Duplicate File Names-A Novel Steganographic Data Hiding Technique

Avinash Srinivasan[1] and Jie Wu[2]

[1] PA Center for Digital Forensics, Bloomsburg University, Bloomsburg PA 17815
[2] Center for Networked Computing, Temple University, Philadelphia, PA 19122

Abstract. Data hiding has been an integral part of human society from the very early days dating back to BC. It has played its role for both good and bad purposes. First instances of data hiding dates back to 440 B.C. and has been cited in several works as one of the first known and recorded use of steganography. Several complicated Steganographic techniques have been proposed in the past decade to deceive the detection mechanisms. Steganalysis has also been one of the corner stones of research in the recent past to thwart such attempts of the adversary to subterfuge detection. In this paper we present a novel, simple, and easy to implement data hiding technique for hiding files with duplicate names. The proposed file hiding technique *Duplicate File Names* uses an innocuous file as the cover medium exploiting its name and reputation as a good file. This vulnerability was first discovered on a Windows 98 machine with DOS 6.1. We have tested this vulnerability on several different file systems to confirm that the vulnerability exists across file systems and not specific to older Windows file systems. Finally, we have discussed using this method for legitimate data hiding as well as detecting when employed for illegitimate data hiding.

Keywords: Digital forensics, duplicate file name, file hiding, identity and data theft, steganography.

1 Introduction

Steganography has been a great challenge to the digital forensic community from the very beginning. However, one has to be unbiased and recognize the good side to Steganography like digital copyrighting and watermarking. Several techniques have been developed to detect information hiding accomplished by various Steganographic tools employing a limited number of Steganographic algorithms. However the adversary has been consistently successful in developing new techniques to achieve the same. In this paper we expose a potentially serous vulnerability which was first discovered on a Windows 98 machine with DOS 6.1.

The problem was identified while recovering deleted files on a FAT12 formatted floppy disk using DiskEdit. Norton Diskedit is a hexeditor for logical and physical disk drives on all Windows filesystems. It is an undocumented utility that comes along with the standard Norton Utilities package for Windows. The aforementioned vulnerability persists across the FAT file system family- FAT12, FAT16, and FAT32. The vulnerability can be formally stated as follows-

"A malicious file can be renamed, using a simple Hex editor tool, to bear the same name as that of a known good file on the media to evade simple detection schemes including visual examination".

This vulnerability is as powerful as it appears simple. An average computer user with the knowledge of the underlying file systems' structure and layout can easily traffic important files in and out of a room, building, or the country. To accomplish this, all he needs is a simple hex editor tool such as DiskEdit or HxD. Such files can range anywhere from a simple and not so critical data like coworkers' salary and bonus package to important business data like design and development blueprints and Intellectual Property. From a national security perspective, this could a document with classified information or a terrorist plot. None-the-less, these files can also be potentially dangerous viruses, malware, child porn image and video files.

The question that many of us want an answer to is *"Is this the most sophisticated data hiding technique"?* and the simple answer is "NO". However, the answer neither mitigates the risk nor eliminates the threat from such a simple data hiding technique.

In this paper we will discuss the structure of a simple FAT file system- FAT12. We then discuss the steps by which malicious files can be hidden in plain sight there by easily evading detection and visual inspection techniques employed. Simple and routine inspections are commonly deployed at the periphery of an organization like the security guard who can be directed to inspect the files carried out by employees working in certain restricted areas. The idea of this research work is to develop a simple and easy to use tool that can be used to detect and thwart such simple information theft that can potentially cause irreversible business losses and jeopardize the national security. We then discuss in detail the reverse engineering process of extracting such files.

The reminder of this paper is organized as follows. In Sec.2 we will review some of the important works in the field of steganography relevant to our work. We then present two application scenarios discussing in detail the presented data hiding technique in Sec.4. In Sec.3, we discuss the requirements for this method of data hiding to work, the categorization of storage devices, and related issues. We also present in this section the various areas on a disk where data can be hidden that usually would not hold user data otherwise. Later in Sec.5 we present some fundamental information of file systems with FAT12 as an example because of its simplicity. In Sec.6 we will discuss the details on how files can be hidden in plain sight by exploiting the vulnerability presented in this paper. We present a detailed detection and recovery process in Sec.7. Finally, in Sec.8, we conclude our work with directions for future research.

2 Related Work

Steganography can be used to insert plain or encrypted data in a cover file to avoid detection. The sole purpose of steganography is to conceal the very fact that something exists as opposed to cryptography which aims at rendering the contents uninterpretable.

MaDonald and Kuhn's StegFS [MK2000] hides encrypted data in the unused blocks of a Linux ext2 file system. Consequently, it makes the data look like a partition in which unused blocks have recently been overwritten. Furthermore, the proposed method of overwriting with random bytes mimics disk wiping tool.

Metasploit Anti-Forensics Project [MetaSplolt] seeks to develop tools and techniques for removing forensic evidence from computer systems. This project includes a number of tools, including Timestomp, Slacker, and SAM Juicer, many of which have been integrated in the Metasploit Framework. Metasploits Slacker hides data within the slack space of FAT or NTFS file system.

FragFS [TM2006] hides data within the NTFS Master File Table. It scans the MFT table for suitable MFT entries that have not been modified within the last year. It then calculates how much free space is available and divides it into 16byte chunks for hiding data.

RuneFS [G2005] stores files on blocks it assigns to the bad blocks inode which happens to be inode 1 in ext2. Forensic programs are not specifically designed to look at the bad blocks inode. Newer versions of RuneFS also encrypt files before hiding them making the problem a two fold problem.

3 Hiding Information on Storage Devices

In this section we will list the the requirements for successful data hiding and various areas on the storage volume where data can be hidden.

3.1 Requirements

For successful data hiding using the *Duplicate File Names* method, the following requirements have to be met.

1. The cover file should always have a lower starting cluster number compared to the file to be hidden. This is because the OS, when you access a file will always open the file with the lower starting cluster number. This is true and has been verified on all three FAT file systems.

2. The cover file and the hidden file have to be at the same hierarchical level in the directory structure. In light of this point, we have to ask the following question-

"Is it possible to have two files with the same name but different contents at the same hierarchical level- i.e., on the same drive, inside the same partition, and inside the same folder?

The answer to this question is "No". Trivially, there are two ways to attempting to create two files with the same name-

1. **Renaming an existing file**-Two files already exists inside a folder with different names. Try to rename one of the them to have the same name as the other by either right clicking or by opening the file and using the "save as" option under file menu. An error message will pop up.

2. **Creating a new file**- A file already exists. Try to create a new file and save it in the same folder as the existing one with the same name. This is same as opening an existing file and using "save as" option. Once again you will see an error message pop up.

In summary, one cannot save two file with the same name inside the same directory without overwriting. Once overwritten, the original file content will be lost forever although parts of it may be recovered from slack space. None-the-less, creating multiple files with duplicate names can be easily accomplished with the use of any freely available HeX editor. This requires some knowledge of the underlying file system and the associated OS. With the help of a HeX editor, the adversary can rename multiple files with a single name. Since, with a HexEditor, we work below the file system, the OS will not complain about the file already existing. Neither does the OS overwrite the contents of the original file. This way, there can be several files with the same name inside the same directory. This has been illustrated in Fig.1

There are several common areas on the disk that are either unused or reserved and can serve the purpose of hiding data without interfering with the intended primary operations of the storage partition. Below is the a list of areas common to both OS partition and non-OS partition.

- Slack Space- RAM and File Slack
- Boot Sector of non-bootable partition
- Unallocated Space
- Volume Slack

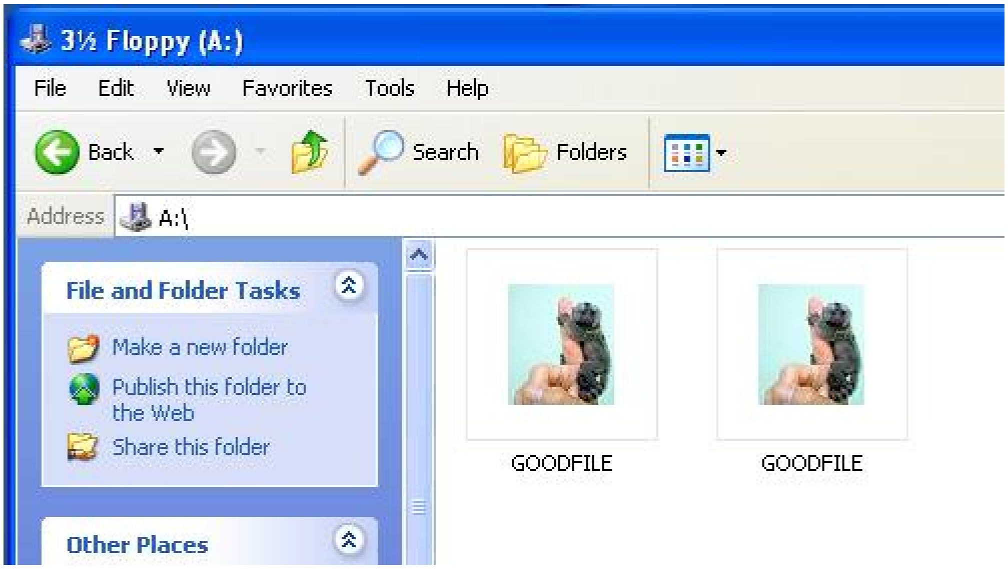

Fig. 1. Screenshot of a diskette storing two files with exactly the same name and extension at the at the same hierarchical level.

4 Application Scenario

In this section we present two application scenarios in different domain to emphasize the potential threat that *Duplicate File Name* data hiding technique can pose.

1. Scenario- 1: Child Pornography: A child pornographer can hide child porn images and/or videos using the same name as that of an innocuous looking image and/or video file respectively. The child pornographer can be doing this at his work place or at home. Since two files have the same name and clicking on either will always open the known good cover file.

2. Scenario- 2: Information Theft: A company employee easily steal confidential and proprietary data. The employee can steal the data very easily. He can save it on to his system with the name of a file he has privilege to access. Then copy both the original file and the file he is stealing with the Duplicate Name and walk out. Even if there is any sceurity screeningNo body would immediately wonder as to how two files with the same name can be copied to the same directory.

The following two situations have to be clearly differentiated. Duplicate files can have the same name and or different names. If they have the same name and are inside the same volume on a drive, then there will be only one root directory entry for all copies of the file with the same name. However, if duplicate copies have different names, then there will be a separate root directory entry for each copy with a different name irrespective of the hierarchy they reside at. In the former situation, as long as duplicate copies are inside the same volume, copies with the same name will have consistent data as long as they are duplicate. However, in the later scenario, modifying a file will not update the duplicate copies with different file names.

As already mentioned, in this paper we are trying to resolve the first scenario. There are commercially available tools to handle the second and third scenario. The fourth scenario is benign and poses no threat as such.

5 Hiding On Floppy Disk

For simplicity, we consider the example of hiding a malicious file on a floppy disk formatted with FAT12 file system. Additionally, to enable the reader in appreciating and understanding the file hiding technique presented in this paper, we will briefly discuss the layout of a floppy disk formatted with FAT12 file system and important data structures as shown in Fig.2.

The entire floppy disk can be divided into two main regions.

1. System Region
2. Data Region

System region consists of important system areas and data structures as follows-

1. Boot Sector
2. File Allocation Table
   (a) Primary FAT
   (b) Secondary FAT
3. Root Directory

For file recovery, the two most critical regions are the File Allocation Table and the Root Directory. The standard, default size of a root directory entry is 32 bytes and is consistent across the three FAT file systems-12, 16 and 32. In this paper we will restrict our discussions to FAT file systems for simplicity of conveying the idea. The 32 byte directory entry of a file stored on a FAT formatted volume has some critical information, which are listed below, that can be useful in detecting different files with duplicate names.

1. File Name
2. File Extension
3. File Attribute(s)
4. Create Date
5. Created Time
6. Last Accessed Date
7. Modified Date
8. Modified Time
9. Start Cluster Number
10. File Size

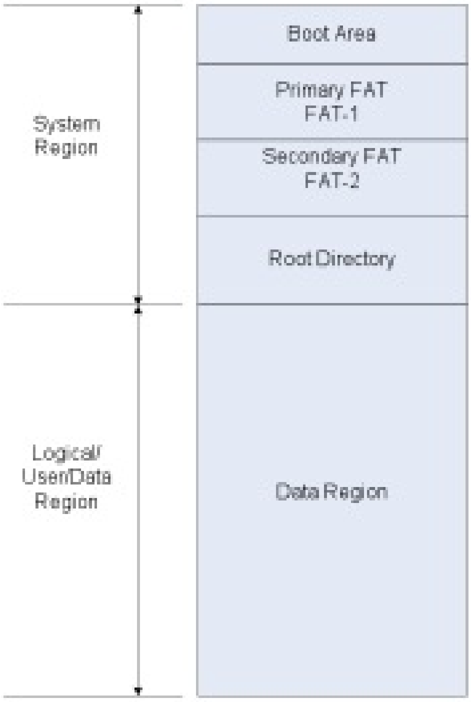

Fig. 2. The two main regions of a FAT12 formatted floppy disk and regions and data structures within the system region of a FAT12 formatted floppy disk.

In particular, for files that have different content but the same name and extension, the start cluster numbers have to be unique. The file size, in almost all cases should be different as well, however it cannot serve as an evidence to trigger suspicion nor serve as a confirmatory litmus test.

The same vulnerability can be seen from another perspective of having positive applications including hiding password files in plain sight. Such file can be accessed and opened on the fly by the methods presented later in this paper.

6 The Process of Hiding

In this section, we will discuss the method of hiding files with duplicate name using HeX Editor tool. The file uses size as the key requirement for choosing a cover file. Extension is not a key concern when choosing cover file since extension can be easily modified for the malicious file to match that of the cover file.

Without of loss of generality, we will use “Good File” to refer to the cover file being used whose name will not cause any suspicion or raise flags and “Bad File” to refer to the file being hidden which can be proprietary information of a corporate, child pornography image or video, etc.

The tool scans the entire root directory and returns the top five files whose size match the given file in size and attributes very closely. Then the user can choose a file whose name and extension will be used as the cover for hiding the malicious file. Once the user makes his choice, the rest is very simple.

1. The user initially save the file to be hidden on the storage device.

2. The user then loads the storage device into a hex editor and opens it.

3. User locates the entry in the root directory for the file to be hidden.

4. User over writes the name and extension of the file to be hidden with name and extension of the cover file.

5. User saves the changes made to the storage device.

6. Now, when the storage device is opened on any system, you can see two files with the exact name and extension at the same hierarchical level.

7 The Process of Detection and Recovery

Detecting files with duplicate names but different content can be performed in two different ways. Both these methods are described in detail below. Once two or more files are detected to have the same name but different content using the method below, then they have to be recovered with out loosing data for their potential evidentiary value.

7.1 Renaming Method of Detection

1. Open the disk in a hex editor tool.

2. Scan the root directory entries on the entire disk including subdirectories for duplicate file names including the extension. If there is more than one file with the same name and extension, then cross check their start cluster number and logical file size.

3. Two files with the same name and extension, if they are exactly the same in content, should have the exact same start cluster number and logical size.

4. If the result of this test confirms the files under scrutiny have the same start cluster number, then it can be ignored since it represents duplicate files.

5. If the result of this test confirms that the file with duplicate names have different start cluster numbers, then they are clearly different.

6. The logical size cannot be used as a confirmatory test since two files with same name but different contents can have the same size.

7. Both these files can be now retrieved to a different location such that original content is not altered, rename them as DIRTY-1.EXT and DIRTY-2.EXT. Now open both files. Having named them differently, the malicious file will not be protected any longer since accessing it now will reveal the actual content.

8 Conclusion and Future Work

In this paper, we have exposed a subtle yet important vulnerability in file systems, specifically FAT, that can be exploited to hide files in plain sight and evade detection. We have also proposed simple solutions to overcome such data hiding techniques and detect hidden files. We will continue to investigate along these lines to uncover any such data hiding techniques that have been either unknown or have been dismissed as too trivial. We have shown strong reasons through example application scenarios where such simple techniques can have a big payoff for the adversary with minimum risk. In the second phase of this project we will be developing a tool that can be used to hide information in plain sight exploiting the same vulnerability. The tool will be primarily targeted for education and training purposes.

As part of our future work we will be investigating anti-forensics techniques-techniques that are specifically designed to hinder or thwart forensic detection of criminal activities involving digital equipment and data. Also on our agenda of future research is Denial-of-Service attacks exploiting file system knowledge.

References

[HBW2006] Huebnera E., Bema D., Wee C K. :"Data hiding in the NTFS file system". In *Digital Investigation*, Vol. 3, Issue 4, December 2006, Pages 211-226.

[P1998] Petitcolas F. A. P.: "Attacks on Copyright Marking Systems". In *Information Hiding: Second International Workshop*, D. Aucsmith, Editor, Lecture Notes in Computer Science 1525, Springer-Verlag, Portland, OR (April 1517, 1998), pp. 219 - 239.

[AB1992] Abramson N. and Bender W.: "Context-Sensitive Multimedia". In *Proceedings of the International Society for Optical Engineering*, (SPIE) 1785, Washington, DC (September 1011,1992), pp. 12232.

[BGM1995] Bender W., Gruhl D. and Morimoto N.: "Datahiding Techniques". In *Proceedings of SPIE 2420*, 1995.

[BGML1996] Bender W., Gruhl D., Morimoto N. and A. Lu.: "Techniques for Data hiding. In *IBM System Journal*, Vol. 35, 3 & 4, 1996.

[BL2006] Buskrik and Liu:."Digital Evidence: Challenging the Presumption of Reliability. In *Journal of Digital Forensic Practice*, 1:1926, 2006. DOI 10.1080/15567280500541421.

[GM2005] Garfinkel and Malan:. "One Big File is Not Enough: A Critical Evaluation of the Dominant Free-Space Sanitization Technique". In *The 6th Workshop on Privacy Enhancing Technologies*, Robinson College, Cambridge, United Kingdom, June 28 - June 30.

[LB2006] Li u and Brown:. "Bleeding-Edge Anti-Forensics. In *Infosec World Conference & Expo, MIS Training Institute.*

[MK2000] McDonald, A. and Kuhn, M:. "StegFS: A Steganographic File System for Linux. In A. Pfitzmann (Ed.), IH99, LNCS 1768, pp. 463-477.

[TM2006] Thompson, I., and Monroe, M:. "FragFS: An Advanced Data Hiding Technique. In BlackHat Federal, Wang (2004).

[MetaSplolt] Metasploit Anti Forensics Project.
Online-http://www.metasploit.com/research/projects/antiforensics/

[G2005] Grugq : The Art of Defiling, Black Hat 2005.
Online- http://www.blackhat.com/presentations/bh-usa- 05/bh-us-05-grugq.pdf.