# A Comparison of Storage Hard Disk Drive Used by the Windows Master File Table and Cluster Table to Highlight Inconsistencies Which May Indicate the Existence of Malware

***Abstract –*** It is known that some Advanced Persistent Threats store malware at the end of a Microsoft Windows partition. It is not known if the Operating System documents the allocation of this used disk space in the Master File Table and, or, Cluster Table. This paper presents a comparison of Hard Disk Drive storage listed as allocated in the Master File Table with that listed as allocated in the Cluster Table. Five machines were analysed, one in two different states, and discrepancies between the two tables were found on all machines and states: disk Space not flagged as being used by the Master File Table but flagged as being used by the Cluster Table was found being used within, and at the end of, the partition.

***Keywords*** – Microsoft Windows; Master File Table; $MFT; Cluster Table; $Bitmap; Malware; Advanced Persistent Threat (APT); Hard Disk Drive; Hiding in Plain Sight.

## 1. Introduction

This paper provides proof of concept for identification of areas used by Advanced Persistent Threats (APT) to store malware on a Hard Disk Drive (HDD) that are inconsistent with storage data documented by the Operating System (OS). The malware may be stored outside the boundaries of the Master File Table ($MFT) space allocation but within the boundaries of the Cluster Table ($Bitmap) space allocation.

The cyber security industry publishes the outcome of their analysis of APTs mainly in the form of white papers. During previous research it was noted from these papers that APTs may hide their malware in plain sight at the end of a partition. Some work was performed to demonstrate this (Bentley, 2021, pp. 253, 254). However, in the light of publication in the white papers it is possible that APTs may have moved the location to elsewhere on the HDD. Changes in APT Modus Operandi are known to occur (Horejsi, 2018, p. 17).

The Null Hypothesis is that $MFT and $Bitmap are consistent and that after the comparison there should be no non-zero positions in $Bitmap. The Alternative Hypothesis is that $MFT and $Bitmap data are inconsistent and there are non-zero positions in $Bitmap. Note that technique this will pick up inconsistencies through the HDD partition and not just at the end.

This paper is agnostic towards the origin and intent of APTs.

Access dates for some of the References is the date of access from the local repository and not internet access.

## 2. Background

It is known that malware authors hide malware at the end of the HDD (Berghel, 2008).

> "The initial Stage 1 driver is the only plainly visible code on the computer. All other stages are stored as encrypted data blobs, as a file, or within a non-traditional file storage area such as the registry, extended attributes, or raw sectors at the end of disk."
>
> (Symantec-Security-Response, 2015, p. 8)

Kaspersky have seen an APT which stored malware at the end of the last partition on disk (Kaspersky, 2014, p. 6) and on a USB (Universal Serial Bus) "The USBs are formatted to reduce the USB partition size and reserve several hundred megabytes of hidden data at the end of the disk for malicious purposes." (Kaspersky, 2016, p. 8).

Mandiant (Mandiant-Consulting, 2016, pp. 30-31) have identified at least one MBR (Master Boot Record) bootkit which iterates over all NTFS formatted logical drives and attempts to store malware, in two places – one as a disk file and another in unallocated sectors near the end of the file system. The latter is a backup in the event the former is removed. The installer then overwrites sections of the malicious MBR over the legitimate MBR, preserving the original partition table and error messages. The malware ensures that the MBR is only modified on the physical drive that contains the file system where %WinDir% (i.e. where the Windows operating system is installed) is located and that the MBR has not been previously modified.

Given that there are two places in the OS where data existence is known, a comparison of these two places might yield discrepancies and, hence, possible malware.

## 3. Data and Literature Review

In concert with Hard Disk Drive (HDD) geometry (Bentley, 2021, pp. 55-61) the Microsoft Windows Operating System (OS) stores and manages system and user files. This is achieved using the ($Bitmap) (Microsoft, 2023b) and the $MFT (Microsoft, 2023c), (Sammes, 2007, pp. 215-275, 389-410). The Windows OS treats everything as a file:

> “NTFS is based round a relational database. This is the MFT or Master File Table. All “objects” stored on the volume are regarded as files, except the Partition Boot Record”
>
> (Sammes, 2007, p. 217)

No reference from Microsoft for this assertion could be found but work on this paper enforces the concept of “everything is a file”.

$Bitmap stores cluster use in little endian bytes. Each cluster contains eight sectors, each sector contains 512 bytes.

Sammes et al state that in $MFT files are Resident or Non-Resident. The former means that the file is small enough to fit in the $MFT entry; the latter means that the file is too big for this and is stored in clusters elsewhere on the HDD. The positions of these clusters are given by data runs in the $MFT entry.

Any malware not modifying the cluster table means that APTs risking perpetual existence on the HDD and may be erased by a programs like cipher (Microsoft, 2023a) or SDelete (Microsoft, 2022b).

## 4. The Analysis

Given the structure of the HDD and $MFT all that needs to be done is read $MFT and $Bitmap, step through the $MFT entries looking for non-resident data and use the associated data runs to set to zero in $Bitmap every $MFT cluster that has been denoted as used by these data runs. $MFT is also a file and part of this process so Resident data is inherently incorporated into the process as part of te $MFT data runs.

Kusano’s program (Kusano, 2015) was modified in accordance with the licence. This modified program read $MFT and $Bitmap and for any non-resident $MFT data and set the relevant bit in $Bitmap to zero. For this paper, the resulting $Bitmap table is denoted as £Bitmap. Each entry in £Bitmap is an eight-but byte. The program looks for non-zero bytes. Therefore, storage flagged as being used will be one of eight clusters.

Another program was written to test the cluster table overwrite. This program produced a simulated cluster table where the used clusters were prime numbers positions. It was reasoned that a non-repeating set of data would be needed (e.g. not all even numbers to be set) and not an unrepeatable set which could not be set to zero. Another method could have used, for example, srand with the same seed for generation and overwriting (Open_Group, 2023)..

Five machine states were analysed: five machines, A, B and C as well as A after it had been optimised (defragmented). The Table below shows where, after the above process, the $Bitmap entry was seen four times for A, B, C and A optimised and extended to machines D and E:

| £Bitmap entry | Machine | | | | | | |
|---|---|---|---|---|---|---|---|
| | A | B | C | A (Optimised) | A (Rebuilt) | D (Rebuilt) | E (Rebuilt) |
| 1 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
| 51 | | 4 | | | | 1 | 1 |
| 53 | 1 | | 1 | 1 | 1 | x | x |
| 56 | 8 | 8 | 8 | 8 | 5 | 5 | 5 |
| 68 | 2 | 2 | 2 | 2 | 46 | 46 | 46 |
| 94 | 26 | 26 | 26 | 26 | 46 | 46 | 46 |
| 130452 | 1 | 2 | 2 | 1 | x | x | x |
| 175666 | 1 | 1 | 1 | 1 | x | x | x |
| 7716063 | 1 | 1 | 1 | 1 | x | x | x |
| No of Counts of 1 | 314 | 646 | 627 | 315 | 223 | 151 | 112 |
| Longest Run | 539243 | 661775 | 538209 | 539243 | 883246 | 880328 | 958896 |
| Penultimate Run Start | 5144309 | 3811638 | 5851739 | 5144309 | 3782080 | 3729242 | 3739558 |
| Final Run Start | 7716063 | 7716063 | 7716063 | 7716063 | 7716063 | 7716063 | 7716063 |

**Table 1 - Cluster Set Usage**

The £Bitmap entry is the cluster set position; cells are the length of the entry. Entry 53 was seen in machines A, B, C but masked but a longer entry run on Machine B.

There are inconsistencies between the HDD usage in $MFT and $Bitmap. The data at 7716063 is consistent with the observation of malware at the end of the partition.

## 5. Discussion

What may be the source of theses inconsistencies?

The inconsistencies may be valid Windows OS conditions and the $MFT and $Bitmap are expected to be inconsistent. i.e. there are Windows OS files which are deliberately not referenced by $MFT but need to be referenced by $Bitmap.

Some, or all, of the inconsistencies may be malware. The final cluster set is almost at the end of the partition and, therefore, aligns with the Alternative Hypothesis.

The OS disk management software may have bugs in it. This is not as far-fetched as it sounds as Microsoft issue patches for the OS. There are also known reliability issues with file date/times as highlighted by Sammes et al (Sammes, 2007, pp. 400-403).

## 6. A Critique of this Work and Suggested Further Lines of Work

The idea behind this paper was to find clusters on the HDD which may host malware.

Potential problems with the approach:

- As always, the software developed may contain bugs;
- The Microsoft routine to read $Bitmap (Microsoft, 2022a) with the FSCTL_GET_VOLUME_BITMAP control code (Microsoft, 2021) could have been maliciously modified. It would have been better to build $Bitmap from $MFT. Again, such a modification is not far-fetched as it is known that system commands have been modified by an APT (Bentley, 2021, p. 139);
- Using POSIX C and working as close to the OS and data as possible to reduce the possibility of malicious interference would be better. However, this paper is trying to achieve proof of concept, so time and ease of programming were chosen;
- The program looked for non-zero bytes in $Bitmap and hence eight-byte cluster sets, not individual clusters. It could be modified to look at each bit to narrow the cluster usage but as a first pass non-zero-bytes was considered sufficient.

Given that this aligns with known a known APT technique further work on the contents of this data will be done as part of this series of papers.

## 7. Conclusion

It has been demonstrated that for five machines, one with two states, there is an inconsistency between the storage allocated by $MFT and $Bitmap. The inconsistency could be faulty analysis, OS disk management issues, deliberately OS design or malware. The Null Hypothesis is therefore rejected and the Alternative Hypothesis accepted.

## References

Bentley, P. (2021) *The Treatment of Advanced Persistent Threats on Windows Based Systems.*

Berghel, H. H., David ;Sthultz, Michael. (2008) *Chapter 1 Data Hiding Tactics for Windows and Unix File Systems.* Vol. 74: Academic Press, Inc.6277 Sea Harbor Drive Orlando, FL. United States.

Horejsi, J. (2018) *New Powershell-Based Backdoor Found in Turkey, Strikingly Similar to Muddywater Tools.* Available at: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections

Kaspersky (2014) *The Epic Turla Operation: Solving Some of the Mysteries of Snake/Uroboros.* Available at: https://cdn.securelist.com/files/2014/08/KL_Epic_Turla_Technical_Appendix_20140806.pdf

Kaspersky (2016) *The Project Sauron Apt.* Kaspersky. Available at: https://cdn.securelist.com/files/2016/07/The-ProjectSauron-APT_research_KL.pdf

Kusano. (2015) *Ntfsdump*. Available at: https://github.com/kusano/ntfsdump (Accessed: 2023).

Mandiant-Consulting (2016) *M-Trends 2016.* Available at: https://marketingcentral.fireeye.com/ResourceFiles/9c255c9d-3a6a-4086-a000-bf614d811fee.pdf (Accessed: Accessed: 4th June 2016).

Microsoft. (2021) *Fsctl_Get_Volume_Bitmap Ioctl (Winioctl.H)*. Available at: https://learn.microsoft.com/en-us/windows/win32/api/winioctl/ni-winioctl-fsctl_get_volume_bitmap (Accessed: 19th August 2023).

Microsoft. (2022a) *Deviceiocontrol Function (Ioapiset.H)*. Available at: https://learn.microsoft.com/en-gb/windows/win32/api/ioapiset/nf-ioapiset-deviceiocontrol (Accessed: 19th August 2023).

Microsoft. (2022b) *Sdelete V2.04*. Available at: https://learn.microsoft.com/en-us/sysinternals/downloads/sdelete (Accessed: 26th August 2023).

Microsoft. (2023a) *Cipher*. Available at: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/cipher (Accessed: 26th August 2023).

Microsoft. (2023b) *How Ntfs Reserves Space for Its Master File Table (Mft)*. Available at: https://learn.microsoft.com/en-us/troubleshoot/windows-server/backup-and-storage/ntfs-reserves-space-for-mft (Accessed: 16th August 2023).

Microsoft. (2023c) *Master File Table*. Available at: https://learn.microsoft.com/en-us/windows/win32/devnotes/master-file-table (Accessed: 16th August 2023).

Open_Group. (2023) *Rand*. Available at: https://pubs.opengroup.org/onlinepubs/9699919799/functions/rand.html (Accessed: 27th August 2023).

Sammes, T. J., B. (2007) *Forensic Computing a Practioner's Guide.* London 2010: Springer-Verlag.

Symantec-Security-Response (2015) *Regin: Top-Tier Espionage Tool Enables Stealthy Surveillance.* Symantec. Available at: Regin: Top-Tier Espionage Tool Enables Stealthy Surveillance.