EFFECTIVE DIGITAL FORENSIC ANALYSIS OF THE NTFS DISK IMAGE

Mamoun Alazab, Sitalakshmi Venkatraman, Paul Watters

University of Ballarat, Australia
{m.alazab, s.venkatraman, p.watters} @ballarat.edu.au

ABSTRACT

Forensic analysis of the Windows NT File System (NTFS) could provide useful information leading towards malware detection and presentation of digital evidence for the court of law. Since NTFS records every event of the system, forensic tools are required to process an enormous amount of information related to user / kernel environment, buffer overflows, trace conditions, network stack, etc. This has led to imperfect forensic tools that are practical for implementation and hence become popular, but are not comprehensive and effective. Many existing techniques have failed to identify malicious code in hidden data of the NTFS disk image. This research discusses the analysis technique we have adopted to successfully detect maliciousness in hidden data, by investigating the NTFS boot sector. We have conducted experimental studies with some of the existing popular forensics tools and have identified their limitations. Further, through our proposed three-stage forensic analysis process, our experimental investigation attempts to unearth the vulnerabilities of NTFS disk image and the weaknesses of the current forensic techniques.

Keywords: 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]. While forensic investigation 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 have taken 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 [2, 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 abovementioned 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 system 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, malware writers 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 FORENSIC ANALYSIS PROCESS

In this section, we describe the forensic analysis process we had adopted 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.

For the experimental investigation of the effectiveness of the above tools, we created test data 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 pilot empirical study, we focused on the boot sector of the NTFS disk image. We adopted the following three stages to perform digital forensic analysis in a comprehensive manner:

Stage 1: Hard disk data acquisition,
Stage 2: Evidence searching and
Stage 3: Analysis of NTFS file system.

2.1 Stage 1 - Hard Disk Data Acquisition

As the first stage in forensic analysis, 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 device. This utility was selected for investigation since it provides simple and flexible acquisition tools. The main advantage of using these tools is that we could extract the data in or between partitions to a separate file for more analysis. In addition, this utility 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 which values of data have been maliciously changed as well as to explore known data objects [19].

2.2 Stage 2 - Evidence searching

The next stage involved searching for evidences with respect to system tampering. 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 stage, we used the 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 Stage 3 - Analysis of NTFS File System

In the final stage of the experimental study, we analyzed the data obtained from the NTFS disk image that contributed towards meaningful conclusions of the forensic investigation. We adopted a collection of tools such as the Sleuth Kit (TSK), 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 FORENSIC INVESTIGATION STEPS

Many aspects must be taken into consideration when conducting a computer forensic investigation. There are different approaches adopted by an investigator while examining a crime scene. From the literature, we find five steps adopted, such as, Policy and procedure development, Evidence assessment, Evidence acquisition, Evidence examination, and documenting and reporting [26]. In our proposed approach for the digital forensic investigation, we adopted the following nine steps as shown in Figure 1:

Step 1: Policy and Procedure Development – In this step, suitable tools that are needed in the digital scene are determined as part of administrative considerations. All aspects of policy and procedure development are considered to determine the mission statement, skills and knowledge, funding, personal requirement, evidence handling and support from management.

Step 2: Hard Disk Acquisition – This step involves forensic duplication that could be achieved by obtaining NTFS image of the original disk using DD tool command. This step is for obtaining sector-by-sector mirror image of the disk and the output of the image file is created as Image.dd.

Step 3: Check the Data Integrity – This step ensures the integrity of data acquired through reporting of a hash function. We used MD5 tool to guarantee the integrity of the original media and the resulting image file.

Step 4: Extract MFT in the Boot Sector – In this step, the MFT is extracted from the boot sector. We analyzed the MFT using WinHex hexeditor tool and checked number of sectors allocated to the NTFS file system using NTFSINO.

Step 5: Extract $Boot file and Backup boot sector – In this step, the $Boot file is extracted to investigate hidden data. We analyzed the hidden data in the $Boot metadata file system using WinHex, TSK and Autopsy tools.

Step 6: Compare Boot sector and Backup – A

comparison of the original and backup boot sectors is performed in this step. We obtained another 2 Images from the original Image using DD tool. The output generated resulted in two image files named, backupbootsector.dd and bootsector.dd. We analyzed the two image file named backupbootsector.dd and bootsector.dd using WinHex hex-editor tool, TSK and Autopsy tools.

Step 7: Check the Data Integrity – In this step the integrity of data is verified again for test of congruence. We adopted the hashing technique using MD5 tool for the two created image files to check the data integrity.

Step 8: Extract the ASCII and UNICODE –This step involves extracting the ASCII and UNICODE characters from the binary files in the disk image. We used the Strings command tool and keyword search for matching text or hexadecimal values recorded on the disk. Through keyword search, we could find even files that contain specific words.

Step 9: Physical Presentation – In this final step, all the findings from the forensic investigation are documented. It involves presenting the digital evidence through documentation and reporting procedures.

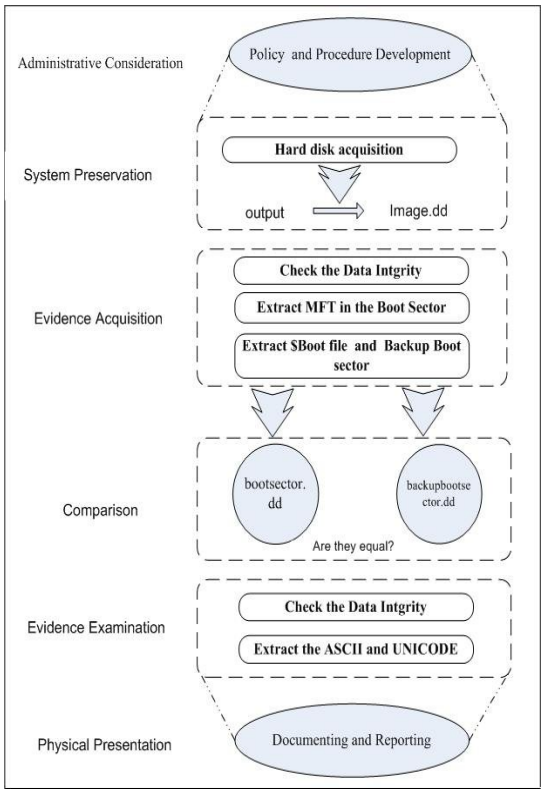


Figure 1: Forensic investigation steps

4 BOOT SECTOR ANALYSIS OF NTFS

4.1 NTFS Disk Image

As mentioned in the previous section, the first step to be adopted by a digital forensic investigator is to acquire a duplicate copy of the NTFS disk image before beginning the analysis. This is 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 from the disk image in order to extract the evidence that could be found on the electronic storage devices. By conducting investigations on the disk image, we could unearth any hidden intrusions since 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].

4.2 Master File Table

To investigate how intrusions result in data hiding, data deletion and other obfuscations, 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]. Hence, a good knowledge of the MFT layout structure also facilitates the disk recovery process. Each MFT entry has a fixed size which is 1 KB (at byte offset 64 in the boot sector one could identify the MFT record size). We provide the MFT layout and represent the plan of the NTFS file system using Figure 2. The main purpose of NTFS is to facilitate reading and writing of the file attributes and 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.

The key feature to note is that MFT entry within the MFT contains attributes that can have any format and any size. Further, as shown in Figure 2, 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.

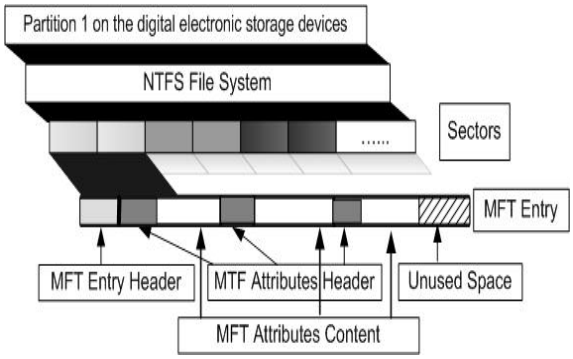


Figure 2: MFT layout structure

4.3 Boot Sector Analysis and Results

We performed boot sector analysis by investigating metadata files that are used to describe the file system. We followed the steps described in previous section (Figure 1) by first creating a NTFS disk image of the test computer using the dd utility for investigating 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. Such data structure examination enables us to view the MFT information, allocation size, volume size and metadata files. We extracted useful 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.

Table 1: NTFS Information Details.

Volume Size

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

From the information gained above, we followed the steps in Figure 1 to analyze the boot sector image. As shown in Figure 3, we performed an analysis of the data structure of this boot sector and the results of the investigation conducted using existing forensic tools is summarized in Table 2. From these results, we could conclude that the existing forensic tools do not check possible infections that could take place in certain hidden data of the boot sector. Hence, we describe the hidden data analysis technique that we had adopted in the next section.

5 HIDDEN DATA ANALYSIS AND RESULTS

The recent cyber crime trends are to use different obfuscated techniques such as disguising file names, hiding attributes and deleting files to intrude the computer 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 have analyzed the hidden data in the $Boot file structure. 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].

In order to investigate the NTFS file system, one requires to possess substantial knowledge and experience to analyze the data structure and the hidden data [24]. The $Boot metadata file structure 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 structure has four attributes, namely, $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 attributes values for the test sample are shown in Table 2 as an illustration. To achieve this, we used the following TSK command tools:

```
Istat –f ntfs c:\image.dd 7
```

From our investigations of the resulting attribute values, we find that, the $Boot data structure 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].

Table 2: Results of $Boot Analysis

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

Analysis of the $Boot data structure 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 in the $Boot data structure. 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:

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 data structure could not be detected directly by the existing tools used in this study and manual inspections were required alongside these forensic tools. Hence, through the analysis conducted with various existing utilities and tools, we arrived at the following results:

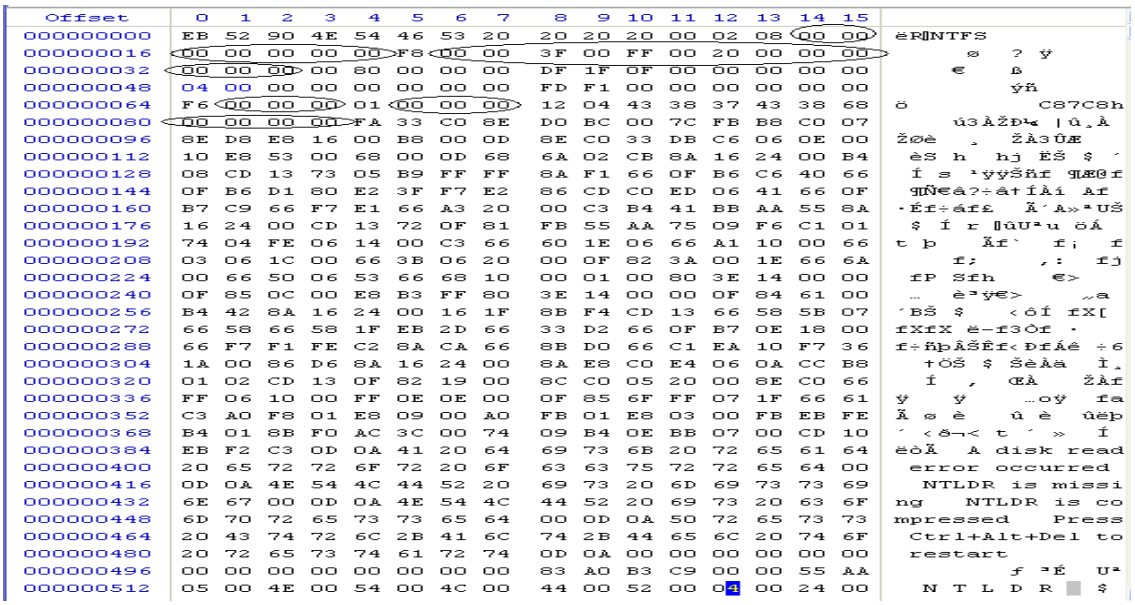
i) Since NTFS stores all events that take place on a computer system, there is a huge amount of data analysis required while scanning the entire NTFS disk image for forensic purposes. In this empirical study, by merely focusing on the hidden data of the $Boot file, we have shown that a variety of tools and utilities had to be adopted along with manual inspections. Hence, it takes an enormous amount of time to analyze the data derived with such tools.

ii) The existing forensic tools are not comprehensive and effective in identifying the recent computer threats. Not all computer infections are detected by forensic tools, especially intrusions that are in the form of hidden data in the $Boot file go unchecked.

iii) It was mandatory to perform manual investigations alongside the existing tools. By adopting a manual introspection of the $Boot file using the three-stage approach of i) hard disk acquisition, ii) evidence searching and iii) analysis of the NTFS file system, we could successfully identify hidden data in the $Boot file.

iv) Intelligent search techniques could be adopted 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.

v) One of the main reasons for having varying tools is that Microsoft has different versions of the NTFS file system to be catered for. While Windows XP and Windows Server 2003 use the same NTFS 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. Most of the existing tools do not work with all the different versions of NTFS file system, and hence a comprehensive tool is warranted to cope with these changes.



Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000000000	EB	52	90	4E	54	46	53	20	20	20	20	00	02	08	00	00
0000000016	00	00	00	00	00	F8	00	00	3F	00	FF	00	20	00	00	00
0000000032	00	00	00	00	80	00	00	00	DF	1F	0F	00	00	00	00	00
0000000048	04	00	00	00	00	00	00	00	FD	F1	00	00	00	00	00	00
0000000064	F6	00	00	00	01	00	00	00	12	04	43	38	37	43	38	68
0000000080	00	00	00	00	FA	33	C0	8E	D0	BC	00	7C	FB	B8	C0	07
0000000096	8E	D8	E8	16	00	B8	00	0D	8E	C0	33	DB	C6	06	0E	00
0000000112	10	E8	53	00	68	00	0D	68	6A	02	CB	8A	16	24	00	B4
0000000128	08	CD	13	73	05	B9	FF	FF	8A	F1	66	0F	B6	C6	40	66
0000000144	0F	B6	D1	80	E2	3F	F7	E2	86	CD	C0	ED	06	41	66	0F
0000000160	B7	C9	66	F7	E1	66	A3	20	00	C3	B4	41	BB	AA	55	8A
0000000176	16	24	00	CD	13	72	0F	81	FB	55	AA	75	09	F6	C1	01
0000000192	74	04	FE	06	14	00	C3	66	60	1E	06	66	A1	10	00	66
0000000208	03	06	1C	00	66	3B	06	20	00	0F	82	3A	00	1E	66	6A
0000000224	00	66	50	06	53	66	68	10	00	01	00	80	3E	14	00	00
0000000240	0F	85	0C	00	E8	B3	FF	80	3E	14	00	00	0F	84	61	00
0000000256	B4	42	8A	16	24	00	16	1F	8B	F4	CD	13	66	58	5B	07
0000000272	66	58	66	58	1F	EB	2D	66	33	D2	66	0F	B7	0E	18	00
0000000288	66	F7	F1	FE	C2	8A	CA	66	8B	D0	66	C1	EA	10	F7	36
0000000304	1A	00	86	D6	8A	16	24	00	8A	E8	C0	E4	06	0A	CC	B8
0000000320	01	02	CD	13	0F	82	19	00	8C	C0	05	20	00	8E	C0	66
0000000336	FF	06	10	00	FF	0E	0E	00	0F	85	6F	FF	07	1F	66	61
0000000352	C3	A0	F8	01	E8	09	00	A0	FB	01	E8	03	00	FB	EB	FE
0000000368	B4	01	8B	F0	AC	3C	00	74	09	B4	0E	BB	07	00	CD	10
0000000384	EB	F2	C3	0D	0A	41	20	64	69	73	6B	20	72	65	61	64
0000000400	20	65	72	72	6F	72	20	6F	63	63	75	72	72	65	64	00
0000000416	0D	0A	4E	54	4C	44	52	20	69	73	20	6D	69	73	73	69
0000000432	6E	67	00	0D	0A	4E	54	4C	44	52	20	69	73	20	63	6F
0000000448	6D	70	72	65	73	73	65	64	00	0D	0A	50	72	65	73	73
0000000464	20	43	74	72	6C	2B	41	6C	74	2B	44	65	6C	20	74	6F
0000000480	20	72	65	73	74	61	72	74	0D	0A	00	00	00	00	00	00
0000000496	00	00	00	00	00	00	00	00	83	A0	B3	C9	00	00	55	AA
0000000512	05	00	4E	00	54	00	4C	00	44	00	52	00	04	00	24	00

Figure 3: Analysis of the test boot Sector

Table 2: Results from the analysis of the test boot sector.

Byte Range	*Size*	*Description*	*Value*	*Action / Result*
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 – Possible Infection
16 -- 20	5	Unused	0	Unused – Possible Infection
21 -- 21	1	Media descriptor	0	
22 -- 23	2	Unused	0	Unused – Possible Infection
24 -- 25	2	Sectors per track	63	No Check – Possible Infection
26 -- 27	2	Number of heads	255	No Check – Possible Infection
28 -- 31	4	Unused	32	No Check – Possible Infection
32 -- 35	4	Unused	0	Unused – Possible Infection
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^{10}=1024	
65 -- 67	3	Unused	0	Unused – Possible Infection
68 -- 68	1	Size of index record	01h	
69 -- 71	3	Unused	0	Unused – Possible Infection
72 -- 79	8	Serial number	C87C8h	
80 -- 83	4	Unused	0	Unused – Possible Infection
84 -- 509	426	Boot code	~	
510 --511	2	Boot signature	0xAA55	

6 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

Recent methods adopted by computer intruders, attackers and malwares are to target hidden and deleted data so that they could evade from virus scanners and become even difficult to be identified using existing digital forensic tools. This paper has attempted to explore the difficulties involved in digital forensics, especially in conducting NTFS disk image analysis and to propose an effective digital forensic analysis.

In this empirical study, we have found that the boot sector of the NTFS file system could be used as a vehicle to hide data by computer attackers as there is a potential weakness. We have emphasized the knowledge and importance of file systems for digital forensics, as several techniques to hide data such as slack space and hidden attributes are being recently adopted by attackers. This is an important NTFS file system weakness to be addressed and research in this domain area could lead to effective solution for the open problem of detecting new malicious codes that make use of such an obfuscated mode of attack. We have shown that the existing forensic software tools are not competent enough to comprehensively detect all hidden data in boot sectors.

As a first step to address this problem, we have proposed a three-stage forensic analysis process consisting of nine steps to facilitate the experimental study. We have reported the results gathered by following these proposed steps. By adopting effective search techniques, we were successful in identifying some unknown malicious hidden data in the $Boot file that were undetected by current forensic tools.

In this pilot study we had adopted a few forensic techniques and effective manual inspections of the NTFS file image. Our future research directions 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 and extrapolate malware signatures effectively as well as intelligently for any existing and even new malware that use hidden and obfuscated modes of attack. We would automate the knowledge of how to extract data from hidden data structures and how to reclaim deleted data and we believe this would extensively benefit the digital evidence collection and recovery process.

7 REFERENCES

[1] M. Reith, C. Carr, & G. Gunsch: An examination of digital forensic models, International Journal of Digital Evidence, 1, pp. 1-12 (2002).

[2] M. Alazab, S. Venkatraman & P. Watters: Digital forensic techniques for static analysis of NTFS images, Proceedings of ICIT2009, Fourth International Conference on Information Technology, IEEE Xplore (2009).

[3] B. Carrier: File system forensic analysis, Addison-Wesley Professional, USA, (2008).

[4] S. Ardisson: Producing a Forensic Image of Your Client's Hard Drive? What You Need to Know, Qubit, 1, pp. 1-2 (2007).

[5] M. Andrew: Defining a Process Model for Forensic Analysis of Digital Devices and Storage Media, Proceedings of SADFE2007, Second International Workshop on Systematic Approaches to Digital Forensic Engineering, pp. 16-30 (2007).

[6] E Investigation: Electronic Crime Scene Investigation: A Guide for First Responders, US Department of Justice, NCJ, (2001).

[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] D. Purcell & S. Lang: Forensic Artifacts of Microsoft Windows Vista System, Lecture Notes in Computer Science, Springer, 5075, pp. 304-319 (2008).

[10] T. Newsham, C. Palmer, A; Stamos & J. Burns: Breaking forensics software: Weaknesses in critical evidence collection, Proceedings of the 2007 Black Hat Conference, (2007).

[11] DD tool, George Garner's site, Retrieved January, 2009 from http://users.erols.com/gmgarner/forensics/.

[12] DCFL tool, Nicholas Harbour, http://dcfldd.sourceforge.net/, accessed on 14/1/2009.

[13] WinHex tool, X-Ways Software Technology AG, Retrieved January, 2009 from http://www.x-ways.net/winhex/.

[14] FRHED tool, Raihan Kibria site, http://frhed.sourceforge.net/, 14/1/2009.

[15] STRINGS, Mark Russinovich, Retrieved January, 2009 from http://technet.microsoft.com/en-us/sysinternals/bb897439.aspx.

[16] TSK tools, Brian Carrier site, http://www.sleuthkit.org/sleuthkit/, 14/1/2009.

[17] Autopsy tools, Brian Carrier site, Retrieved January, 2009 from http://www.sleuthkit.org/autopsy/.

[18] NTFSINFO tool, Mark Russinovich, Retrieved January, 2009 from http://technet.microsoft.com/en-au/sysinternals/bb897424.aspx.

[19] V. Roussev, Y.Chen, T. Bourg & G. Richard: Forensic file system hashing revisited, Digital Investigation, Elsevier, 3, pp. 82-90 (2006).

[20] K. Chow, F. Law, M. Kwan & K. Lai: The

Rules of Time on NTFS File System, Proceedings of the Second International Workshop on Systematic Approaches to Digital Forensic Engineering, pp. 71-85(2007).

[21] K.; Jones, R. Bejtlich & C. Rose: Real digital forensics: computer security and incident response, Addison-Wesley Professional, USA, (2008).

[22] H. Carvey: Windows Forensic Analysis DVD Toolkit, Syngress Press, USA, (2007).

[23] L. Naiqi, W. Yujie & H. QinKe: Computer Forensics Research and Implementation Based on NTFS File System, CCCM'08, ISECS International Colloquium on Computing, Communication, Control, and Management, (2008).

[24] J. Aquilina, E. Casey & C. Malin: Malware Forensics Investigating and Analyzing Malicious Code, Syngress Publishing,USA, (2008).

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

[26] S. Hart, J. Ashcroft & D. Daniels: Forensic examination of digital evidence: a guide for law enforcement, National Institute of Justice NIJ-US, Washington DC, USA, Tech. Rep. NCJ, (2004).