input) which provides the ability to validate files and test integrity.

Besides the fundamental programming techniques used (e.g. for, if, echo), the basic Unix commands used include:

ls – ls was used to get the attributes of the files when creating the validation file as well as in the process of testing the current data.

awk – awk was used in combination with ls to extract the required information from the given output.

grep – grep was used to check whether a file is part of the validation list or if it has been added as well as to parse through the validation file to check for differences during testing.

stat – stat was used to output the timestamps for last modification or status changes.

wc – wc is a simple word count command used to get the word count of a file.

readlink – readlink is used to return the path name of a file.

tee – tee is used for outputting to multiple sources. Within the IDS it is used to output to both the console as well as a log file.

3 IMPLEMENTATION

Implementation worked in 3 stages: Validation File Generation, Intrusion Detection and Overall Program Improvement.

3.1 VALIDATION FILE GENERATION -C

The first factor to implement is function of looping through the given directory and accessing all files, symbolic links and folders. The challenge faced with this stemmed from the use of ls instead of a find command. It introduced the difficulty of implementing recursion as well as differentiating between the different types of files returned.

The second part of the file generation implemented the reading in of the file attributes into the file. This was done by creating/clearing the validation file and then using ls to read in the attributes and awk to pull the exact value required out of the returned line.

Each different file is placed on a new line in order to differentiate between files. There is a difference between the last few values of a file depending on whether the entry represents a file, directory or symbolic link. All of this functionality is placed in a getValues() function in case of reuse at a later date. The main values used to loop through are also kept as parameters so that different values can define what the function does.

3.2 INTRUSION DETECTION -O

The Intrusion Detection was harder to implement than the generation of the validation file. On top of reading in the attributes of a file and comparing to the validation file it is also necessary to both check for new files within the given folder as well as check if any previous files have been deleted.

getChanges()
This function loops through the current files and reads their attributes, passing them to the checkFile() function after determining what type of file it is (file, folder, symbolic link) and changing the given data accordingly.

checkDelete()
This function reads through the validation file and for each entry checks whether the file still exists at the relevant path. It is used to both check for file deletion as well as moving or even changing the file name.

checkFile()
This function first ensures that the file is part of the validation file and if it isn't returns that the file has been added after the validation file creation. If the file is found to exist, then with the use of grep discrepancies are detected between the validation files attributes and the current attributes. Any changes result in the compromised data being outputted.

compareToFile()
This function is the main controller of the intrusion detection system and calls the above functions. It lays everything out nicely for easy readability and basically manages the other intrusion functions.

3.3 Overall Program

Overall the program uses these two modules to accomplish the goals of an IDS. They follow the same format of data in order to minimalize complications when they interact with it separately (i.e. when comparing data is laid out in a predictable way). The actual use of the system is managed by a simple case loop which waits for a user's input to dictate what function to perform.

4 Testing

The testing of the program was exhaustive and determined that all implemented features are working. One run of these test can be seen in the images located in the Appendix 1-5 and are described below:

Appendix 1 demonstrates the basic prompting of the program. As well as this it shows what a user sees upon successful creation of a validation file (-c) as well as the (-e) exit command in action.

The validation file is generated from the -c command. Each file is represented by a line, and the values of the file are separated with spaces. (**Appendix 2**)

During testing the validationFile.txt was generated, and screenshots of the test directory are before any changes are made can be seen in **Appendix 3**.

The changes that were made to the test directory can be seen in **Appendix 4**. These cover almost every conceivable action to be performed on the various files in the test directory and allow for an extensive test of the IDS.

Appendix 5 shows the program checking the new data against the values in the validation file. If a file has been changed it will display the value of the changed attribute and goes on to output both to the screen as well as to an output file. As seen by the highlighting it also shows added files as well as deleted files on top of a count of the various performed operation at the end for informative purposes.

5 Discussion

5.1 Findings and Recommendations

I quite enjoyed this task as it let me learn more about Unix and shell and let me apply skills learnt in class to a practical project. I can see how a person could affect a system badly if the right security measures aren't in place and while an IDS doesn't prevent damage it definitely provides a valuable function when determining the integrity of data. Only problem with the one implemented is the md5 requirement which has been shown to be vulnerable to collisions compared to other hash functions.

5.2 Difficulties

Overall Implementation of an IDS was successful, however difficulties where encountered during design. Implementing a recursive function in the format I have in shell is not common as far as I know. This took a lot of trial and error.

The biggest difficulty faced was probably the confusion of using 3 different sets of quotes. When attempting to read in variables much time was spent with the use of different combinations of quoting because of the command interpretation needed in some parts of an echo statement whereas over parts had to be interpreted as a string otherwise permissions got involved. This was possibly the most frustrating thing about the assignment which was eventually solved by using the $() to output the result of a command into a string.

Overall design of the system was easy, the only difficulties faced laid with my inexperience with coding in shell. I found I knew what I wanted to do but actually writing a command to do it sometimes caused difficulties.

5.3 LIMITATIONS OF THE IDS

The IDS provides all required functionality but still has a few limitations.

The first and worst limitation which wasn't discovered until very late (read: last minute) in the design stems from how the layout of the verification file is designed. Because of the use of one line equaling one file, whitespaces in filenames will break the IDS, meaning that within the testFolder no spaces can be in filenames otherwise my use of awk will cause invalid results.

The second limitation (or feature depending on how you look at it) is the capture of backup files when reading in to the validation file. Some programs, mainly text editors, generate a hidden backup file designated by a ~ after the filename (e.g. "file.txt~"). This doesn't affect the program overmuch but could be considered a limitation.

A minor limitation which can be changed given the time is the ability for users to designate what directory they want to use with the IDS regardless of where the program itself is located. At the moment this program is made to work solely with the intended directory with the right filenames. Steps have been taken with the use of function parameters so that it would be possible for the user to input their own directory but no prompt is given to the user for that functionality at this time.

6 CONCLUSION

Overall implementation of the two main functions of the IDS was successful, with the only real problem/drawback found being the lack of support for spaces within filenames. This is quite a big problem with a relatively simple fix however due to time constraints won't be implemented.

The IDS can successfully detect any changes made within the given directory included added and deleted files as well as attributes of files themselves changing. IT provides strong security with the use of timestamps and hashing to ensure a very difficult time for someone wanting to change data without alerting the IDS.

Difficulties faced weren't from lack of understanding but were due to inexperience with shell and Linux as a whole. Furthermore, a practical application of what we have learnt is relatively enjoyable.

7 APPENDIX

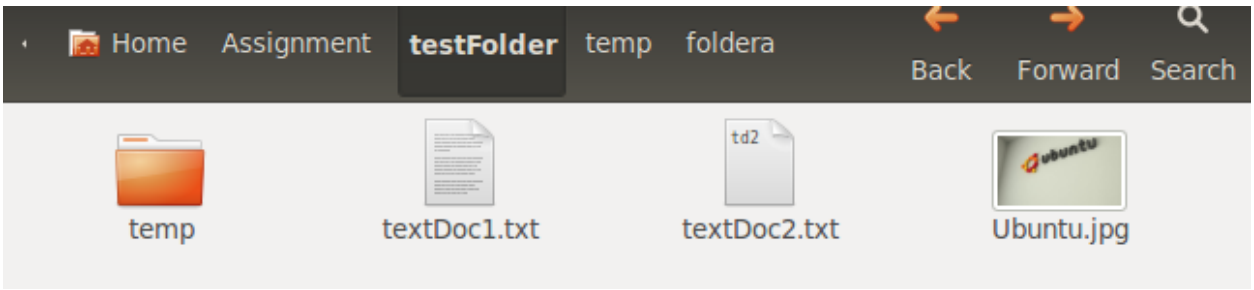
7.1 APPENDIX 1

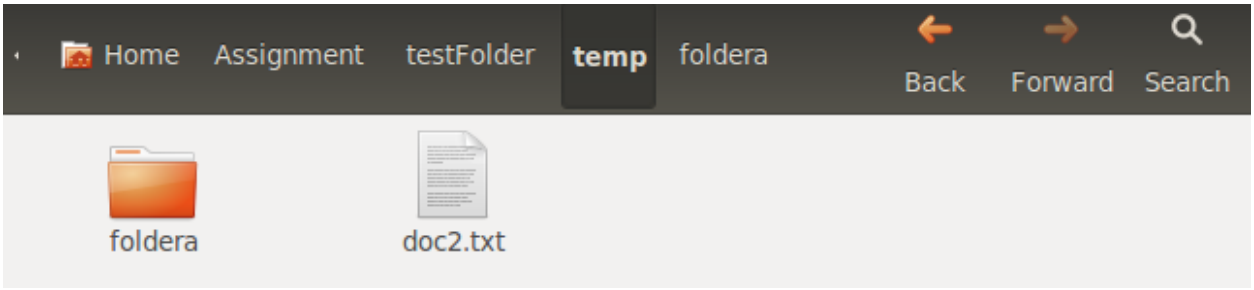
```
[05/22/2016 02:35] seed@ubuntu:~/Assignment$ sh ./script.sh
Commands:
 -c to generate validation file.
 -o to check for changes
 -e to exit application
 Enter a Command: -c
Validation File Created

Commands:
 -c to generate validation file.
 -o to check for changes
 -e to exit application
 Enter a Command: -e
[05/22/2016 02:37] seed@ubuntu:~/Assignment$
```

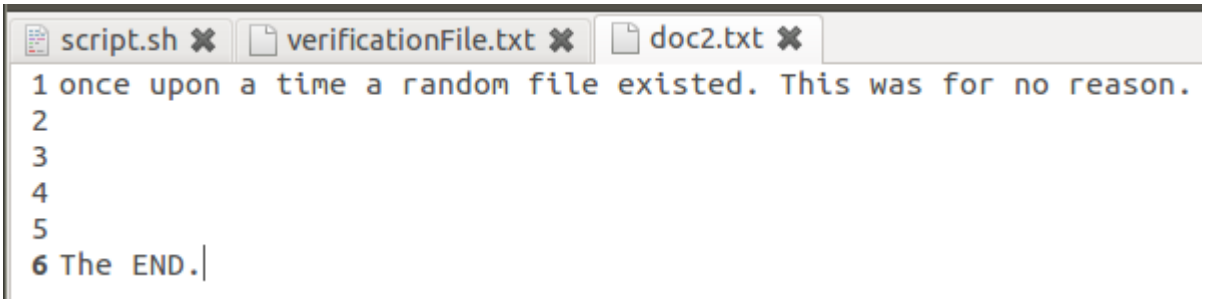
7.2 APPENDIX 2

```
1 /home/seed/Assignment/testFolder/Ubuntu.jpg Ubuntu.jpg rwxrw-rw- seed seed file 2013-08-25 09:58:56.000000000 -0700 2016-05-21
  21:31:31.499456003 -0700 77129 2256ef0a5e9d26cec4aea2bbbdf87511
2 /home/seed/Assignment/testFolder/script.sh~ script.sh~ rwxr-xr-x seed seed file 2016-05-15 23:00:23.367286034 -0700 2016-05-15
  23:01:44.935286034 -0700 126 e7659692ad4f886e4bd220f3028f222a
3 /home/seed/Assignment/testFolder/temp temp rwxrwxr-x seed seed d 2016-05-22 02:38:43.943456002 -0700 2016-05-22 02:38:43.943456002 -0700
4 /home/seed/Assignment/testFolder/temp/doc2.txt doc2.txt rw-rw-r-- seed seed file 2016-05-22 02:38:43.939456004 -0700 2016-05-22
  02:38:43.943456002 -0700 77 e2c851effe8a9f64f6b27cb934dffc85
5 /home/seed/Assignment/testFolder/temp/doc2.txt~ doc2.txt~ rw-rw-r-- seed seed file 2016-05-22 02:37:49.823456003 -0700 2016-05-22
  02:38:43.943456002 -0700 0 d41d8cd98f00b204e9800998ecf8427e
6 /home/seed/Assignment/testFolder/temp/foldera foldera rwxrwxr-x seed seed d 2016-05-22 02:38:18.663456003 -0700 2016-05-22
  02:38:18.663456003 -0700
7 /home/seed/Assignment/testFolder/temp/foldera/words.txt words.txt rw-rw-r-- seed seed file 2016-05-22 02:38:18.643456002 -0700 2016-05-22
  02:38:18.663456003 -0700 28 b36e1b7586a1aeef1c784549a0414dce
8 /home/seed/Assignment/testFolder/temp/foldera/words.txt~ words.txt~ rw-rw-r-- seed seed file 2016-05-22 02:37:58.663456002 -0700 2016-05-22
  02:38:18.663456003 -0700 0 d41d8cd98f00b204e9800998ecf8427e
9 /home/seed/Assignment/testFolder/testText.txt~ testText.txt~ rw-rw-r-- seed seed file 2016-05-15 20:04:23.626744090 -0700 2016-05-15
  20:04:54.202024000 -0700 0 d41d8cd98f00b204e9800998ecf8427e
10 /home/seed/Assignment/testFolder/textDoc1.txt textDoc1.txt rw-rw-r-- seed seed file 2016-05-22 02:39:16.339456002 -0700 2016-05-22
   02:39:16.395456001 -0700 10 7931a56bc0100176574d25196c23f6f7
11 /home/seed/Assignment/testFolder/textDoc1.txt~ textDoc1.txt~ rw-rw-r-- seed seed file 2016-05-22 02:37:27.231456003 -0700 2016-05-22
   02:39:16.395456001 -0700 0 d41d8cd98f00b204e9800998ecf8427e
12 /home/seed/Assignment/testFolder/textDoc2.txt textDoc2.txt rw-rw-r-- seed seed file 2016-05-22 02:39:09.191456003 -0700 2016-05-22
   02:39:09.195456003 -0700 4 26ff1e6c6fe697602efa85f9004f5a9f
13 /home/seed/Assignment/testFolder/textDoc2.txt~ textDoc2.txt~ rw-rw-r-- seed seed file 2016-05-22 02:38:56.099456003 -0700 2016-05-22
   02:39:09.195456003 -0700 0 d41d8cd98f00b204e9800998ecf8427e
```

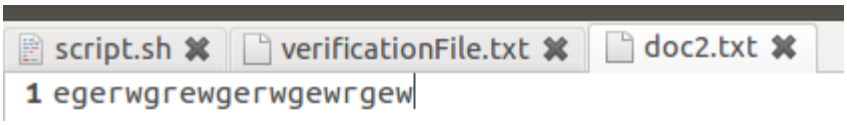
7.3 APPENDIX 3

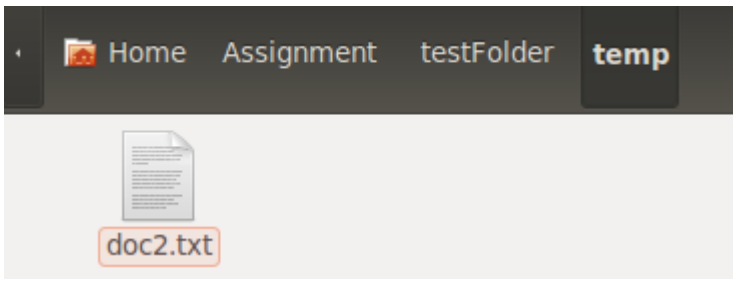


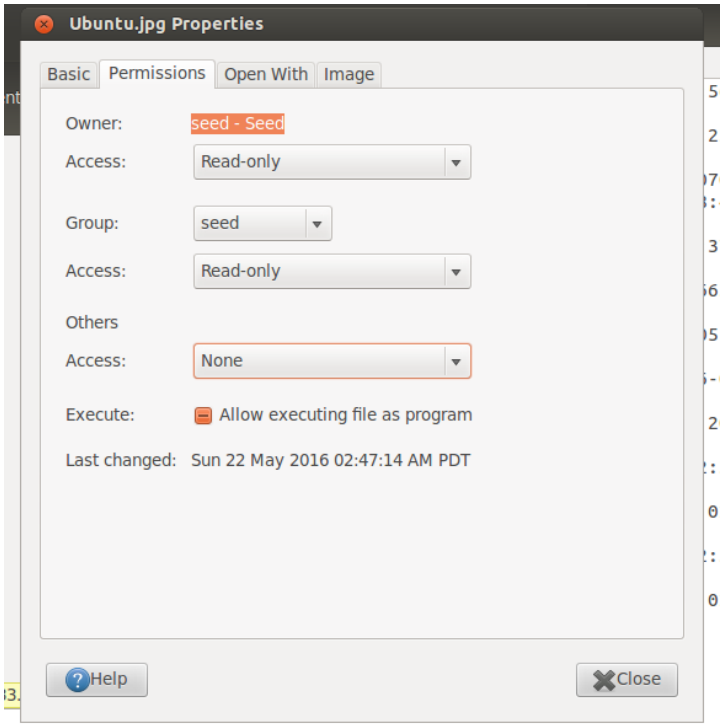


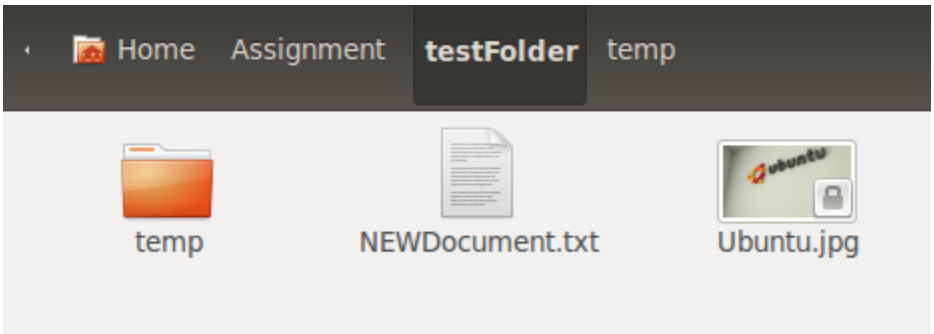


7.4 APPENDIX 4

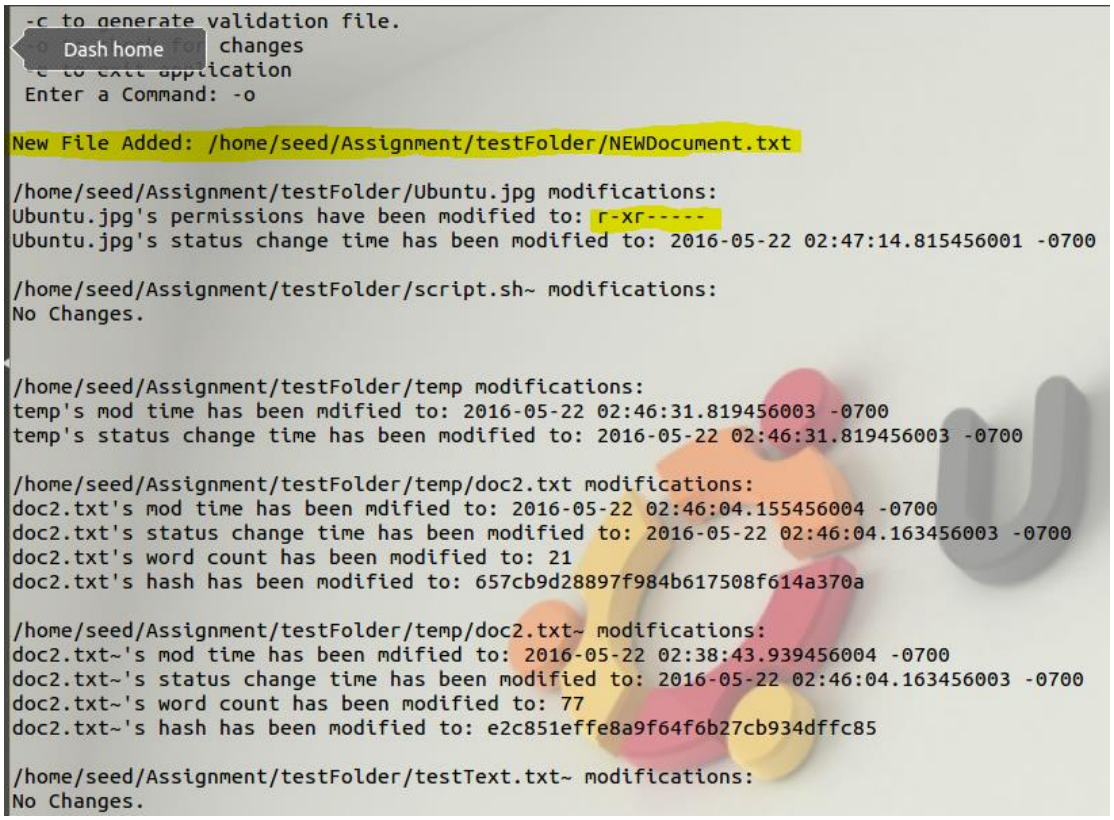


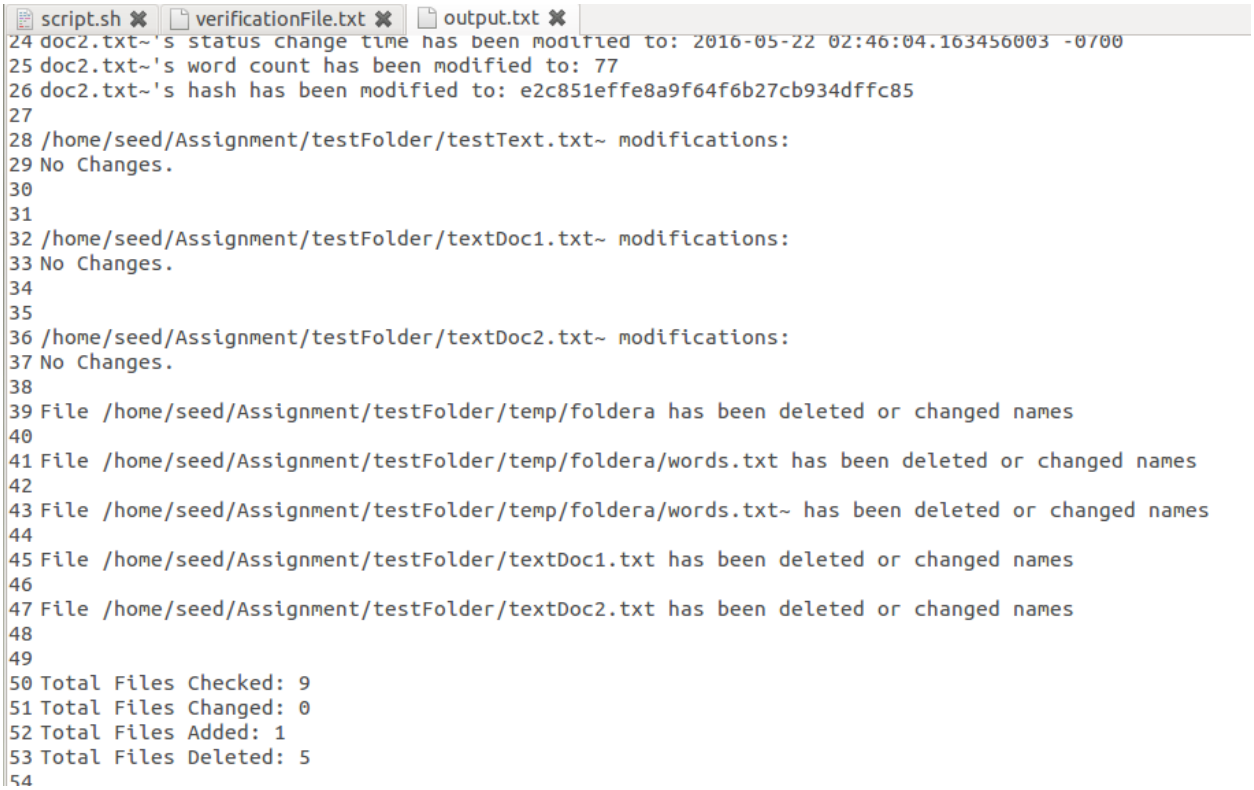






7.5 APPENDIX 5





7.6 APPENDIX 6

```
#!/bin/sh

if [ ! -f "/home/seed/Assignment/verificationFile.txt" ]
        then touch /home/seed/Assignment/verificationFile.txt
fi

getValues () {
        for i in $1
                do
                        type=`ls -ld $i | awk '{ print $1 }' | cut -c -1`
                        name=`basename $i`
                        perm=`ls -ld $i | awk '{ print $1 }' | cut -c 2-`
                        type=`ls -ld $i | awk '{ print $1 }' | cut -c -1`
                        owner=`ls -ld $i | awk '{ print $3 }'`
                        group=`ls -ld $i | awk '{ print $4 }'`
                        modTime=`stat --printf="%y" $i`
                        changeTime=`stat --printf="%z" $i`
```

```
                        if [ $type = "l" ]
                                then
                                        type="symLink"
                                        path="$3`ls -ld $i | awk '{ print $9 }' | cut -c 2-`"
                        elif [ $type != "l" ]
                                then
                                        path=`readlink -f $i`
                        fi

                        if [ $type = "d" ]
                                then
                                        l=$i
                                        l=$l"/*"
                                        echo "$path $name $perm $owner $group $type $modTime
$changeTime"
                                        if [ 0 != $(ls -1 $i | wc -l) ]
                                                then
                                                        getValues "$l" "$2" "$3"
                                        fi

                        elif [ $type != "d" ]
                                then
                                        if [ $type != "symLink" ]
                                                then type="file"
                                        fi
                                        hash=`md5sum $i | awk '{print $1 }'`
                                        count=`wc -m $i | awk '{print $1 }'`
                                        echo "$path $name $perm $owner $group $type $modTime
$changeTime $count $hash"
                        fi

        done >> $2
}

generateFile() {
        >"$2"
        getValues "$1" "$2" "$3"
        echo "Validation File Created \n"
}

checkDelete(){
        while read line
                do
                        path=$(echo "$line" | awk '{ printf $1 }')
                        if [ ! -e $path ]
                                then
                                        echo "File $path has been deleted or changed names \n" | tee -a
output.txt
                                        delCount=`expr $delCount + 1`
                        fi
        done < verificationFile.txt
        </dev/null
}

checkFile() {
```

```
        if grep -q "$path" /home/seed/Assignment/verificationFile.txt
                then

                        echo "\n$1 modifications:" | tee -a output.txt
                        if [ "$perm" != "$(grep "^$1\s" /home/seed/Assignment/verificationFile.txt | awk
'{ printf $3 }')" ]
                                then
                                        echo "$name's permissions have been modified to: $perm" | tee -
a output.txt
                                        modified=true
                        fi
                        if [ "$owner" != "$(grep "^$1\s" /home/seed/Assignment/verificationFile.txt | awk
'{ printf $4 }')" ]
                                then
                                        echo "$name's owner has been modified to: $owner" | tee -a
output.txt
                                        modified=true
                        fi
                        if [ "$group" != "$(grep "^$1\s" /home/seed/Assignment/verificationFile.txt | awk
'{ printf $5 }')" ]
                                then
                                        echo "$name's group has been modified to: $group" | tee -a
output.txt
                                        modified=true
                        fi
                        if [ "$type" != "$(grep "^$1\s" /home/seed/Assignment/verificationFile.txt | awk
'{ printf $6 }')" ]
                                then
                                        echo "$name's type has been modified to: $type" | tee -a
output.txt
                                        modified=true
                        fi
                        if [ "$modTime" != "$(grep "^$1\s" /home/seed/Assignment/verificationFile.txt |
awk '{ printf $7 " "$8 " " $9 }')" ]
                                then
                                        echo "$name's mod time has been mdified to: $modTime" | tee -a
output.txt
                                        modified=true
                        fi
                        if [ "$changeTime" != "$(grep "^$1\s" /home/seed/Assignment/verificationFile.txt |
awk '{ printf $10 " "$11 " " $12 }')" ]
                                then
                                        echo "$name's status change time has been modified to:
$changeTime" | tee -a output.txt
                                        modified=true
                        fi

                        if $2
                                then
                                        if [ "$count" != "$(grep "^$path\s"
/home/seed/Assignment/verificationFile.txt | awk '{ printf $13 }')" ]
                                                then
                                                        echo "$name's word count has been modified to:
$count" | tee -a output.txt
                                                        modified=true
                                        fi
```

```
                                        if [ "$hash" != "$(grep "^$path\s"
/home/seed/Assignment/verificationFile.txt | awk '{ printf $14 }')" ]
                                                then
                                                        echo "$name's hash has been modified to:
$hash" | tee -a output.txt
                                                        modified=true
                                        fi
                        fi

                        if ! $modified
                                then
                                        echo "No Changes. \n" | tee -a output.txt
                        fi
        elif ! grep -q "$path" /home/seed/Assignment/verificationFile.txt
                then
                        echo "\nNew File Added: $path" | tee -a output.txt
                        addCount=`expr $addCount + 1`
        fi

}

getChanges () {
        for i in $1
                do
                        totCount=`expr $totCount + 1`
                        modified=false
                        type=`ls -ld $i | awk '{ print $1 }' | cut -c -1`
                        name=`basename $i`
                        perm=`ls -ld $i | awk '{ print $1 }' | cut -c 2-`
                        owner=`ls -ld $i | awk '{ print $3 }'`
                        group=`ls -ld $i | awk '{ print $4 }'`
                        modTime=`stat --printf="%y" $i`
                        changeTime=`stat --printf="%z" $i`

                        if [ $type = "l" ]
                                then
                                        type="symLink"
                                        path="$2`ls -ld $i | awk '{ print $9 }' | cut -c 2-`"
                        elif [ $type != "l" ]
                                then
                                        path=`readlink -f $i`
                        fi
                        if [ $type = "d" ]
                                then
                                        l=$i
                                        l=$l"/*"
                                        checkFile "$path" "false"
                                        if [ 0 != $(ls -1 $i | wc -l) ]
                                                then
                                                        getChanges "$l"
                                        fi

                        elif [ $type != "d" ]
                                then
                                        if [ $type != "symLink" ]
                                                then type="file"
```

```
                                        fi
                                        hash=`md5sum $i | awk '{print $1 }'`
                                        count=`wc -m $i | awk '{print $1 }'`
                                        checkFile "$path" "true"
                        fi

        done
}

compareToFile()         {
        >output.txt
        delCount=0
        addCount=0
        totCount=0
        chngCount=0

        getChanges "$1" "$2"

        checkDelete
        #checkNew "$1"

        echo "\nTotal Files Checked: $totCount" | tee -a output.txt
        echo "Total Files Changed: $chngCount" | tee -a output.txt
        echo "Total Files Added: $addCount" | tee -a output.txt
        echo "Total Files Deleted: $delCount \n" | tee -a output.txt
}



while true; do
        read -p "Commands:`echo '\n '`-c to generate validation file.`echo '\n '`-o to check for
changes`echo '\n '`-e to exit application`echo '\n '`Enter a Command: " inp
        case $inp in
        -c ) generateFile "./testFolder/*" "verificationFile.txt" "/home/seed/Assignment"; ;;
        -o ) compareToFile "./testFolder/*" "/home/seed/Assignment"; ;;
        -t ) testCommands; ;;
        -e ) exit;;
        * ) echo "Command not valid. Please Try Again";;
esac
done
```