Reverse-Wipe Data on a Storage Medium

G. Fragkos

Faculty of Advanced Technology, University of Glamorgan, Wales, UK
e-mail: gfragkos@glam.ac.uk

Abstract

This experimental study suggests an alternative method of data deletion which can be considered secure up to an acceptable level for most purposes while maintaining anti-forensics characteristics. It is a quick and "dirty" solution to remove data from a storage medium while achieving to confuse the investigator about the contents of the i.e. the hard disk. More specifically, working upon the file system of the storage medium it fills all the empty space leaving no slack space available. Consequently, this approach could provide an insight today, as in how it could be used as an anti-forensics method tomorrow.

Keywords

Computer Forensics, Anti-Forensics, Data Wiping, File Headers

1. Introduction

The capacity of the storage mediums is increasing rapidly. The cost is low and that makes them suitable for most purposes. Data remaining on storage mediums is a serious issue (Jones, 2006) especially when they are needed to be disposed. Several types of data wipe software are available for secure erase of the contents. However, some times the whole process costs time and money. In this experimental study it is proposed to use another approach slightly different from the know ones. Knowing how time consuming a three times data wipe can be, it is proposed to perform overwrite, based on the file system of the storage medium. In other words, in the same amount of time or faster sometimes a complete overwrite of the minimum cluster size set by the Operating System can be achieved. Considering within the equation the fact that instead of the typical overwrites, with a standard value like 0xA, a random value is used which is part of a random valid header or part of valid random file.

2. Working with the file system's structure

From forensics point of view there are various methods to wipe clean the remaining data on a storage medium. There are quick, slow and very slow methods according to the quality of the results you are after. The most advanced and secure wiping method is the use the US Department of Defence (DoD) data wiping process which suggests that a device needs to be overwritten seven times before the original data can be considered un recoverable (Gutmann, 1996). The data wiping process starts from the beginning of the storage device and continues all the way up to the last bit. Each individual byte is overwritten with zeros, then with some random value and again with zeros.

This process can be considered secure enough when it passes through the whole device about 3-4 times. All this process is very time consuming depending on the capacity of the device. Moreover, this approach will erase everything from the device, along with any file system present and of course requires advanced technical knowledge in order to be sure it is done correctly and none of any other disk/partitions that need to be kept unharmed, have been wiped clean accidentally.

Depending on the file system's type used on each one of these storage devices, there is a minimum allocation unit (considering a block of predefined blank space named as **cluster**) which will be used to store data. If the volume of the data needs to be stored is less than the capacity of that cluster, then the space left within this cluster is remaining unused (known as slack space). Obviously, if the volume of the data to be stored exceeds the capacity of a cluster, then the first cluster will be filled in completely and any data remaining will be store into the next one.

Just a few really large files can easily occupy (mark the storage medium as full) the whole available storage space on the device. In reality there is some more space available on the device but it is actually reserved for another use. Depending on the capacity of the storage medium there is also space allocated specifically for storing the filenames of each file stored. Modern Operating Systems support long filenames up to 255 characters long including any file extensions might need to be used. The reserved space for storing filenames on a medium is relative to the overall storage capacity of the device. Thus, using 255 character long filenames for each file will fill in your index table leaving no space for more files to be written on the device. Eventually, the empty space of your device will be marked as unavailable even though the data stored on it is far more less than the actual capacity of the device.

Overwriting in full both domains of a storage device (data domain and the index table domain) relies on the understanding the file system's design principles and how we can take advantage of it. Consequently, the number of files to be created along with the length of filenames to be used can be calculated in advance.

3. Understanding the process

In order to avoid occupying the device's capacity either with just large files by leaving empty space on the index table side, or either with long filenames that will leave most of the device's storage space unavailable, the appropriate number between files to be stored and their relative filename's length need to calculated. By dividing the device's capacity with the cluster size used by its file system the number of files that can be stored on this device can be calculated. The OS is responsible of performing all the required operations in order for a file to be written on a device. As mentioned in section II, the appropriate file size in order to force the operating system to leave no slack space available, is to create files which are exactly the same in capacity as the cluster size used on the storing device. If the calculated number of files to be created is a three digits number (i.e. 784), then the correct filename length to be used for these number of files must not exceed or be less than that length. Effectively, starting with a filename like 001, for the first file, and increasing it sequentially, the storage device will end up with 784 files named respectively.

This approach is going to leave no slack space available for each cluster, and the index table allocated for filenames will be totally used without affecting the actual number of files that the device can actually store. The number of files that can be stored in the root partition or in any of the subdirectories of a storage device, for example a hard disk that has NTFS file system, is 65535 files. Thus, if more files need to be stored, then a subdirectory or subdirectories need to be created by subtracting two files in order to create one directory.

4. Data Wiping and Anti-Forensics

The idea described in this paper is a ‘quick and dirty’ approach. It can maintain the file system structure along with any data that must not be deleted. It works with the Operating System thus it can be performed at run time without effecting or disturbing the work it is been done on the same system. It will only overwrite the blank and slack space remaining on the disk without altering/deleting any data that need to be kept. It does not require direct access to the hard disk allocation table, thus no administrative privileged required, neither installations. There is no need to buy any special commercial product that uses some kind of advanced algorithm in order to perform this basic task. The source code is very simple and can remain open source for further expansion the codes capabilities. In overall, it is faster than using an advance method of wiping a storage device.

The main advantage of this ‘reverse wiping’ method is that each file generated in order to be written, has a cluster size of data available to be manipulated. This block of available data, instead of being zeros like most of the applications do, can be altered to the most well known file headers. File headers that point to picture files, document files, any type of files that most computers uses these days but containing no actual data within them. Consequently, when an investigator will try analyse the contents of this particular device instead of finding zeros everywhere and being easy to spot any data remains, he/she can be get really frustrated when a search for picture files return a few thousand hits, that in reality contain no valid data. In this case it is easier to hide or avoid detection of any real data that have been left of the device accidentally.

The proof of concept application developed was a very simple one. The source code is just a few lines [Fig 1] which prove the simplicity of the idea. The application’s input requires knowing in advance the path to the intended drive to be wiped (overwritten), the cluster size of the file system used, the number of characters to use in order to generate the appropriate filename length and finally the value to be written throughout the cluster size data block.

```
#!/usr/bin/python
#G.Fragkos 2007
TARGET_LOCATION="I:\\" # Set Target Drive Letter
BYTES=1024*4           # Cluster Size in KB
FILENAME_LENGTH=6      # Filename Length
CHARACTER="0"          # Overwrite using this character

# Initialising Variables
enumList=['bytes', 'KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB']
COUNTA=1
loopcounter=0
try:
     while True:
          FILENAME=(FILENAME_LENGTH-len(str(COUNTA)))*CHARACTER+str(COUNTA)
          DATA=CHARACTER*BYTES
          PATH=TARGET_LOCATION + FILENAME
          print COUNTA
          f=open(PATH, 'w')
          f.write(DATA)
          COUNTA+=1
except IOError:
     print "!Disk Full"
     COUNTA-=1
     CSIZE=BYTES*COUNTA
     while CSIZE % 1024 == 0 or CSIZE > 1024:
          CSIZE = CSIZE / 1024
          loopcounter+=1
     print  "R-Wiped  Using:  "  +  str(COUNTA)  +  "  files,  "  +  str(CSIZE)  +  enumList[loopcounter]  +  "("  +
str(BYTES*COUNTA) + " bytes)"
```

Figure 1: Source Code

Instead of using a specific character repeatedly, which in this example is zero, a whole file header can be used in order to overwrite existing data. Trying to wipe the data contained on a 32 MB removable storage device using the DoD method in comparison with the proposed method, proved that the latter was scientifically faster and of course easier to be performed even from the most inexperienced user. On the other hand, the DoD method was undoubtedly the one which totally erased data to an unrecoverable state. Even though the proposed method is not a match to the DoD approach managed to overwrite most of the disk's space, leaving almost no data recoverable from previous uses. Mainly, the data that was able to be identified was the file system information. The use of random values was able to scramble the contents of the device and demonstrated how complicated can get, for a forensic analyst, to locate data that were originated from previous existing data. In a real life example it would be very difficult for a forensics analyst to investigate a million files (picture files) which contain randomly generated colorful pictures which actually mean nothing and within this vast amount of data try to identify any valid data remains. Extending this process to use a wide range of different file types becomes exponentially difficult to investigate.

5. Conclusions

Tests were performed in order to conclude if the proposed idea has any potential in the area of forensics and anti-forensics. At this stage of the experiment, the outcome can be considered successful. The results proved that the original data were erased through overwriting, up to an acceptable level, as no sensitive, profiling or back tracing data could be recovered afterwards. Furthermore, speed-wise, even though this method cannot be considered as a sophisticated one, was able to perform well and efficiently. The method, as it stands at the moment, needs to be expanded and further amendments need to be included to the functionality. Calculating automatically the appropriate variable values for different file systems is essential. Finally, as this idea is still in an experimental state, further tests need to take place which will include different types of storage devices, taking under consideration performance results when dealing with high capacity devices.

6. References

Gutmann P., 1996, Secure Deletion of Data from Magnetic and Solid-State Memory, Sixth USENIX Security Symposium Proceedings, San Jose, California, July 22-25

Jones A., Valli C., Sutherland I., Thomas P., 2006, An Analysis of Information Remaining on Disks offered for sale on the second hand market, Journal of Digital Forensics, Security and Law, Vol. 1(3).