File Integrity

Estimated completion time: 1 hour

One difficult question facing system administrators is this: how can I tell if my system has been hacked? Yet another difficult question is: in the event that an attack is identified, how can I tell which files were modified, or deleted, or added? With some advance planning, these questions can be answered with confidence. Having logging enabled and architected is one step in that direction. This lab exposes the student to some additional tools that can help.

The objective of this exercise is to provide the student with a hands-on way to appreciate the concept of system integrity.

Heads up: A simple list of Unix commands is given in an Appendix at the end of the document.

I. Getting Started

Boot your Linux system or VM. If necessary, log in and then open a terminal window and cd to the labtainer/labtainer-student directory. The pre-packaged Labtainer VM will start with such a terminal open for you. Then start the lab:

labtainer file-integrity

Note the terminal displays the paths to two files on your Linux host:

1) This lab manual

2) The lab report template

On most Linux systems, these are links that you can right click on and select "Open Link". **If you chose to edit the lab report on a different system, you are responsible for copying the completed report back to the displayed path on your Linux system before using "stoplab" to stop the lab for the last time.**

Use `su` to elevate your privileges to root, using badpassword as the password.

II. Poor Mans' Integrity

This section starts the introduction into system integrity by building your own simple integrity system using only the utilities provided by a typical CentOS installation, and which are typically found on other Unix variants, though they may have different names.

Most freeware versions of Unix come with a utility that will calculate a digest of a file. In CentOS one such utility is known as `sha1sum`.

1. Create a new file named `tempfile` with some content of your choosing.

2. Calculate a digest for this new file by doing the following:

   ```
   sha1sum  tempfile
   ```

   The digest for `tempfile` is displayed on the screen.

3. Now redirect the output of `sha1sum` to a file called `hashes.txt`, as shown below:

   ```
   sha1sum  tempfile >  hashes.txt
   ```

4. Modify the content of `tempfile` and then save the changes.

5. Recalculate the digest for the changed file:

   ```
   sha1sum  tempfile
   ```

   By looking at the two digests (which should be different) you can determine that the file has changed.

6. Use an automated approach

   Instead of a visual inspection of digests, you can rely on an automated approach by using a `sha1sum` option to detect if the file has changed. In step #3 above you saved the original digest for `tempfile` in the file called `hashes.txt`. Use `sha1sum` as shown below to determine whether the file has changed:

   ```
   sha1sum --check hashes.txt
   ```

7. To get a bigger view of the system, build on this idea by calculating digests for a lot of the critical files by executing the following commands[1]:

   ```
   sha1sum   /usr/bin/*     >  hashes.txt
   sha1sum   /usr/sbin/*    >> hashes.txt
   sha1sum   /etc/*         >> hashes.txt

   cat  hashes.txt
   ```

   Notice the appending!
   (Here and in other places)

1 You will see some errors from `sha1sum` complaining about directories, but that is OK.

8. To determine how many files were hashed, you can use the following to count the number of lines in the file[2]:

   ```
   wc -l hashes.txt
   ```

   The output from the above command should have been more than 800. [If you have a lot less, then you may not have been appending the output in step #7.]

   If this was a production system this would not be a complete list of files to worry about, but it is a good start. More files really ought to be added to the list, and some files ought to be removed because they change often due to normal activity.

9. Verify that the digests calculate properly:

   ```
   sha1sum  --check  hashes.txt
   ```

   Because the digests were computed only seconds earlier, there should be no failures. But any time in the future that any of those files change, the above command will display it.

   The astute observer will have noticed that this will only help to identify when existing files have been modified or deleted. How can you tell which files or directories have been *added* by a hacker? To answer this question we must return to the `find` command.

   `find` can be used to display all the files and directories in a hierarchy, which can be redirected to a file and saved. This can then be compared at a future time using the `diff` command, which takes two text files as input and displays the differences between them.

10. Execute the following commands to build a list of existing files and directories:

    ```
    find /usr/bin  -print    >   myfiles
    find /usr/sbin -print    >>  myfiles
    find /etc      -print    >>  myfiles

    cat myfiles
    ```

---

2 `wc` stands for *word count*. Without any options `wc` will tell you how many words are in a text file. Using the `"-l"` option (lower-case L, not the number 1), it counts the number of *lines* in a file.

11. Imagine some significant time has passed since creating the list of files above. Acting as a hacker, add a new (empty) file into the `/bin` directory by using the `touch` command:

    ```
    touch  /usr/bin/dummyfile
    ```

12. Imagine again that some time has passed, and you (as the system administrator) want to verify that no new files have been added. Recreate the hierarchy of files by doing the following:

    ```
    find /usr/bin  -print    >   tempfile
    find /usr/sbin -print    >>  tempfile
    find /etc      -print    >>  tempfile
    ```

13. Now compare the "old" list with the new list:

    ```
    diff  myfiles  tempfile
    ```

    You have now identified the added file.

    In a real environment, the "hashes.txt" and "myfiles" files would be copied onto a read-only removable medium. Then a script could be written to execute on a regular basis to check the integrity of the system; if a mismatch is observed, then the cause of the change can be investigated. Or in another scenario, you have confirmed hacker activity, but you don't know what the hacker changed or deleted; this approach would tell you.

III. AIDE

In this section you will be using an open source integrity product called Advanced Intrusion Detection Environment (AIDE), which was introduced to you in the lecture. It saves a lot more than digests. AIDE is not usually installed in the CentOS distribution, but it is installed on this Labtainer.

1. First, to make sure you have a somewhat repeatable experiment, tell the OS to drop all its caches:

    ```
    sysctl -w vm.drop_caches=2
    ```

2. Build the database

   Using the default AIDE configuration, build the integrity database by executing the following exactly as shown:[3] [4]:

   ```
   date ;  aide --init ;  date
   ```

   Record in Item #1 of the Worksheet how long it took to complete the building of the database using the default AIDE configuration. [Provide the information in minutes and seconds whenever you are asked to write "how long?".]

3. Change the AIDE configuration to maximize digest calculations.

   You will now change the default AIDE configuration such that two digests are saved per any given file, instead of the default action of saving one SHA256 digest for each file.

   a. As root, open the `/etc/aide.conf` AIDE configuration file.
   b. Go to line in the file, that reads:
      `CONTENT_EX=sha256+ftype+p+u+g+n+acl+selinux+xattrs`
      And change it by adding "sha512+" in front of "sha256".

   c. Save the file and exit the editor.

4. Once again, tell the OS to drop all its caches:

   ```
   sysctl –w vm.drop_caches=2
   ```

5. Execute the following to rebuild the database using the modified configuration:

   ```
   date ;  aide --init ;  date
   ```

   Record in Item #2 of the Worksheet how long it took to complete the building of the database using an additional digest.

   Note that Items #3 and #4 of the Worksheet ask you to analyze the times you recorded in Items #1 and #2.

---

3 Multiple commands can be put on one command line by separating them with a semi-colon. In this example the shell will execute the first `date` command, and when the `date` command has completed it will execute the `aide` command, and then finally the second `date` command. This will allow you to see how long it takes the `aide` command to execute.

4 Be warned that this can take anywhere between 2 and 15 minutes, depending on the power of your host OS and the underlying hardware. Note that this would probably take less time if the OS were running on "bare metal" instead of a VM.

6. Execute the following to make a few changes to the file system:

```
cp  /etc/rsyslog.conf  /etc/rsyslog.copy
echo "# another comment" >> /etc/rsyslog.conf

cp  /usr/bin/passwd  /usr/bin/passwd.copy
echo "  " >> /usr/bin/passwd

chmod ugo=rwx /bin/logger
```

7. Execute the following to generate an AIDE report on the current state of the files.

```
cd /var/lib/aide
mv  aide.db.new.gz  aide.db.gz
aide --check
```

   When done, the AIDE report will eventually scroll off the screen, but do not worry because the results will also be written to a log file. Do not continue with the instructions until the report has been generated. [The verification will take longer than the database generation.]

8. As root, open **`/var/log/aide/aide.log`**, which is the location where the above results were written.

   The file starts with a short summary of the detected changes, followed by a summary of the files that were added, followed by a summary of the files that changed in some way, followed by the details about each change.

   In step 6 above, you created two new files, modified the content of two files, and changed the permissions on one file.

   By looking through the AIDE log file, record in Item #5 of the Worksheet the change(s) that were not detected by AIDE.

14. After finding the change(s) that were not detected by AIDE, refer to the AIDE configuration file at `/etc/aide.conf` to explain why the above change(s) were not detected.

   Record your explanation in Item #6 of the Worksheet.

15. Refer back to the AIDE log file at `/var/log/aide/aide.log` to find the detailed information about the changes made to the `logger` command. What would an administrator learn about this change from the information provided?

   Record your answer in Item #7 of the Worksheet.

16. Assume your boss is very paranoid about the shadow password file at `/etc/shadow`. He is OK with having only one digest for all the other files on the system, but he wants to have several digests generated for the shadow password file (in addition to everything else that is being saved for the file). How could you change the AIDE configuration file to meet his request? Be specific.

   Record your answer in Item #8 of the Worksheet.

IV. Submission

After finishing the lab, go to the terminal on your Linux system that was used to start the lab and type:

```
stoplab file-integrity
```

If you modified the lab report on a different system, you must copy that completed file into the directory path displayed when you started the lab, and you must do that before typing “stoplab”. When you stop the lab, the system will display a path to the zipped lab results on your Linux system.
Provide the zip file to your instructor, e.g., via the Sakai site.

Appendix – Some Unix Commands

`cat` — Display the contents of a text file to the terminal.

```
cat filename
```

`cd` — Change directory

```
cd  location
```

With no "location", you will be taken to your home directory.

`chmod` — Change the DAC permissions on a file or directory.

```
chmod permissions objectname
```

Consult Lab 1 for examples.

`diff` — Show the difference between two text files

```
diff file1 file2
```

`echo` — Display a string on the terminal.

```
echo string
```

Often used in scripts or to overwrite/create a file, such as:

```
echo string > file
```

Or to append to an existing file, such as

```
echo string >> file
```

`find` — Find an object with a given kind of attribute. The basic use of find is to list all the files and directories in a given hierarchy:

```
find directorypath –print
```

`ls` — List the contents and/or attributes of a directory or file

```
ls  location
ls  file
```

With no "location" or "file" it will display the contents of the current working directory.

`man` — Manual

```
man command
```

Displays the manual page for the given "command". To see another page press the space bar. To see one more line press the Enter key. To quit before reaching the end of the file enter 'q'.

`more` — Display a page of a text file at a time in the terminal

```
more file
```

To see another page press the space bar. To see one more line press the Enter key. To quit before reaching the end of the file enter 'q'.

`mv`	Move a file and/or change its name. `mv currentname newlocation_andor_name`
`pwd`	Display the present working directory `pwd`
`su`	Super user (change to root)
`touch`	Change the modification date on the given file. If the file does not exist, it will be created. `touch filename`
`wc`	Count the number of words in a given text file. Given the "-l" option (for "lines"), it will return the number of lines in a text file: `wc –l filename`