Host Forensics Concepts

Bratosin Bogdan Alexandru

Proposed agenda

Forensic Basics

Physical and Logical Disk Structures

Windows Artifacts / Evidence of …

Windows Registry

What is Forensic?

Why is it needed?

Forensic science, also known as criminalistics, is the *application of science to criminal and civil laws*, mainly on the criminal side – during criminal investigation, as governed by the legal standards of admissible evidence and criminal procedure. Forensic scientists collect, preserve, and analyze scientific evidence during the course of an investigation.

Digital forensics, also known as computer and network forensics, has many definitions. Generally, it is considered the *application of science to the identification, collection, examination, and analysis of data while preserving the integrity of the information and maintaining a strict chain of custody for the data*.
Data refers to distinct pieces of digital information that have been formatted in a specific way.

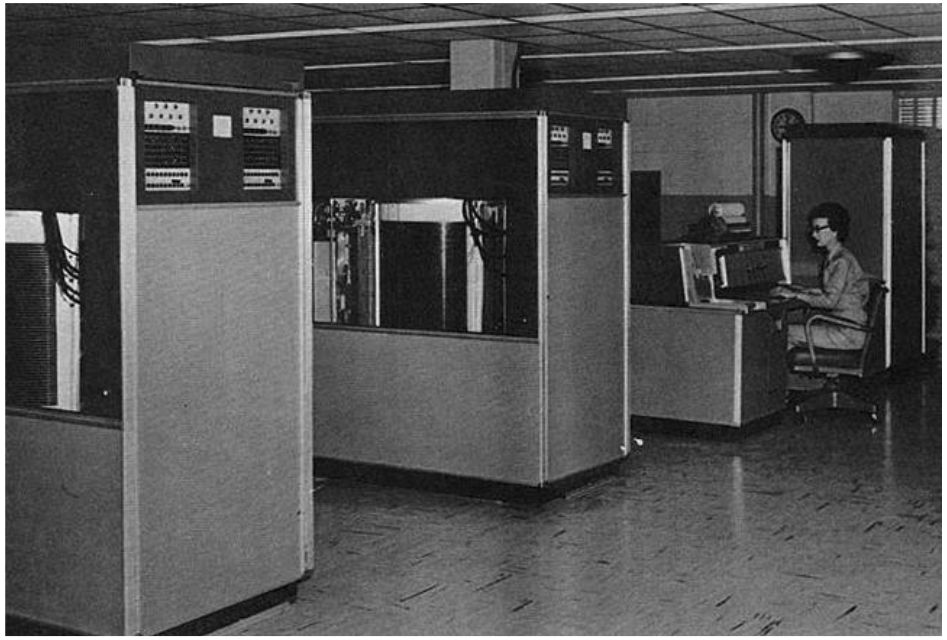
Digital forensic techniques can be used for many purposes, such as:

- investigating crimes and internal policy violations,
- reconstructing computer security incidents,
- troubleshooting operational problems,
- recovering from accidental system damage.

Back in the Day

The fist **H**ard **D**isk **D**rive (**HDD**) in the world was shipped in 1956.
The drive contained fifty 24-inch platters, stored 5 Mb of data and took more room than two refrigerators.

The price? Just 50.000 USD



Typical technologies

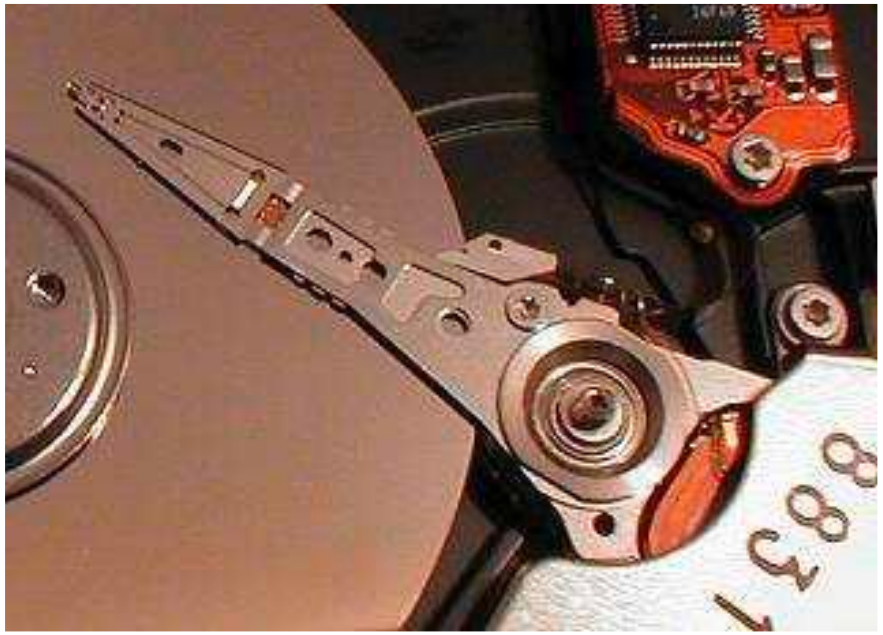
IDE / EIDE	**E**nhanced **I**ntegrated **D**rive **E**lectronics – deprecated
SCSI	**S**mall **C**omputer **S**ystems **I**nterface – found in servers and systems where high performance and data integrity is critical
SATA	**S**erial **A**dvanced **T**echnology **A**rchitecture – the most commonly used type of HDD in personal computers; reached its 3rd revision (SATA III)

HDD Internal Characteristics

Drives will have one or more spinning platters made of rigid aluminum or glass coated with various forms of a magnetic substrate. There's an actuator arm with read/write heads attached to it. This arm positions the read-write heads over the correct area of the drive to read or write information.

The drive may need to read from multiple locations in order to launch a program or load a file, which means it may have to wait for the platters to spin and the arm to be moved into the proper position multiple times before it can complete the command.

If a drive is asleep or in a low-power state, it can take several seconds more for the disk to spin up to full power and begin operating.



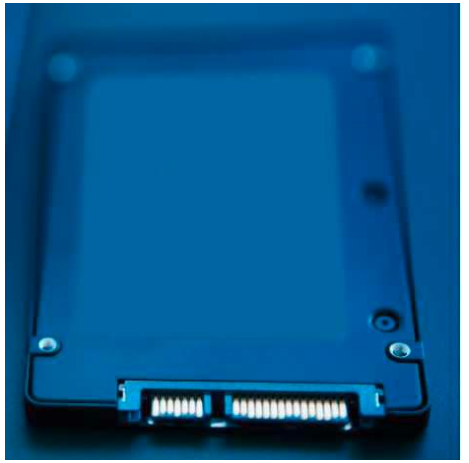
SSD Solid State Drive

Solid state: electrical term that refers to electronic circuitry that is built entirely out of semiconductors.

In terms of an SSD: it refers to the fact that the primary storage medium is through semiconductors rather than a magnetic media such as a hard drive.

An SSD on the outside looks almost no different than a traditional hard drive. The design is to allow the SSD drive to be put in a notebook or desktop computer in place of a hard drive.

Standard dimension: 1.8, 2.5 or 3.5-inch.
Uses either the ATA or SATA drive interfaces so that there is a compatible interface (other form-factors exists).



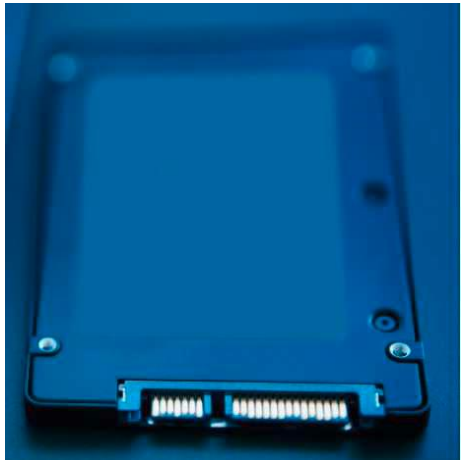
SSD Advantages

The drive does not have any moving parts.

While a traditional drive has drive motors to spin up the magnetic platters and the drive heads, all the storage on a solid state drive is handled by flash memory chips.

This provide three distinct advantages:

- Less power usage
- Faster data access
- Higher reliability



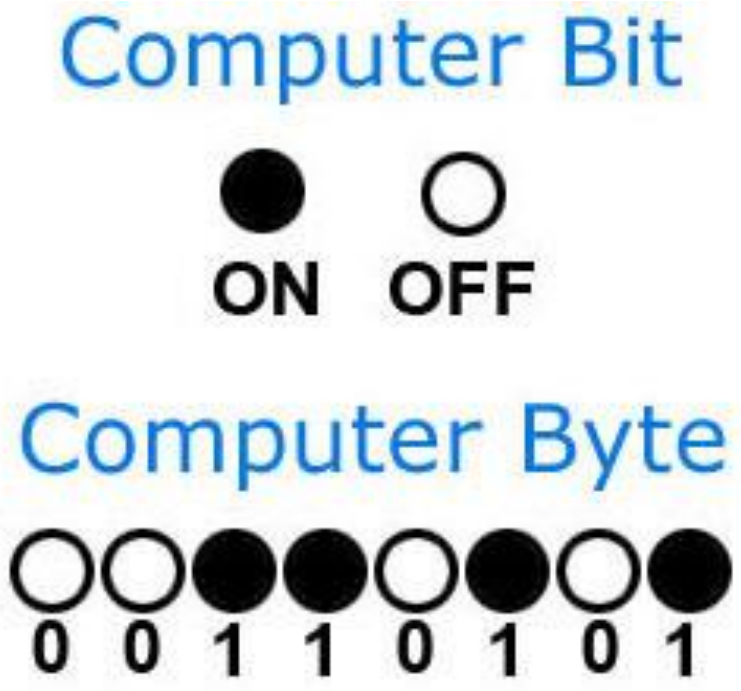
Logical Disk Structure

In every memory storage device, the data is written in the form of a magnetic field or electrical charge that represents an **on** or **off** value, which we know of as a **b**inary dig**it** or **bit**.

In order to make data storage device useable, some form of order needs to be applied to it, which refers to as the Logical Disk Structure.

For data to be written to a disk, there are three processes that must be undertaken:

- Low-level formatting
- Partitioning
- High-level formatting



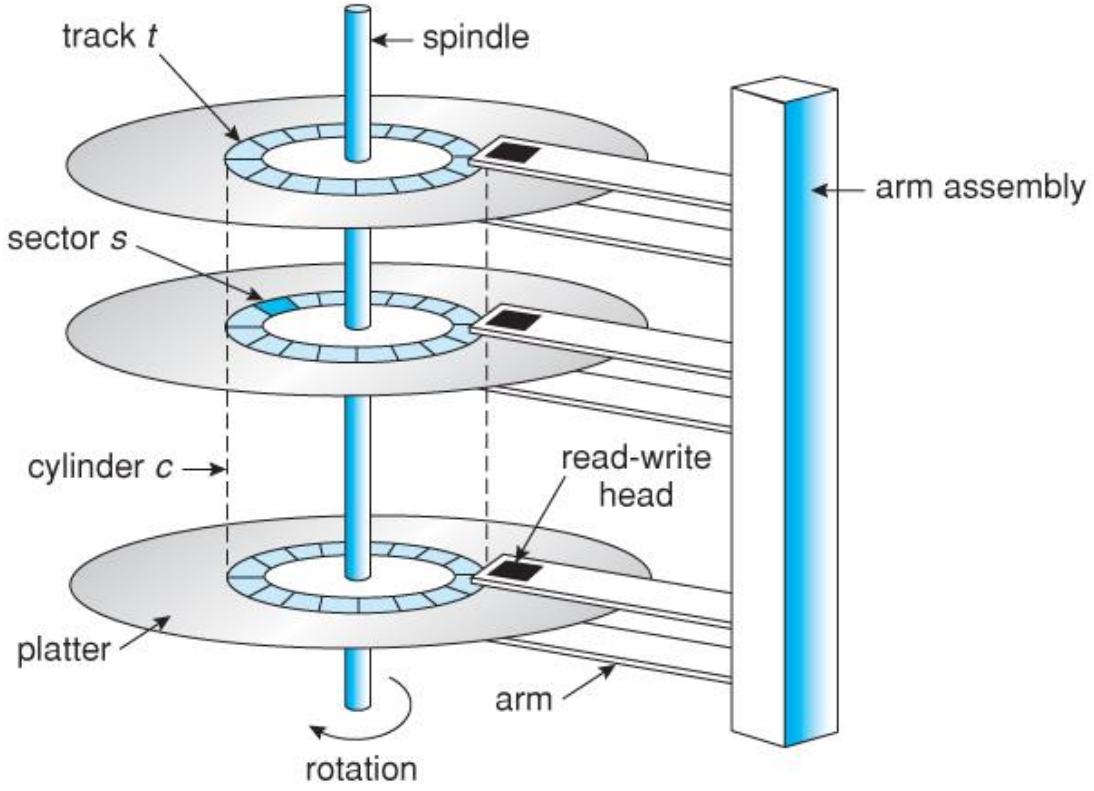
Low-level formatting / Sectors

When data is written in a circular pattern there is a problem identifying *where that data begins* and *where it ends*. To overcome this the track is divided into smaller chunks known as **sectors**.

A sector is the smallest storage unit that is writeable by an HDD; the most common physical sector size is 520 bytes, from which only 512 bytes are used for the actual storage of data and 8 bytes are used for error checking.

The process of creating sectors is called Low-level formatting and can only be done by the HDD manufacturer.

As SSD doesn't have physical platters and tracks, the sector structure is created also by the manufacturer.



Low-level formatting / Clusters

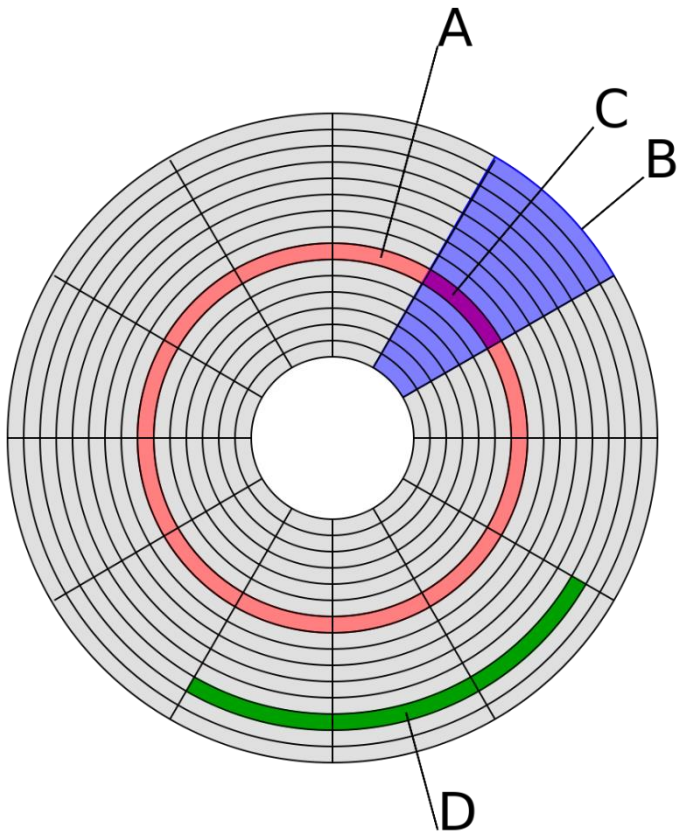
The smallest unit of disk space that can be allocated to a file is called a **cluster** (aka allocation unit).

A cluster consists of one or more consecutive sectors.

Every file must be allocated an integer number of clusters.

If a cluster size is 4096 bytes then a 4000 bytes file will use one cluster or 4096 bytes on the disk.
A 5000 bytes file will use two clusters, or 8192 bytes on the disk.

This “wasted” space is called **slack space**.



A = track
B = sector
C = sector of a track
D = cluster

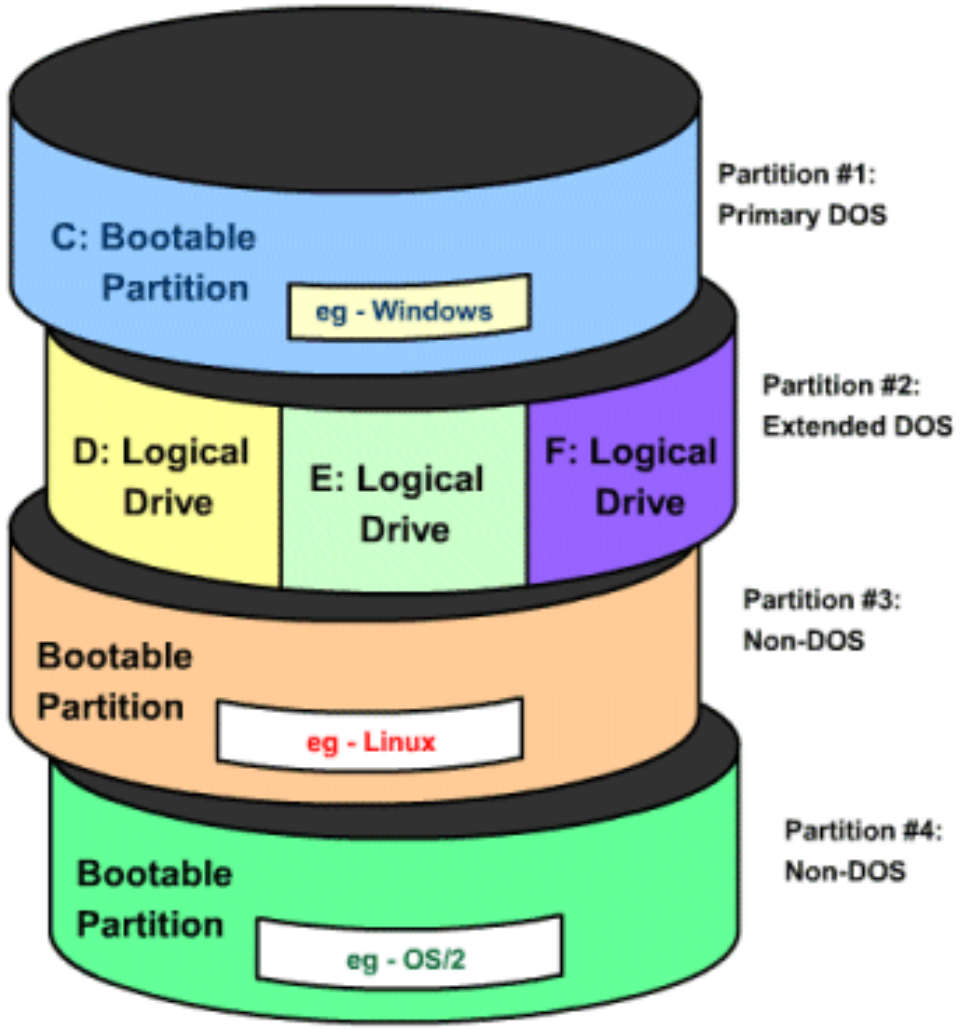
Partitioning

Involves logically dividing the hard disk up into a number of pieces, each piece being called a **partition**.

Partitioning is typically the first step of preparing a newly manufactured disk, *before* any files or directories have been created.

The disk stores the information about partitions locations and sizes in an area known as the partition table that the operating system reads before any other part of the disk.

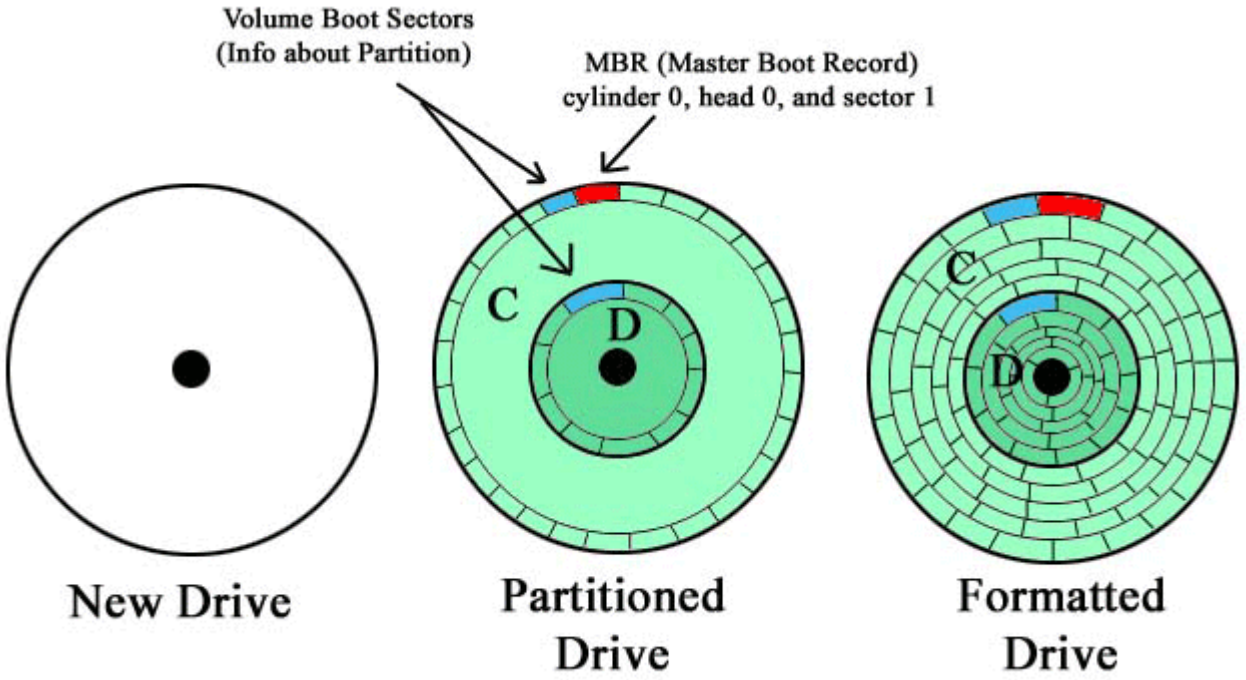
Each partition then appears in the operating system as a distinct "logical" disk that uses part of the actual physical drive.



High-level formating

Represents the process of writing the file system structures on the disk, such as the master boot record and the file allocation tables, that let the disk be used for storing programs and data.

High-level formatting is done after the hard disk has been partitioned (even if only one partition is to be used).

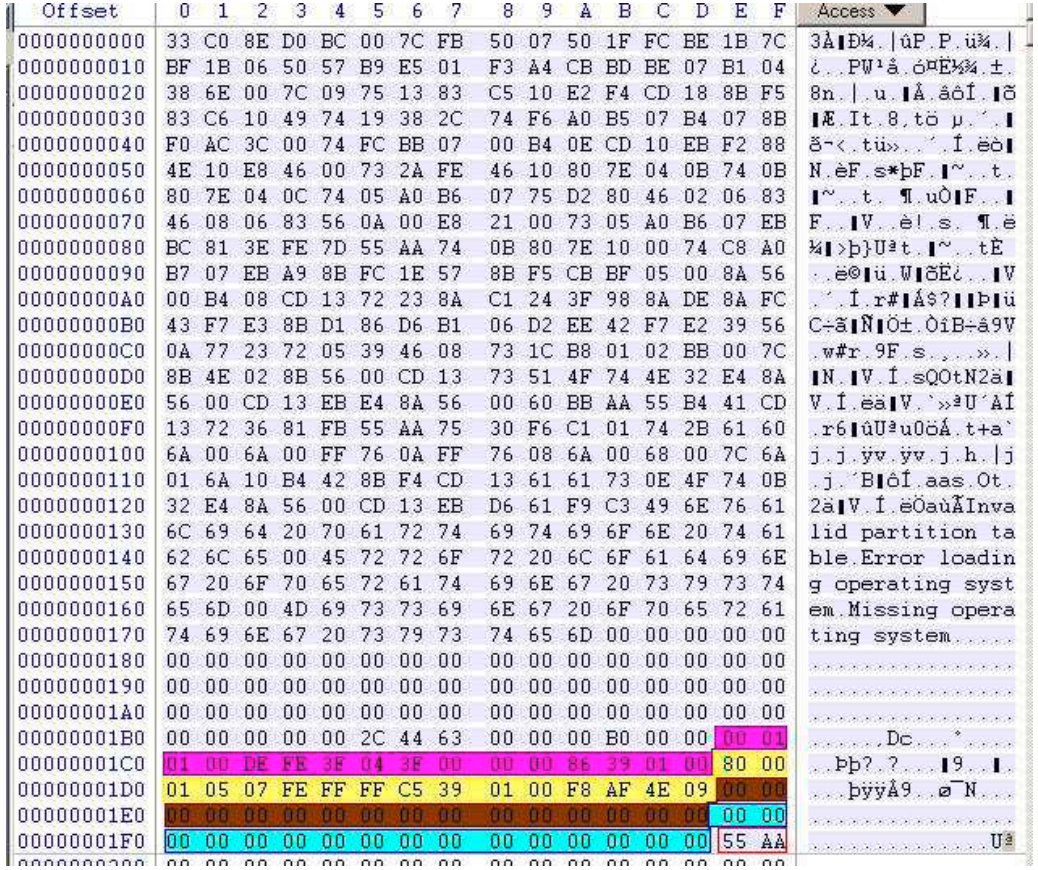


Master Boot Record

Begins in the very first physical sector of the hard drive, usually referred to PS0 or Physical Sector 0.
Typically the MBR will only fill one sector or 512 bytes of information; the remaining sectors in the track are reserved.

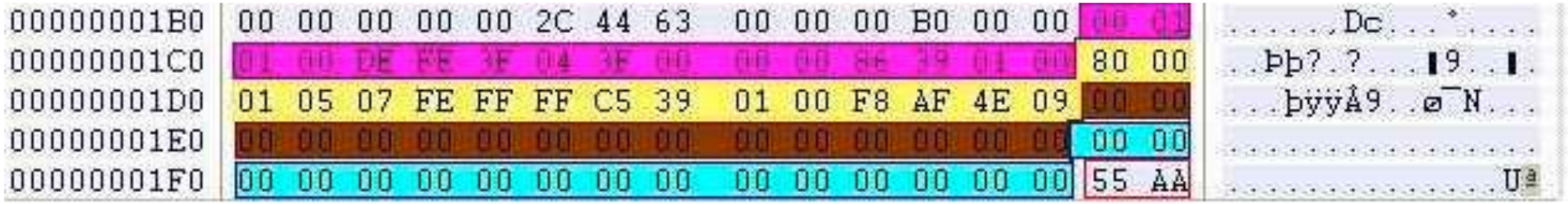
The first 446 bytes (0 to 445) of information are actual programming code or boot code. This code identifies the drive and instruct the system on the structure of the drive.
The next 64 bytes consist of the Master Partition Table (four 16 byte-entries). The last two bytes of the sector are always hexadecimal 55 AA (0x55AA) (bytes 510 and 511).

Without an MBR, the computer would stop after the BIOS finished executing its data because it wouldn't know where to go next to find its instruction about what to load next.



Master Partition Table

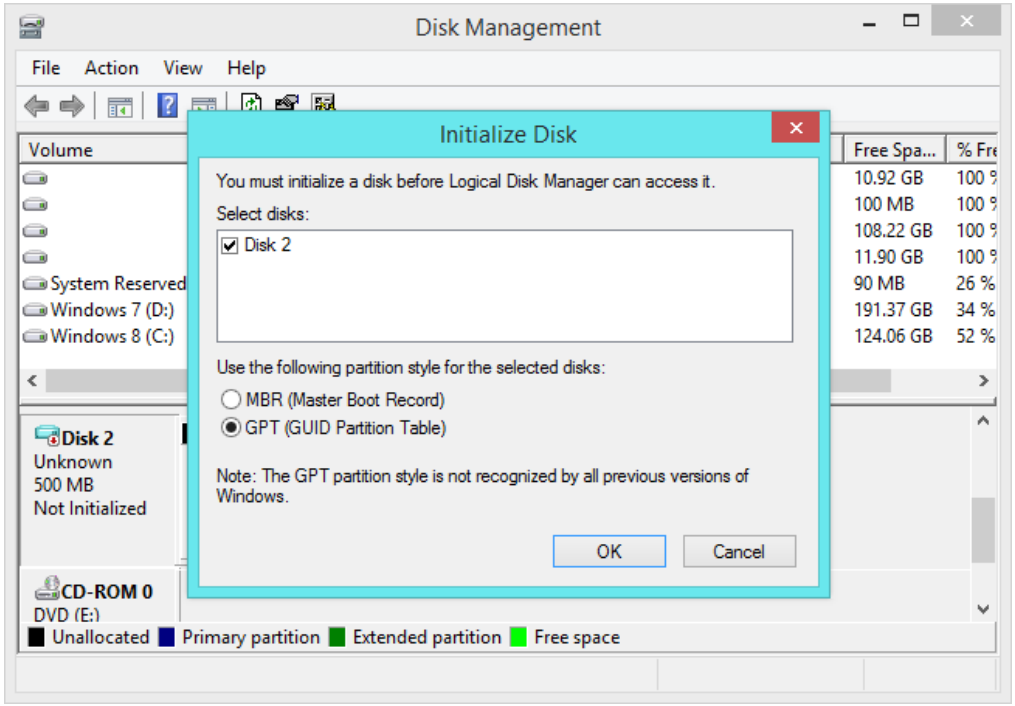
The MPT is 64 bytes in length and contains four 16 bytes entries. Each striped line represents a single partition table entry:



This allows for up to four “**Primary Partitions**” on a drive (a Primary Partition is a partition that can contain the computer boot files). There must be at least one Primary Partition within the MBR and *only primary partitions are **bootable*** (can be used to boot the computer to an operating system).

To overcome the limitations of only having four partitions, a different type of partition is allowed, this is known as an “Extended Primary Partition”, that can be further split into several smaller portions (“**Logical Partitions**”), thus allowing up to a total of twenty-four partition, each partition assigned an alphabetical drive letter (drive letter A and B are reserved for floppy disk).

GPT Partitions



After more than 30 years of supremacy, the BIOS (Basic Input Output System) – has been replaced with a complete re-engineered boot environment, known as UEFI (Unified Extensible Firmware Interface). While the BIOS is fundamentally a solid piece of firmware, UEFI is a programmable software interface that sits on top of a computer's hardware and firmware.

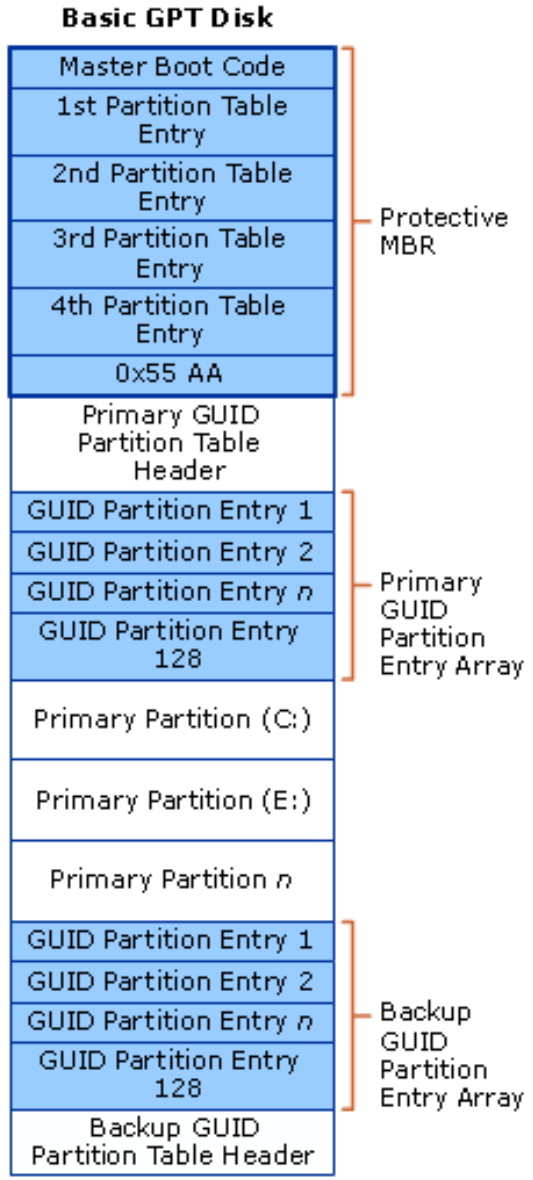
UEFI uses a partition system called the GUID Partition Table (GPT). GUID stands for Globally Unique Identifier (a 128-bit number used to identify information in computer systems).

The GPT Partition table can support up to 128 partitions and uses 64-bit LBA addresses (LBA = Logical Block Addressing where every sector in the drive is given a linear address – the first sector is Sector 0 and they are numbered sequentially to the end of the drive).
The GPT system has the ability to support very large hard drives, which are becoming increasingly available and cheap to purchase.

GPT Partitions

There are five major parts to a GPT partitioned disk:

- The Protective MBR
- Primary GUID partition table header
- GUID partition entries
- Partition area
- Backup Area



GPT Partitions

Protective MBR is located at the very beginning of the disk (Physical Sector 0) and like the previous system it is usually 512 bytes in length. There will be no Boot Code present in the Protective MBR sector. This structure is for backward compatibility with disk management utilities that operate on MBR.

The **primary GUID Partition Table Header** will always immediately follow the MBR in Physical Sector 1. The GPT header always begins with the 8-byte EDI signature string: 0x45 0x46 0x49 0x20 0x50 0x41 0x52 0x54 (ASCII: “EFI PART”). Using the information contained in the GPT header it is possible to determine the layout of the disk. This includes the location of the partition table, partition data areas and backup copies.

The **partition tables** are usually located in Physical Sector 2. Each partition entry is 128 bytes in length and the entry provides much information about the partition.

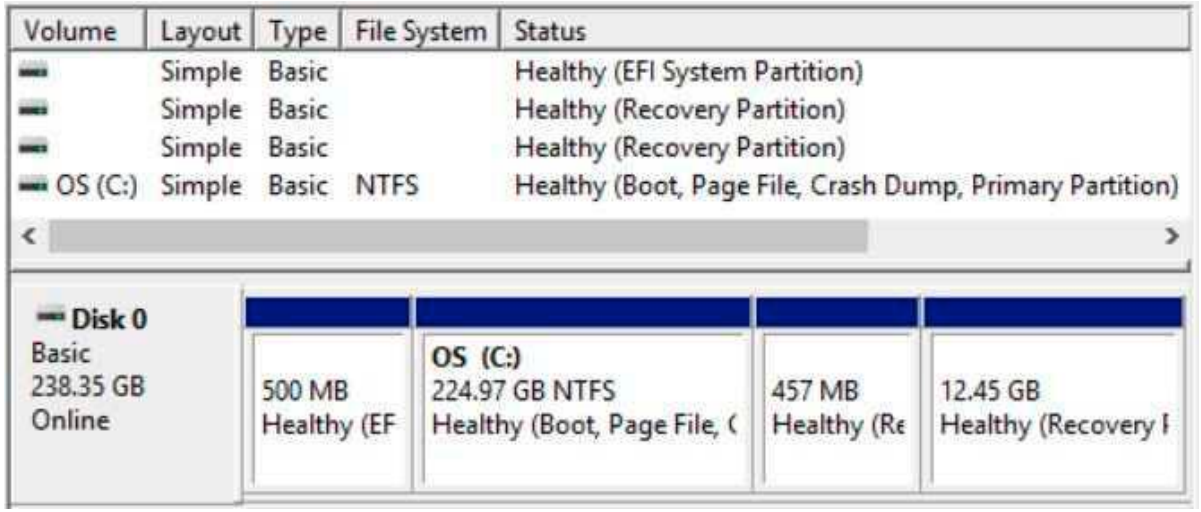
GPT Backup – one of the advantages of using a GUID partition structure is the additional resilience it provides. Located at the end of the disk is an entire backup of the Primary partition entries and the GPT Header.

NTFS (New Technology File System)

NTFS was developed in 1990’s as a replacement for FAT and it has undergone four upgrades, most recent version being v3.1

The main features and advantages over FAT:

- Support for mixed case filenames;
- Support for long filenames (255 characters as opposed to FAT’s 8+3 char.);
- Less fragmentation of data;
- Transaction journaling for crash recovery;
- Support for compression, encryption and quota enforcement.



NTFS

In NTFS, all the data is stored as files (including system data).

Basically there are **regular files** (containing user data) and **“special” files** (containing system data).
These are referred to as **Metadata files** (metadata = “data about data”).
These special files are hidden to users on a live NTFS volume and are not accessible through the file system itself.

The most important metadata file in an NTFS volume is the Master File Table (**$MFT**).
NTFS uses the $MFT to track and store all information about every file within the volume – including itself.

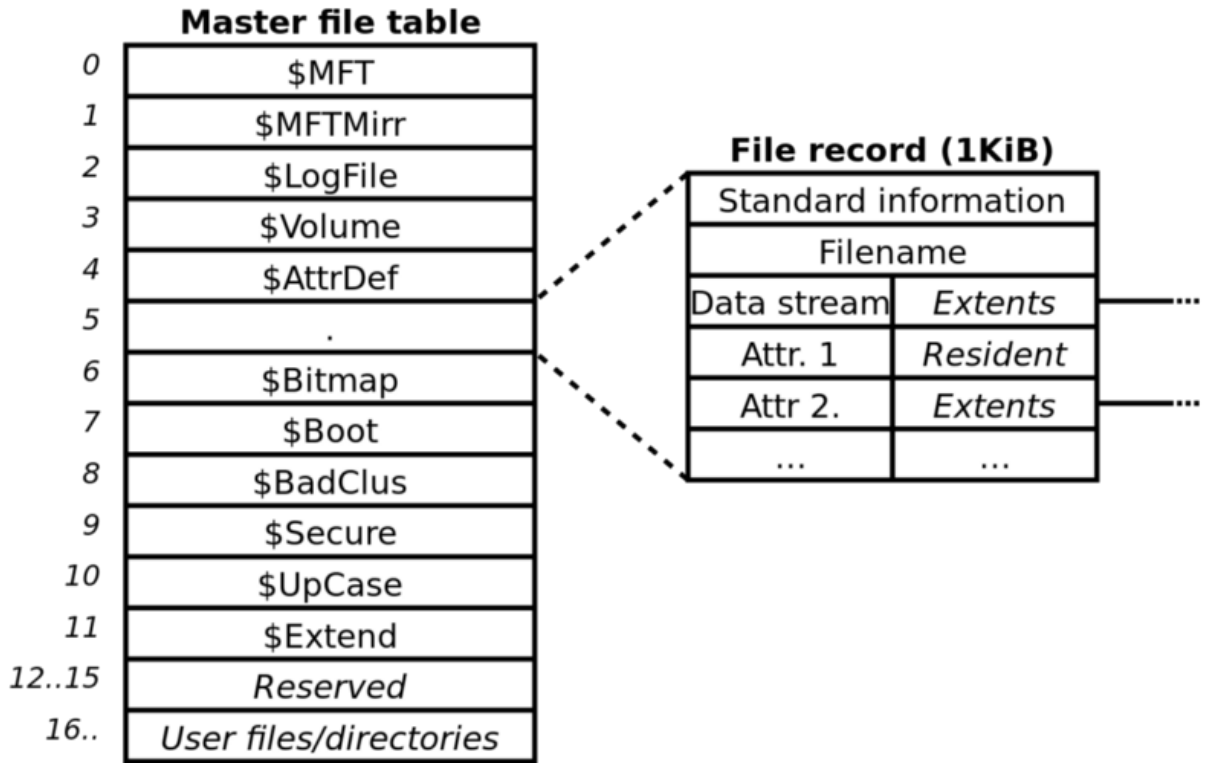
Additional metadata files are used to track storage space allocation, security issues, accessibility permissions, journaling and encryption.

NTFS / MFT

NTFS uses Master File Table to track every file within the volume. It does this by having (at least) one entry for each file, called a **File Record**, which is given a unique number.
The MFT is similar to a relational database table, containing various attributes about different files.

The first 16 entries in the MFT are metadata files, the first record listed being for the $MFT itself, which describes its location and size on the NTFS volume. Therefore, the $MFT needs to be processed in order to know its own size and location on the disk.

All files on an NTFS volume, including the root directory, have a record entry in the $MFT describing their size and location in the same manner.



NTFS / File Records

The MFT lists each file in individual entries called **File Records**. Every file record has its own record number, which is used as an identification number for the file that the file records refers to.
As files are added to the volume, a record for each file is added in sequential order within the $MFT.

If a file is deleted, the file record that was used to track that file becomes unused; the $MFT will reuse “deleted” records before creating new records.
As a result, “deleted” MFT records can be over-written very quickly.

A file record is divided into separate blocks of data that contain information about the file record itself and the file or directory to which that file record points to. Information about the file or folder is stored in discrete blocks called “**Attributes**”. Each attribute stores a certain type of information about the file.

The signature (beginning) of a file record is “**FILE**” while the end is marked with **0xFF FF FF FF**

NTFS / Attributes

Attributes come in different types and contain different information about the file, or the file content itself.
Attributes have their own data structure, which comprises of headers and content.
The attribute headers and content are all discrete blocks of data and must be interpreted individually to determine what information the attribute holds about the file.

Attributes that are *mandatory* for all files and directories are:

- **$Standard_Information**
- **$File_Name**

If a file name is longer than eight characters or contains special characters, Windows will also create a DOS-compatible 8.3 name and save this as a second $File_Name attribute.

Files will also have a **$Data** attribute to hold their content. This attribute can be either *resident* (completely within the MFT Record – for files less than 700 bytes) or *nonresident* (header in MFT will point to the cluster the content is in).

NTFS / Attributes

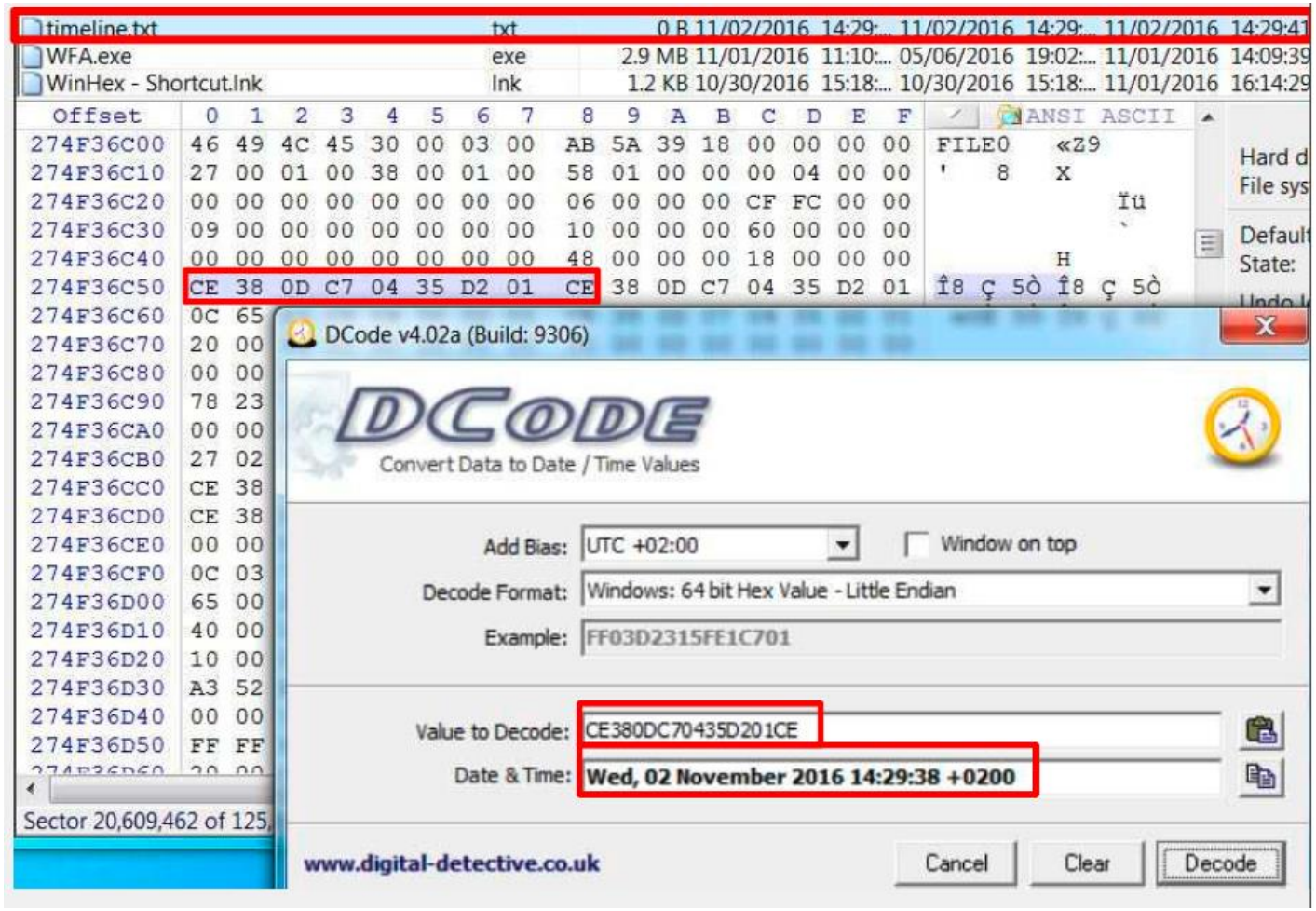
Attribute Identifier	Attribute Name	Description
10 00 00 00	**$Standard_Information**	**Contains File permissions, time stamps, security and administrative information.**
20 00 00 00	$Attribute List	Location of all attributes that do not fit in a single file record entry.
30 00 00 00	**$File_Name**	**The name of the file.**
40 00 00 00	$Object_ID	Contains a Globally Unique Identifier for the file.
50 00 00 00	$Security_Descriptor	Access control and security properties of the file.
60 00 00 00	$Volume_Name	This contains the volume label.
70 00 00 00	$Volume_Information	This attribute contains the NTFS version information.
80 00 00 00	**$Data**	**The actual file ‘s data.**
90 00 00 00	**$Index_Root**	**List of directory’s child files**
A0 00 00 00	$Index_Allocation	Points to the location the Index Buffers of a large directory
B0 00 00 00	$Bitmap	Tracks the allocation status
C0 00 00 00	$Symbolic_Link	Soft link information
D0 00 00 00	$Reparse_Point	Similar to a soft link (legacy)
E0 00 00 00	$EA_Information	Allows compatibility with HPFS (legacy)
00 01 00 00	$Logged_Utility_Stream	Information and keys for encryption attributes

NTFS / Date and Time

Dates and Times are stored in $Standard_Information attribute, in what is called a FILETIME structure.

FILETIME is a 64-bit value representing the number of 100 - nanosecond intervals since January 1, 1601 in Coordinated Universal Time (UTC).

All file dates and times are written to the NTFS file system in UTC



Windows Artifacts / Evidence of …

- File Download
- Program Execution
- File / Folder Opening
- Physical Location
- Account Usage
- External Device / USB Usage
- Browser Usage

The categories on the left map specific Windows artifacts to the analysis questions that the forensic investigation will help to answer, be it malware infection, intellectual property theft or other cyber-crime investigations.

These artifacts will help in painting a clear picture of **which** user was involved, **what** the user was doing, **when** the user was doing it, **why** and **how** the nefarious activities were performed.

File / Folder Opening

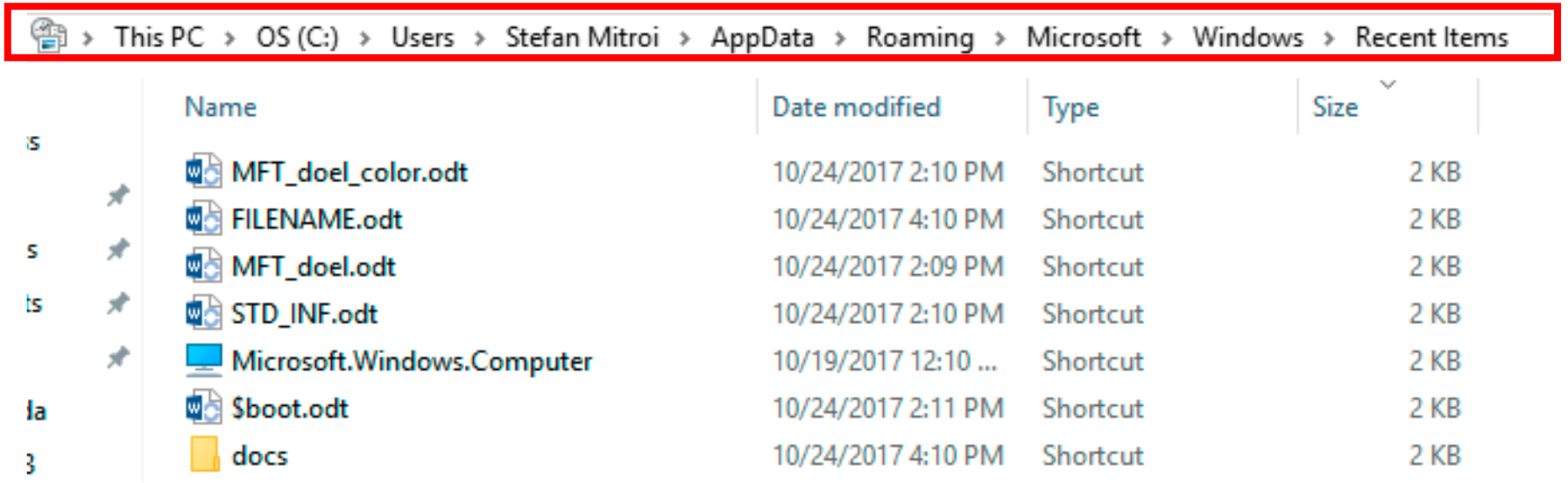
Shell Link Files

A **shell link file** is commonly referred to as a *link file* or *shortcut*. It is a special file that contains “links” or ”pointers” to other resources, for example, programs, data files and folders.

By default, when a file or document is opened in Windows, a link (.lnk) file is created in the Recent folder.

The best thing about a Link file is that it will often demonstrate a ***user’s knowledge of a file and their interaction with that file***.

In Windows 7 – 10 these files are displayed in “Recent Items”, which is a virtual folder when viewed with Windows Explorer.



Windows Artifacts

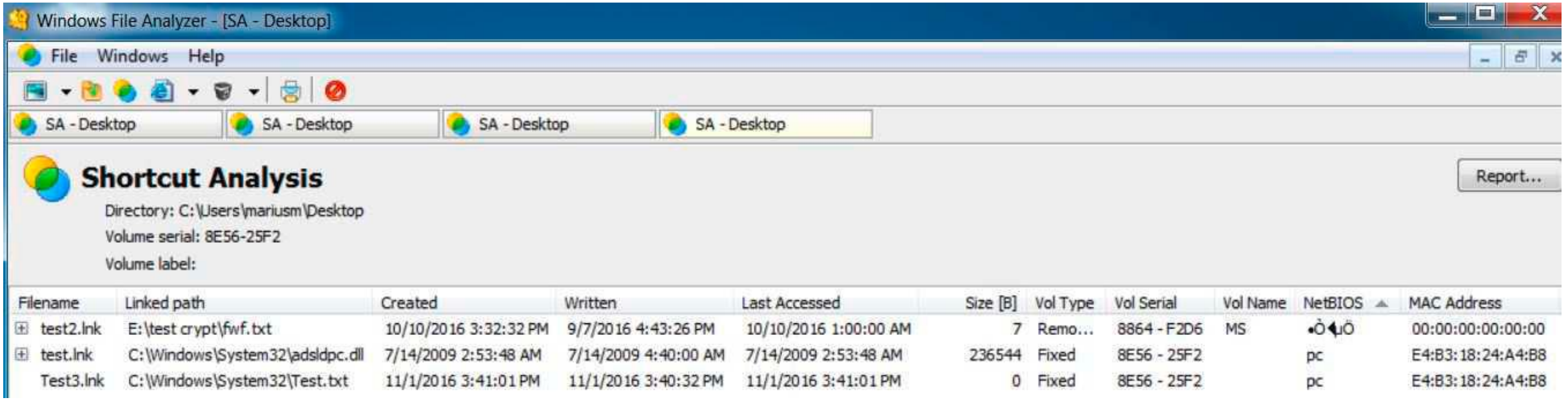
File / Folder Opening

Shell Link Files

These files contain some very useful information about the target file including:

File Attributes	MAC Times	File Size	Volume Type
Volume Serial Number	Volume Label	Original File Path	

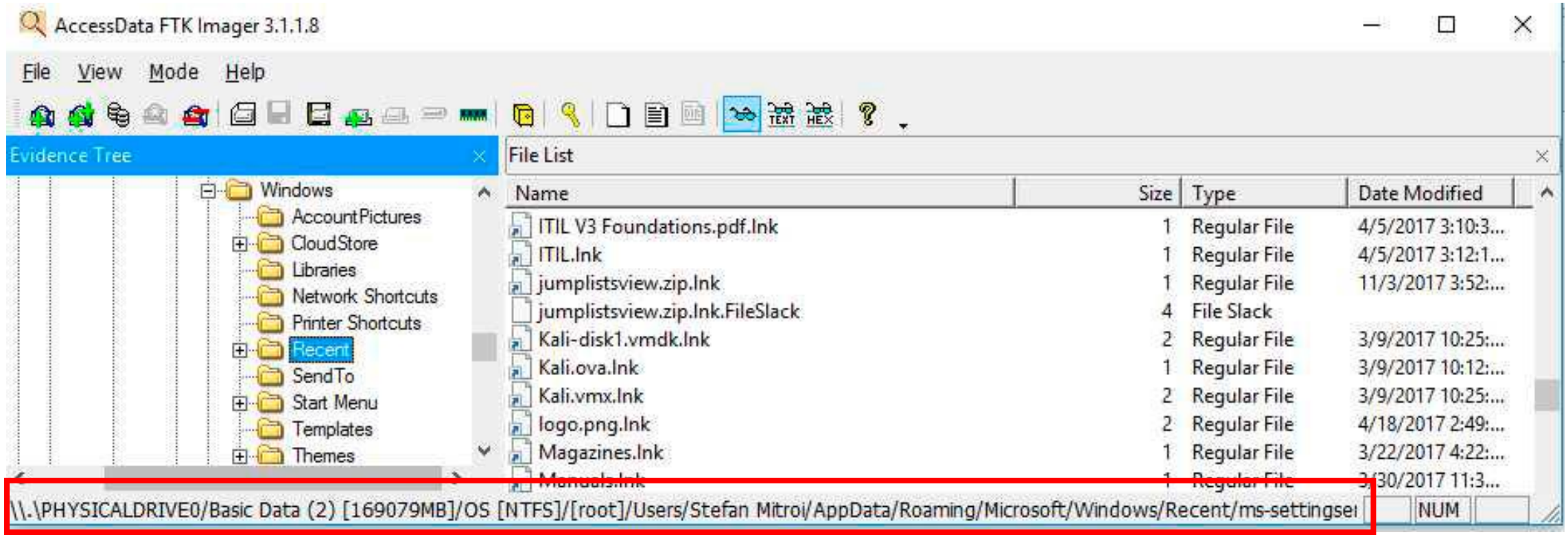
A useful and free tool for parsing .lnk files is Windows File Analyzer (http://mitec.cz/wfa.html)



File / Folder Opening

Shell Link Files

There are many forensics implications relating to the content of these files.
The Volume Serial Number can be used to tie a specific thumb drive, USB drive, memory card or other removable media to a specific computer system.



Windows Artifacts

File / Folder Opening

Jump Lists

Windows 7 introduced a new feature called "**Jump Lists**", which are essentially a list of every file that has been opened (or attempted to open) by a particular application. It's similar to the "Recent" folder, except that each list only applies to one program. The Jump List feature provides the user with a graphical interface associated with each installed application which lists files that have been previously accessed by that application.

This artifact often provides significant insight to user activity and is especially beneficial if entries in the Recent folder have been deleted, or even if the application has been deleted or uninstalled.

Clearing the items in the Recent folder does not eliminate the Jump List data unless the user first reveals the hidden folders containing the Jump Lists data and manually delete them, which is not easy as they are "Super Hidden".

There are two main types of Jump Lists:

- **Automatic** – this Jump List is automatically populated by the system
- **Custom** – this Jump List is maintained by the individual application

File / Folder Opening

Jump Lists

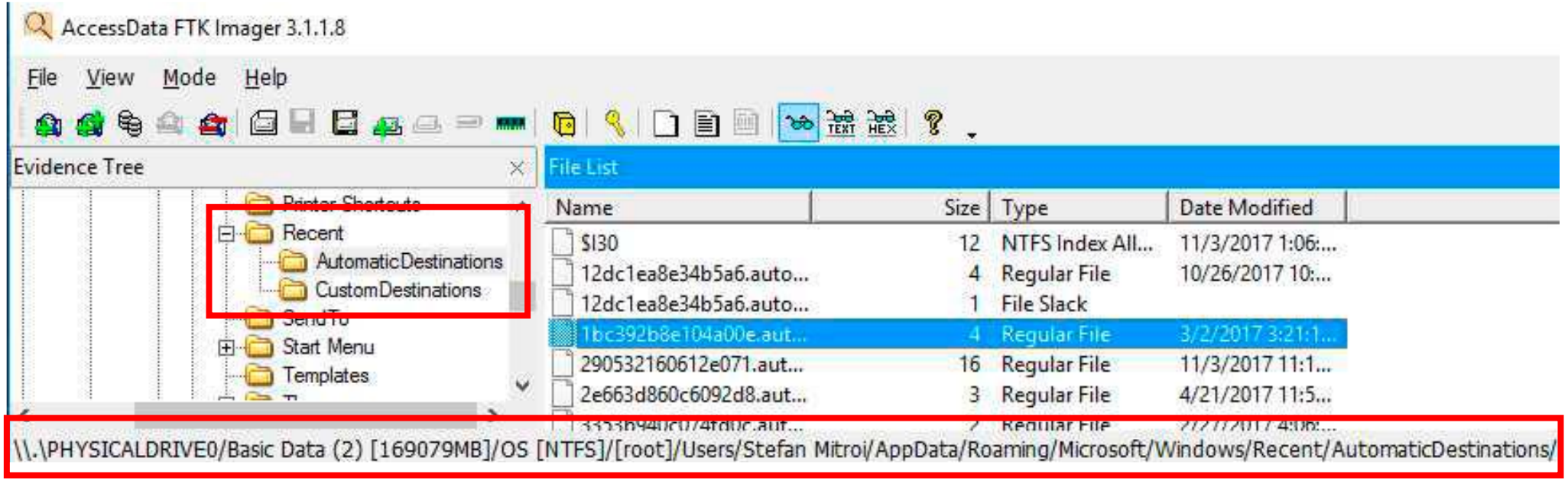
Jump List data for all applications is stored in the users profile path:

%UserProfile%\AppData\Roaming\Microsoft\Windows\Recent

When this folder is viewed with a forensic tool, additional folders appear:

%UserProfile%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations

%UserProfile%\AppData\Roaming\Microsoft\Windows\Recent\CustomDestinations



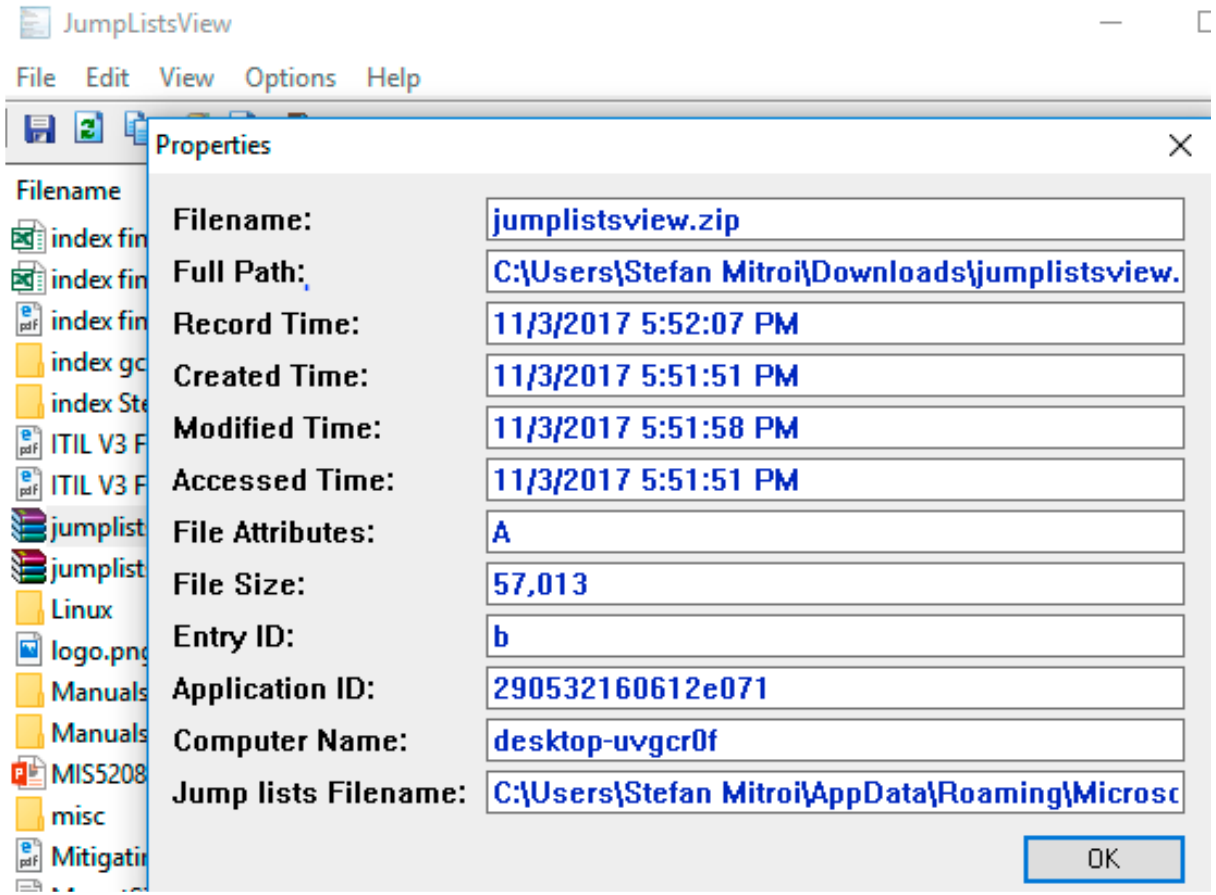
File / Folder Opening

Jump Lists

The recent item data from the Jump Lists populate those two folders. Each program will have its own file name, referred to as a ”Jump List ID”. By examining each file with a text editor, it can be determined which file links correspond to which program’s Jump List entry.
More jump list IDs can be found at:
http://www.forensicswiki.org/wiki/List_of_Jump_List_IDs

A useful tool is available for examining “lists”, called JumpListView, is available at:
http://www.nirsoft.net/utils/jump_lists_view.html

JumpListView will display a list of all items in the Jump List, their path, date and time and the entry number for each item.



Windows Artifacts

Account Usage

Event Logs

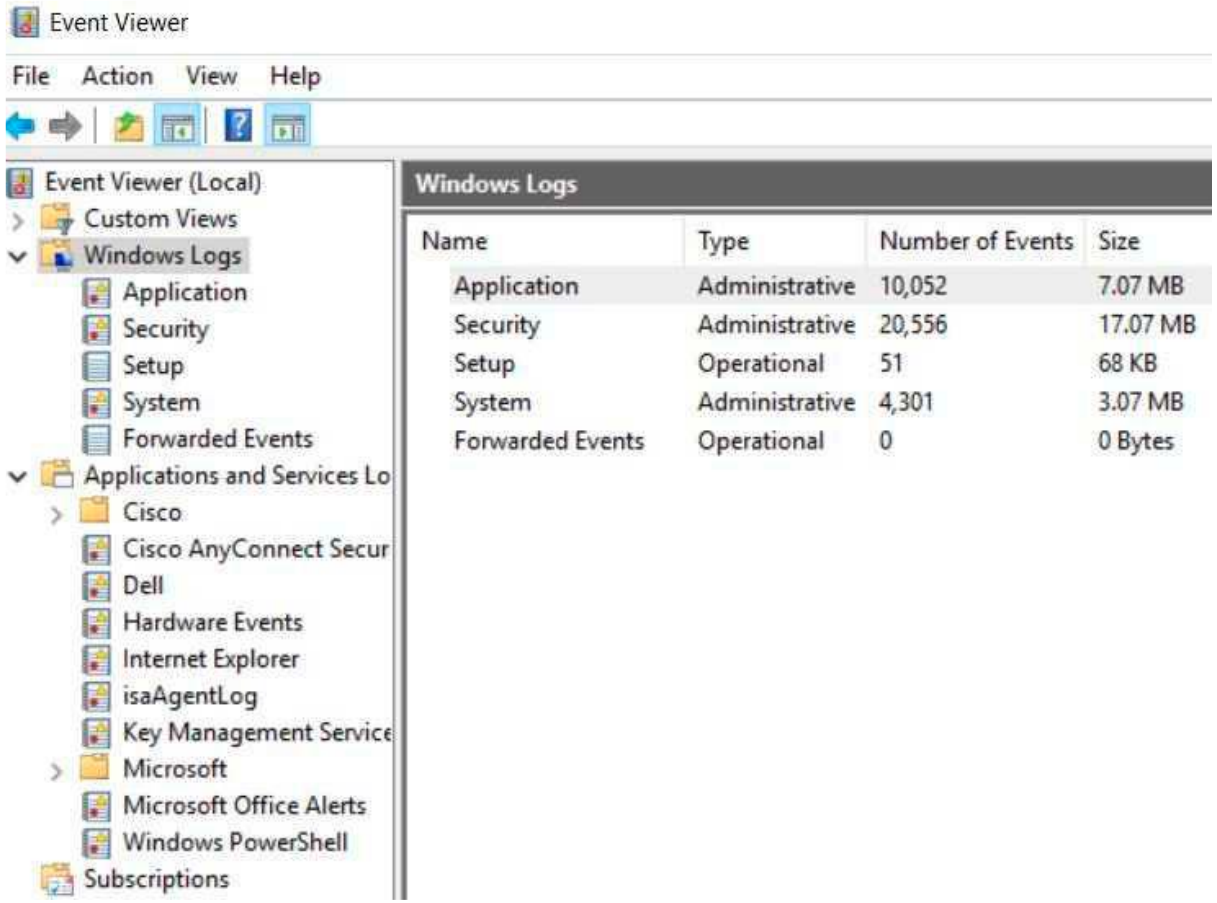
Microsoft defines an “event” as any occurrence that is potentially noteworthy either to the user, the operating system or to an application.

The logs are stored in C:\Windows\system32\winevt\logs

Event logs have the extension .evtx and utilize the .xml format

Events are categorized into 2 main classes:

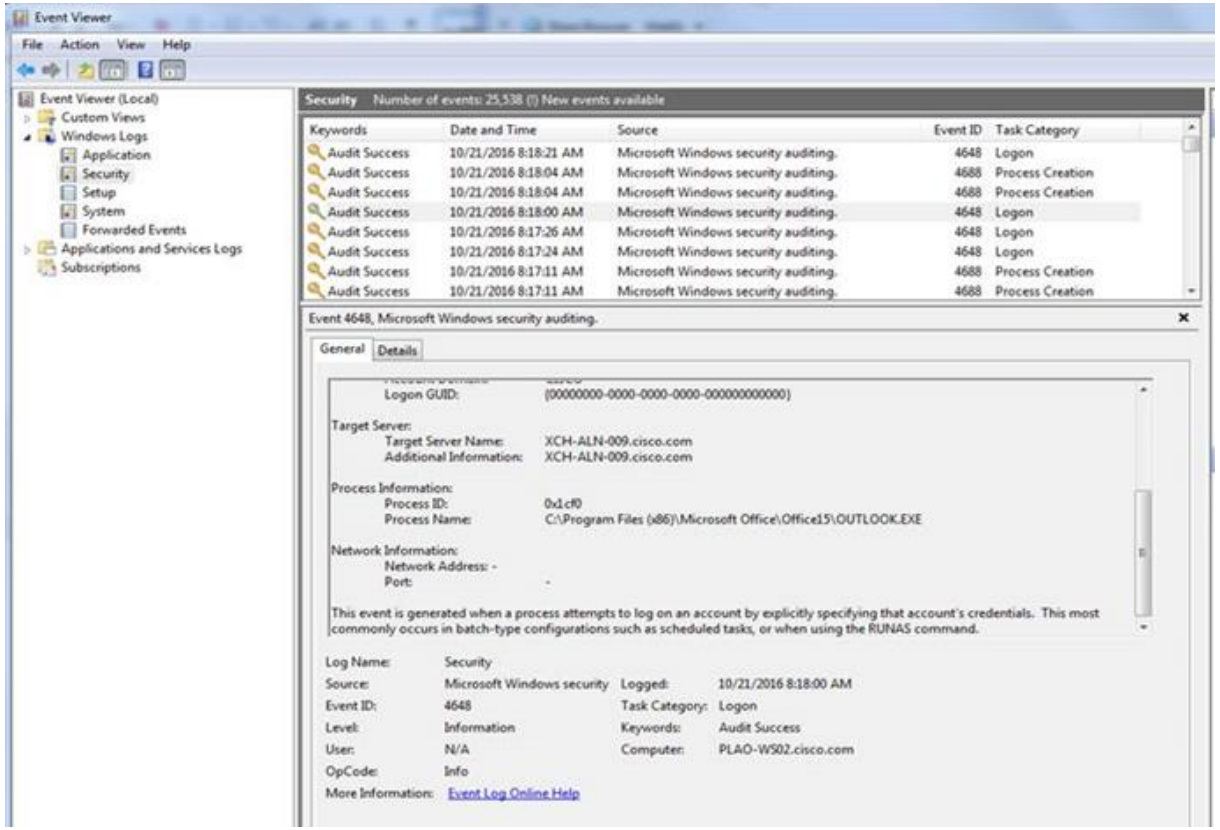
- Windows Logs
- Application and Services Logs



Event Logs

Windows Logs consists of:

- **Security Logs** – provide details on a variety of actions like user authentication (logons, “Run as” commands, remote access etc.) and what a particular user did on a system after authentication (like Files / Folders / Share access).
- **System Logs** – contains events related to Windows services, system components, drivers, resources (ex. Service stopped; System Rebooted).
- **Application Logs** – contains software events unrelated to operating system (ex. MS Office alert or informational messages).

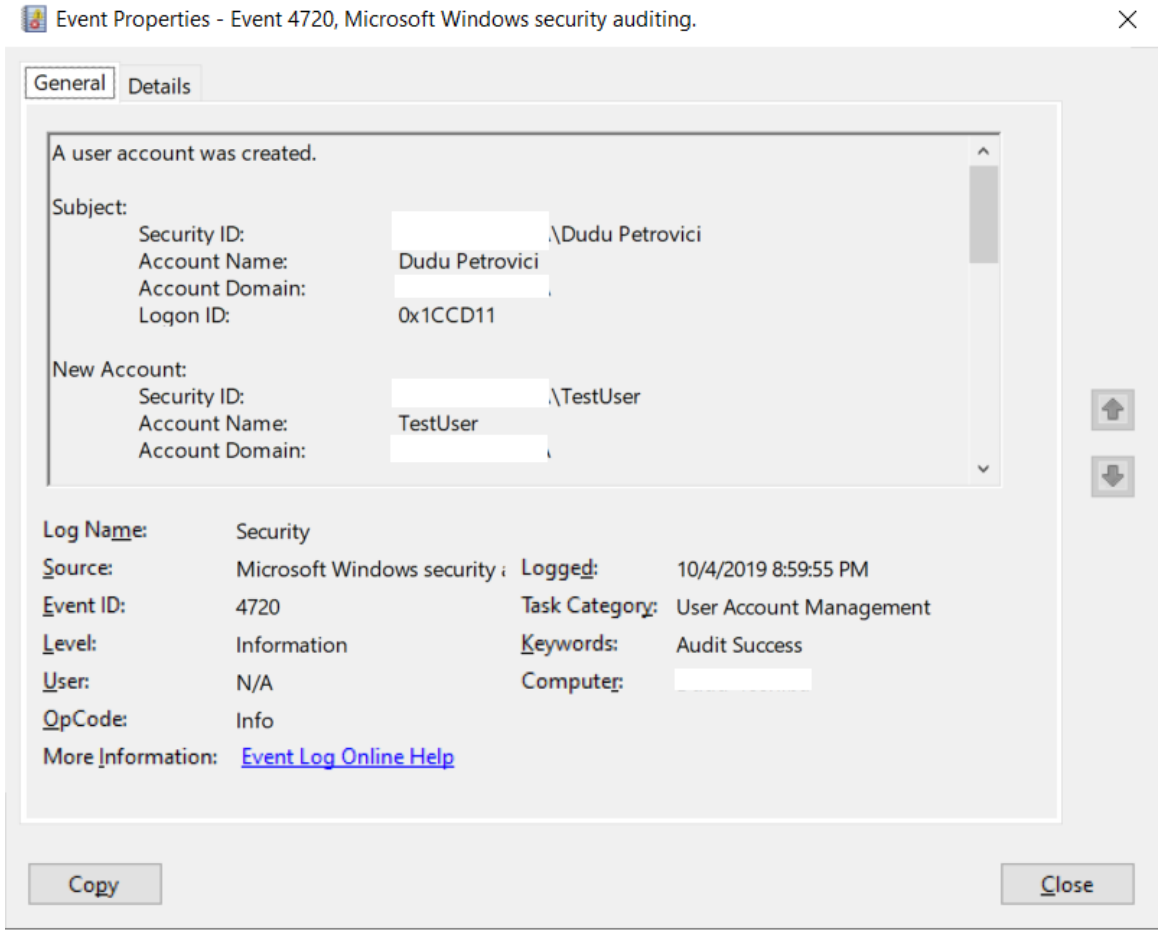


Event Logs

Event logs can be useful in a forensics examination to show that a user may or may not have performed a particular action at a specific time.

They can be useful for several things, such as:

- Tracing logs in the case of logging into a restricted network, proving the computer was running during a particular time;
- Proving the computer was running during a particular time;
- Showing time change events;
- USB driver installation;
- Wireless connections;
- Identify rogue accounts created by threat actors;



Event Logs

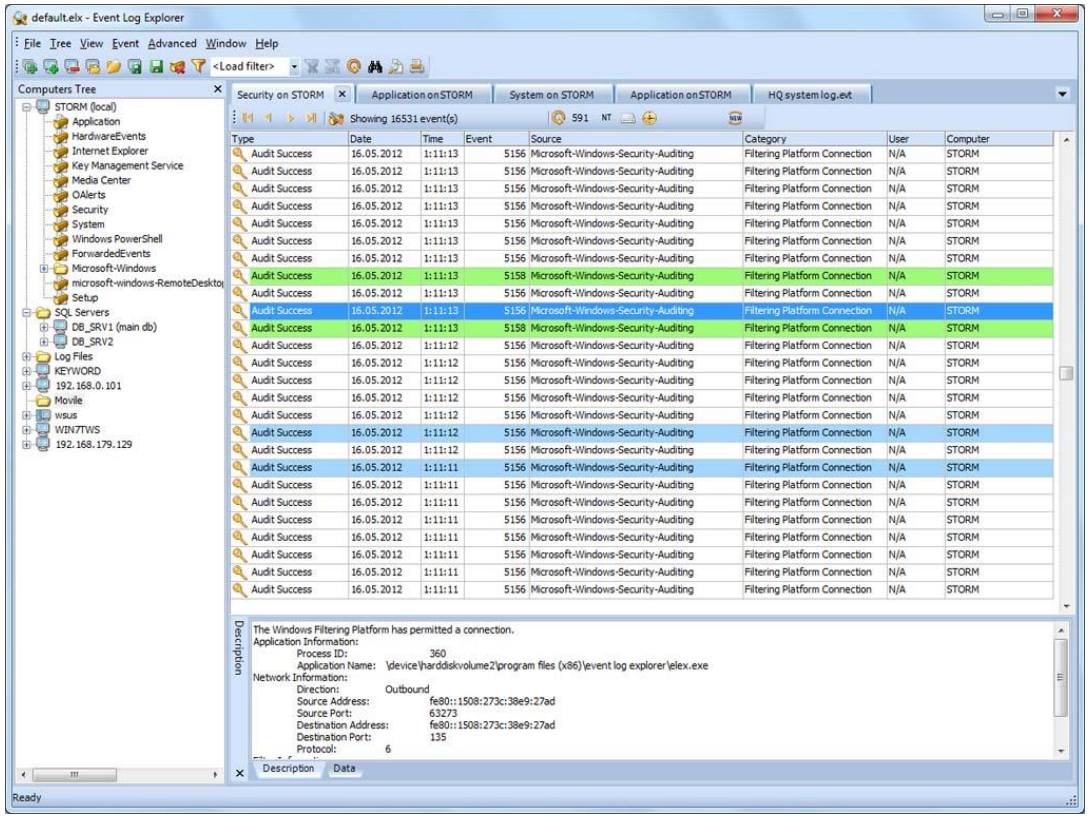
Most common Windows Event IDs of interest:

- 4626 An account successfully logged on
- 4625 Account failed to log in
- 4634 / 4647 Successful logoff
- 4648 Logon using explicit credentials (Runas)
- 4672 Special privileges assigned to new logon (Administrator)
- 4720 An account was created

A detailed description for a wide range of Event IDs can be found at

https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/

A useful tool for aggregating, searching and correlating various event logs is Event Log Explorer (https://eventlogxp.com)



Windows Artifacts

Physical Location

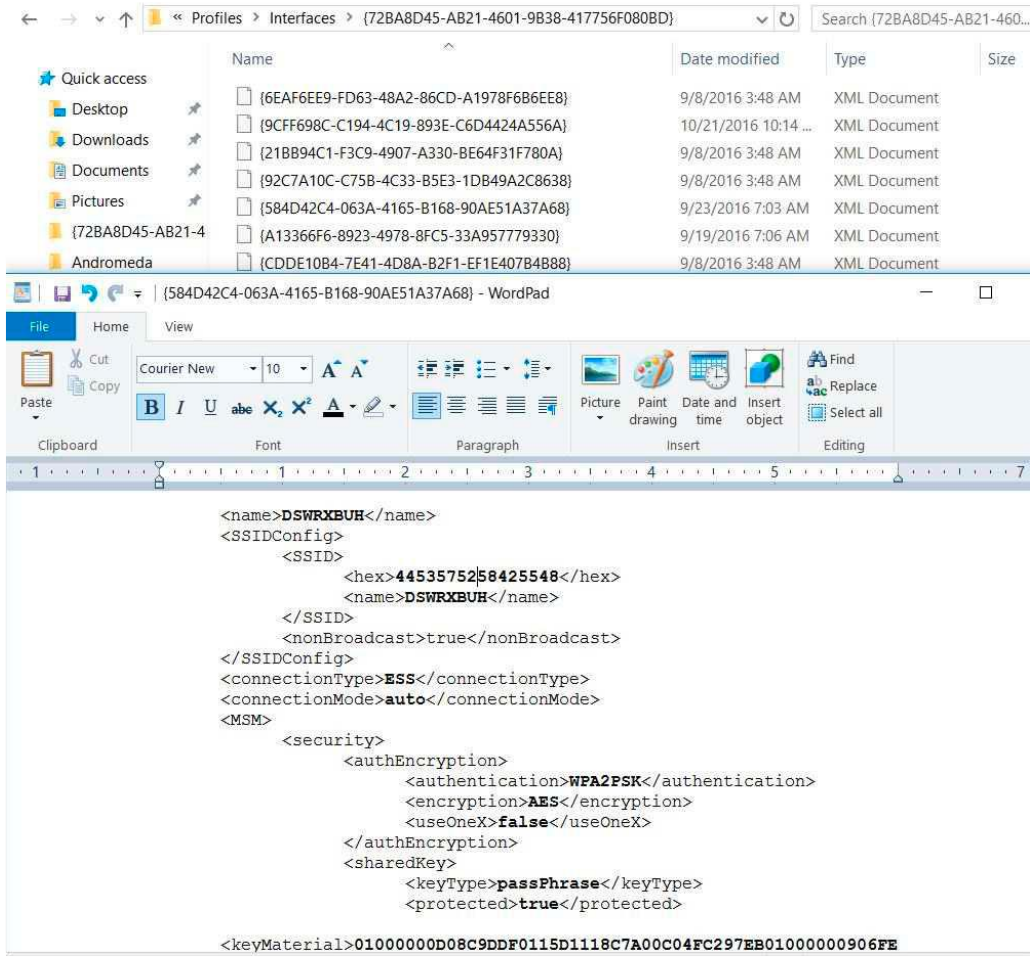
Wireless Network History

When a Windows system is connected to a wireless (WiFi) network, a record is kept in C:\ProgramData\Microsoft\Wlansvc\Profiles\Interfaces

A series of sub-folders will be created for each interface; the folder name is in GUID format and contains files named %GUID%.xml

The XML file will include the Service Set Identifier (SSID) of the network the device connected to. The *.xml extension is used by Extensible Markup Language (XML), a programming language that is readable by both humans and computers.

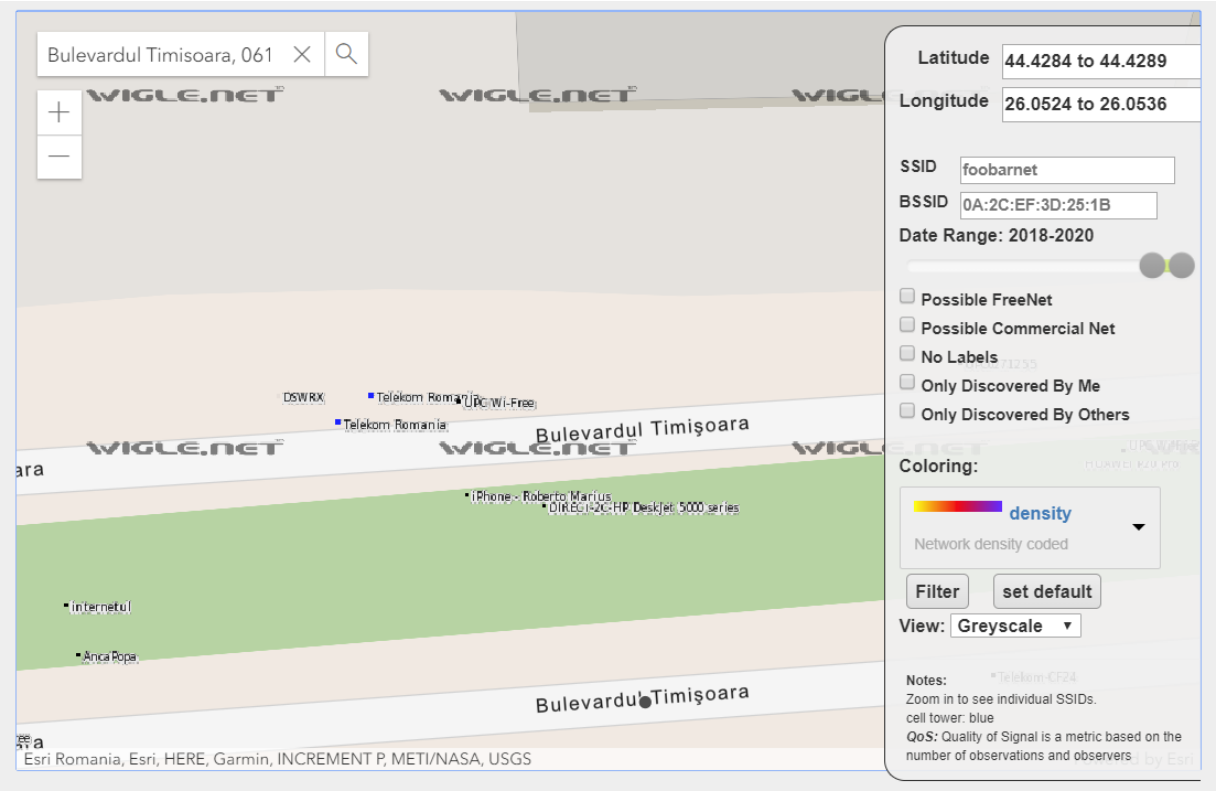
Testing has shown that files relating to previously connected wireless network may be deleted



Physical Location

Wireless Network History

Using a public service such as https://wigle.net, the SSID can be used to pinpoint the geo-location of the access point used for that connection



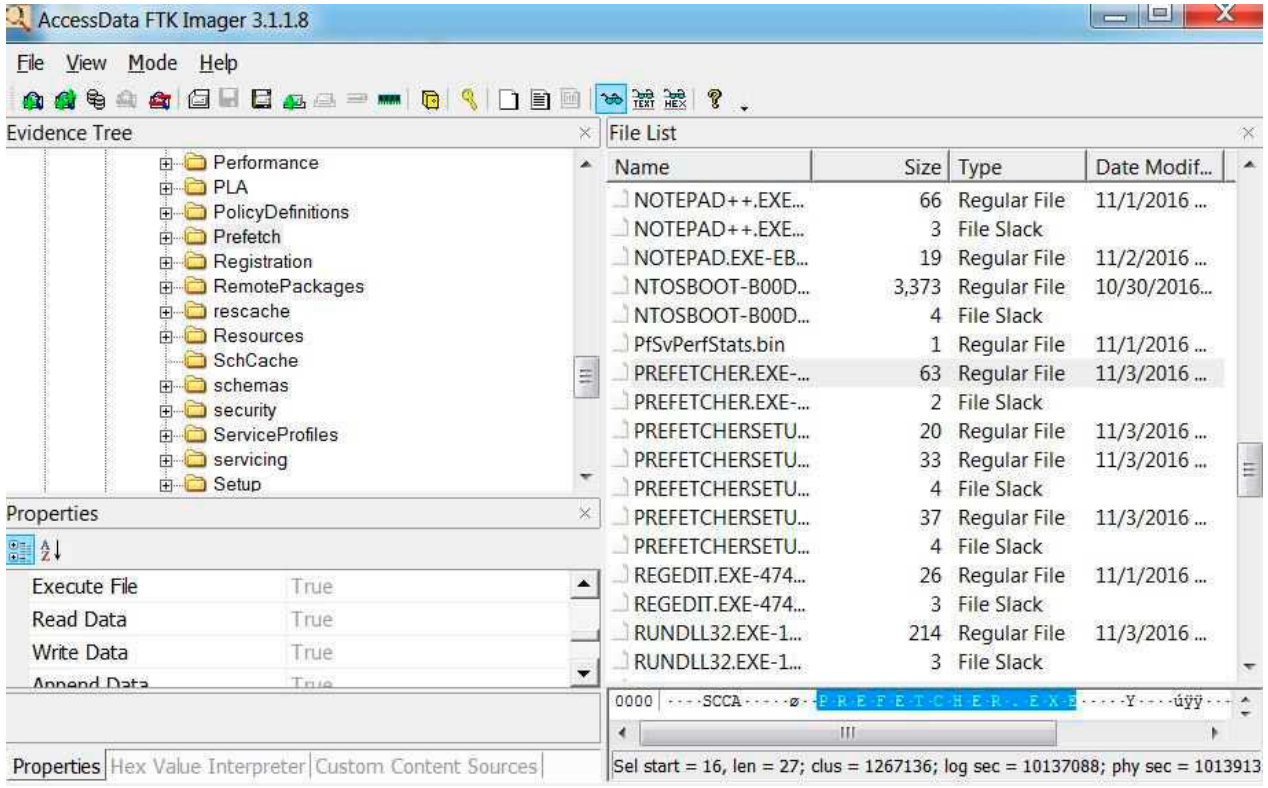
Program Execution

Prefetch / SuperFetch

Prefetching speeds up computer performance by bringing the data and code pages of programs used during boot process and in subsequent program launches into memory from the disk before that data and code is actually demanded.

The prefetch files that are created as a result of the tracing process that occurs are located in %WINDOWS%\Prefetch.

The file's name is the name of the application to which the trace applies followed by a dash and the hexadecimal representation of a hash of the file's path ending with a .PF file extension.
The prefetch folder will never grow larger than 129 entries.

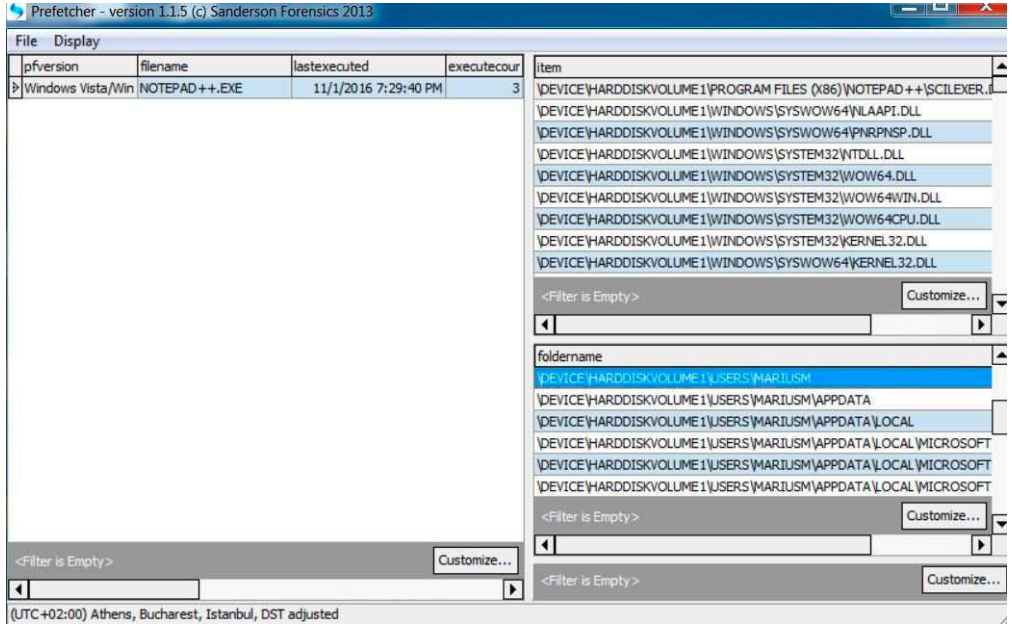


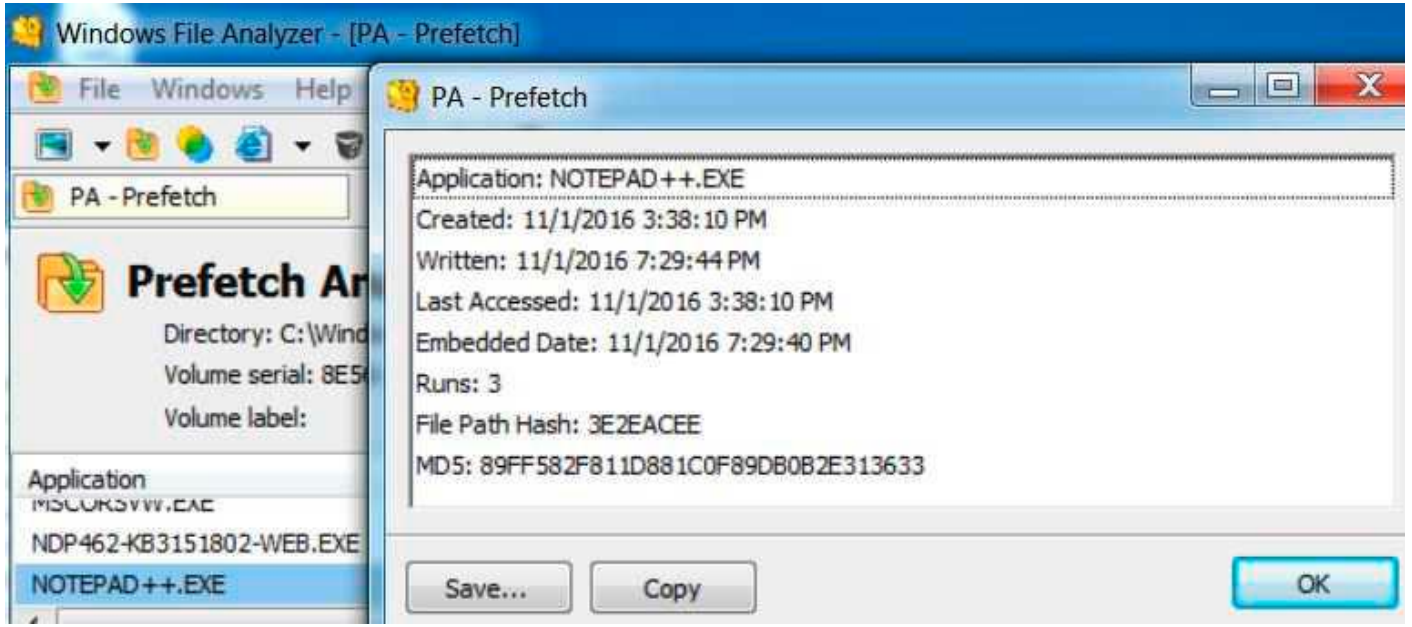
Looking at the actual content of the .PF file, the name of the executable file being traced is located at offset 10h an is visible in plaintext. The file will also contain the run count, last run date and a list of files used by the application when it loads.

Program Execution

Prefetch / SuperFetch

SuperFetch was introduced in Windows Vista. It adds additional functionality by keeping track of when and how often a program is run.
In a forensic examination, prefetch files can be used to help determine when an application was last run. This is useful for creating a timeline of events or if attempting to determine if a virus or other exploit is active on a computer.
Examining the files and directories accessed during the launch of an application can reveal hidden directories, point to user accounts or show that an application was accessed from an external storage drive.





Forensics and the Windows Registry

Windows registry is a central hierarchical database in which Microsoft Windows stores information that is necessary to configure the system for one or more users, applications, and hardware devices (profiles for each user, applications that are installed on the computer, types of documents that each can create, property sheet settings for folders and application icons, hardware that exists on the system, and the ports that are being used etc.)

The highest element of the hierarchy is known as a **hive**, which maps to one or more files in the file system that contains a binary database of registry keys and values. Hives are designed to store specific types of information.

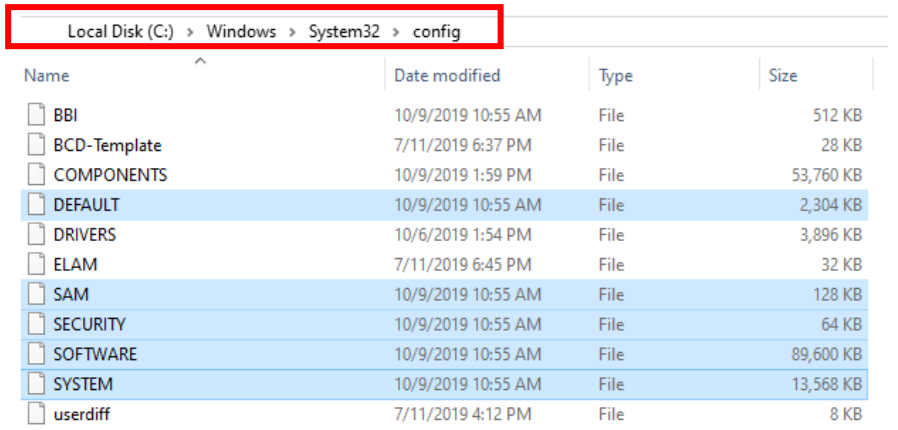
The registry contains useful information about the computer system and its users:

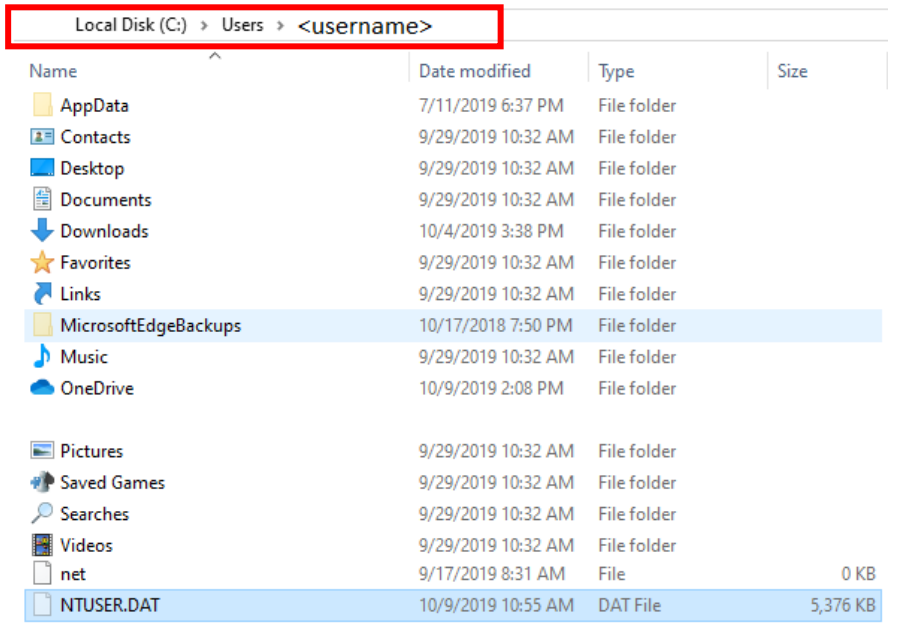
- System Settings
- Hardware Information
- USB Devices
- User Names and Security Identifiers (SID)
- Personal User Settings
- Browser Preferences
- Web Browsing Activity
- Recently Opened Items
- Programs Execution
- Windows Firewall

Registry Hive Files

The registry is built upon a series of hive files.
A registry hive is a group of keys, subkeys, and values in the registry that has a set of supporting files.

Registry Hive	Supporting Files
HKEY_CURRENT_CONFIG	**System**, System.alt, System.log, System.sav
HKEY_CURRENT_USER	**Ntuser.dat**, Ntuser.dat.log
HKEY_LOCAL_MACHINE\SAM	**Sam**, Sam.log, Sam.sav
HKEY_LOCAL_MACHINE\Security	**Security**, Security.log, Security.sav
HKEY_LOCAL_MACHINE\Software	**Software**, Software.log, Software.sav
HKEY_LOCAL_MACHINE\System	**System**, System.alt, System.log, System.sav
HKEY_USERS\.DEFAULT	Default, Default.log, Default.sav



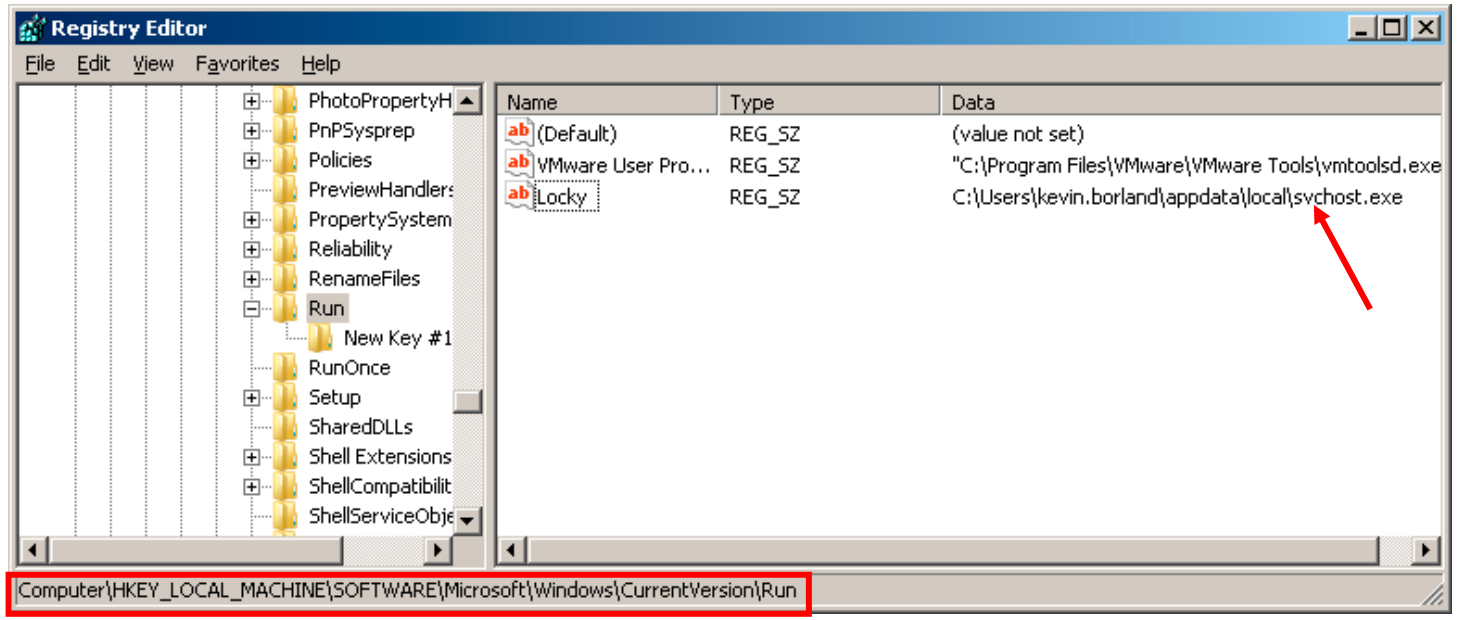


Registry Autorun Locations

Often referred to as ***Autoruns*** or ***Autostart locations***, these registry entries are used to automatically start programs, either when the system boots or a user logs in.

- Indicates executables locations
- Last written time can indicate when a potential infection occurred

Can be used by both legitimate applications and by malware such as trojans, viruses, worms, spyware or adware.



Registry Autorun Locations

Autoruns from Sysinternals (https://docs.microsoft.com/en-us/sysinternals/downloads/autoruns) is a great tool that shows you what programs are configured to run during system bootup or user login.

File hashes can be automatically submitted to VirusTotal to determine if a program is malicious or bening.