CMMC Hashing Guide

Version –2.13 | September 2024

DoD-CIO-00008 (ZRIN 0790-ZA24)

24-T-2768

NOTICES

The contents of this document do not have the force and effect of law and are not meant to bind the public in any way. This document is intended only to provide clarity to the public regarding existing requirements under the law or departmental policies.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

CMMC Artifact Hashing Tool User Guide

Audience

This guide assumes that the reader has a basic understanding of command-line tools and scripting. Given the proprietary nature of the artifacts generated during a CMMC assessment, it also assumes that the Organization Seeking Assessment (OSA) has staff with sufficient technical background to use the hashing tool independently on an approved organizational system. If the OSA lacks staff with the requisite background, they may request assistance from the assessor or another party in order to complete the process of artifact hashing. Step-by-step instructions are provided below.

Scope and Purpose

When doing self-assessments, OSAs are not required to generate hashes for artifacts. Hashing is only required for assessments by C3PAOs and DCMA DIBCAC.

During the performance of a CMMC assessment, the assessment team will collect objective evidence using a combination of three assessment methods:

- examination of artifacts,
- affirmations through interviews, and
- observations of actions.

Because these OSA artifacts may be proprietary, the assessment team will not take or retain OSA artifacts offsite at the conclusion of the assessment. For the protection of all stakeholders, the OSA must retain the artifacts. The artifacts used as evidence for the assessment must be retained by the OSA for six (6) years from the date of assessment.

Because the artifacts will remain with the OSA, a tool has been developed to provide a cryptographic reference (or hash) for each artifact used in the assessment as discussed in 32 CFR § 170.17 and 32 CFR § 170.18. If needed, the integrity of the assessment artifacts may be checked by verifying the hash generated during the assessment. If an artifact has not been modified, the hash will remain the same.

Hashing is Different From Encryption

Do not confuse hashing with encryption. Both are cryptographic functions, but hashing does *not* provide confidentiality for the artifacts. It provides only a mechanism to track the integrity of the artifacts. Confidentiality of the artifacts needs to be handled separately by the OSA, using a different mechanism, such as encryption. When choosing a location to archive the artifacts, the OSA should consider data protection requirements.

The Artifact Hashing Tool is a Microsoft PowerShell script that uses the SHA-256 algorithm to generate a hash of each artifact. Next, it generates a list of artifact filenames and associated hashes, then completes the process by generating a hash of the list. At the conclusion of the assessment, the OSA and the assessor will each have the list of artifact names, artifact hashes, and a hash of the list.

System Requirements

A computer capable of running Microsoft PowerShell is required for this tool. PowerShell is available for Windows, Linux, and macOS. Please refer to Microsoft PowerShell instructions for installation, if the software is not already on the system. The execution of PowerShell scripts may be restricted by the organization. Microsoft's instructions explain how to temporarily bypass such restrictions to use the tool. It may be necessary to speak to an administrator to obtain the necessary permissions to execute PowerShell scripts. Additional details can be found in the Supplemental Information section.

This tool was tested on Windows 11 (version 23H2), Windows 10 (version 1904), Linux (Ubuntu 20.04), and macOS (10.15.7).

Process Overview

During the assessment planning and preparation, the OSA and assessment team should decide jointly how they will store artifact files during the assessment. The agreed-upon location should be secure and accessible only to those with a need to know because the artifacts may contain sensitive or proprietary information.

During the course of the assessment, the team collects information through three assessment methods: interviews, artifact examination, and observation. This collection may include such activities as interviewing organization staff, examining the documents or the configuration of a device, and observing organization staff performing actions (Figure 1). It is important to collect evidence while performing these actions, to substantiate MET or NOT MET decisions for each CMMC requirement.

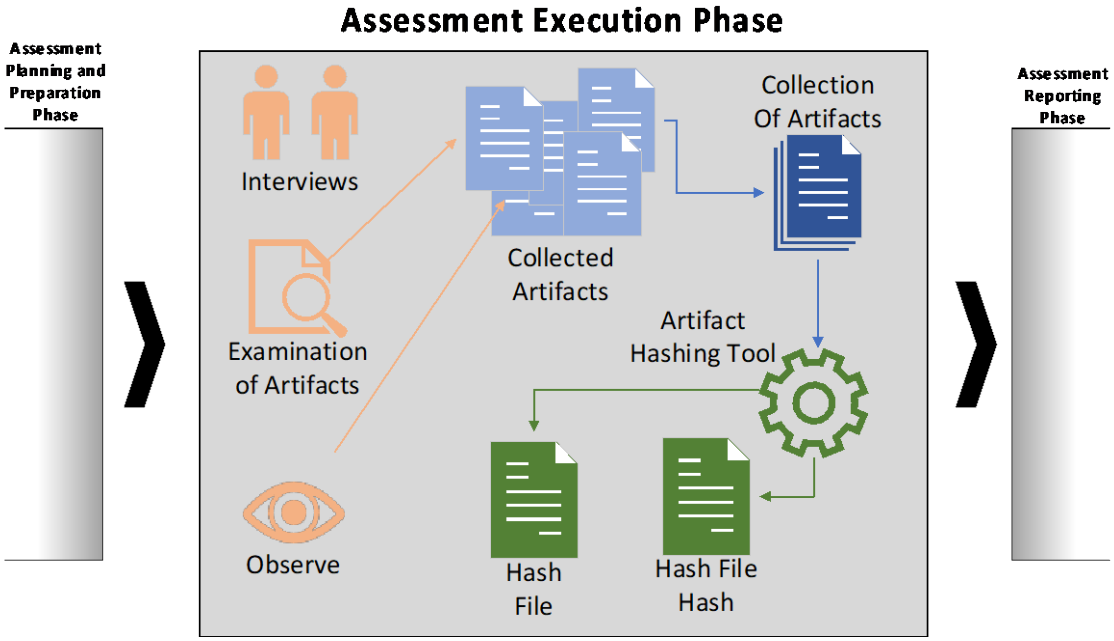

Figure 1 - Assessment Execution

The central location where collected assessment artifacts are stored may be a single root directory (Figure 2, Scenario 1) where all documents are stored. Optionally, the root directory may have subdirectories within it (Figure 2, Scenario 2). The Artifact Hashing Tool can operate in either scenario.

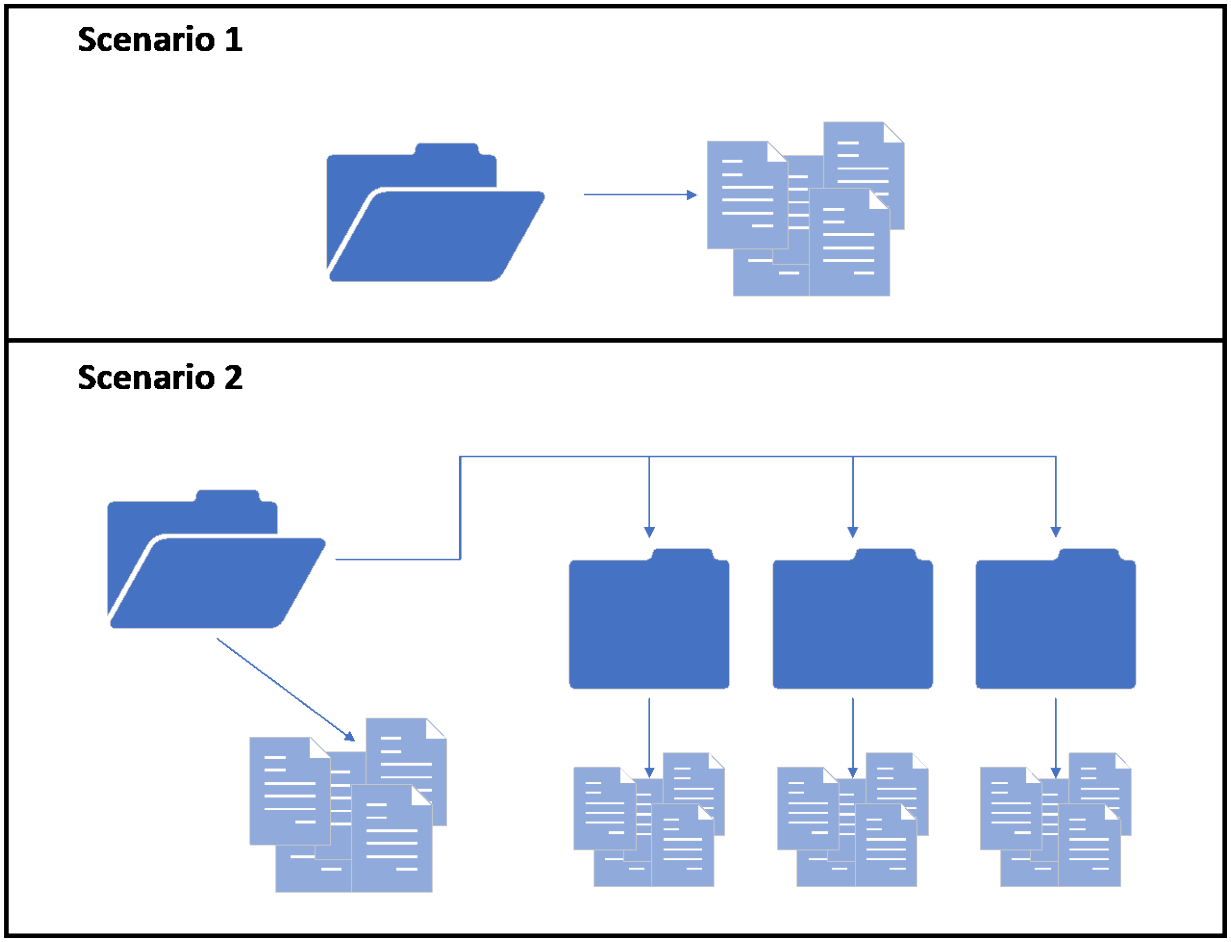

Figure 2 - Folder Hierarchy Scenarios

Clearly naming artifacts will aid in the event of an audit or retrospective reviews of assessment data. Artifact filenames should follow a standardized naming pattern or be grouped by CMMC requirement.

After all artifacts reviewed by the assessment team are consolidated into the central location, the OSA may run the artifact hashing tool. Both the OSA and assessor should retain a copy of the file log, file hashes, and the integrity hash. The following section details the process for generating hashes for all collected assessment artifacts.

Tool Usage Process

Use the commands listed in the instructions below to execute the Artifact Hashing Tool on a computer running Microsoft Windows. If the computer running the tool is operating on Linux or macOS, minor command modifications will need to be made (e.g., in Linux and macOS, use `mv` instead of `ren`, respectively).

Preparation

1. Create the ArtifactHash.txt file from the content located in Appendix A. The ArtifactHash.txt file location should be the root directory of the collected assessment artifacts. Ensure you have access to the root directory location.
2. Locate the root directory where collected assessment artifacts are stored. In this instance, "root directory" refers to the directory in which all of the assessment artifacts and/or other folders containing assessment artifacts have been stored.
3. Modify the file extension of the script file created from the Appendix A content. The script content is located as text within Appendix A. You should have a copy of the ArtifactHash.txt file in the root directory of collected assessment artifacts. You can use another location, but this guide assumes that the script file has been copied to this directory.
4. Open **Windows Command Prompt** or a terminal window for macOS/Linux, then navigate to the location of the script file.
5. Change the file extension of the script file to read as follows:

> **Note:**
> The command line entries in this guide can be copied and pasted into the respective OS Command Prompt.

```
Windows:
      ren ArtifactHash.txt ArtifactHash.ps1

Linux/maxOS:
      mv ArtifactHash.txt ArtifactHash.ps1
```

Execution of Tool

1. After renaming the script, run the tool. The script has three parameters:
    - `ExecutionPolicy`: This parameter allows the script to run unrestricted. It is recommended that the ByPass value be retained.
    - `ArtifactRootDirectory`: This specifies the root directory path of the CMMC assessment artifacts. This location can be represented by a traditional Windows file path, a UNC path, or even .\ to indicate the current directory. The default value

is the current directory. If the script is located in the root of the artifact repository, this parameter does not need to be specified on the command line.

- `ArtifactOutputDirectory`: This specifies the directory where the script will write two log files. The first log is the listing of all files within the `ArtifactRootDirectory` as well as the corresponding hash. The second log is a hashed value of the first log. This is a simple way to help preserve the integrity of the artifact listing without requiring the maintenance of a public/private key pair or a password for an HMAC. The default value for this parameter is the current directory. If the script is located in the desired output location, this parameter does not need to be specified on the command line.

2. Execute the following command, along with the determined values for the two directory parameters:

```
Windows:
  powershell -ExecutionPolicy ByPass .\ArtifactHash.ps1 -
ArtifactRootDirectory .\ -ArtifactOutputDirectory .\

Linux / macOS:
  pwsh -ExecutionPolicy ByPass ./ArtifactHash.ps1 -
ArtifactRootDirectory ./ -ArtifactOutputDirectory ./
```

Important

The command above assumes that the script file is located in the root assessment artifact directory and that the output hash files typically goes into to the same directory. The ".\" following the parameters should be modified if the script is located in a different directory or to output the hash files to a different directory. In addition, this command assumes usage of the `ExecutionPolicy` cmdlet, which may not be necessary. See the Supplemental Information section for details.

Note:
Contact the system administrator to address permission errors or other restrictions when running this script.

3. If the tool has run successfully, SCRIPT COMPLETE will be displayed in the command prompt. At this time, verify that the files (CMMCAssessmentArtifacts.log) and (CMMCAssessmentLogHash.log) have been generated in the output directory specified by the second script parameter.

Use Example

In this simple example, a C3PAO has used four files provided by an OSC to support an assessment. The four assessment related files are in a directory along with the PowerShell script as shown in Figure 3.

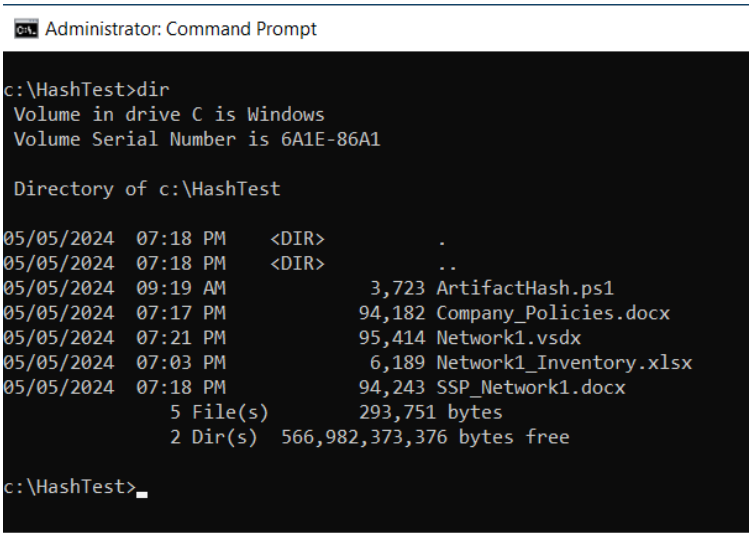

Figure 3 - Assessment file directory before hash

Figure 4 shows the successful execution of the PowerShell script and Figure 5 shows the same directory with the addition of the two PowerShell output files.

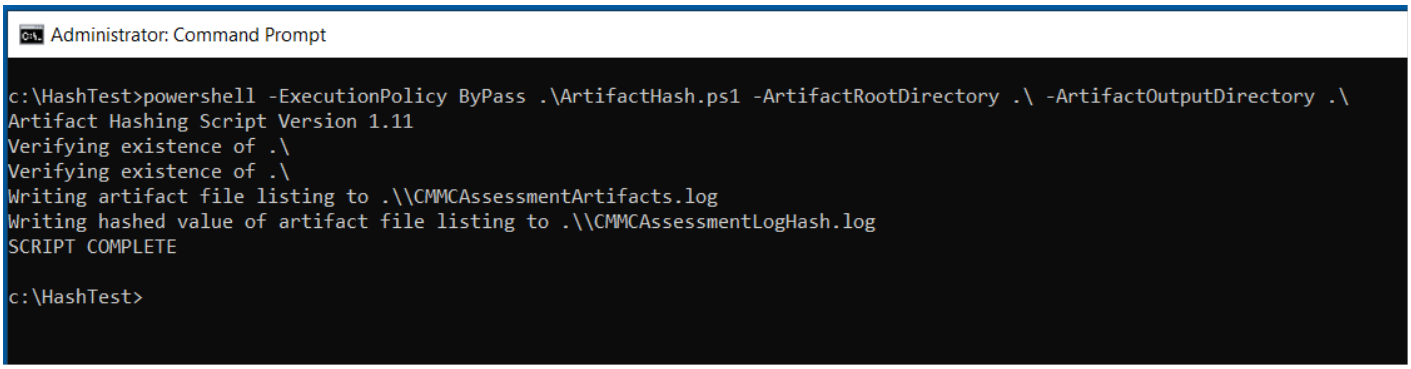

Figure 4 - Successful execution of ArtifactHash.ps1

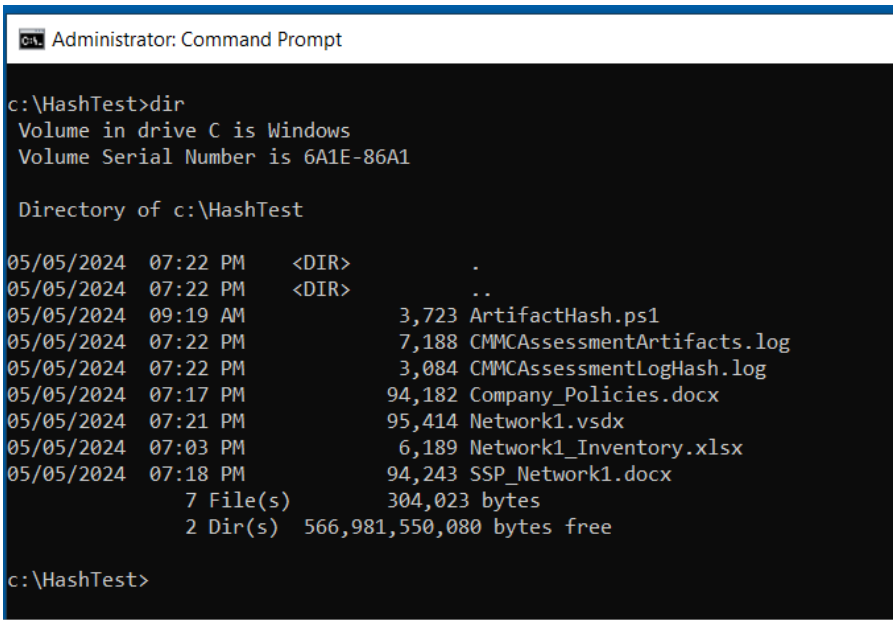

Figure 5 - Assessment file directory after script execution

CMMCAssessmentArtifacts.log, in Figure 6, is a text file that contains the hashing algorithm used, hash value of each individual file in the directory, and the filename and path. This text file contains the list of artifacts. The filename is entered into the Hashed Data List data field in the CMMC instantiation of eMASS.

CMMCAssessmentArtifacts.log

Algorithm	Hash	Path
SHA256	43DA1D2EFE64C6C6CA20FF9226104BB3AF6737BE5FBF7BA36E7727F7FAC354	C:\HashTest\ArtifactHash.ps1
SHA256	E3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855	C:\HashTest\Company_Policies.docx
SHA256	E3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855	C:\HashTest\Network1.vsdx
SHA256	0D68C066F78B237D50D663A335053E337B63AD8D05130C07997E64262DF29D07	C:\HashTest\Network1_Inventory.xlsx
SHA256	E3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855	C:\HashTest\SSP_Network1.docx

Figure 6 - CMMCAssessmentArtifacts.log

CMMCAssessmentLogHash.log in Figure 7, is a text file that contains the single return value generated by creating a hash of CMMCAssessmentArtifacts.log. This hash is the string to enter in the Hash Value data field in the CMMC instantiation of eMASS.

CMMCAssessmentLogHash.log

Algorithm	Hash	Path
SHA256	58B521F97FFE61659B7300AAA9916142D3BA77CAF1DCE76512DA32198F8D177C	C:\HashTest\CMMCAssessmentArtifacts.log

Figure 7 - CMMCAssessmentLogHash.log

Supplemental Information

- For parameters that include spaces in the paths, surround the entire path name in single quotes. During testing, double quotes produced an error.
- If the script file is not placed in the root assessment artifact directory, specify the path of the script, for example, Z:\Files\Tool\ArtifactHash.ps1.
- In certain instances, the organization may restrict the execution of PowerShell scripts. The ByPass value of the `ExecutionPolicy` cmdlet within the command should temporarily bypass these restrictions. In addition, it is possible to manually verify and modify the PowerShell script execution policy of the current user as follows.

  Note: The content following the # (hashtag) symbol represents a comment in the script.

1. Verify the policy that is set.

```
Windows:
  powershell get-ExecutionPolicy -Scope CurrentUser #make note of the
current setting
```

2. Set the execution policy for the user to bypass, and verify that "ByPass" is reflected.

```
Windows:
  set-ExecutionPolicy ByPass -Scope CurrentUser
  get-ExecutionPolicy -Scope CurrentUser #verify the setting was updated
```

3. After completion of the hashing process, change the execution policy back to the default state.

```
Windows:
  set-ExecutionPolicy Default -Scope CurrentUser
```

Appendix A: ArtifactHash.txt File Content

The blue Courier text below is the PowerShell script needed for this task. Use cut and paste to copy all of the blue Courier content into a text editor and store the file with the name: ArtifactHash.txt.

```
<#
.SYNOPSIS
    Hash artifacts for a CMMC Assessment to maintain integrity in the event any files are needed
in the future
.DESCRIPTION
    This script will recursively evaluate all files in a local or UNC path.  Each file will be
hashed and written to a text file.  Additionally, the record is hashed to preserve the integrity
of the output
.PARAMETER ArtifactRootDirectory
    Specifies the root path of the CMMC assessment artifacts.  This location can be represented
by a traditional Windows file path, a UNC path, or even .\
.PARAMETER ArtifactOutputDirectory
    Specifies the directory where the script will write two log files.  The first log is the
listing of all files within the ArtifactRootDirectory as well as the corresponding hash. The
second log, is a hashed value of the first log.  This is a simple way to help preserve the
integrity of the artifact listing without requiring the maintenance of a public/private key pair
or a password for an HMAC
#>
#VERSION 1.11
param
(
    [Parameter(mandatory=$false)][string]$ArtifactRootDirectory = ".\",
    [Parameter(mandatory=$false)][string]$ArtifactOutputDirectory = ".\"
)

function GetFileHashes ([string] $rootLocation, [boolean] $isDirectory)
{
    if ($isDirectory)
    {
        $hashList = Get-ChildItem -path $rootLocation -Recurse -Force -File | Get-FileHash
    }
    else
    {
        $hashList = Get-FileHash $rootLocation
    }
    return $hashList
}


function WriteASCIIFile ([string] $filePath, [object] $fileContent)
{
    Out-File -FilePath $filePath -Force -Encoding ASCII -InputObject $fileContent -Width 1024
}

function VerifyLocationExist ([string] $location)
{
    try
    {   $doesExist = Test-Path $location
        if (-Not $doesExist)
        {
            ECHO "Location $location does not exist"
            throw
        }
    }
    catch
    {
        ECHO "The program failed to evaluate the path.  Perhaps you specified an incorrectly
formatted command line parameter?"
        EXIT
    }
```

```
}

function IsDirectory ([string] $location)
{
    $isDirectory = (get-item $location) -is [System.IO.DirectoryInfo]
    return $isDirectory
}

$version = "1.11"
ECHO "Artifact Hashing Script Version $version"
#Just making sure locations are legit
ECHO "Verifying existence of $ArtifactRootDirectory"
VerifyLocationExist $ArtifactRootDirectory
ECHO "Verifying existence of $ArtifactOutputDirectory"
VerifyLocationExist $ArtifactOutputDirectory

#determine if the input provided is for a single file or for a directory of files
$artifactLocationIsDir = IsDirectory($ArtifactRootDirectory)
$logFileLocationIsDir = IsDirectory($ArtifactOutputDirectory)

if($logFileLocationIsDir)
{
    $logFileLocation = $ArtifactOutputDirectory + "\CMMCAssessmentArtifacts.log"
    $hashedLogFileLocation = $ArtifactOutputDirectory + "\CMMCAssessmentLogHash.log"
}
else
{
    $endOfString = $ArtifactOutputDirectory.LastIndexOf("\")
    $logFileLocation = $ArtifactOutputDirectory.Substring(0,$endOfString) +
"\CMMCAssessmentArtifacts.log"
    $hashedLogFileLocation = $ArtifactOutputDirectory.Substring(0,$endOfString) +
"\CMMCAssessmentLogHash.log"
}

#return the list of artifacts with their hashed values
$hashedFiles = GetFileHashes $ArtifactRootDirectory $artifactLocationIsDir
ECHO "Writing artifact file listing to $logFileLocation"
WriteASCIIFile $logFileLocation $hashedFiles

#Now, I'm going to create a second file hashing the artifacts file
$hashTheHash = GetFileHashes $logFileLocation $false
ECHO "Writing hashed value of artifact file listing to $hashedLogFileLocation"
WriteASCIIFile $hashedLogFileLocation $hashTheHash
ECHO "SCRIPT COMPLETE"
```

This page intentionally left blank.


DEPARTMENT OF DEFENSE
CHIEF INFORMATION OFFICER
CYBERSECURITY MATURITY MODEL
CERTIFICATION