Forensic and Anti-Forensic on modern Computer Systems

Alexander Krenhuber[1] and Andreas Niederschick[1]

Johannes Kepler Universität Linz,
Institut für Informationsverarbeitung und Mikroprozessortechnik
alex.krenhuber@liwest.at, andi_niederschick@gmx.at

Abstract. The intention of this paper is to show, how data can be hidden on modern computer systems.
There exist many possible places, starting at the hardware itself and ending up using different places in the file system used.

1 Introduction

This paper shows which possibilities exist to hide data on modern computer systems. First field to be observed is hardware and how manufacturers design certain hidden places. Afterwards operating system properties concerning data management will be discussed. Next step is the analysis of file systems. Last topic is how to securely delete data.

2 Hardware

Storing data on hard disk, independent from running operating systems, seems to be a nice spot to hide data. Since hard disk manufacturers try to keep these options confidental, it is especially important for forensic analysts to know about the existence of these features, how to detect them and how commonly available forensic software treats them. It is important to note that not every forensic tool supports these areas, and so a generated forensic image may not contain them.[1]

2.1 HPA

The Host Protected Area (HPA), also known as Hidden Protected Area, is defined as a reserved protected area on modern hard disks. HPA was introduced as an optional feature in the ATA-4 standard[2] in 1998, making all currently available hard disks support this feature. The protected area is used to store data and configuration files, which neither the user nor the operating system can display or change. However there are certain tools available, like hdat2[3], that are able to modify an existing HPA.

The original goal of HPA was to store recovery images on the hard drive, to put them back into the regular field of the hard disk if necessary. Especially

notebook manufacturers often used them to save important system tools and an initial state of the system to save the costs of included recovery DVDs.

The central command in HPA is SET MAX ADDRESS, which is used to tell the hard disk where the user available space ends and where the protected area starts. For hard disks larger than 137GB (268.435.454 sectors), that use 48bit logical block addressing[4] the appropriate command is SET MAX ADDRESS EXTEND. To determine the size of a hard disk, the operation system examines the ATA- commando IDENTIFIY-DEVICE. This commando will no longer return the actual size of the disk, but instead the address set by the SET MAX ADDRESS command.

However each hard drive that is HPA capable must also support the ATA command READ NATIVE MAX ADDRESS, and respectively READ NATIVE ADDRESS EXTEND, and so it is possible to detect that a HPA is being used by comparing the output from IDENTIFIY DEVICE and the READ NATIVE MAX ADDRESS commands. If the two commands show different sizes, it is highly likely that there exists a protected area on the hard disk. [5]

A simple rerun of the SET MAX ADDRESS command will restore the factory default size of the hard drive. Newer versions of the Linux kernel will display a detected HPA at boot time, or even reset it until the next reboot, to gain access to the whole disk.

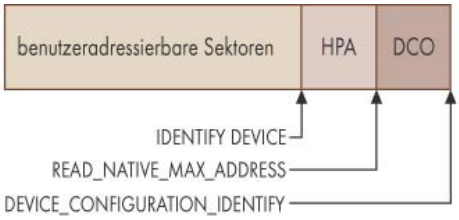
2.2 DCO

Device Configuration Overlay (DCO) is another available optional feature set to hide a data area from the user and reduce the available user space. It was introduced as an optional feature in the ATA-6 standard in 2002[2]. Device Configuration Overlay is much more powerful than HPA and also less well-known and thus an interesting opportunity to hide data.

The primary goal of DCO was to allow pc manufacturres to purchase differently sized hard disks and then make every drive have exactly the same number of sectors. It can also be used to enable or disable features of the hard disk, for example the SMART feature set.[1]

With the DEVICE CONFIGURATION SET command, the capacity of the disk can be reduced. Using this method, modifying the disk size is more powerful than doing it with HPA, because now even the earlier mentioned READ NATIVE MAX ADDRESS command returns the size defined by the DEVICE CONFIGURATION SET and not the original size of the hard disk. Now an access behind the set maximum size ends with an Identify Drive Command Address Not Found Error. If there is already a HPA before activating the DEVICE CONFIGURATION SET, it fails to prevent possible loss of data of the HPA.[1]

To use both ATA features (HPA and DCO), DCO must be defined first. Then in addition an HPA can be generated, whereas the maximum address of the HPA must always be less than the DCO configuration.



[6]

The only way to detect a DCO, is to use the command DEVICE CONFIGURATION IDENTIFY, which shows the true size of the hard disk. The detection of the hidden data is relatively easy by comparing the output of the commands DEVICE CONFIGURATION IDENTIFY and READ NATIVE MAX. To restore the original configuration, the command DEVICE CONFIGURATION RESTORE needs to be invoked, which restores the logical size of the hard disk to its original state. In existence of an additional HPA, this must be reset before the DCO, because a reverse execution leads to an error. During a forensic analysis of a hard disk, it is highly recommended to compare the printed size on the hard disk with the size returned from the DCO and HPA commands to identify possible hidden areas.

3 Operation system

3.1 Slack Space

Not every single bit on a hard disk is accessible due to the logical structure of data. These areas which are not accessible are called slack space. Several different reasons lead to slack space. For example, a file normally does not fit exactly into a sector on the platter. The unused space of the sector can not be addressed by the operating system system and is known as RAM slack. Not every file ends within the last sector of a cluster or block, which is the smallest unit an operating system has access to. So the sectors within a cluster which are unused are not ready to use. This is referred to as drive slack.[7]

Another form of slack is volume slack. A partition does not need to use all space designated to it. The unused space can not be accessed by the operating system and is therefore slack space.[8] In other words, "‘volume slack is the unused space between the end of the file system and the end of the partition where the file system resides."[9]

4 NTFS

The NTFS (New Technology File System), introduced in 1993 with Windows NT 3.1, became the standard file system for Microsoft’s operating systems. There are many ways of hiding data in the NTFS file system. In this section we try to

explain various methods to hide data and show how hidden data can be detected. Methods easy detectable or with storage capacities of only a few bytes are not part of this paper.

4.1 Faked bad clusters

A defect sector has a physical error and thus it is no longer usable. Although the hard disk and the operating system can handle it, any data in this sector is lost. Older hard disks did not have the possibility to detect defective sectors, and so this was done by the operating system.[9] NTFS identifies defect clusters in the metadata file $BadClus, which is stored in the Master File Table (MFT). Now in order to hide files, it is sufficient to mark some clusters as defect and use them to hide data. To accomplish this, the clusters are added to the $Bad attribute of the $BadClus metadata file. In addition the size of the $Bad attribute and the size of the MFT file record need to be modified. The size of the data that can be hidden with this method is unlimited.[9]

Modern hard disks manage the defective sectors by themselves and divide them into several categories.

- Primary defect list (PLIST). Here all sectors are listed that are already defect after manufacture. The PLIST is located inside a reserved area, and once created it should not be changed.[3]
- The grown defect list (GLIST) contains all sectors that get defect during operation. If a defect sector is detected it is remapped to a different part of the medium. In the usual case, a defect that has been reassigned no longer has an LBA, and thus can‘t be accessed again.[3]

Thus it is unlikely that the operating system detects bad sectors before the disk does. So if there are any clusters marked as bad in the $Badclus file, one should be suspicious and further analysis of the content in the bad marked sectors or a surface scan of the hard disk to verify the existence of bad sectors should be done.

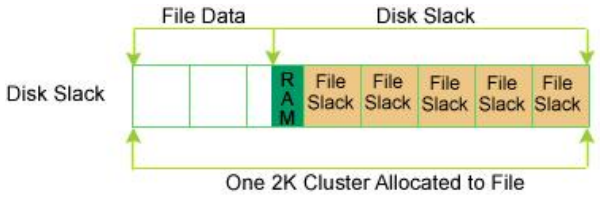
4.2 Additional clusters allocated to a file

This method uses additionally added clusters to a file, although they are not needed for size reasons. For example, a file needs to allocate 3 clusters to store its content, but there are 5 clusters allocated. There are now 2 clusters that can be used to store information and can‘t be overwritten by other files, since these sectors are already allocated to a file. Because there is no limit of additionally allocated clusters the available space for hidden information is only bound to the available disk space. But a quite serious disadvantage of this method is that if the original file grows in space, the hidden information will be overwritten and so no longer usable. To prevent this, the use of files that do not change is preferred. Program executables or operating system files are a good point to start.[9]

4.3 File slack

When a file is created, its actual size depends on the content to store and some additional metadata information. Since the file system, due to simplifying the administration of available sectors, combines sectors to a larger logical unit called cluster most of the time, a file does not fill the entire last cluster of its allocated space. Due to the general use of a cluster size of 4 sectors, every file generates a usable space from 0 to 3 sectors to hide information. This type of file slack is called drive slack, it spans from the first unused sector of the last allocated cluster to the end of this cluster. Since Microsoft Windows ignores the information stored in the drive slack, stored information won't be detected by the operating system itself.

Another type of file slack is the so-called RAM slack. RAM slack spans from the end of the file to the end of the sector. It depends on the operating system how ram slack is handled. For example, Linux and Microsoft Windows write a series of 0 into RAM slack.[8]



[8]

So RAM slack can also be used to store secret information, but since the available space is not that big and any 0Bit under Windows operating system would be suspicious to an investigator, it is not an ideal place. Since the hidden information gets overwritten when the original file grows in space, stable files are a preferred target.

4.4 Alternate data streams

The data content of a file is usually stored in the $Data attribute. By default, all data from a file is stored in a single $Data attribute, even if it is a large file and the $Data attribute must be split into many parts. A file may have more than one $Data attribute, as Alternate data stream uses them, and even a directory, although unnecessary, may have one or more.[9]

Whenever a file or folder has more than one $Data attribute, the additional attribute is a so-called Alternate Data Stream (ADS). The main purpose of this feature was to develop a compatibility with the former Apple file system HFS and their Resource Forks[10]. During a data transfer from a Mac to a Windows PC with NTFS file system in use, this additional information from Resource Forks is stored in ADS.[11] Although ADS has been available since Windows 2000 , few know about this feature and its function. For example each, with Internet Explorer 6 and above, downloaded file has an ADS containing the region from where it was downloaded.

```
G:\>dir
04.05.2008  11:05            40.319 kusss.ics
               1 Datei(en)         40.319 Bytes
               0 Verzeichnis(se),  4.088.881.152 Bytes frei
```

With the dir command or Windows Explorer, only the file itself is visible. Even the original file size does not change, no matter how much space the additional ADS need. Still there exist some free programs like ADS-Locator[12], or even from Microsoft itself, the so called Streams[13], that can search for and display if there exists any ADS.

```
G:\>streams -s g:
G:\kusss.ics:
   :Zone.Identifier:$DATA       26
```

In addition to the original file, a 26Byte Alternate Data Stream is added. Now that we know the exact name of the ADS we can extract it to a file or simply display its content.

```
G:\>more < kusss.ics:Zone.Identifier
[ZoneTransfer]
ZoneId=3
```

ZoneID 3 means that this file was downloaded from the Internet. A list auf possible values is posted on the Windows Power Shell Blog[14]. Even the Windows Indexing service uses ADS to add a thumbnail to a picture for previewing in Windows Explorer. Creating an ADS is relatively simple, and the only need are existing Windows programs.

```
type kusss.gif > g:\kusss.ics:kusss.gif
```

The above command generates an Alternate Data Stream in the kusss.ics file, containing an image. Now without knowing the specific name of the ADS, it can*t be discovered. The only telltale sign is that timestamp, when the file was the last time edited, changed to the time the ADS was created. As you can see through ADS it is very simple to effectively hide data from the users view. Even a comparison of the hashes (MD5, SHA1) of the file before and after the ADS adding reveals no difference.

Unfortunately to delete an Alternate Data Stream without deleting the original file is pretty difficult. Only, moving the file to a FAT formatted partition would destroy the ADS. However special NTFS access rights added to file will be lost too.

5 Linux

Many different file systems are out there, when using Linux as operating system. Therefore, in this paper we will describe some general ways of hiding data on Linux systems, regardless of the file system used.

5.1 Mounting on non-empty directories

Probably the easiest and a quite effective method of hiding data is mounting a file system on a non-empty directory. After successful mounting, no further access to the underlying, now hidden, data is possible. To make the data visible again, it is enough to unmount the file system. The resulting maximum storage capacity for hidden file depends only on the remaining space. To recognize this method of hiding data, it requires the continuous control of which file systems are currently mounted on the system. This can be achieved by the use of tools like mount or df. Likewise frequent mount and unmount operations should appear suspicious. To prevent the abuse from users, user mountable file systems should be avoided.[15]

5.2 Slack Space

Similar to NTFS, the existing storage space is divided into logical blocks (like NTFS clusters). A data structure may not use an entire logical disk block, this leads to some available free space for hiding data.[8] There exist some free tools like bmap or slacker that access this free space and hide data in it. However, to use this tools administrator privileges are needed. The drawbacks are similar to those using NTFS and so stable files are preferred.

5.3 Extended file attributes

Similar to the Alternate Data Streams in NTFS, Linux supports a feature called Extended Attributes (xattr) in various file systems (extX, ReiserFS, JFS, XFS). To use this function, the libattr feature must be activated in the kernel configuration. Each file can now have a list of extend attributes. Each of these attributes has a distinguished name and related data. The name must be prefixed with a special namespace identifier (user, root and security namespace are available) and a following point. Extended Attributes are easy to create, but for a person who is aware of this feature they are easy to detect.

5.4 Removal of open files

Deleting a file, even if it is opened by a program, will immediately remove the file name from the directory containing the file. The data itself won‘t be deleted until the last program (last opened file descriptor) that has access to the file, was closed. In the past this was a common way to hide data, since for this method of hiding data no special rights are needed. However, to access the deleted content after the last file descriptor was closed, requires direct access to the file system. To detect deleted files that are still opened, a Linux tool called lsof (List open files) can be used. The available space for hiding data is only limited to the free space on the hard disk.[16]

6 Secure Deletion

Nowadays computers hold a large amount of data. Parts of this data are not stored intentionally by the user. Access to deleted files and files created by the operating system or applications lead to unwanted information which can be abused. The following pages show which kinds of unwanted data exist and how to delete it safely.

6.1 Sources of unintentionally created data

Microsoft Windows uses a swap or page file to store data which does not fit into the main memory. This file can include every piece of information that can exist on the computer, e.g. databases or internet settings. As page files can get quite big (up to 200 million bytes) and there happen to be more than one page file on the computer, a large amount of data can be found on the hard disk without the user noticing it. [17]

Applications often create temporary files to improve their performance. These files are normally deleted after the program is closed, but can be accessed because they remain on the hard disk until other data overwrites it.[17]

To describe the characteristics of data files contain metadata. This metadata can hold crucial information like names of users, organizations, computers or networks. Hackers might use this information, so metadata needs to be safely removed as well. [17]

Deleted files normally remain on the hard drive in their original form. The memory addresses are just made available for overwriting by the file system. So the operating system and therefore normal users cannot access these files anymore, but with adequate tools they can be restored. To safely remove data its location must be overwritten several times. Therefore removing data completely takes much time.[7]

6.2 Methods of secure deletion

As mentioned above, normally deleted files are still accessible with special tools because the file content remains on the hard disk. In order to truly remove data, its memory has to be overwritten. It can be simply overwritten once with ones, zeros or random data, called Single Pass. But there are some sophisticated methods to ensure secure deletion as well.[17]
The Department of Defense (DoD) of the USA invented a method which first overwrites everything with zeros, then with ones and afterwards with a pseudo-random pattern. This process can take place as many times as you like, always repeating these three steps.[17]

The RCMP TSSIT OPS-II method alternately overwrites the addressable memory locations first with ones and then with zeros. After repeating this for three or more times random data is written to the disk. Another method is to overwrite memory with a specific data pattern and then with that patterns complement. Afterwards a random pattern is written. The overwriting process has to cover allocation tables, directories, block maps and file space.[17]

The Guttman method overwrites data 35 times. It uses bit patterns which seem to efficiently cover old data encoded with different coding methods like FM or RLL (will be addressed later on).[17]

It has to be said that these methods only reduce the likelihood of data recovery, but will not fully sanitize harddrives. But: the better the overwriting method, the more time will be needed to recover data. So an attacker or forensic worker will have to relate their spent effort to the importance of recovering files.[18] Only the destruction of the harddrive through disintegration, incineration or pulverization will ensure sanitation of a disk.[17]

6.3 Methods of recovering deleted Data

On magnetic media ones and zeros are represented by different small magnetic fields. Ones are coded with a higher magnetic force then zeros. If you overwrite a zero with a one, the magnetic force should be equal to the level that indicates a one. But the real effect will be about 0.95 of the level of a one. So with a proper technique old data can be retrieved keeping in mind this inaccuracy.[18]

Another source of inaccuracy is the drive head. Its exact position varies within small but perceivable borders. With drive heads which work more precisely these divergences and therefore the previous bits can be identified.[17]

Two methods to retrieve the bit information from the hard drive are magnetic force microscopy (MFM) and scanning tunneling microscopy (STM). MFM uses a sharp magnetic tip which interacts with the magnetic fields of the platter. The tip is moved over the tracks of the platter and measures the magnetic force of every bit. STM is a new variant of MFM. The tip in this case has a small electrical potential to tunnel electrons form the surface of the platter to the probe on which the tip is mounted. The variation in current is transformed into a image of the disk.[18]

With these methods old layers of data can often be recovered. But how efficient they work and how often the old data has to be overwritten depends on encoding techniques. Stored data shall be coded so that minimal overhead is needed and synchronisation can be achieved.[18]

Frequency modulation (FM) and modified frequency modulation (MFM) are

encoding methods which are quite straight forward. As drive heads do not measure absolute magnetic strength FM and MFM code reversals in the magnetic fields of two adjacent bits. FM represents a one as two transitions and a zero as one transition. After a bit a transition is added for synchronisation. MFM enhances this technique by reducing the number of addes clock transitions. Only if a zero is followed by another zero synchronisation has to be considered.[18] These two methods overwrite old layers of bits with new patterns, so the problem mentioned above that a zero overwritten with one will be about 0.95 occurs. Therefore some old layers will be possible to be recovered.

Another way of encoding data are run-length-limited methods (RLL). Instead of coding every single bit RLL encodes patterns of bits. For example, 11 is encoded with TNNN, where T is a transition and N is no transition. These patterns are designed to bring along synchronisation. RLL is a family of encoding methods, where the difference shows up in the minimum and maximum number of bits without transitions between 2 transitions.[18]

A newer version of RLL is partial-response maximum-likelihood (PRML). In contrast to other methods which read the peaks of analog signals taken from the platter through the drive head, PRML samples the signal and chooses the most likely pattern matching the retrieved samples.[18] Nowadays in use is extended PRML, which is a method similar to PRML but with more efficient algorithms.

The usage of patterns offers to pack data with a higher density on the platter. Therefore it is harder to recover the data due to the need of very precise hardware. Overwriting old data with a random pattern should be enough to remove files safely due to the lack of good enough signal processing.[18]

7 Conclusion

As this paper showed there are many different spots where data can be hidden on a computer. These places cover the whole range from hardware up to logical file systems. Allthough this data cannot be addressed through normal computer usage, there are many ways and programs to hide and seek it. Therefore computer forensics is important to avoid or detect abuse of hidden data areas.

References

1. Mayank R. Gupta, Michael D. Hoeschele, M.K.R.: Hidden disk areas: Hpa and dco. International Journal of Digital Evidence **5 Issue 1** (2006)
2. http://www.t13.org/
3. Cabla, L.: hdat2 user‘s manual. http://www.hdat2.com/
4. http://www.48bitlba.com/
5. Vidström, A.: Computer forensics and the ata interface. Swedish Defence Research Agency Command and Control Systems (2005)
6. Tennert, O.: Post-mortem-analyse von filesystemen unter linux. iX (3 2006)
7. Dillon, S.: Hide and seek: Concealing and recovering hard disk data. (2006)
8. Hal Berghel, David Hoelzer, M.S.: Data hiding tactics for windows and unix file systems. Identity Theft and Financial Fraud Research and Operation Center (2006)
9. Ewa Huebner, Derek Bem, C.K.W.: Analysis of hidden data in ntfs file system. Digital Investigation The International Journal of Digital Forensics Incident Response **3 Issue 4** (2006) 211–226
10. Apple Computer, I.: The data fork and the resource fork. http://developer.apple.com/documentation/mac/MoreToolbox/MoreToolbox-11.html
11. Berghel, H., Brajkovska, N.: Wading into alternate data streams. Commun. ACM **47**(4) (2004) 21–27
12. ADS-Locator. http://www.safer-networking.org/de/paragraphs/tools_ads.html
13. Russinovich, M. http://www.microsoft.com/technet/sysinternals/FileAndDisk/Streams.mspx (2007)
14. Windows-PowerShell-team: How does the remotesigned execution policy work? http://blogs.msdn.com/powershell/archive/2007/03/07/how-does-the-remotesigned-execution-policy-work.aspx
15. Knut Eckstein, M.J.: Data hiding in journaling file systems. In: Digital Forensic Research Workshop. (2005)
16. Chuvakin, A.: Linux data hiding and recovery. (2002)
17. Mallery, J.R.: Secure file deletion: Fact or fiction? (2006)
18. Gutmann, P.: Secure deletion of data from magnetic and solid-state memory. (1996)