CRITERIA FOR VALIDATING SECURE WIPING TOOLS

Muhammad Zareen, Baber Aslam, Monis Akhlaq

Chapter 19

CRITERIA FOR VALIDATING SECURE WIPING TOOLS

Muhammad Sharjeel Zareen, Baber Aslam and Monis Akhlaq

Abstract The validation of forensic tools is an important requirement in digital forensics. The National Institute of Standards and Technology has defined standards for many digital forensic tools. However, a standard has not yet been specified for secure wiping tools. This chapter defines secure wiping functionality criteria for NTFS specific to Windows 7 and magnetic hard drives. The criteria were created based on the remnants of user actions – file creation, modification and deletion – in $MFT records, the $LogFile and the hard disk. Of particular relevance is the fact that the $LogFile, which holds considerable forensic artifacts of user actions, is not wiped properly by many tools. The use of the proposed functionality criteria is demonstrated in an evaluation of the Eraser secure wiping tool.

Keywords: Tool validation, secure wiping functionality, NTFS, $LogFile forensics

1. Introduction

To ensure the integrity of digital evidence and the results of forensic investigations, digital forensic professionals must use validated tools based on state-of-the-art technology and tried and tested procedures [1, 8]. In addition to ensuring the integrity of evidence, this protects the credibility of digital forensic professionals [5].

The U.S. National Institute of Standards and Technology (NIST) conducts extensive digital forensic tool validation under its Computer Forensics Tool Testing (CFTT) Program [4]. Prior to validating the functionality of a forensic tool, it is necessary to define standards (i.e., validation criteria) for tool functionality. NIST standardization procedures involve defining tool specifications, test assertions, test cases and a test methodology [4].

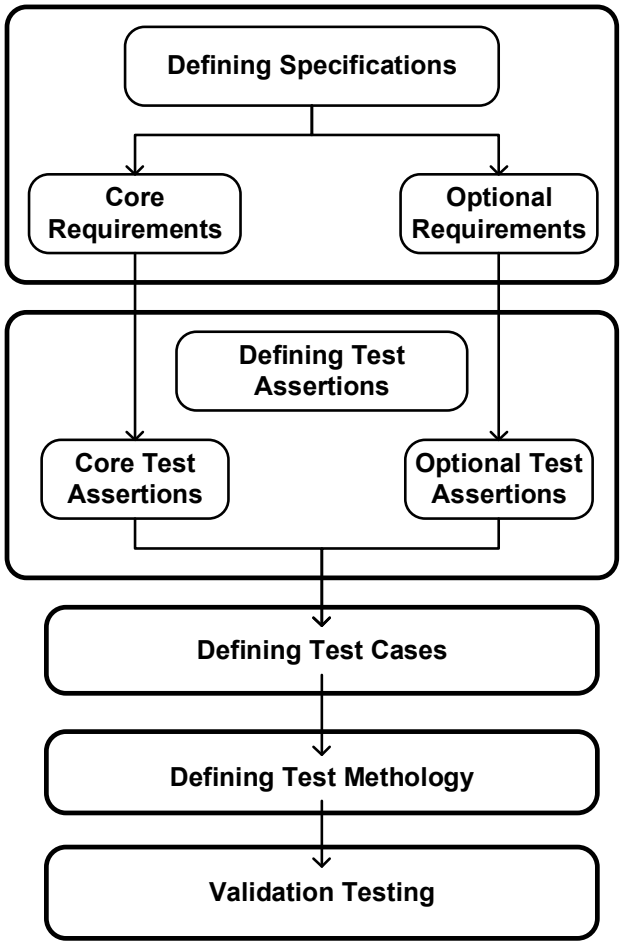

Figure 1. NIST validation cycle.

Each specification includes the required functions and technical features. These provide a basis for defining test assertions and test cases. Test assertions are defined to help ascertain that a tool meets each specification. Based on the test assertions, test cases and a methodology for testing tool functionality are defined. Figure 1 shows the NIST tool validation cycle.

Unfortunately, a standard has not as yet been specified for secure wiping tools. This chapter defines secure wiping criteria for the New Technology File System (NTFS) specific to Windows 7 and magnetic hard drives.

Data deletion is the process of removing digital data from digital storage. The goal of deletion is to remove all the data and its related information from digital storage in order to ensure data confidentiality. When data is deleted from NTFS using standard deletion functions, the operating system only marks the master file table ($MFT) records as deleted and makes minor changes such as removing links in cluster maps and directories [3].

Since no data overwriting is done, the data is recoverable using digital forensic tools, provided that the original data has not been overwritten by new data. Secure wiping tools delete files in order to prevent data retrieval. A general perception is that once files have been deleted using secure wiping tools, no information about the files can be extracted. However, this chapter shows that, even after secure wiping, critical information is left behind, especially in the $LogFile, including the names, contents and temporal information of files and information about the wiping operation. Moreover, each wiping tool leaves behind a signature pertaining to how the data was overwritten; this signature can be used to identify the tool. This chapter evaluates the functionality of Eraser, a popular secure wiping tool, and identifies the artifacts left behind after a secure wiping operation.

Validation criteria for secure wiping functionality are defined in this chapter to ensure that all the data and related information are wiped from $MFT records, the $LogFile and the hard disk (clusters containing non-resident data, index entries and other non-resident payloads of attributes). The criteria also ensure that no signature remains that could reveal information about the wiping operation and the tool that was used.

Digital forensic examinations of deleted data involve collecting evidence that includes: (i) what was deleted; (ii) when it was deleted; (iii) when the deleted files were created; (iv) what the deleted files contained; and (v) what were the sizes of the deleted files. In the light of these objectives, the forensic artifacts of a deleted file are divided into three main categories: (i) name; (ii) time; and (iii) contents. Information about the name of a file and its parent folder falls in the name category. Information about the file creation, modification and deletion times fall in the time category. Information about file content and size comes under the contents category.

User actions have effects on different areas of an NTFS. When a file is created or modified, changes are made to:

- $MFT records of the file and folder and related records ($BitMap, $MFT, etc.) that store resident data and metadata under various attributes.

- $LogFile logs that record transactions related to user actions.
- The hard disk that contains non-resident data and non-resident payloads of various attributes.

When a file has been dormant for an extended period of time, its operational records in the $LogFile are overwritten; the exact duration depends on the volume of the activities performed on the hard drive. However, related information remains in $MFT records and on the hard disk. When a file is deleted, it leaves artifacts in $MFT records, the $LogFile and hard disk. Moreover, after deletion, all previous artifacts of file creation and modification are available in the $LogFile and on the hard disk, and partially in $MFT records (new user actions generate new attributes that overwrite previous attributes). This chapter only discusses the deletion process in detail because artifacts that remain after file deletion must be considered when characterizing the secure wiping process.

The primary contribution of this work is the identification of the effects of user actions performed on a file in various segments of an NTFS, including the $MFT records, $LogFile and hard disk, and the identification of the remnants in these entities after a deletion process. These remnants are used to define the validation criteria for secure wiping functionality; specifically, to ensure that all file data and information about the wiped file and the wiping tool are removed from the filesystem (NTFS in a Windows 7 environment).

2. Deletion Effects and Deletion Types

This section briefly discusses the effects of user actions on $MFT records, the $LogFile and the hard disk. Note that only the remnants of deletion processes are discussed because these remnants must be removed by secure wiping tools. Also, the section describes two types of deletion.

2.1 Deletion Effects

A user action on a file in an NTFS has the following effects:

- **$MFT Record:** The end state of a user action is recorded in the $MFT record of the file and associated $MFT records such as the parent folder, $BitMap and $MFT. The $MFT record carries the $LogFile sequence number (LSN) in its header, which links the $MFT record to the corresponding chain of operational records in the $LogFile.

- **$LogFile:** A user action generates a chain of operational records in the $LogFile. Each record represents an NTFS transaction that is performed to complete the user action. When a new action is performed, the records in the $LogFile are not replaced with new records corresponding to the file. Instead, new records are stored in the next available space in the $LogFile [7]. The writing of records in the $LogFile is circular in nature. After the $LogFile is filled, new records are written starting from the beginning of the $LogFile [6]. The $LogFile thus contains artifacts of user actions that constitute valuable digital evidence. Interested readers are referred to [10] for details about the $LogFile structure and how it is read. Details about chains of operational records in the $LogFile for user actions and the linking of the chains with corresponding $MFT records are discussed in [9].

- **Hard Disk:** The hard disk, as well as the $MFT and $LogFile, which carry non-resident data and non-resident index entries, are updated.

2.2 Deletion Types

NTFS provides two types of file deletion: (i) deletion via the recycle bin; and (ii) permanent deletion.

Deletion via the Recycle Bin. This deletion process has two stages. In the first stage, the file is deleted and sent to the recycle bin. In the second stage, the file is removed from the recycle bin.

During the first stage, the following actions are performed by NTFS on $MFT records:

- **Deleted File:** The name of the deleted file is changed to a random name of seven characters starting with $ and followed by the letter "R" (R is for the renamed file in the recycle bin; this file is referred to as $Rxxx in this chapter). The renamed file is moved to the recycle bin and its temporal information in attribute `0x10` $Standard_ Information and attribute `0x30` $File_Name of the $MFT record are updated.

- **New File Generation:** At the same time, a new file with the same random name $Rxxx is also generated in the current user folder of the recycle bin, but it starts with $I (I is for information). The new file is referred to as $Ixxx in this chapter. A corresponding entry is made in the $MFT for the newly-generated $MFT record of $Ixxx. The file contains the file name, location and temporal

information about the original file and the file deletion process. The $Ixxx file facilitates the NTFS recovery process when the file is to be recovered from the recycle bin.

- **Deleted File Index Entry:** The index entry of the deleted file is moved from the parent folder to the current user folder of the recycle bin. Temporal information in attribute `0x10` of the $MFT record of the current user folder and the parent folder are updated to show the deletion time. The parent file index entry in its own parent folder is also updated with the time and size. In addition, if the name was indexed at the end of attribute `0x90` $Index_Root or attribute `0xA0` $Index_Allocation, then the deletion leaves the index entry (name + time + size) in the slack space.

A chain of operational records is generated in the $LogFile in which all the above transactions are logged. If a deleted file is non-resident, no changes occur to the non-resident data contents of attribute `0x80` $Data (i.e., data residing on the hard disk outside the $MFT record). Details of the changes are discussed in [9].

Clearing the file from the recycle bin during the second stage results in the following changes:

- **$Rxxx and $Ixxx:** The $MFT record headers of the $Rxxx and corresponding $Ixxx files are updated to show the new $LogFile sequence number, record sequence number, flag with the $MFT record status as deleted and update sequence number. In addition, the update sequence numbers of the sector boundaries of the $MFT record (`0x1FE-0x1FF` and `0x3FE-0x3FF`) are updated accordingly. The remaining attributes of the $MFT record are not altered or wiped. This makes information contained in the attributes available to digital forensic professionals.

- **Parent Folder:** The $MFT record of the current user folder of the recycle bin is updated to show its modified size (after the removal of two files) and temporal information in attribute `0x10` of the $MFT record. The index entries of the deleted files are removed and the next file records in line are shifted up to these locations.

A chain of operational records is accordingly generated in the $LogFile in which all the above transactions are logged. If the deleted file is non-resident, no changes occur to the non-resident data contents of attribute `0x80` $Data (i.e., data residing on the hard disk outside the $MFT record). This data is recoverable using digital forensic tools.

Permanent File Deletion. When a file is deleted via a command prompt or using `shift + delete`, the file does not go to the recycle bin, but is deleted permanently. This type of deletion leaves more forensic information than a recycle bin deletion. Almost all the attributes of the $MFT records of the deleted file are unaltered, except for the $MFT record header. The deletion remnants include:

- **Deleted File:** The $MFT record header of the deleted file is updated showing the updated $LogFile sequence number, record sequence number, flag with the $MFT record status as deleted and update sequence number. In addition, the update sequence numbers of the sector boundaries of the $MFT record (`0x1FE-0x1FF` and `0x3FE-0x3FF`) are updated accordingly. The remaining attributes of the $MFT record are not altered or wiped. This makes all the information contained in the attributes available to digital forensic professionals.
- **Index Entry of Deleted File:** The index entry of the deleted file in the $MFT record of the parent folder is removed and its attribute `0x10` is updated. This reveals the time of file deletion. In addition, if the name was indexed at the end of attribute `0x90` or the index of attribute `0xA0`, then the deletion leaves the index entry (name + time + size) in slack space. The parent file index entry in its own parent folder is also updated with the new time and size.

A chain of operational records is accordingly generated in the $LogFile in which all the above transactions are logged. If the deleted file is non-resident, no changes occur to the non-resident data contents of attribute `0x80` $Data (i.e., data residing on the hard disk outside the $MFT record). This data is recoverable using digital forensic tools.

3. File Deletion Artifacts

Forensic artifacts are divided into three categories based on their locations: (i) $MFT records; (ii) $LogFile; and (iii) hard disk. The following sections discuss each category of artifacts for the two deletion methods.

3.1 $MFT Records (Recycle Bin Deletion)

The following artifacts are present in $MFT records.

- **Name Information:** The name of the $Rxxx deleted file is present in attribute `0x30` of the $MFT record of the deleted file. The actual name and location of the parent folder under which the original file

was indexed are available in the $MFT record of $Ixxx. This file also reveals the name of the original file and its original path. The altered file name is also indexed in the current user folder of the recycle bin. If the original/renamed file is indexed at the end of attribute `0x90` or index of attribute `0xA0`, then the deletion process leaves the index entry (name + time + size) in the slack space of the original parent folder/current user folder; this information is recoverable using digital forensic tools.

- **Temporal Information:** Temporal information about file creation and deletion (file modification time when the file was renamed to $Rxxx and the $MFT record update time) is available in attribute `0x10` $Standard_Information. Temporal information about the renaming of the file to $Rxxx is also available in attribute `0x30` $File_Name. The renamed index entry (name + time + size) of the file in the current user folder of the recycle bin also reveals the deletion time (i.e., file deletion to the recycle bin). Temporal information about the original/renamed file is also available in the slack space if the original/renamed file is indexed at the end as mentioned above.

- **Content Information:** Information about file contents is available in attribute `0x80` $Data of the $MFT record of $Rxxx. In the case of resident data, the content is also available in the same attribute. In the case of non-resident data, information about the size of the data and clusters is available in attribute `0x80`. The original file size is also available in the $MFT record of $Ixxx.

3.2 $LogFile (Recycle Bin Deletion)

The $LogFile is a gold mine when it comes to forensic artifacts. All the transactions related to file creation, modification and deletion are logged here. As mentioned above, the $LogFile records produced during file creation are unchanged during file modification and deletion. These records are only overwritten from the start in a circular manner after the $LogFile is filled.

The following artifacts are present in the $LogFile:

- **Name Information:** The file name is present in previous records related to file creation, file renaming and file deletion. The records related to adding and deleting the index entry of a deleted file to/from parent folder also hold file name information.

- **Temporal Information:** Temporal information is logged about the creation, renaming and access times of the file along with the

$MFT record update time. These logs are related to changes in the $MFT records of the associated files (original file, $Rxxx and $Ixxx) and their parent folders and the parents of their parent folders. Indirect information, such as the bit representing $MFT record allocation in the non-resident location of attribute `0xB0` $BitMap of the $MFT, also provides temporal information.

- **Content Information:** File size information is logged to show the data payload in various logs related to $MFT changes in the file and its parent folder, and the parents of the parent folder all the way up to the $Root filename index. Data contents are not logged in the $LogFile except when a resident file is converted to a non-resident file, in which case the old resident content is logged in the $LogFile and is retrievable as discussed in [9]. However, if the data content is modified, it is logged in the $LogFile and the sizes of the old and new data are both available. All these records are not altered during the deletion process and are, therefore, available to digital forensic professionals.

3.3 Hard Disk (Recycle Bin Deletion)

Non-resident data of attributes stored outside $MFT records on the hard disk are included in this category. All this data is available to digital forensic professionals because it not overwritten during the deletion process. Even when the $MFT record of non-resident data is overwritten, the non-resident data may exist if the new overwritten $MFT record is resident or, in the case of a non-resident record, some other clusters are allocated if the size of new non-resident data is different from the previously-deleted data (if the number of clusters required is the same, then NTFS reuses the clusters and overwrites the data).

3.4 $MFT Records (Permanent File Deletion)

Almost all the attributes of the $MFT record of a deleted file are left unaltered, except for the header of the $MFT record. The following artifacts remain after the deletion process:

- **Name Information:** All the information in attribute `0x30` of the $MFT record of the deleted file is left unaltered, thus giving away the name of the deleted file and name of the parent folder under which the file was indexed. If the deleted file is indexed at the end of attribute `0x90` or index of attribute `0xA0` of the parent folder, the deletion process leaves the index entry (name + time + size) in the slack space of the parent folder, thus revealing the file name.

Information in attribute 0x40 $Object_ID in the $MFT record of the deleted file reveals the file GUID.

- **Temporal Information:** All the information in attribute 0x10 in the $MFT record of the deleted file is unaltered, thus giving away temporal information of the $MFT record update, owner ID and security ID. In the $MFT record of the parent folder, the index entry of the deleted file is removed and its attribute 0x10 is updated; this reveals the file deletion time. If the original deleted file is indexed at the end of attribute 0x90 or index of attribute 0xA0 of the parent folder, the deletion process leaves the index entry (name + time + size) in the slack space of the parent folder, thus revealing temporal information about the deleted file.

- **Content Information:** All the information in attribute 0x80 in the $MFT record of the deleted file is left behind, thus giving away information about the content size and contents (for resident data) and the location (for non-resident data).

3.5 $LogFile (Permanent File Deletion)

The $LogFile chain of operational records generated as a result of permanent deletion is smaller than the chain generated by recycle bin deletion. However, the artifacts related to name, temporal and content information are similar. The missing items are the changes related to the creation of the $Rxxx and $Ixxx files and the moving of the deleted file to the recycle bin.

3.6 Hard Disk (Permanent File Deletion)

The artifacts of this deletion process are the same as those of the recycle bin deletion process.

4. Validation Criteria

Under its Computer Forensics Tool Testing Program, the U.S. National Institute of Standards and Technology has defined validation criteria and standards (i.e., specifications, test assertions, test cases, test plans and test images) for a number of forensic functions [4]. However, no standards or criteria are available for secure wiping functionality. As a result, the claims made by vendors about the accuracy and efficacy of their secure wiping tools cannot be validated.

Three validation criteria for secure wiping tools that meet the standards of digital forensic investigations are:

- Specifications of the capabilities of secure wiping tools. These are further divided into core requirements (i.e., mandatory features of secure wiping tools) and optional requirements (i.e., requirements that are applicable only if a vendor claims that its tool has the features).
- Test assertions that ascertain the secure wiping tool capabilities based on the defined specifications. These are divided into core test assertions and optional test assertions.
- Test cases defined for particular environments in order to test the claimed secure wiping tool functionality based on the test assertions.

4.1 Specifications

The specifications fall into two categories: (i) core requirements; and (ii) optional features.

Secure Wiping Core Requirements (SW-CR). The following secure wiping core requirements must be met by a tool claiming to have secure wiping functionality:

- **SW-CR-01:** The tool shall overwrite the entire $MFT record of the file being wiped and accordingly make the record available in $MFT.
- **SW-CR-02:** The tool shall overwrite the entire allocated cluster(s) of non-resident data of the file being wiped and accordingly make the cluster(s) available in $BitMap.
- **SW-CR-03:** The tool shall overwrite the index entry of the file in the parent folder.
- **SW-CR-04:** The tool shall overwrite the entire chain of operational records in the $LogFile of the file being wiped.
- **SW-CR-05:** The secure wiping of a file and its parent folder shall not generate a chain of operational logs in the $LogFile of the file being wiped.
- **SW-CR-06:** The tool shall overwrite all the previous chains of operational logs in the $LogFile of the file being wiped.
- **SW-CR-07:** The tool shall overwrite the entire $MFT record of the folder being wiped along with the $MFT records of all its files and make their corresponding records available in $MFT.

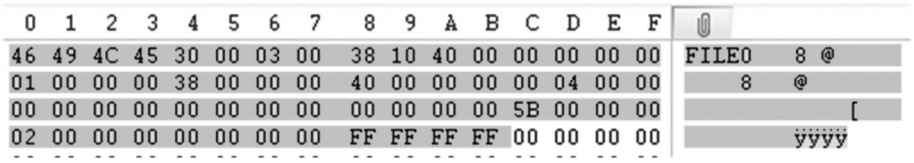

Figure 2. Empty unallocated $MFT record.

Secure Wiping Optional Features (SW-OF). The following secure wiping optional features may be met by a tool claiming to have secure wiping features:

- **SW-OF-01:** If the tool supports the overwriting of index entries of the parents of the parent folder (until $Root filename index), then it shall change the file sizes and temporal information in the index entries accordingly.

- **SW-OF-02:** If the tool supports the overwriting of deleted files, then it shall follow the SW-CR-01 and SW-CR-02 requirements for the wiped files.

4.2 Test Assertions

Assertions are used to ascertain tool functionality vis-a-vis specifications. The assertions fall into two categories: (i) core test assertions; and optional test assertions.

Secure Wiping Core Test Assertions (SW-CA). The following core test assertions are specified for testing and validating secure wiping functionality:

- **SW-CA-01:** When a tool overwrites an entire $MFT record, the first `0x3C` bytes shall be written with the standard information of an empty $MFT record as shown in Figure 2. The remaining bytes shall be written with all zeros with the update sequence number at the sector boundaries at offsets `1xFE` and `3xFE` as entered in offset `0x30`.
  Justification: Some wiping tools leave signatures based on the manner in which they overwrite $MFT records. The complete wiping of the record with zeros removes the signature.

- **SW-CA-02:** The tool shall set the record allocation bit in the $MFT for the file being wiped.
  Justification: This removes any mismatch between the $MFT

and status flag in the $MFT record after it has been wiped and marked as available.

- **SW-CA-03:** The tool shall delete the corresponding index entry and, if the deleted entry is not the last entry, shift the remaining index entries to occupy the space.
  Justification: This removes traces of the wiped file from the folder under which it is indexed. Shifting the remaining index entries upwards to occupy the space vacated by the wiped file removes all traces of the wiping operation.

- **SW-CA-04:** The tool shall overwrite the $MFT records of $Rxxx and $Ixxx and their corresponding record allocation bits in $MFT as mentioned in SW-CA-01 and SW-CA-02. Where applicable, SW-CA-05 shall also be followed.
  Justification: This removes traces of the wiped file from the recycle bin. Also, it removes any mismatch between the $MFT and the status flag in the $MFT records of the $Rxxx and $Ixxx files after the file has been wiped and its space is marked as available.

- **SW-CA-05:** For non-resident files, the tool shall overwrite the entire cluster containing non-resident data with all zeros and set the corresponding cluster allocation bit in $BitMap to zero.
  Justification: This ensures the wiping of non-resident data; data that is left in slack space is recoverable. Moreover, if the data is not wiped with all zeros, traces of the wiping operation remain. Some tools may also leave signatures based on the way they overwrite non-resident data; overwriting with all zeros removes the signatures. This also removes any mismatch between the cluster allocation status of $BitMap and the wiped cluster.

- **SW-CA-06:** The tool shall locate the entire chain in the $LogFile associated with the file being wiped and copy randomly-selected contents from the $LogFile so that the entire chain is overwritten. Also, record page headers that specify the last $LogFile sequence number or file offset and last end $LogFile sequence number and update sequence numbers at sector boundaries shall be placed carefully.
  Justification: This ensures that the chain of operational records that log the latest actions performed on the file before secure wiping are overwritten; this removes all traces of the actions. The adjustments to the various fields eliminate mismatches and ensure secure wiping.

- **SW-CA-07:** The tool shall search and identify the previous chains of operational records associated with the file being wiped and overwrite them according to SW-CA-06.
  Justification: This ensures that no information related to the name, time and contents of the wiped file is present in $LogFile.

- **SW-CA-08:** The secure wiping operation shall not generate a chain of operational records in $LogFile.
  Justification: This ensures that no traces of wiping functionality are present in $LogFile. Otherwise, it is possible to obtain information about the wiping operation and the file that was wiped.

- **SW-CA-09:** The tool shall overwrite all $MFT records of a folder and its files and their corresponding record allocation bits in $MFT as mentioned in SW-CA-01 and SW-CA-02. Where applicable, SW-CA-05 shall also be followed.
  Justification: This ensures that, when a folder is being wiped, all the files and folders contained in the folder are also wiped. This removes all traces of the folder and the wiping operation.

Secure Wiping Optional Test Assertions (SW-OA). The following optional test assertions are specified for testing and validating secure wiping functionality:

- **SW-OA-02:** If the tool supports the removal of artifacts from the parents of the parent folder, then it shall adjust their index entries according to the reduced sizes of the parent folders (i.e., without the wiped file index) and accordingly adjust their temporal information.

- **SW-OA-02:** If the tool supports the secure wiping of deleted files, then it shall perform SW-CA-04, SW-CA-06 and SW-CA-07.

4.3 Test Cases (SW-TC)

The following test cases are defined to test the functionality of secure wiping tools:

- **SW-TC-01:** Overwriting the $MFT record of the resident file when it is not in the recycle bin.

- **SW-TC-02:** Overwriting the $MFT record of non-resident data and overwriting the non-resident data when the file is not in the recycle bin and setting the corresponding record allocation bit in the $MFT of the wiped file to zero.

Table 1. Relational summary table.

No.	Requirements	Test Assertion	Test Cases
1	SW-CR-01	SW-CA-01, SW-CA-02, SW-CA-04	SW-TC-01 or SW-TC-02 or SW-TC-03 or SW-TC-04 or SW-TC-05 or SW-TC-06
2	SW-CR-02	SW-CA-05	SW-TC-02 or SW-TC-04
3	SW-CR-03	SW-CA-03	SW-TC-01 or SW-TC-02 or
			SW-TC-03 or SW-TC-04 or SW-TC-05 or SW-TC-06
4	SW-CR-04	SW-CA-06	-do-
5	SW-CR-05	SW-CA-08	-do-
6	SW-CR-06	SW-CA-07	-do-
7	SW-CR-07	SW-CA-09	SW-TC-07
8	SW-OF-01	SW-OA-01	SW-TC-01 or SW-TC-02 or SW-TC-03 or SW-TC-04 or SW-TC-05 or SW-TC-06
9	SW-OF-02	SW-OA-02	SW-TC-05, SW-TC-06

- **SW-TC-03:** Overwriting the $MFT record of the resident file when the file is in the recycle bin.
- **SW-TC-04:** Overwriting the $MFT record of the non-resident data and overwriting the non-resident data when it is in the recycle bin.
- **SW-TC-05:** Overwriting the $MFT record of the resident file after it has been deleted.
- **SW-TC-06:** Overwriting the $MFT record of the non-resident data after it has been deleted.
- **SW-TC-07:** Overwriting the $MFT records of a folder and all its files.

4.4 Relational Summary Table

Table 1 shows the relational table of specifications, assertions and test cases.

5. Validation Testing of Eraser

Eraser is an open source software tool that is often used to securely wipe files and folders [2]. The tool was selected for validation testing because it is one of the top secure wiping tools. Experiments were conducted on Eraser 6.0.10.2620 to check its secure wiping functionality on NTFS (Windows 7 Professional, Service Pack 1) against the verification standard for secure wiping functionality presented in this chapter.

Eraser was configured with the following settings:

- Default file erasure method of German VSIRT (seven passes).
- Default unused space erasure method using pseudorandom data (one pass).
- Random data source of RNGCryptoServiceProvider. A higher number of passes was not required because hardware data extraction is out of scope.

A non-resident file (converted from a resident file) was used as a test case. The target file, which was stored in a folder in the root directory of a physical drive, was securely wiped using Eraser. WinHex 16.9 x86 was used to view and analyze the information in the $MFT records, non-resident locations (including index entries) and $LogFile records. The following observations were made:

- Eraser marked the $MFT record of the target file as deleted and the corresponding record allocation bit in $MFT was set to zero.
- Eraser overwrote all the attributes of the $MFT record of the target file.
- The target file name in attribute `0x30` was overwritten with a random name.
- Eraser overwrote the entire cluster containing non-resident data with random entries. The corresponding cluster allocation bit in $BitMap was set to zero.
- Eraser replaced the corresponding index entries in the parent folder with random entries.
- Eraser did not wipe any entries from the $LogFile. Instead, it generated its own chain of operational records in the $LogFile.

The performance of Eraser with regard to overwriting data in $MFT records and non-resident data locations was satisfactory. However, it

Table 2. Eraser compliance matrix.

No.	Requirements	Test Cases	Results of Test Assertions
1	SW-CR-01	SW-TC-02	– SW-CA-01 non-compliant as the $MFT record was not overwritten as required – SW-CA-02 compliant – SW-CA-04 non-applicable
2	SW-CR-02	SW-TC-02	SW-CA-05 compliant
3	SW-CR-03	SW-TC-02	SW-CA-03 non-compliant as the corresponding entry was not deleted and the remaining entries moved up; instead, the entries were replaced with with random entries
4	SW-CR-04	-do-	SW-CA-06 non-compliant as the latest chain of operational records in the $LogFile related to the target file was not wiped
5	SW-CR-05	-do-	SW-CA-08 non-compliant as chains of operational records were generated in the $LogFile
6	SW-CR-06	-do-	SW-CA-07 non-compliant as previous chains were not wiped
7	SW-CR-07	SW-TC-07	SW-CA-09 non-applicable
8	SW-OF-01	SW-TC-07	SW-OA-01 non-applicable
9	SW-OF-02	SW-TC-05 SW-TC-06	SW-OA-02 non-applicable

failed to wipe critical forensic artifacts residing in the $LogFile. Table 2 shows the compliance matrix for Eraser based on the secure wiping criteria presented in this chapter. Researchers can replicate this validation test by creating a resident file in Notepad. The resident file must then be converted to a non-resident file by adding data to the file.

6. Conclusions

The $LogFile is a valuable source of information about deleted files and the tools used to delete the files. However, a standard has not as yet been specified for secure wiping tools. The validation criteria for secure wiping functionality defined in this chapter ensure that all the data and

related information are wiped from the $MFT records, $LogFile and hard disk. The criteria also help ensure that no signature remains that could reveal information about the wiping operation and the wiping tool that were used.

The popular Eraser secure wiping tool was evaluated against the validation criteria. Eraser's performance with regard to overwriting data in $MFT records and non-resident data locations was satisfactory. However, it failed to wipe critical forensic artifacts residing in the $LogFile.

Future research will attempt to specify similar criteria for the Resilient File System (ReFS) of Windows 8. This research will advance digital forensic efforts as well as efforts focused on protecting sensitive personal and proprietary information.

References

[1] J. Beckett and J. Slay, Digital forensics: Validation and verification in a dynamic work environment, *Proceedings of the Fortieth Annual Hawaii International Conference on System Sciences*, 2007.

[2] T. Fisher, 36 free file shredder software programs, About Tech (`www.pcsupport.about.com/od/data-destruction/tp/free-file-shredder-software.htm`), 2015.

[3] M. Miroshnichenko, Why deleted files can be recovered? Hetman Software, Walnut, California (`www.hetmanrecovery.com/recovery_news/why-deleted-files-can-be-recovered.htm`), 2012.

[4] National Institute of Standards and Technology, Computer Forensics Tool Testing Program, Gaithersburg, Maryland (`www.cftt.nist.gov/documents.htm`), 2015.

[5] B. Nelson, A. Phillips and C. Steuart, *Guide to Computer Forensics and Investigations*, Course Technology, Boston, Massachusetts, 2009.

[6] J. Oh, NTFS Log Tracker (`www.forensicinsight.org/wp-content/uploads/2013/06/F-INSIGHT-NTFS-Log-TrackerEnglish.pdf`), 2013.

[7] R. Russon and Y. Fledel, NTFS Documentation (`www.dubeyko.com/development/FileSystems/NTFS/ntfsdoc.pdf`), 2005.

[8] J. Vacca, *Computer Forensics: Computer Crime Scene Investigation*, Charles River Media, Hingham, Massachusetts, 2005.

[9] M. Zareen, Provision of Validated Toolset of Freeware Computer Forensic Tools, M.S. Thesis, Department of Information Security, Military College of Signals, National University of Sciences and Technology, Islamabad, Pakistan, 2014.

[10] M. Zareen and B. Aslam, $LogFile of NTFS: A blueprint of activities, presented at the *Seventeenth IEEE International Multi-Topic Conference*, 2014.