Reliable Time Based Forensics in NTFS

Xiaoqin Ding, Hengming Zou
School of Software, Shanghai Jiao Tong University
{micang1987, zou}@sjtu.edu.cn

ABSTRACT

Temporal evidence plays a crucial role in forensic investigations. Numerous techniques have been proposed to extract, analysis, and correlate temporal information for data forensics. But such forensic results are not reliable because timestamps of files/ directories can be tampered by anti-forensic tools. To overcome this problem, this paper proposes a reliable time based forensics approach for NTFS taking advantages of the inherent rules that govern the timing relationship between files, directories, and their metadata under various access scenarios. Since tampering of a file's timestamp often causes inconsistency in the overall timing relationship, our approach can reliably identify malicious access, modification, copy and tampering of timestamps.

1. INTRODUCTION

Interacting with computer will always leave footprints behind. Among various evidences, temporal footprints are particularly important because they could be used by investigators to confirm the intrusion time and reconstruct the events timeline. Acutely aware of this use of temporal evidences, professional intruders have tried to distort or erase time-based information from the computer using anti-forensic utilities and thus render the normal forensics ineffective or useless. For example, *Timestomp*, a tool of Metasploit Project, allows modification on MACE timestamps (last modified time, last access time, created time and MFT entry modified time) of NTFS files and often brings naïve forensic investigation into a halt.

Though evidence tampering is an important threat, state-of-the-art time based forensics seldom deal with this issue. This paper proposes a reliable time based forensics approach for NTFS by taking advantages of the inherent rules that govern the timing relationship between files, directories, and their metadata under various access scenarios. Since tampering of a file's timestamp often causes inconsistency in the overall timing relationship, our approach can reliably identify such malicious activities as illegal access, modification, copy and tampering of timestamps.

Compared to other time based forensic methods, our scheme has the following distinct features: collected time information are more reliable than that queried using APIs such as GetFileTime() which can be hooked by hackers; countermeasures are enacted to guard against anti-forensics during the whole investigation; Derived results are more accurate than those determined from simple characteristics analysis of timestamps.

2. APPROACH

Our reliable time based forensics uses a three-step process to conduct postmortem investigations: First, it extracts relevant temporal artifacts from file metadata without interfering with the data. Second, it cross-references both the discrepancies and similarities among these evidences depending on the file types. Finally, certain intrusion activities that include malicious access, modification, copy and tampering of timestamps are identified.

2.1 Timestamps Extraction

NTFS treats everything as a file and maintains at least one MFT entry containing various attributes to store relevant metadata including MACE times for each file. These timestamps are stored in the FILETIME structure, which contains a 64-bit value representing the number of 100-nanosecond intervals elapsed since January 1, 1601(UTC).

There are two attributes in the MFT that hold MACE times: $STANDARD_INFORMATION ($SI) and $FILE_NAME ($FN). Various timing information obtained from file property or Windows APIs are the $SI MAC time (the E time is hidden from users). The $SI MACE times are updated when the file is accessed or operated on. However, Windows does not typically update the MACE times in the $FN attribute. These values are updated only when the file is created, renamed or moved. The reason is that in NTFS, $FN attributes of all the files on each volume are sorted into a B+ tree as an index to facilitate fast data access, hence when a file's name or relative location in the volume changes, the corresponding $FN attribute in the index will update accordingly, as are the timestamps in it.

To extract both sets of MACE timestamps from a file/directory, we obtain the MFT ID by parsing the index entry corresponding to the file's path, search attribute type identifiers 0x10 ($SI) and 0x30 ($FN) in the MFT entry, get the FILETIME objects, and finally convert these objects to local timestamps.

2.2 Cross-reference on Evidences

The management rules for MACE timestamps in NTFS are very complicated and are dependent on both the attributes and file type as well as the applications used for opening/editing the files. To analyze these rules, we classify files into three categories:

- *Portable Executable (PE) files*: files with extensions like .exe, .dll, .bat, .com, etc.
- *Documents*: including text files (.txt, .rtf), Office files (doc, xsl, and ppt) and web files (.html).
- *Directories.*

Next we discuss timestamps changes for each of the above types.

2.2.1 $SI.MACE Times

In this set, $SI.C stands for the original creation time of the file in the computer, and will not be changed by any legal operations. The MAE times, however, are updated under different scenarios. Specifically, for any file, $SI.M changes only if the file's content or summary properties are modified. When copying or replacing operations are performed, the M time of the target file will inherit from that of the source file. For directories, the MAE times are updated to the time of operation when adding, deleting, or renaming files/subdirectories in the directory, or replacing the directory with another one with the same name. Literally speaking, $SI.A is updated while the file is accessed. However, from Vista on, Windows operating system, by taking into

account the performance requirements, does not update $SI.A by default until the file is moved to another volume or replaced by a new file. An exception to this rule is, while a document is modified by Office tools, the A time will updated to the modification time as the M and E times. In a nutshell, the E time updates when the MFT entry changes and the updating rules of this timestamp depend highly on corresponding file types. In our tests, we found that the E time of the .exe file is most sensitive which actually represents the file's last access time. For other PE files and documents, when copying or replacing happens, the E time of the target file inherits that from the source file.

2.2.2 $FN.MACE

We divide all the operations that affect this set of timestamps into three categories. The first category includes rename and intra-volume move, both are activities performed on existing files/directories within the current volume. Modification of the file name triggers an update of the $FN attribute as well as the MACE times in it. When a file is moved to another directory, the location of the file in the index will change, then the OS will search a new node in the B+ tree to put the corresponding $FN attribute in, which also causes an update of times in the $FN attribute. In these situations, the $FN.MACE times are set to the values from $SI.MACE before the operation. For instance, when a file is renamed, the $FN.MACE times are set to the $SI.MACE values prior to the renaming, then the $SI. E time is updated to the time when the renaming occurs.

The second category consists of three operations: new, inter-volume move, and copy, which are all file creation in the target volume. Under these circumstances, All four (for directories, all eight) timestamps stored in $FN are updated to the time of operation. The final category of operation contains intra-volume replace of files. The $FN.MAC of target file copy values of $FN.MAC in the original file while the $FN.E of target file inherits $SI.E of the original file. For directories, replacement will not change $FN times but set $SI.MAC to replacing time.

It can be concluded that $FN.C represents the creation time of the file in the current volume, so it should be later than or equal to $SI.C (for directories, $FN.C always equals to $SI.C).

2.3 Intrusion Activities Determination

We determine the intrusion activities based on previous analysis.

2.3.1 All file types

Normally, timestamps should satisfy conditions $SI.M <=$SI.E, $SI.C<=$FN.C, $SI.C<=$SI.A. If any is false, the timestamps were probably tampered by anti-forensic tools. But some intra-volume replacement may cause $SI.C>$FN.C or $SI.C>$SI.A and the corresponding $SI.E indicates the time of replacement. If $SI.E< $FN.E, the timestamps are unreliable.

If $SI.C<$FN.C and $FN.M=$FN.A=$FN.C=$FN.E, we can conclude that the file was moved from another volume and $FN.C is the last moving time. Meanwhile the file has not been renamed or moved after being relocated to the current volume.

If the condition $FN.M=$FN.A=$FN.C=$FN.E is false, while $SI.C<$FN.C and $SI.E>$FN.E is true, the file is replaced by another file with the same name and type in the same volume, and this file is created after the creation of the file been replaced. If $FN.M=$FN.A=$FN.C=$FN.E is false, but $SI.C=$FN.C, then $FN.MACE is copied from the $SI.MACE before last renaming or moving within the volume.

If $SI.C>$SI.M, then the contents and the summary property of the file have not been modified in the current volume; If $SI.C=$SI.M, the file have not been modified since its creation; If $SI.C<$SI.M, the $SI.M is the last modification time of content or summary property of the file in the current volume.

2.3.2 Files other than Office files and .exe files

If $SI.C>$SI.M and $SI.C>$SI.E, we can conclude that the file was copied from another volume. The $SI.M and $SI.E were inherited from those of original file.

2.3.3 Office files

If $FN.M=$FN.A=$FN.E>$SI.C, then $FN.M is the last modification time of the file contents in the current volume. If $SI.E=$FN.M also holds, then no renaming or intra-volume move happened after the modification. If $SI.E>$SI.M, the $SI.E is the time of last renaming/intra-volume moving/general property modification. If $SI.M=$FN.M is not satisfied, the timestamps must be altered.

If $SI.M=$SI.E>$SI.A>=$SI.C, the $SI.M is the time of last modification on general property of the file.

2.3.4 exe files

$SI.E time of the .exe file must be newer or equal than the other seven timestamps. If this is not true, then they are not reliable.

2.3.5 Directories

The timestamps should satisfy $SI.M<=$SI.E, $SI.C=$FN.C, $SI.C<=$SI.A. If anyone is false, the timestamps are unreliable.

If $SI.M=$SI.A=$SI.E>$SI.C, $SI.E indicates the last time of content change in the target directory. For each file or subdirectory, if its $SI.E times is the same as that of the parent directory, then this file or subdirectory is recently added or renamed. If no such file or subdirectory is found, then delete operations must have been performed within the directory.

3. CASE STUDY

Suppose that an investigation gives us the relevant timestamps of directory D:\test and file b.doc and c.txt (Table 1).

Table 1. Timestamps of suspicious files

Path	Att.	C	M	E	A
D:\test	$SI	2010/05/09 17:23:03	**2010/06/06 05:20:05**	**2010/06/06 05:20:05**	**2010/06/06 05:20:05**
	$FN	2010/05/09 17:23:03	2010/05/09 17:23:03	2010/05/09 17:23:03	2010/05/09 17:23:03
D:\test \b.doc	$SI	2010/05/09 18:10:21	*2010/05/09 19:52:03*	*2010/05/09 19:52:03*	*2010/05/09 19:52:03*
	$FN	2010/05/09 18:10:21	**2010/06/06 04:52:03**	**2010/06/06 04:52:03**	**2010/06/06 04:52:03**
D:\test \c.txt	$SI	*2010/05/25 19:25:54*	*2010/05/25 19:25:54*	*2010/05/25 19:25:54*	*2010/05/25 19:25:54*
	$FN	**2010/06/06 05:20:05**	**2010/06/06 05:20:05**	**2010/06/06 05:20:05**	**2010/06/06 05:20:05**

Since directory *D:\test* was modified at time 05:20 and the $FN MACE times of file c.txt are also 05:20, an intruder must have copied *c.txt* into *D:\test* at 05:20 and then modified $SI MACE times to hide the fact that the file was created on May 25th. For file *b.doc*, since $FN.M = $FN.A=$FN.C≠$SI.MAE, someone must have modified it at 04:52 and changed its $SI timestamps.