Time Based Data Forensic and Cross-Reference Analysis

Xiaoqin Ding
School of Software, Shanghai Jiao Tong University
800 Dong Chuan Road, Min Hang District, Shanghai
micang1987@sjtu.edu.cn

Hengming Zou
School of Software, Shanghai Jiao Tong University
800 Dong Chuan Road, Min Hang District, Shanghai
zou@sjtu.edu.cn

ABSTRACT
Data forensics is becoming increasingly important as computer related crimes intensify. In forensic investigations, temporal evidence plays a crucial role. However, the inherent volatility of time information and the tampering of such information through anti-forensic techniques have significantly lowered the reliability of temporal evidences, and posed great challenges to simple time-based forensics. To overcome this problem, this paper proposes a cross-reference time-based forensics approach for NTFS by analyzing both the discrepancies and similarities among various temporal evidences associated with file metadata and the registry. Experiment results show that our approach can reliably identify certain intrusion activities such as malicious access, modification, copy and tampering of timestamps. Some thought about dealing with anti-forensics is also provided in our analysis.

Categories and Subject Descriptors
K.4.2 [**Computers and Society**]: Social Issues - *Abuse and Crime involving computers*.

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection - *Unauthorized access.*

General Terms
Security, Reliability, Experimentation, Verification.

Keywords
Data forensics; NTFS; temporal evidence; anti-forensics.

1. INTRODUCTION
Interacting with computer will always leave footprints behind [1], [2], [3]. To discover such trails needs patience and sophistication. The art of data forensics is to collect and analyze adequate and effective evidences from multiple and sometimes numerous sources. Due to the proliferation of anti-forensic techniques and tools, identifying and pinpointing tampering and intrusion activities is becoming ever more difficult [3], [4].

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, sophisticated intruders have tried to distort or erase time-based information from the computer using anti-forensic utilities, most of which focus on the alteration of the MAC timestamps (last modified time, last access time, created time) of files, some goes one step further to modify the less visible E time (MFT entry modified time) in NTFS. For example, Timestomp, a tool of Metasploit Project [5], allows modification on MACE timestamps of NTFS files and often brings naïve forensic investigation into a halt. Besides timestamps tampering, some anti-forensic tools can also wipe out evidences such as event logs, temporary files, and MRU (Most Recently Used) lists kept by the registry, thus further exacerbate the situation faced by forensic analysts.

This paper proposes a cross-reference time-based forensics scheme to detect evidence tampering by analyzing both the discrepancies and similarities among various temporal evidences associated with file metadata and registry entries. By analyzing the management rules for timestamps in NTFS under different scenarios, extracting relevant artifacts, and cross-referencing all those temporal information, our scheme can not only reliably identify such malicious activities as illegal access and modification of files but also detect tampering of file timestamps.

The layout of the paper is as follows: the following section discusses related work. Section 3 describes sources of evidence we use for investigation followed by a formal presentation of our cross-reference time-based forensics approach in Section 4. Section 5 showcases how to conduct forensics using our approach. Section 6 concludes the paper.

2. RELATED WORK
Time-based data forensics utilizes temporal information stored throughout various places in a computer system to reconstruct past activities. Specific techniques used by current approaches include extracting and analyzing MAC timestamps, parsing web browsing history, and reconstructing events based on timelines.

K.P.Chow [6], provided a set of rules to determine the behavioral characteristics of MAC times on NTFS file system with respect to some commonly used operations that include copy, modify, delete and access. Several factors that may affect MAC time analysis such as sensitivity of times, different system clock setting, and automated scanning tools are also discussed.

Chris Boyd [7] presented a case study in which time structures and their use in Internet Explorer are discussed together with local and UTC time translation issues. In addition, a checklist for date/time analysis is given by the author that forensics examiners can follow when conducting investigation.

Florian Buchholz discussed the role file system metadata plays in data forensics and analyzed different kinds of information desirable for different types of forensic investigations [8]. The author developed a graphical timeline editor named Zeitline [9] that allows investigators to create a timeline of events gathered from many sources and to implement event reconstruction. Other timeline tools include such things as Brian Carrier's Sleuth Kit [10] and Access Data's Forensic Toolkit [11].

Many forensic researches discussed ways to extract forensic evidences from Windows Registry. Harlan Carvey [3], [12] gave numerous key/values that could be referenced when tracing user activities.

One notable thing in data forensics is the lack of discussion of countermeasure to anti-forensic tools. Also, the management rules of MACE times based on different file types are not given. Moreover, temporal information within the registry is not utilized.

3. SOURCES OF EVIDENCE

3.1 File Metadata

File metadata study is a typical approach to detect malicious operations [1], [8]. The timestamps stored in the metadata can be a critical source of information for forensic works.

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) [1], [3].

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

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, etc.), Office files (doc, xls, ppt, etc.) and web files (.html, etc.).
- *Directories*.

We test numerous files of the aforementioned types by extracting temporal information from their metadata and summarize management rules for MACE timestamps under common situations in NTFS in Table 1~2. The tables only list the attributes in which timestamps will be updated by these operations (In some situations timestamps in $FN will not be updated).

3.1.1 $SI.MACE

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 M, A, E times, however, are updated in different situations.

Specifically, for any file, $SI.M changes only if the file's content or summary properties are modified (No.6, 8 in Table 1). When copying or replacing operations (No.5, 9, 10 in Table 1) are performed, the M time of the target file will inherit from that of the source file.

Literally speaking, the A time is updated while the file is accessed. However, from Vista on, Windows operating system, by taking into account the performance requirements, does not update this timestamp by default until the file/directory is moved to another volume or replaced by a new file/directory (No.4, 9 in Table 1, No.4, 7 in Table 2). 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 (No.6 in Table 1). By editing the NtfsDisableLastAccessUpdate value in the registry, the A time can be set to be updated automatically.

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 (No.4, 5, 11 in Table 1). 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 inter-volume replacing happens, the E time of the target file inherits from that of the source file (No.5, 10 in Table 1).

For directories, the M, A, E times are updated to the time of operation when replacing the directory with another one with the same name, or adding, deleting, or renaming files/subdirectories in the directory (No.7, 9 in Table 2).

3.1.2 $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 (No.2, 3 in Table 1, No.2, 3 in Table 2), 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 entry in the index 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 (No.1, 4, 5 in Table 1, No.1, 4, 5 in Table 2), which are all file creation in the target volume actually. Under these circumstances, All four timestamps stored in $FN (for directories, all timestamps stored in both $SI and $FN) are updated to the time of operation.

Table 1. Management rules of FILE MACE timestamps in NTFS

No.	Operation	Type	Attr.	C	M	E	A[1]
1	New	All	Both	Creation time	Creation time	Creation time	Creation time
2	Rename	All	$SI	-	-	Renaming time	Renaming time*
			$FN	$SI.C before renaming	$SI.M before renaming	$SI.E before renaming	$SI.A before renaming
3	Intra-volume move	All	$SI	-	-	Moving time	Moving time*
			$FN	$SI.C before moving	$SI.M before moving	$SI.E before moving	$SI.A before moving
4	Inter-volume move	.exe	$SI	-	-	Moving time	Moving time
			$FN	Moving time	Moving time	Moving time	Moving time
		Others	$SI	-	-	-	Moving time
			$FN	Moving time	Moving time	Moving time	Moving time
5	Copy (target file)	.exe	$SI	Copy time	$SI.M of source file	Copy time	Copy time
			$FN	Copy time	Copy time	Copy time	Copy time
		Others	$SI	Copy time	$SI.M of source file	$SI.E of source file	Copy time
			$FN	Copy time	Copy time	Copy time	Copy time
6	Modify content	Text	$SI	-	Modification time	Modification time	Modification time*
		Office	Both	-	Modification time	Modification time	Modification time
7	Modify general attribute	All	$SI	-	-	Modification time	Modification time*
8	Modify summary attribute	All	$SI	-	Modification time	Modification time	Modification time*
9	Intra-volume replace	All	$SI	-	$SI.M of source file	Replacing time	$SI.A of source file
			$FN	$FN.C of source file	$FN.M of source file	$SI.E of source file	$FN.A of source file
10	Inter-volume replace	All	$SI	-	$SI.M of source file	$SI.E of source file	Replacing time*
11	Touch	.exe	$SI	-	-	Touching time	Touching time*
		Others	$SI	-	-	-	Touching time*
12	Open	All	$SI	-	-	Open time/-	Open time*

Table 2. Management rules of DIRECTORY MACE timestamps in NTFS

No.	Operation	Attr.	C	M	E	A[1]
1	New	Both	Creation time	Creation time	Creation time	Creation time
2	Rename	$SI	-	-	Renaming time	Renaming time*
		$FN	$SI.C before renaming	$SI.M before renaming	$SI.E before renaming	$SI.A before renaming
3	Intra-volume move	$SI	-	-	Moving time	Moving time*
		$FN	$SI.C before moving	$SI.M before moving	$SI.E before moving	$SI.A before moving
4	Inter-volume move	Both	Moving time	Moving time	Moving time	Moving time
5	Copy (target file)	Both	Copy time	Copy time	Copy time	Copy time
6	Modify attribute	$SI	-	-	Modification time	Modification time*
7	Replace	$SI	-	Replacing time	Replacing time	Replacing time
8	Touch /Access	$SI	-	-	-	Operation time*
9	Add\Delete\Rename files/sub-directories	$SI	-	Operation time	Operation time	Operation time

[1] Rules cited by”*” only apply to Windows operating systems in which the last access time is set to automatic updates.

The final category of operation contains intra-volume replace of files (No.9 in Table 1). 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.

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).

3.2 Registry

The Windows registry works as a repository for both system-wide and per-user configurations [13]. As a forensic artifact, abundant evidences can be found within the registry [3], [12].

The registry contains key and value pairs, and the key is a container that can consist of subkey-subvalue pairs while the value is what the key equals to. A registry key has a LastWriteTime which indicates when the key was modified such as new subkeys/values added or deleted. In addition, there are many locations in registry containing temporal information and they are list in Table 3. When the operating system starts up and a user logs on, LastWriteTimes of some keys are updated to the logon time.

Meanwhile, there exist other values containing the logon or logoff time (No.1, 2 in Table 3). Under normal circumstances, these timestamps should be very close to each other. If any of them deviate from others significantly, it is an indication that it is modified by someone.

As mentioned above, there are numerous MRU lists located throughout the registry logging specific actions performed by the users (No.3 in Table 3): UserAssist records the execution history of applications launched from Internet toolbar and active desktop; RecentDocs maintains a list of recently opened documents; MUICache stores absolute paths of executable files once they run; OpenSaveMRU keeps a MRU list of files opened and saved through Windows Shell; LastVisitedMRU is MRU of applications which have opened files in OpenSaveMRU. In addition, Windows Office also maintains MRU lists.

Another important key in the registry is USBSTOR which consists of subkeys listing all the installed USB storage devices, and the LastWriteTimes of these subkeys record the time when the devices were lastly plugged in (No.4 in Table 3) and can thus be utilized by investigators [3], [12].

Table 3. Temporal information for forensics in registry

No.	Indication	Source	Location
1	Log on Time	Value	HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Prefetcher\StartTime
			HKLM\SOFTWARE\Microsoft\Windows NT \CurrentVersion\ProfileList\S-1-5-19\ProfileLoadTime
			HKLM\SOFTWARE\Microsoft\Windows NT \CurrentVersion\ProfileList\S-1-5-20\ProfileLoadTime
		Last Write Time	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
			HKLM\SYSTEM\CurrentControlSet\Control\ComputerName
2	Log off Time	Value	HKLM\SYSTEM\CurrentControlSet\Control\Windows\Shutdown Time
			HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Prefetcher\ExitTime
3	MRU	Last Write Time	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist
			HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
			HKCU\Software\Microsoft\Windows\ShellNoRoam\MUICache
			HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU
			HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU
			HKCU\Software\Microsoft\Office\12.0\WORD\File Name MRU
4	Removable disk	Last Write Time	HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR

4. OUR FORENSICS APPROACH

Based on the above analysis, we propose a cross-reference time-based forensics scheme that consists of three procedures:

- *Temporal evidence extraction*
- *Metadata-registry cross-reference*
- *Metadata-metadata cross-reference*

Next we describe the three procedures in detail.

4.1 Temporal Evidences Extraction

The first step in our approach is to extract temporal evidences from the registry and file metadata.

First we make an image of the registry (using FTK Imager of AccessData [11]) and extract from the image those significant values and LastWriteTimes given in Table 3 (using Harlan Carvey's Regripper [14]).

Then we extract temporal information from the metadata of the suspicious files/directories that are indicated in the MRU lists of the registry. To acquire all MACE timestamps in both $SI and $FN attributes, we obtain the MFT ID by parsing the index entry corresponding to the file's path, search attribute type identifiers 0x10 ($SI) and 0x30 ($FN) in the MFT entry, get the FILETIME objects, and convert these objects to local timestamps. If the MRU lists were wiped by the intruder, we will parse all MFT entries.

4.2 Metadata-Registry Cross-Reference

The second step of our approach is to cross-reference the temporal evidences extracted from file metadata and registry in the previous step. For example, if a new subkey of USBSTOR is found and the user is unaware of the existence of the relevant device, the subkey's LastWriteTime implies the existence of the intrusion and can be referenced as an important time point.

If the last access time is updated automatically, we compare the $SI.A times of the most recently accessed document recorded in the key RecentDocs with the LastWriteTimes of the key and its subkeys. If a document's $SI.A is much earlier than the corresponding LastWriteTime, the timestamps of the document must have been altered. If a subkey's LastWriteTime is the same as RecentDocs's LastWriteTime, the time is the last modified time of the most recently accessed document the subkey recorded. If the time is different from $SI.M of the document, the timestamps in the file metadata must have been falsified.

Associating these temporal evidences with the intrusion time the user reports, the investigator can determine a suspicious intrusion time period.

4.3 Metadata-Metadata Cross-Reference

The final step checks the reliability of file timestamps extracted from metadata and cross-references them to determine the intrusion activities based on different file types.

4.3.1 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.

4.3.2 Files other than Office files and .exe file

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 the source files.

4.3.3 Office files

If $FN.M=$FN.A=$FN.E>$SI.C, $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.

4.3.4 .exe files

Under all circumstances, $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.

If the last access time is set to automatic updates, the $SI.A should be the same as $SI.E. Or else the timestamps must have been altered.

4.3.5 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.

5. CASE STUDY

Next we showcase our approach to forensics with a case study. The target host runs Windows XP (the last access time is updated automatically by default) and the file system is NTFS. A user reports that the host suffered intrusion attack on June 6th, 2010 and he wants to know what happened.

First we use the FTK Imager to make an image of the registry. For the sake of protecting the host machine, we copy the image to our computer and run the Regripper to analyze the image. We find that a new subkey appeared under the key USBSTOR, of which LastWriteTime happens to be June 6th:

```
Disk&Ven_Generic&Prod_USB_Flash_Drive&Rev_1.00
[Sun Jun  6 04:32:42 2010]
  S/N: 0707212210380038&0 [Sun Jun  6 04:32:46 2010]
  FriendlyName : Generic USB Flash Drive USB Device
  ParentIdPrefix : 8&2ad21ad7&0
```

We can, therefore, conclude that the intruder plugged a removable disk in the target computer on June 6th, 2010. Moreover, the LastWriteTime of the RecentDocs has the same date:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\Recent
Docs LastWriteTime Sun Jun 6 05:52:03 2010
  10 = b.xlsx
  63 = c.txt
  8 = a
  …
```

Table 4. Timestamps of suspicious files/directories

Path	Att.	C	M	E	A
D:\a	$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:\a\b.xlsx	$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 05:52:03**	**2010/06/06 05:52:03**	**2010/06/06 05:52:03**
D:\a\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**

Now, it can be deduced that the intrusion happened between 4:00 and 6:00 on June 6th, 2010. Then we extract MACE timestamps of these files in Table 4.

We find that the $SI.MAE of directory D:\a are all 05:20, which indicates that the directory D:\a had its content changed at 05:20. Besides, the .txt subkey of RecentDocs also records the same timestamp:

Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.txt LastWriteTime ***Sun Jun 6 05:20:05 2010***

6 = c.txt
…

Then we check c.txt in the directory and find its $FN.MACE are also 05:20. This tells us that the intruder created c.txt in D:\a (or copy from another volume) at 05:20 and modified $SI.MACE to May 25th manually to hide the fact.

For b.xlsx, we find that $FN.M=$FN.A=$FN.E≠$SI.MAE, while the information about .xlsx subkey of RecentDocs is as follows:

Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.xlsx LastWriteTime ***Sun Jun 6 05:52:03 2010***

4 = b.xlsx
…

From the information above we know that the intruder modified b.xlsx at 05:52, and this timestamp is the same as the LastWriteTime of the RecentDocs which indicates that the file is the most recently modified document in the system. Then the intruder falsified the $SI timestamps of b.xlsx to an earlier time.

6. SUMMARY

This paper proposes a *Time Based Data Forensic and Cross-Reference Analysis* approach for Windows operating systems. The proposed scheme traces intruders' malicious operation trail and uncovers evidence tampering by cross-referencing and parsing temporal information in file metadata and the Windows registry.

7. REFERENCES

[1] Carrier, B. 2005. *File System Forensic Analysis*. Addison-Wesley Professional.

[2] Carvey, H. 2004. *Windows Forensics and Incident Recovery*. Addison-Wesley Professional.

[3] Carvey, H. 2007. *Windows Forensic Analysis*. Syngress Publishing.

[4] Hilley, S. 2007. Anti-forensics with a small army of exploits. *Digital Investigation*. 4, 1 (2007), 13-15.

[5] Foster, J and Liu, V. 2005. Catch me if you can. In *Blackhat Briefings 2005* (Las Vegas, NV, August 2005).

[6] Chow, K. P., Law, F. Y., Kwan, M. Y., and Lai, P. K. 2007. The Rules of Time on NTFS File System. In *Proceedings of the Second international Workshop on Systematic Approaches To Digital Forensic Engineering* (April 10 - 12, 2007). SADFE. IEEE Computer Society, Washington, DC, 71-85. DOI= http://dx.doi.org/10.1109/SADFE.2007.22.

[7] Boyd, C. and Forster, P. 2004. Time and date issues in forensic computing-a case study. *Digital Investigation*. 1, 1 (2004), 18-23.

[8] Buchholz, F and Spafford, E. 2004. On the role of file system metadata in digital forensics. *Digital Investigation*. 1, 4 (2004).

[9] Buchholz, F and Falk, C. 2005. Design and implementation of zeitline: a forensic timeline editor. In *Digital Forensics Research Conference*, New Orleans, LA, 2005.

[10] Carrier, B. The Sleuth Kit. http://www.sleuthkit.org.

[11] AccessData Corp. Forensic Toolkit. http://www.accessdata.com/forensictoolkit.html.

[12] Carvey, H. 2005. The Windows registry as a forensic resource. *Digital Investigation*. 2 (2005), 201-205.

[13] Russinovich, M. E. and Solomon, D. A. 2004. *Microsoft Windows Internals, Fourth edition*. Microsoft Press.

[14] Carvey, H. The Regripper. http://regripper.net/.