

# Digital forensic techniques for static analysis of NTFS images

Mamoun Alazab

Internet Commerce Security Laboratory  
University of Ballarat, Australia  
m.alazab@ballarat.edu.au

Sitalakshmi Venkatraman

Internet Commerce Security Laboratory  
University of Ballarat, Australia  
s.venkatraman@ballarat.edu.au

Paul Watters

Internet Commerce Security Laboratory  
University of Ballarat, Australia  
p.watters@ballarat.edu.au

## ABSTRACT

Static analysis of the Windows NT File System (NTFS) which is the standard and most commonly used file system could provide useful information for digital forensics. However, since the NTFS disk image records every event in the system, forensic tools need to process an enormous amount of information related to user / kernel environment, buffer overflows, trace conditions, network stack and other related subsystems. This leads to imperfect forensic tools that are practical for implementation but not comprehensive and effective. This research discusses the analysis technique to detect data hidden based on the internal structure of the NTFS file system in the boot sector. Further, it attempts to unearth the vulnerabilities of NTFS disk image and the weaknesses of the current forensic techniques. This paper argues that a comprehensive tool with improved techniques is warranted for a successful forensic analysis.

Key Words: NTFS, Forensics, disk image, data hiding.

## 1. Introduction

Digital forensics is the science of identifying, extracting, analyzing and presenting the digital evidence that has been stored in the digital electronic storage devices to be used in a court of law [1, 2, 3]. It attempts to provide full descriptions of a digital crime scene. In computer systems, the primary goals of digital forensic analysis are fivefold: i) to identify all the unwanted events that took place, ii) to ascertain their effect on the system, iii) to acquire the necessary evidence to support a lawsuit, iv) to prevent future incidents by detecting the malicious techniques used and v) to recognize the incitement reasons and

intendance of the attacker for future predictions [4]. The general component in digital forensic process are; acquisition, preservation, and analysis [5].

Digital electronic evidence could be described as the information and data of investigative value that are stored by an electric device, such evidence [6]. This research focuses on the third goal of acquiring the necessary evidence of intrusions that take place on a computer system. In particular, this paper investigates the digital forensic techniques that could be used to analyze and acquire evidences from the most commonly used file systems on

computers, namely, Windows NT File System (NTFS).

Today, NTFS file system is the basis of predominant operating systems in use, such as Windows 2000, Windows XP, Windows Server 2003, Windows Server 2008, Windows Vista, Windows 7 and even in most free UNIX distributions [7, 8, 9]. Hence, attackers try to target on NTFS as this could result in affecting more computer users. Another compelling reason for witnessing a strong relationship between computer crime and the NTFS file system is the lack of literature that unearth the vulnerabilities of NTFS and the weaknesses of the present digital forensic techniques [10]. This paper attempts to fill this gap by studying the techniques used in the analysis of the NTFS disk image. Our objectives are i) to explore the NTFS disk image structure and its vulnerabilities, ii) to investigate different commonly used digital forensic techniques such as signatures, data hiding, timestamp, etc. and their weaknesses, and iii) finally to suggest improvements in static analysis of NTFS disk image.

## 2. Research Methodology

In order to achieve the above mentioned objectives of this research work, we conducted an empirical study using selected digital forensic tools that are predominantly used in practice. Several factors such as effectiveness, uniqueness and robustness in analyzing NTFS disk image were considered in selecting the tools / utilities required for this empirical study. Since each utility does some specific functionality, a collection of such tools were necessary to perform a comprehensive set of functionalities. Hence, the following forensic utilities / tools were adopted to conduct the experimental investigation in this research work:

- i) Disk imaging utilities such as dd [11] or dcfldd V1.3.4-1 [12] for obtaining sector-by-sector mirror image of the disk;

- ii) Evidence collection using utilities such as Hexedit [13], Frhed 1.4.0[14] and Strings V2.41[15] to introspect the binary code of the NTFS disk image;
- iii) NTFS disk analysis using software tools such as The Sleuth KIT (TSK) 3.01[16] and Autopsy [17] and NTFSINFO v1.0 [18] to explore and extract intruded data as well as hidden data for performing forensic analysis.

Test data for the experimental investigation with the above tools was created on a Pentium (R) Core (TM) 2 Due CPU, 2.19 GHz, 2.98 of RAM with Windows XP professional that adopts the NTFS file system partition. In this research, we focus on the boot sector of the NTFS disk image for the empirical study. We adopt the following three steps to perform digital forensic analysis in a comprehensive manner:

- i) Hard disk acquisition,
- ii) Evidence searching and
- iii) Analysis of NTFS file system.

### 2.1 Hard Disk Data Acquisition

In this step, we used the dcfldd developed by Nicholas Harbour and dd utility from George Garner to acquire the NTFS disk image from the digital electronic storage devices since they are simple and flexible acquisition tools. The advantage of using these tools is that we could extract the data in or between partitions to a separate file for more analysis, and in addition, it provides built-in MD5 hashing features. Some of its salient features allow the analyst to calculate, save, and verify the MD5 hash values. In digital forensic analysis, using hashing technique is important to ensure data integrity and to identify whether the value of data has been changed as well as explore known data objects [19].

### 2.2 Evidence searching

An evidence of intrusion could be gained by looking for some known signatures, timestamps as well as even searching for hidden data [20]. In this step, we used Strings command by Mark Russinovich, Frhed hexeditor tool by Rihan Kibria and WinHex hexeditor tool by X-Ways Software Technology AG to detect a keyword or phrase from the disk image.

### 2.3 Analysis of NTFS File System

The final step in the experimental investigation is to analyze the data obtained from the NTFS disk image that contribute towards meaningful conclusions of the forensic investigation. We adopted a collection of tools such as the Sleuth Kit (TSK) and Autopsy Forensic by Brian Carrier and NTFSINFO v1.0 from Microsoft Sysinternals by Mark Russinovich to perform different aspects of the NTFS file system analysis.

### 3. Analysis of the boot sector of the NTFS disk image

The first step for a digital forensic investigator is to acquire a duplicate copy of the NTFS disk image before beginning the analysis so as to ensure that the data on the original devices have not been changed during the analysis. Therefore, it is required to isolate the original infected computer to extract the evidence that could be found on the electronic storage devices from the disk image as the image captures the invisible information as well [21]. The advantages of analyzing disk images are that the investigators can: a) preserve the digital crime-scene, b) obtain the information in slack space, c) access unallocated space, free space, and used space, d) recover file fragments, hidden or deleted files and directories, e) view the partition structure and f) get date-stamp and ownership of files and folders [3, 22].

To understand how intrusions can lead to data hiding, deleting, etc. and to facilitate recovery, it is essential to understand the physical characteristics of the Microsoft NTFS file system. Master File Table (MFT) is the core of NTFS since it contains details of every file and folder on the volume and allocates two sectors for every MFT entry [23]. Each MFT entry has a fixed sized which is 1 KB (At byte offset 64 in the boot sector to identify the MFT record size). We provide the MFT layout and represent the plan of the NTFS file system using Figure 1. NTFS exists to read and write the attributes instead of read and write the file content. The MFT enables a forensic analyst to examine in some detail the structure and working of the NTFS volume. Therefore, it's important to understand how the attributes are stored in the MFT entry.

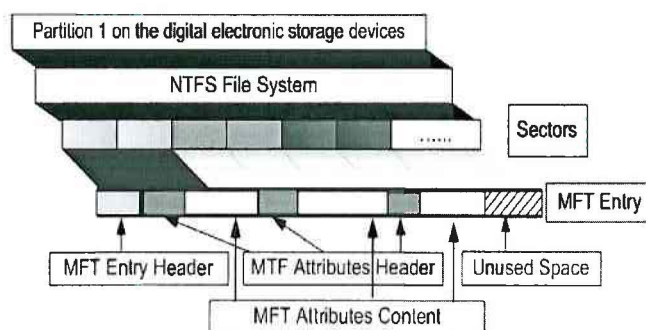


Figure 1: MFT layout structure.

MFT entry within the MFT contains attributes that can have any format and any size. Further, as it shows in Figure 1, every attribute contains an entry header which is allocated in the first 42 bytes of a file record, and it contains an attribute header and attribute content. The attribute header is used to identify the size, name and the flag value. The attribute content can reside in the MFT followed by the attribute header if the size is less than 700 bytes (known as a resident attribute), otherwise it will store the attribute content in an external cluster called cluster run (known as a non-resident attribute). This is because; the MFT entry is 1KB in

size and hence cannot fit anything that occupies more than 700 bytes.

Metadata files are used to describe the file system. We created a NTFS disk image of the test computer using the dd utility and investigated the boot sector. We used NTFSINFO tool on the disk image as shown in Table 1 which shows the boot sector of the test device and information about the on-disk structure: it enables you to view the MFT information, allocation size, volume size and metadata files. We extracted information such as the size of clusters, sector numbers in the file system, starting cluster address of the MFT, the size of each MFT entry and the serial number given for the file system.

Free clusters : 106696  
Free space : 416 MB (86% of drive)

Allocation Size

Bytes per sector : 512  
Bytes per cluster : 4096  
Bytes per MFT record : 1024  
Clusters per MFT record: 0

MFT Information

MFT size : 0 MB (0% of drive)  
MFT start cluster : 41300  
MFT zone clusters : 41344 - 56800  
MFT zone size : 60 MB (12% of drive)  
MFT mirror start : 61949

Meta-Data files

Table 1: NTFS Information Details.

From the information gained above and from analyzing the boot sector image as shown in Figure 2, we performed an analysis of the data structure of this boot sector and this is summarized in Table 2.

Volume Size  
-----  
Volume size : 483 MB  
Total sectors : 991199  
Total clusters : 123899

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
000000000	EB	52	90	4E	54	46	53	20	20	20	00	02	08	00	00	00	®\NTFS
000000016	00	00	00	00	00	F8	00	00	3F	00	FF	00	20	00	00	00	ø ? ý
000000032	00	00	00	00	80	00	00	00	DF	1F	0F	00	00	00	00	00	€ ¢
000000048	04	00	00	00	00	00	00	00	FD	F1	00	00	00	00	00	00	ýñ
000000064	F6	00	00	00	01	00	00	00	12	04	43	38	37	43	38	68	ü C87C8h
000000080	00	00	00	00	FA	33	C0	8E	DO	BC	00	7C	FB	B8	C0	07	ú3ÀŽĐ¼  ú,À
000000096	8E	D8	E8	16	00	B8	00	0D	8E	C0	33	DB	C6	06	0E	00	Ž0è , ŽÀ30ÙE
000000112	10	E8	53	00	68	00	0D	68	6A	02	CB	8A	16	24	00	B4	èS h hj ÈŠ \$
000000128	08	CD	13	73	05	B9	FF	FF	8A	F1	66	0F	B6	C6	40	66	í s 'ýýŠñf ¶Æ0f
000000144	0F	B6	D1	80	E2	3F	F7	E2	86	CD	C0	ED	06	41	66	0F	¶Ñeá?+átÍÁÍ Af
000000160	B7	C9	66	F7	E1	66	A3	20	00	C3	B4	41	BB	AA	55	8A	·Éf+áfE Á'A»'UŠ
000000176	16	24	00	CD	13	72	0F	81	FB	55	AA	75	09	F6	C1	01	\$ í r ÐáU'á u óÁ
000000192	74	04	FE	06	14	00	C3	66	60	1E	06	66	A1	10	00	66	t p Áf' f; f
000000208	03	06	1C	00	66	3B	06	20	00	0F	82	3A	00	1E	66	6A	f; ,: fj
000000224	00	66	50	06	53	66	68	10	00	01	00	80	3E	14	00	00	fP Sfh €>
000000240	0F	85	0C	00	E8	B3	FF	80	3E	14	00	00	0F	84	61	00	... è=ýe> „a
000000256	B4	42	8A	16	24	00	16	1F	8B	F4	CD	13	66	58	5B	07	'BŠ \$ <óí rX[
000000272	66	58	66	58	1F	EB	2D	66	33	D2	66	0F	B7	0E	18	00	rXfX e-f3Óf .
000000288	66	F7	F1	FE	C2	8A	CA	66	8B	DO	66	C1	EA	10	F7	36	f+RpÁŠÉf<DrÁE +6
000000304	1A	00	86	D6	8A	16	24	00	8A	E8	C0	E4	06	0A	CC	B8	+0Š \$ ŠèÀè ì,
000000320	01	02	CD	13	0F	82	19	00	8C	C0	05	20	00	8E	C0	66	í , eÀ ŽÀf
000000336	FF	06	10	00	FF	0E	0E	00	0F	85	6F	FF	07	1F	66	61	ý ý ...oy fa
000000352	C3	A0	F8	01	E8	09	00	A0	FB	01	E8	03	00	FB	EB	FE	Á ø è ú è úèp
000000368	B4	01	8B	FO	AC	3C	00	74	09	B4	0E	BB	07	00	CD	10	' <ó-< t ' » í
000000384	EB	F2	C3	0D	0A	41	20	64	69	73	6B	20	72	65	61	64	èóÁ A disk read
000000400	20	65	72	72	6F	72	20	6F	63	63	75	72	72	65	64	00	error occurred
000000416	0D	0A	4E	54	4C	44	52	20	69	73	20	6D	69	73	73	69	NTLDR is missi
000000432	6E	67	00	0D	0A	4E	54	4C	44	52	20	69	73	20	63	6F	ng NTLDR is co
000000448	6D	70	72	65	73	73	65	64	00	0D	0A	50	72	65	73	73	mpressed Press
000000464	20	43	74	72	6C	2B	41	6C	74	2B	44	65	6C	20	74	6F	Ctrl+Alt+Del to
000000480	20	72	65	73	74	61	72	74	0D	0A	00	00	00	00	00	00	restart
000000496	00	00	00	00	00	00	00	00	83	A0	B3	C9	00	00	55	AA	f =É U'
000000512	05	00	4E	00	54	00	4C	00	44	00	52	00	04	00	24	00	N T L D R \$

Figure 2: First Sector of the test boot Sector.

<i>Byte Range</i>	<i>Size</i>	<i>Description</i>	<i>Value</i>	<i>Note</i>
0 -- 2	3	Jump to boot code	9458411	If bootable, jump. If non-bootable, used to store error message
3 -- 10	8	OEM Name – System ID	NTFS	
11 -- 12	2	Bytes per sector:	512	
13 -- 13	1	Sectors per cluster	8	
14 -- 15	2	Reserved sectors	0	Unused
16 -- 20	5	Unused	0	Unused
21 -- 21	1	Media descriptor	0	
22 -- 23	2	Unused	0	Unused
24 -- 25	2	Sectors per track	63	Not Check
26 -- 27	2	Number of heads	255	Not Check
28 -- 31	4	Unused	32	Not Check
32 -- 35	4	Unused	0	Unused
36 -- 39	4	Drive type check	80 00 00 00	For USB thumb drive
40 -- 47	8	Number of sectors in file system (volume)	0.47264 GB	
48 -- 55	8	Starting cluster address of \$MFT	4*8=32	
56 -- 63	8	Starting cluster address of MFT Mirror \$DATA attribute	619,49	
64 -- 64	1	Size of record - MFT entry	2 <sup>10</sup> =1024	
65 -- 67	3	Unused	0	Unused
68 -- 68	1	Size of index record	01h	
69 -- 71	3	Unused	0	Unused
72 -- 79	8	Serial number	C87C8h	
80 -- 83	4	Unused	0	Unused
84 -- 509	426	Boot code	~	
510 --511	2	Boot signature	0xAA55	

**Table 2:** Data structure for the test boot sector

#### 4. Analysis of the Hidden Data in the \$Boot metadata file system

Attackers use different techniques such as disguising file names, hiding attributes and deleting files to intrude the system. Since the Windows operating system does not zero the slack space, it becomes a vehicle to hide data, especially in \$Boot file. Hence, in this study, we analyze the hidden data in the \$Boot file. The \$Boot entry is stored in a metadata file at the first cluster in sector 0 of the file system called \$Boot from where the system boots. It is the only metadata file that has a static location so that it cannot be relocated. Microsoft allocates the first 16 sectors of the file system to \$Boot and only

half of these sectors contains non-zero values [3].

NTFS file system requires knowledge and experience to analyze the data structure and the hidden data [24]. The \$Boot metadata file is located in MFT entry 7 and contains the boot sector of the file system. It contains information about the size of the volume, clusters and the MFT. The \$Boot metadata file has four attributes: \$STANDARD\_INFORMATION, \$FILE\_NAME, \$SECURITY\_DESCRIPTION and \$DATA. The \$STANDARD\_INFORMATION attribute contains temporal information such

as flags, owner, security ID and the last accessed, written, and created times. The \$FILE\_NAME attribute contains the file name in Unicode, the size and temporal information as well. The \$SECURITY\_DESCRIPTION attribute contains information about the access control and security properties. Finally, the \$DATA attribute contains the file contents. These are illustrated for the test sample as shown in Table 2 using the following TSK command tools:

```
Istat -f ntfs c:\image.dd 7
```

---

MFT Entry Header Values:

```
Entry: 7      Sequence: 7
$LogFile Sequence Number: 0
Allocated File
Links: 1
```

\$STANDARD\_INFORMATION Attribute Values:

```
Flags: Hidden, System
Owner ID: 0
Created: Mon Feb 09 12:09:06 2009
File Modified: Mon Feb 09 12:09:06 2009
MFT Modified: Mon Feb 09 12:09:06 2009
Accessed: Mon Feb 09 12:09:06 2009
```

\$FILE\_NAME Attribute Values:

```
Flags: Hidden, System
Name: $Boot
Parent MFT Entry: 5      Sequence: 5
Allocated Size: 8192     Actual Size: 8192
Created: Mon Feb 09 12:09:06 2009
File Modified: Mon Feb 09 12:09:06 2009
MFT Modified: Mon Feb 09 12:09:06 2009
Accessed: Mon Feb 09 12:09:06 2009
```

Attributes:

```
Type: $STANDARD_INFORMATION (16-0)
Name: N/A Resident size: 48
Type: $FILE_NAME (48-2) Name: N/A Resident
size: 76
Type: $SECURITY_DESCRIPTOR (80-3) Name:
N/A Resident size: 116
Type: $DATA (128-1) Name: $Data Non-Resident
size: 8192
0 1
```

---

Table 2: "\$Boot Attributes"

Hence the \$Boot attribute of the NTFS file system could be used to hide data. By analyzing the hidden data in the boot sector,

one could provide useful information for digital forensics. The size of the data that could be hidden in the boot sector is limited by the number of non-zero that Microsoft allocated in the first 16 sectors of the file system. The data could be hidden in the \$Boot metadata files without raising suspicion and without affecting the functionality of the system [25].

Analysis of the \$Boot attribute of the NTFS file system will identify any hidden data. The analyzer should start by making a comparison between the boot sector and the backup boot sector. The image with the boot sector and backup boot sector are supposed to be identical; otherwise there is some data hidden on the \$Boot file. One method is to check the integrity of the backup boot sector and the boot sector by calculating the MD5 for both of them. A difference in checksum indicates that there is some hidden data. We performed this comparison by adopting the following commands on the \$Boot image file and the backup boot image, see the applied below:

```
dd if=image.dd bs=512 count=1 skip=61949
of=c:\backupbootsector.dd -md5sum -
verifymd5 -md5out=c:\hash1.md5
```

```
dd if=image.dd bs=512 count=1
of=c:\bootsector.dd -md5sum -verifymd5 -
md5out=c:\hash2.md5
```

We found that hidden data in the \$Boot file was not detected directly by the tools used in this study and manual inspections were required alongside these forensic tools. Hence, through the analysis conducted with various utilities and tools, we arrived at the following results:

1. There is a huge amount of data analysis required while scanning the entire NTFS disk image for forensic purposes. Just by focusing on the hidden data in the \$Boot, this empirical study showed that many

- tools and utilities have to be adopted and it takes an immense amount of time to analyze the data derived.
2. Not all computer infections are detected by forensic tools, especially intrusions that are in the form of hidden data in the \$Boot file.
  3. By adopting a manual introspection of the \$Boot file using the three-step approach of i) hard disk acquisition, ii) evidence searching and iii) analysis of the NTFS file system, we could identify hidden data in the \$Boot file.
  4. Searches can be performed to extract the ASCII and UNICODE characters from binary files in the disk image on either the full file system image or just the unallocated space, which could speed-up the process of identifying hidden data.
  5. Microsoft has different versions of the NTFS file system. While Windows XP and Windows Server 2003 use the same version, Windows Vista uses the NTFS 3.1 version [7]. The new NTFS 3.1 has changed the on-disk structure. For example, the location of the volume boot record is at physical sector 2,048. Not all existing tools work with all the different versions of NTFS file system, hence a comprehensive tool is warranted even with the changes in the NTFS file structure.

## 5. Conclusions and Future Work

This paper has attempted to explore the difficulties involved in digital forensics, especially in conducting static analysis of NTFS disk images and propose a solution method. In this empirical study, we have found the boot sector of the NTFS file system could be used as a vehicle to hide data by computer attackers. This is an important NTFS file system weakness to be addressed as research in this domain area could lead to effective methods for the open

problem of detecting new malicious codes that use this mode of attack. The existing forensic software tools are not competent enough to comprehensively detect hidden data in boot sectors. As a first step to address this problem, we have formulated a three-step forensic analysis process to facilitate the research methodology. We have reported the results gathered by adopting this process. One clear achievement through this research study is that we were successful in identifying some unknown malicious hidden data in the \$Boot file that were hidden from current well-known virus scanners. The research methodology reported in this paper could be adopted to analyze other sectors of the NTFS file system as well.

In this initial research investigation conducted, we had adopted a few forensic techniques and manual inspections of the NTFS file image. Our next stage of this research work would be to automate the proposed process so as to facilitate forensic analysis of the NTFS disk image in an efficient and comprehensive manner. We plan to extract signatures intelligently so as to detect efficiently new malware that use hidden and obfuscated modes of attack. This would help trigger more research to be conducted in satisfying the objective of automatically and proactively identifying unseen malware that try to evade detection.

## References:

- [1] Reith, M.; Carr, C. & Gunsch, G., "An examination of digital forensic models", International Journal of Digital Evidence, 2002, 1, 1-12.
- [2] Technical Working Group for Electric Crime Scene Investigation. "Electronic Crime Scene Investigation: A Guide for First Responders", 2001.

- [3] Carrier, B., *"File system forensic analysis"*, Addison-Wesley Professional, USA, 2008.
- [4] Ardisson, S., *"Producing a Forensic Image of Your Client's Hard Drive? What You Need to Know"*, Qubit, 2007, 1, 1-2.
- [5] Andrew, M., *"Defining a Process Model for Forensic Analysis of Digital Devices and Storage Media"*, Systematic Approaches to Digital Forensic Engineering, 2007, SADFE 2007. Second International Workshop on, 2007, 16-30.
- [6] Investigation, E., *"Electronic Crime Scene Investigation: A Guide for First Responders"*, US Department of Justice, NCJ, 2001, 187736.
- [7] Svensson, A., *"Computer Forensic Applied to Windows NTFS Computers"*, Stockholm's University, Royal Institute of Technology, 2005.
- [8] NTFS, <http://www.ntfs.com>, 22/2/2009.
- [9] Purcell, D. & Lang, S., *"Forensic Artifacts of Microsoft Windows Vista System"*, Lecture Notes in Computer Science, Springer, 2008, 5075, 304-319.
- [10] Newsham, T.; Palmer, C.; Stamos, A.; Burns, J. & iSEC Partners, I., *"Breaking forensics software: Weaknesses in critical evidence collection"*, Proceedings of the 2007 Black Hat Conference, 2007.
- [11] DD tool, George Garner's site <http://users.erols.com/gmgarner/forensics/>, 14/1/2009.
- [12] DCFL tool, Nicholas Harbour, <http://dcfldd.sourceforge.net/>, 14/1/2009.
- [13] WinHex tool, X-Ways Software Technology AG, <http://www.x-ways.net/winhex/>, 14/1/2009.
- [14] FRHED tool, Raihan Kibria site, <http://frhed.sourceforge.net/>, 14/1/2009.
- [15] STRINGS, Mark Russinovich, <http://technet.microsoft.com/en-us/sysinternals/bb897439.aspx>, 14/1/2009.
- [16] TSK tools, Brian Carrier site, <http://www.sleuthkit.org/sleuthkit/>, 14/1/2009.
- [17] Autopsy tools, Brian Carrier site, <http://www.sleuthkit.org/autopsy/>, 14,1,2009.
- [18] NTFSINFO tool, Mark Russinovich, <http://technet.microsoft.com/en-au/sysinternals/bb897424.aspx>, 14/1/2009.
- [19] Roussev, V.; Chen, Y.; Bourg, T. & Richard, G., md5bloom: Forensic file system hashing revisited, Digital Investigation, Elsevier, 2006, 3, 82-90.
- [20] Chow, K.; Law, F.; Kwan, M. & Lai, K., *"The Rules of Time on NTFS File System"* Proceedings of the Second International Workshop on Systematic Approaches to Digital Forensic Engineering, 2007, 71-85.
- [21] Jones, K.; Bejtlich, R. & Rose, C., *"Real digital forensics: computer security and incident response"*, Addison-Wesley Professional, USA, 2008.
- [22] Carvey, H., *"Windows Forensic Analysis DVD Toolkit"*, Syngress Press, USA, 2007.
- [23] Naiqi, L.; Yujie, W. & QinKe, H., *"Computer Forensics Research and*



*Implementation Based on NTFS File System*", Computing, Communication, Control, and Management, 2008. CCCM'08. ISECS International Colloquium on, 2008, 1.

[24] Aquilina, J.; Casey, E.; Malin, C. & MyLibrary, "*Malware Forensics Investigating and Analyzing Malicious Code*", Syngress Publishing, USA, 2008.

[25] Huebner, E.; Bem, D. & Wee, C., "*Data hiding in the NTFS file system*", Digital Investigation, Elsevier, 2006, 3, 211-226.