**CACIE Tool #00** – ***Fingerprinter Tool (fingerprint.py)***

**Version** **1.1**

**QA**: **QA**

1. **Description and Purpose**

The Fingerprinter tool is a low-level utility that generates a 256-bit hash tag for a specified file or all files and subdirectories in a specified directory. The hash tag represents a unique value and hashes of two sets of data (i.e., two directories or two files) will match if and only if the corresponding data within the directories and/or files also match exactly. Small changes to the data within a file or directory will result in large, unpredictable changes in the hash.

1. **Functional Requirements**

The following are the functional requirements of the Fingerprinter tool.

FR-1: Generate a hash tag for an individual file

FR-2: Generate a hash tag for a file directory and recursively generate a hash tag for each file contained within the file directory and its subdirectory(ies).

FR-3: Generate an output file that lists a datetime stamp and the path and filename and respective hash tag for a file or directory specified to be fingerprinted.

FR-4 Generate a hash tag for a section of a file

FR-5 Specify if the output overwrites or appends to the target output file

1. **Software Requirements Specifications**

The following documents the software requirements for the Fingerprinter tool.

Python >=3.5

Python Standard Libraries:  
hashlib  
os  
argparse  
datetime  
sys

1. **Software Design Description**

The following is a brief description of the required arguments and the output generated by the Fingerprinter tool:

Positional Arguments:

* target: file or directory path to be fingerprinted

Optional Arguments:

* -h, --help
* -o, --output OUTPUT [path and filename of output file; default is fingerprint.txt]
* --outputmode {a, w} Either appends (a) or overwrites (w) the output file.
* --lineranges LINERANGES If this argument is present, the tool will fingerprint a subsection of the target file. LINERANGES must be a nested list of the form [section A, section B, …] where each section is a list of line numbers defining the range of that section. The end line is NOT read. Fore example: --linerange=[[0,1],[2,3]] fingerprints the first and third lines of the file (assuming –lineindexing=0). Likewise, --linerange=[[0,9]] fingerprints the first eight lines only. If this argument is present, the TARGET argument must be a file.
* --lineindexing {0, 1} Determines if the first line of the file is given a value of “0” or “1”. Defaults to 0.

Shell file configuration:

python [directory path]/pylib/fingerprint/fingerprint.py [optional arguments—see above] target

Output: The following information is logged to a path and file (default, i.e. fingerprint.txt) or to a path and file, which is specified by user and is passed to the Fingerprinter as an argument (-o/--output OUTPUT). The optional argument –outputmode controls the write mode (“a” = “append”, “w”=”write/overwrite”). In default mode, the output file contains:

* DateTime stamp
* Total number of files fingerprinted (if a directory is specified)
* Path and file name and corresponding hash for each file fingerprinted

If the argument “—sections” is specified, the output will contain the hash of each specified section instead of the path and file name. If “—sections” is specified, the fingerprinter will not processes a directory.

An independent code walkthrough was conducted by Mitchell Tufford on 11/22/2019. A summary and resolution of the comments is presented in Appendix A, table A-1.

1. **Requirements Traceability Matrix**

The requirements traceability matrix for the Fingerprinter tool is presented in Table 1.

| **Table 1. Fingerprinter Tool Requirements Traceability Matrix** | | |
| --- | --- | --- |
| **Functional Requirement ID** | **Acceptance Test ID** | **Test Case** |
| QA Level | CACIE-fingerprint.py-IT-1 | Installation Test |
| FR-1 FR-3 | CACIE-fingerprint.py-TC-1 | Fingerprint a file |
| FR-2  FR-3 | CACIE-fingerprint.py-TC-2 | Fingerprint a specified directory and the files located in the directory path |
| FR-1  FR-2  FR-3 | CACIE-fingerprint.py-TC-3 | Fingerprint two files (identical except for filename) |
| FR-1  FR-2  FR-3 | CACIE-fingerprint.py-TC-4 | Fingerprint two files (identical except for filename and date stamp) |
| FR-1  FR-2  FR-3 | CACIE-fingerprint.py-TC-5 | Fingerprint two different files |
| FR-1  FR-2  FR-3 | CACIE-fingerprint.py-TC-6 | Fingerprint a compressed file and the contents of the compressed folder after uncompressing |
| FR-4 | CACIE-fingerprint.py-TC-7 | Fingerprint a section of a file |
| FR-5 | CACIE-fingerprint.py-TC-8 | Run fingerprinter in append mode |

1. **Test Plan and Cases**

The test plan for the Fingerprinter tool is presented in Table 2.

| **Table 2. Fingerprinter Tool Test Plan** | | |
| --- | --- | --- |
| **TEST ID** | **Test Case** | **Test Result  (Pass/Fail)** |
| *Note [Testing\_Directory] in acceptance test report* | | |
| *Navigate to [Testing Directory]\CA-CIE-Tools-Testing (code repository)* | | |
| *Verify git branch and repository status by entering the following at the command line:*  *git branch (verify branch is “development”)*  *git pull*  *git status*  *git log -1*  *Note first 6 characters of git SHA-1 hash tag in acceptance test report* | | |
| CACIE-fingerprint.py-IT-1 | *Linux platform:*  *In a Linux command window:*  *Navigate to [Testing\_Directory]\fingerprint\_test* | |
| *Invoke Tool Runner and Fingerprint tool using fingerprint.py\_IT-1\_linux.sh by entering the following at the command line:./fingerprint.py\_IT-1\_linux.sh* | |
| Verify Tool Runner is invoked and executes by checking log and fingerprint files |  |
| Verify Fingerprinter tool is invoked and executes by checking log and fingerprint files |  |
| *Windows platform:*  *In a Windows command window:*  *Navigate to [Testing\_Directory]\fingerprint\_test* | |
| *Invoke Tool Runner and Fingerprint tool using fingerprint.py\_IT-1\_windows.sh by entering the following at the command line:./fingerprint.py\_IT-1\_windows.sh* | |
| Verify Tool Runner is invoked and executes by checking log and fingerprint files |  |
| Verify Fingerprinter tool is invoked and executes by checking log and fingerprint files |  |
| *Navigate to [Testing\_Directory]/fingerprint\_test* | | |
| CACIE-fingerprint.py-TC-1 | *Enter the following command:*  *./fingerprint.py\_TC-1.sh* | |
| Verify that the fingerprinter.py\_TC-1\_fingerprint.txt documents datetime stamp of fingerprint, path (if applicable) and filename and hash tag |  |
| CACIE-fingerprint.py-TC-2 | *Enter the following command:*  *./fingerprint.py\_TC-2.sh* | |
| Verify that the fingerprint.py\_TC-2\_fingerprint.txt documents the following:   * datetime stamp of fingerprint, * a hash for the total number of files in the directory * paths (if applicable) and filenames and hash tag for each individual file in the directory and subdirectories |  |
| CACIE-fingerprint.py-TC-3 | *Copy and paste the file “testfile.txt”.*  *NOTE: directory now includes file “testfile - Copy.txt”* | |
| *Enter the following command:*  *fingerprint.py\_TC-3.sh* | |
| Verify that the fingerprint\_TC-3.txt documents identical hash tags for the following files:   * testfile.txt * testfile - Copy.txt |  |
| CACIE-fingerprint.py-TC-4 | *Copy and paste “testfile.txt”.*    *NOTE: directory now includes file “testfile - copy (2).txt”* | |
| *Open testfile - copy (2).txt in text editor—do not change text.* | |
| *Save file (i.e. change datestamp of file)* | |
| *Enter the following command:*  *./fingerprint.py\_TC-4.sh* | |
| Verify that the fingerprint\_TC-4.txt documents identical hash tags for the following files:   * testfile.txt * testfile - Copy.txt * testfile - Copy (2).txt |  |
| CACIE-fingerprint.py-TC-5 | *Open testfile - Copy (2).txt in text editor.* | |
| *Add a space somewhere in the file.* | |
| *Save file (i.e. change datestamp of file).* | |
| *Enter the following command:*  *./fingerprint.py\_TC-5.sh* | |
| Verify that the fingerprint\_TC-5.txt documents identical hash tags for the following files:   * testfile.txt * testfile - Copy.txt |  |
| Verify that the fingerprint\_TC-5.txt documents a unique hash tag for:   * testfile - Copy (2).txt |  |
| CACIE-fingerprint.py-TC-6 | *Create a compressed (zipped) file named “fingerprint\_test.zip” and add the contents of the fingerprint\_test directory.* | |
| *Enter the following command:*  *./fingerprint.py\_TC-6a.sh* | |
| Verify that the fingerprint\_*TC*-6a.txt documents a unique hash tag for the compressed file:   * *fingerprint\_test.zip* |  |
| *Create the following new subdirectory in the fingerprint\_test directory:*   * *unzipped* | |
| *Unzip the compressed file into the unzipped subdirectory* | |
| *Enter the following command:*  *./fingerprint.py\_TC-6b.sh* | |
| *Compare the hash tags in fingerprint\_TC-5.txt and fingerprint\_TC-6b.txt* | |
| Verify that each file documents the same identical hash tags for the following files:   * testfile.txt * testfile - Copy.txt |  |
| Verify that each file documents the same unique hash tag for:   * testfile - Copy (2).txt |  |
| CACIE-fingerprint.py-TC-7 | Navigate to *[Testing\_Directory]\fingerprint\_test* | |
| Enter the following command:  ./fingerprint.py\_TC-7a.sh | |
| Verify that two fingerprint files were produced:   * *testlog/fingerprint.py\_TC-7\_fingerprintA.txt* * *testlog/fingerprint.py\_TC-7\_fingerprintB.txt* |  |
| Verify that the two hashes in *testlog/fingerprint.py\_TC-7\_fingerprintB.txt* are identical to the hash in *testlog/fingerprint.py\_TC-7\_fingerprintA.txt* labeled with the text “line section [0,1); 0-indexed” |  |
| CACIE-fingerprint.py TC-8 | Navigate to *[Testing\_Directory]\fingerprint\_test* | |
| Run the command ./fingerprint.py\_TC-8.sh | |
| Verify that the file testlog/fingerprint.py\_TC-8.txt contains two identical fingerprint reports |  |

1. **Acceptance Test Report**

* v1

Acceptance testing of the Fingerprinter tool was performed by Neira Mondragon and in accordance with the test plan documented in Section 6. The acceptance testing was performed in the following directory:

\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\fingerprint\_test

The Fingerprinter tool met the functional requirements as documented in Section 2. Verification of the tool’s functionality is documented in Table A-2 and the test logs included in Appendix A. There were no incidents involving the fingerprint.py code requiring resolution and accordingly, there are no unresolved incidents. The Table A-2 does record the steps taken to configure the test machine in order to run the installation test on the Window Platform.

* v1.2

Acceptance testing of the Fingerprinter tool was performed by Christian Hall and in accordance with the test plan documented in Section 6. The acceptance testing was performed in the following directory:

Testing\_dir: [\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\fingerprint\_test](\\\\olive\\backups\\CAVE\\CA-CIE-Tools-TestEnv\\fingerprint_test)

Testing\_repo: [\\olive\backups\CAVE\CA-CIE-Tools-TestRepos/fingerprint1p2](\\\\olive\\backups\\CAVE\\CA-CIE-Tools-TestRepos/fingerprint1p2t)

The Fingerprinter tool met the functional requirements as documented in Section 2. Verification of the tool’s functionality is documented in Table A-2 and the test logs included in the same folder as this document, described below. There were no incidents involving the fingerprint.py code requiring resolution and accordingly, there are no unresolved incidents. The Table A-2 does record the steps taken to configure the test machine in order to run the installation test on the Window Platform.

Installation Test - Nix: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_IT-1\_linux\_fingerprint.txt

Installation Test - Nix: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_IT-1\_linux\_logfile.txt

Installation Test - Win: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_IT-1\_windows\_fingerprint.txt

Installation Test - Win: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_IT-1\_windows\_logfile.txt

Test Case 1: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-1\_fingerprint.txt

Test Case 1: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-1\_logfile.txt

Test Case 2: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-2\_fingerprint.txt

Test Case 2: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-2\_logfile.txt

Test Case 3: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-3\_fingerprint.txt

Test Case 3: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-3\_logfile.txt

Test Case 4: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-4\_fingerprint.txt

Test Case 4: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-4\_logfile.txt

Test Case 5: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-5\_fingerprint.txt

Test Case 5: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-5\_logfile.txt

Test Case 6a: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-6a\_fingerprint.txt

Test Case 6a: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-6a\_logfile.txt

Test Case 6b: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-6b\_fingerprint.txt

Test Case 6b: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-6b\_logfile.txt

Test Case 7: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-7\_fingerprint.txt

Test Case 7: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-7\_logfile.txt

Test Case 8: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-8\_fingerprint.txt

Test Case 8: CA-CIE-Tools/docs/fingerprinter/fingerprinter.py\_TC-8\_logfile.txt

1. **User Guide**

The Fingerprinter tool can be invoked from the command line using the arguments as specified in Section 4 (Software Design) and the arguments for the invoked tool. The Fingerprinter tool can also be invoked using the Tool Runner tool.

**Appendix A**

**Code Review Summary   
and   
Acceptance Testing Logs**

|  | **Table A-1. Fingerprinter Tool Issues/Improvements** | | | | |
| --- | --- | --- | --- | --- | --- |
| **Issue #** | **Code Line** | **Comment** | **Function Impact** | **Suggested Change** | **Resolution** |
| 1 | 72 | The sys lib is imported only when \_\_name\_\_=="\_\_main\_\_": and seems to be unused. | None perceived. | Remove import statement or move to top of file. | Moved import statement to top of code as recommended. |

**Table A-2. Fingerprinter Tool  
Code Reviews**

| **Date** | **Reviewer** | **Comments** |
| --- | --- | --- |
| **02/03/2020** | Mitchell Tufford | No new issues identified. |

| **Table A-3. Fingerprinter Tool Acceptance Test** | | |
| --- | --- | --- |
| **TEST ID** | **Test Case** | **Test Result  (Pass/Fail)** |
| *Note [Testing\_Directory]* [\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\fingerprint\_test](file:///\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\fingerprint_test)  *in acceptance test report*  *Note [Testing\_Repo]* [\\olive\backups\CAVE\CA-CIE-Tools-TestRepos\fingerprint1p2](\\\\olive\\backups\\CAVE\\CA-CIE-Tools-TestRepos\\fingerprint1p2\\)  *in acceptance test report* | | |
| *Navigate to [Testing Repo]* | | |
| *Verify git branch and repository status by entering the following at the command line:*  *git branch (verify branch is “development”)*  *git pull*  *git status*  *git log -1*    *Note first 6 characters of git SHA-1 hash tag in acceptance test report*  *2a5b13* | | |
| CACIE-fingerprint.py-IT-1 | *Linux platform:*  *In a Linux command window:*  *Navigate to [Testing\_Directory]\fingerprint\_test* | |
| *Invoke Tool Runner and Fingerprint tool using fingerprint.py\_IT-1\_linux.sh by entering the following at the command line:*  *./fingerprint.py\_IT-1\_linux.sh*  *First time, permission denied occured*    *After fix, it all worked* | |
| Verify Tool Runner is invoked and executes by checking log and fingerprint files in: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\fingerprint\_test\testlog | *Pass* |
| Verify Fingerprinter tool is invoked and executes by checking log and fingerprint files in: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\fingerprint\_test\testlog | *Pass* |
| *Windows platform:*  *Navigate to [Testing\_Directory]\fingerprint\_test*  *While inside of [Testing\_Directory]\fingerprint\_test in a file explorer window:*   * *Hold shift and press right-click* * *Click “Open PowerShell window here”*   *Alternatively, use Window’s command prompt cmd.exe:*   * *Windows key and type “Command prompt”*   *As a last option, open up git bash to run these commands*   * *Right click in fingerprint\_test* * *Click “Git Bash Here”*   *Lastly, change directories to your mapped drive to run the scripts (Z is a mapped drive to [\\olive\backup\CAVE](\\\\olive\\backup\\CAVE) in christian’s configuration)*  *cd /d Z:\CA-CIE-Tools-TestEnv\fingerprint\_test\* | |
| *Historical Documentation: v1*  *Invoke Tool Runner and Fingerprint tool using fingerprint.py\_IT-1\_windows.sh by entering the following at the command line:./fingerprint.py\_IT-1\_windows.sh*  *Should have gotten an error with this initial test but instead the file would open in notepad*  *In linux putty window enter the following to be mapped after logging into olive:*  *Ln -s /opt/tools ca-cie-tools*  *After running in the same error: “No such file or directory” in windows we did the following:*  *Opened file explorer🡪Easy Access🡪Map as drive*  *In this case Y: was set to* [*\\olive\nmondragon*](file:///\\olive\nmondragon)  *Went back to putty window and entered: ln -s /samba/backups/CAVE CAVE*  *Under settings🡪 “Edit the system environment variables”🡪 Environment Variables 🡪 select “PATH”🡪Edit🡪New🡪 and enter: C:\Program Files\Git\mingw64\bin🡪 OK*  *Git bash windows does not operate properly on my computer. Please note that I used Anaconda Powershell Prompt (anaconda3) and* ***.bat*** *files for this testing* | |
| *V1.2:*  *Invoke Tool Runner and Fingerprint tool using fingerprint.py\_IT-1\_windows.bat or fingerprint.py\_IT-1\_windows.sh (use* ***bat*** *for mapped drive, use* ***sh*** *for no mapped drive)*  *Command prompt (in mapped drive): fingerprint.py\_IT-1\_windows.bat*  *Powershell (in mapped drive): fingerprint.py\_IT-1\_windows.bat*  *Git Bash (no mapped drive):*  *cd //olive/backups/CAVE/CA-CIE-Tools-TestEnv/fingerprint\_test*  *./fingerprint.py\_IT-1\_windows.sh* | |
| Verify Tool Runner is invoked and executes by checking log and fingerprint files | Pass |
| Verify Fingerprinter tool is invoked and executes by checking log and fingerprint files | Pass |
| *Navigate to [Testing\_Directory]/fingerprint\_test* | | |
| CACIE-fingerprint.py-TC-1 | *Enter the following command:*  *./fingerprint.py\_TC-1.sh* | |
| Verify that the fingerprinter.py\_TC-1\_fingerprint.txt documents datetime stamp of fingerprint, path (if applicable) and filename and hash tag | Pass |
| CACIE-fingerprint.py-TC-2 | *Enter the following command:*  *./fingerprint.py\_TC-2.sh* | |
| Verify that the fingerprint.py\_TC-2\_fingerprint.txt documents the following:   * datetime stamp of fingerprint, * a hash for the total number of files in the directory * paths (if applicable) and filenames and hash tag for each individual file in the directory and subdirectories | Pass |
| CACIE-fingerprint.py-TC-3 | *Copy and paste the file “testfile.txt” in the following directory: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\fingerprint\_test*  *NOTE: directory now includes file “testfile - Copy.txt”* | |
| *Enter the following command:*  *./fingerprint.py\_TC-3.sh* | |
| Verify that the fingerprint\_TC-3.txt documents identical hash tags for the following files:   * testfile.txt * testfile - Copy.txt | Pass |
| CACIE-fingerprint.py-TC-4 | *Copy and paste “testfile.txt” in the following directory: \\olive\backups\CAVE\CA-CIE-Tools-TestEnv\fingerprint\_test*    *NOTE: directory now includes file “testfile - copy (2).txt”* | |
| *Open testfile - copy (2).txt in text editor—do not change text.* | |
| *Save file (i.e. change datestamp of file)* | |
| *Enter the following command:*  *./fingerprint.py\_TC-4.sh* | |
| Verify that the fingerprint\_TC-4.txt documents identical hash tags for the following files:   * testfile.txt * testfile - Copy.txt * testfile - Copy (2).txt | Pass |
| CACIE-fingerprint.py-TC-5 | *Copy a fourth file “testfile.txt” into the [Testing\_Directory]* | |
| *Open testfile - Copy (3).txt in text editor* | |
| *Add a space or other new text somewhere in the file.* | |
| *Save file (i.e. change datestamp of file).* | |
| *Enter the following command:*  *./fingerprint.py\_TC-5.sh* | |
| Verify that the fingerprint\_TC-5.txt documents identical hash tags for the following files:   * testfile.txt * testfile - Copy.txt * testfile - Copy(2).txt | Pass |
| Verify that the fingerprint\_TC-5.txt documents a unique hash tag for:   * testfile - Copy (3).txt | Pass |
| CACIE-fingerprint.py-TC-6 | *Create a compressed (zipped) file named “fingerprint\_test.zip” and add the contents of the fingerprint\_test directory.* | |
| *Enter the following command:*  *./fingerprint.py\_TC-6a.sh* | |
| Verify that the fingerprint\_*TC*-6a.txt documents a unique hash tag for the compressed file:   * *fingerprint\_test.zip* | Pass |
| *Create the following new subdirectory in the fingerprint\_test directory:*   * *unzipped* | |
| *Unzip the compressed file into the unzipped subdirectory* | |
| *Enter the following command:*  *./fingerprint.py\_TC-6b.sh* | |
| *Compare the hash tags in fingerprint\_TC-5\_fingerprint.txt and fingerprint\_TC-6b\_fingerprint.txt (since TC-5 has the unzipped folder inside of it as well, they will be different)*  Using *fingerprint\_TC-6b\_fingerprint.txt* verify the following | |
| Verify that each file documents the same identical hash tags for the following files:   * testfile.txt * testfile - Copy.txt * testfile - Copy (2).txt | Pass |
| Verify that each file documents the same unique hash tag for:   * testfile - Copy (3).txt | Pass |
| CACIE-fingerprint.py-TC-7 | Navigate to *[Testing\_Directory]\fingerprint\_test* | |
| Enter the following command:  ./fingerprint.py\_TC-7a.sh | |
| Verify that two fingerprint files were produced:   * *testlog/fingerprint.py\_TC-7\_fingerprintA.txt*   *testlog/fingerprint.py\_TC-7\_fingerprintB.txt* | Pass |
| Verify that the two hashes in *testlog/fingerprint.py\_TC-7\_fingerprintB.txt* are identical to the hash in *testlog/fingerprint.py\_TC-7\_fingerprintA.txt* labeled with the text “line section [0,1); 0-indexed” | Pass |
| CACIE-fingerprint.py TC-8 | Navigate to *[Testing\_Directory]\fingerprint\_test* | |
| Run the command ./fingerprint.py\_TC-8.sh | |
| Verify that the file testlog/fingerprint.py\_TC-8.txt contains two identical fingerprint reports | Pass |