Assignment: Creating a Tripwire Script

Team members: Maryam Bunama, Eric Russon, Roman Kapitoulski

CPRG-217 | Date Due: May 30, 2023 | Instructor: Sanoop Sadique

>> Table of Contents

>> Assignment Problem

In this assignment, we were tasked with creating a tripwire script using Python to check if the files in a specific directory have been tampered with (such as files being added, removed, or altered). The tripwire script has to use hashing to perform the cross-check, accept and identify arguments from the terminal, and incorporate error-handling.

>> Solution

>> Please see the attached code!

Before we even started with working on the assignment, we began by discussing our roles. We each highlighted our strengths and weaknesses and assigned roles for ourselves (more details under "Each Member's Role"). Then, we started by creating the code and pseudocode. Although pseudocode is usually completed first, our initial draft was lacking, and we got started with the code instead. After we completed our code, we wrote down the pseudocode. Then at the end, we created the documentation to wrap up this assignment.

For creating the code, we used Python and utilized the "sys", "os", and "hashlib" modules. The "sys" module allowed us to pass arguments through the terminal shell, the "os" module allowed us to work with files and file paths, and finally, the "hashlib" module allowed us to create the hash with which we compared file information.

We utilized creating functions (5 in total) to simplify the code. The first function we created was one to generate the hash (because it was used in multiple places). The second function was to convert list data into string data. This function was not initially planned but it proved useful when we kept running into an issue where the hash couldn't be created because it couldn't read the list. The third function we created was one for generating the tripwireRecord file. Initially, we didn't need to have this as a function. However, because our code had two instances where it needed to create the record file, we found it was much easier to keep the code in a function. The fourth function was used to pass an error message if the user's input was incorrect. Finally, the fifth function exits the user from the program.

We then structured the program so that it uses sys.args to accept 2 scenarios, one for having the user input 4 arguments, and the other for having the user input 2 arguments.

When the user inputs 4 arguments, the code evaluates the files in the chosen directory using a for loop(which has to be in the same directory as the python file), and returns a hash of the contents of each file. Then, it writes the contents to the record file with proper formatting. This mode checks to make sure that the directory name matches with the available directories. It will print an error if it doesn't. Also, if a tripwireRecord file already exists, the user has the choice to either overwrite it, or exit the program.

When the user inputs 2 arguments, the code takes the record file and creates a new hash of the chosen directory files to evaluate against the record file. It uses a for loop to iterate through each hash and uses if statements to organize the comparison results. Then, it will print on to the terminal if a file was added, modified, or removed. If the file didn't change, the file name won't be printed. This mode only works if the record file already exists. Otherwise, it will print an error and exit.

>> Pseudocode

---

1. Do a check for a valid number of arguments.
    a. Check if there are 4 valid arguments.
        i. If it is, proceed to step 2.
        ii. If not, return an error.
    b. Check if there are 2 valid arguments.
        i. If it is, proceed to step 3.
        ii. If not, return an error.
    c. Check if arguments do not match the requirements.
        i. If arguments are invalid, return error and end program.
2. 4 valid arguments (Record file creation mode)
    a. Do a check for a valid directory argument.
        i. Check to see if the directory exists and that it is a directory.
            1. Give an error if the directory does not exist/is not a directory.

b. Do a check for a pre-existing records file.
    i. Give a notification if the record file already exists.
        1. Check to see if the user would like to override that file.
            a. If so, commence the override.
            b. If not, end the program.
    ii. If none currently exist, continue to step 4.
c. Create a records file for the files found in the directory.
    i. Write the directory's absolute path to the records file.
    ii. Read the contents of the directory.
    iii. For each file, create a key value pair.
        1. Key is the name of the file.
        2. Generate Hash code using a cryptographic algorithm.
            a. Use Hashcode as the Value.

3. 2 Valid Arguments (Record file comparison mode)
a. Check to ensure the record file exists
    i. Give an error if it is not the records file.
b. Compare the directory files with the records file.
    i. Check to see if the file has been modified.
        1. If the hash doesn't match, the file is modified
        2. If the hash matches, do nothing (the file didn't change)
    ii. Check to see if any new files were created.
        1. If the filename is not in the record, a new file was created
    iii. Check to see if any of the files were removed.
        1. If the filename in the record cannot be found, the file was deleted.
c. Show the user the modified, added, and removed files as a terminal output.

>> Each Member's Role

We agreed to divide each task in the programming process so that everyone can focus on a specific aspect based on their abilities.

>> Eric Russon

Eric is pretty comfortable with writing pseudocode and thus, he handled this stage in the process. Going off of what is pointed out

in the assignment document and what our instructor covered in class, he wrote the pseudocode while adjusting to Roman's coding style.

>> Maryam Bunama
Maryam was in charge of creating the documentation for this project, and she also contributed to troubleshooting the code as it had an issue with printing the record file which was caused by a problem in the path handling. and helping to finalize it. Her less formal roles included creating the Github repository and checking in with the team to make sure everyone is in agreement over decisions.

>> Roman Kapitoulski
Having completed two semesters in Software Development, Roman was the most experienced out of the team when it comes to coding. He handled the majority of the coding process by figuring out the code's skeleton, logic, error handling, and comments. The team supported his work through code troubleshooting, final quality checks, adding more comments, and handling the other parts of the assignment.

>> Brief Summary

With everyone's effort and contribution, we successfully created a tripwire script that can run on Linux and Windows. It has two functionalities. The first functionality is for creating a record file containing hash values of a chosen directory. The second functionality is for comparing the created record file against the chosen directory.