Student Name: _____ **Weight:** 20%

Student ID: _____ **Marks:** /20

Assignment: Creating a Tripwire Script

Scenario

You are a software specialist given the task of creating a tripwire script that can be executed on various directories. By comparing the directory content against a static/pre-existing record, the tripwire script will determine whether files have been added, removed or altered.

Review the example below before following the instructions that follow.

Example of how a tripwire script works

1. A script is run as “python tripwire.py tripwireDir tripwireRecord c” where the tripwire.py is your script.
    - “tripwireDir” is the name of the directory that needs to be evaluated.
    - “tripwireRecord” is the name of the record file.
    - “c” stands for “create” in order to create a record file containing the static information (i.e., the four arguments).
2. When the above command is run on a given directory, the script will read all of the files that currently exist in that directory. The files will be read one at a time

    The script will then run a hash algorithm on each file to come up with a checksum, and then write that information into a “tripwireRecord” file in that same directory. The file will contain a heading that specifies the directory that was evaluated. The file will also contain the filename and hash pair. These will look similar to the following example:

```
/home/marceltozser/Documents/test_folder
file1.txt e5ce4db216329f4fccb47b27272f9eb3
file2.txt 86fe6e51d96a44c2f31bedeabb8397be
```

Figure 1: tripwireRecord File

Source: Southern Alberta Institute of Technology

    Ideally, this tripwireRecord file will be stored securely in another directory so that it cannot be altered.

3. The directory content is evaluated against the static “tripwireRecord” content.
    a. The script will run as “python tripwire.py tripwireRecord” where the tripwire.py is your script and the “tripwireRecord” is the name of the record file (two arguments).

   b. Only two arguments are required because the tripwireRecord has the first entry specifying the directory that needs to be evaluated.

4. When the above command is run, your script will do the following by evaluating the directory specified:
   a. For each file name, it will check that the hash has not been changed (which indicates that the file has been modified).
   b. It will keep track of which file names no longer exist and which files have been added.
5. The script will write a report to the console specifying the file names that have been modified, missing, or added in the following format:

```
/home/marceltozser/Documents/test_folder

Modified:
file1.txt

Removed:
file2.txt

Added:
file3.txt
```

Figure 2: Script Report Specifying Files Modified, Missing, or Added

Source: Southern Alberta Institute of Technology

In this example, the file1.txt hash equivalent would have changed, indicating that the file has been modified and that file2.txt was removed from that directory. The example also shows that file3.txt has either been added, or that file2.txt has been renamed as file3.txt.

The purpose of providing this example is to demonstrate that you can run your script on any directory to create a record of the state, and then rerun your script again later with the existing static tripwireRecord file to see if anything has changed in that directory. This technique helps to identify what is going on in a directory so that you can detect problems when troubleshooting system issues.

Instructions

1. In Linux, create a test directory and add at least two different text files with your first and last name inside the content of each file.
2. Create a script called “tripwire.py” that will take in either two arguments (in order to compare the static tripwireRecord content against the current state of a directory) or take in four arguments (in order to create the tripwireRecord).
3. Create a function that will simply read file content and return a hash result. There are many hash algorithms that are available. You can use the following example:

- fileHash = hashlib.md5(data).hexdigest()

4. Read each file content as “rb” to read bytes and then you can run the above statement to create a hash result that you can store and use for comparison purposes.

   Note: Hash algorithms are complex and the results change drastically just by changing one byte in a file. This is why they are used in many applications.

5. Once you have run your script and created a tripwireRecord file containing the file name and hash pair, you should be able to add more functionality. This will enable you to handle the evaluation aspect that compares the content of the tripwireRecord file against the current directory content.

   Note: You will need to check various scenarios by altering the tripwireRecord file content or by changing the directory content by modifying, removing or adding a file.

6. With the tripwireRecord file now available, there are many different approaches to comparing the directory against the tripwireRecord content. Here is one of these approaches, as an example:

   a. Read the tripwireRecord to get the directory name and the filename – hash pairs.

   b. Read the content of the current directory and for each file to check if a record exists in the tripwireRecord. If the record does exist, compare the hash values to check if the file is identical. If the file does not exist in the tripwireRecord, this means that the file has been added to the directory after the tripwireRecord file has been created. Add the results into two different buckets (such as arrays).

   c. After evaluating all of the files in the current directory, there may be some file names that have not been compared in the tripwireRecord. In this case, you can assume that those files in the current directory have been removed.

   d. Keep track of the file names in some data structure (e.g., three different arrays like modified, added, removed).

   e. Write the results to the console.

7. Incorporate error handling such as arguments passed in, file handling, etc.

Deliverables

1. Submit a 10-minute demo of you running your script in your environment. Your demo must meet the following requirements:

   a. Shows you verbally describing your code and what is happening at execution.

   b. Proves that all of the functionalities of your script work.

   c. Do not submit a link to YouTube or other public video sharing service. You must submit your work in D2L. Failure to do this will result in a 0% grade.

      Note: The demo must be no longer than 10 minutes. Content beyond 10 minutes will not be watched.

Python criteria

- Submission of the script that includes Pseudocode.

Note: Pseudocode is in a block comment at the beginning of the code.

Marking Criteria

Categories	0	1	2	Score
Functions	None	Not called or no practical usable code	Properly implemented	**/2**
Lists	None	Limited (no indexing or iteration)	Properly implemented	**/2**
Loops	None	Limited (missing purpose or appropriate looping limits)	Properly implemented	**/2**
Reading and Writing Files	None	Limited (not reading or writing all of the required data)	Properly implemented	**/2**
Error Handing	None	Limited (missing error handling techniques such as working with resources, casting data types, etc.)	Properly implemented	**/2**
Arguments Evaluated Properly	None	Limited (missing error handling or not addressing the various input scenarios)	Properly implemented	**/2**

Proper Output Formatting	None	Limited (output file content is difficult to interpret)	Properly implemented	**/2**
Script Executes as Expected	None	Limited (some aspects of the code do not execute in all cases, and may produce errors)	Properly implemented	**/2**
Demo	None	Limited (problems encountered during the demo without adequate resolution, does not prove that coding solution works, or missing code explanation)	Properly implemented	**/2**
Code Readability	Insufficient (very difficult to follow the coding solution; lacks organization and logical structure)	Limited (missing adequate pseudocode, commenting, or code structure is difficult to follow)	Properly implemented	**/2**
			Total	**/20**