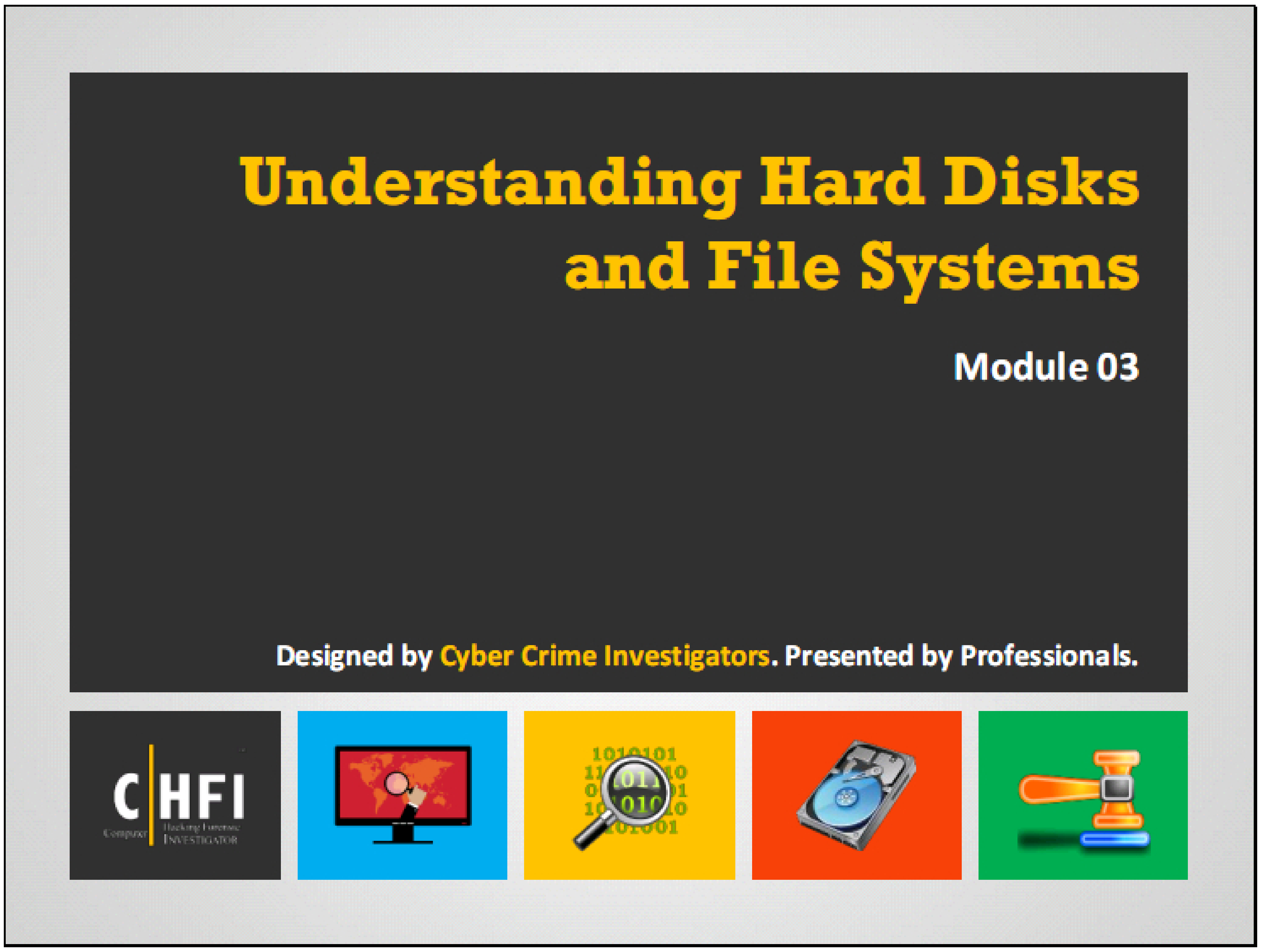
Understanding Hard Disks and File Systems

Module 03



Understanding Hard Disks and File Systems

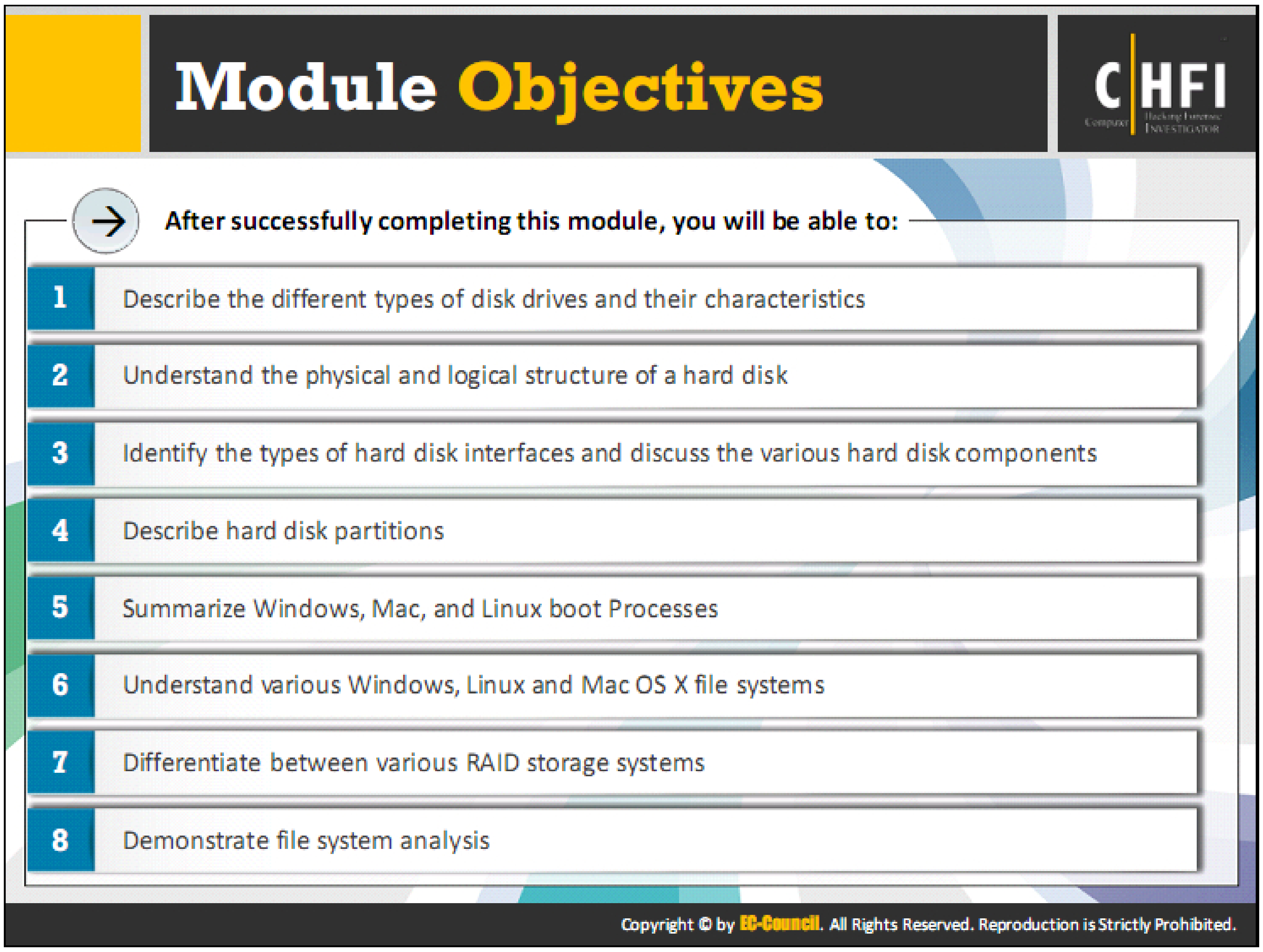
Module 03

Designed by Cyber Crime Investigators. Presented by Professionals.

Computer Hacking Forensic Investigator v9

Module 03: Understanding Hard Disks and File Systems

Exam 312-49

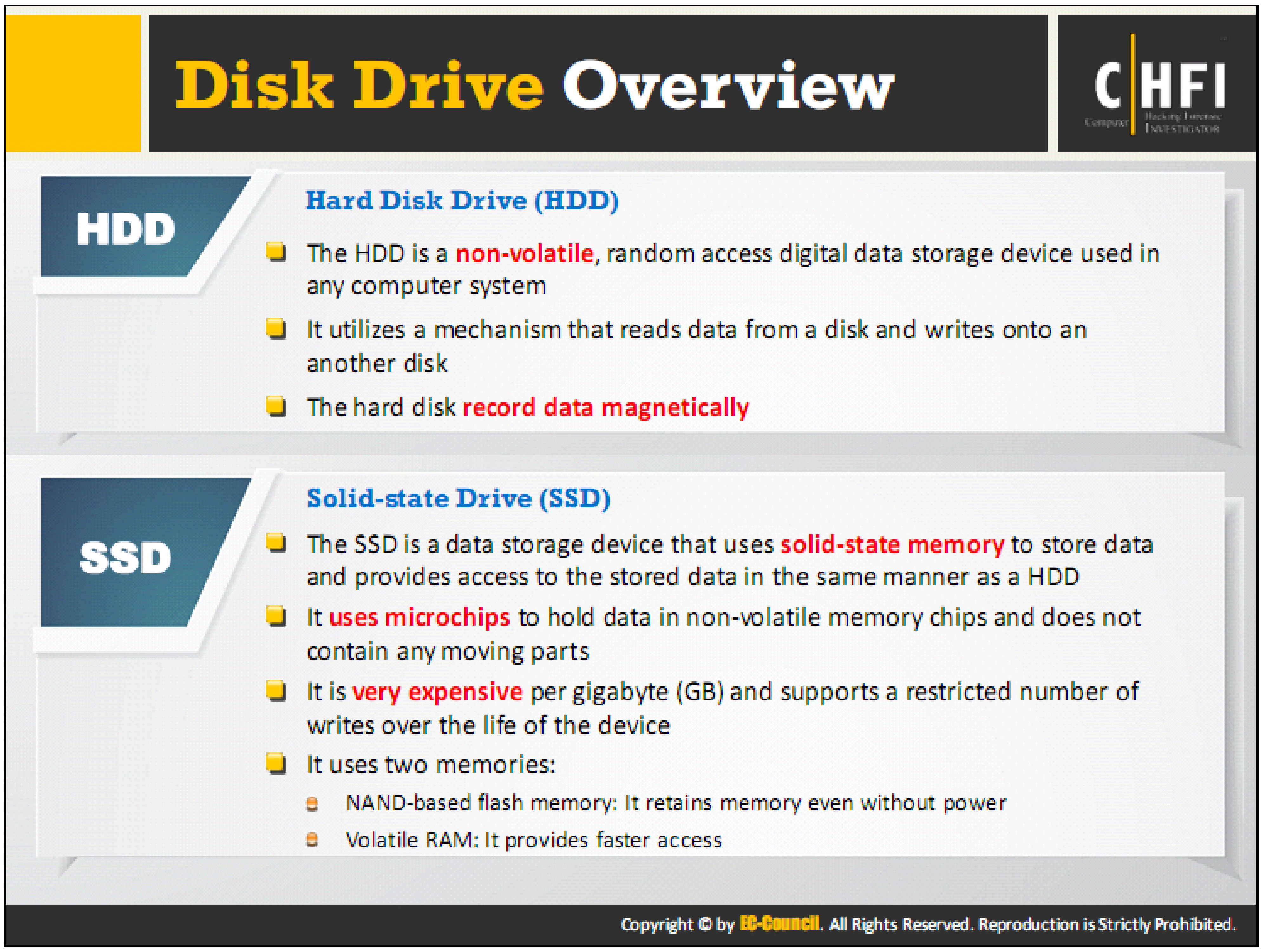


Module Objectives

After successfully completing this module, you will be able to:

1. Describe the different types of disk drives and their characteristics
2. Understand the physical and logical structure of a hard disk
3. Identify the types of hard disk interfaces and discuss the various hard disk components
4. Describe hard disk partitions
5. Summarize Windows, Mac, and Linux boot Processes
6. Understand various Windows, Linux and Mac OS X file systems
7. Differentiate between various RAID storage systems
8. Demonstrate file system analysis

The hard disk is an important source of the information for the investigator. Therefore, an investigator should know the structure and behavior of the hard disk. The investigator should locate and protect the data collected from the hard disk as the evidence. Hence, the investigator should know all the necessary information about working principle of the hard disk. The file system is also important as the storage and distribution of the data in the hard disk is dependent on the file system used.



Disk Drive Overview

HDD

Hard Disk Drive (HDD)

- The HDD is a non-volatile, random access digital data storage device used in any computer system
- It utilizes a mechanism that reads data from a disk and writes onto an another disk
- The hard disk record data magnetically

SSD

Solid-state Drive (SSD)

- The SSD is a data storage device that uses solid-state memory to store data and provides access to the stored data in the same manner as a HDD
- It uses microchips to hold data in non-volatile memory chips and does not contain any moving parts
- It is very expensive per gigabyte (GB) and supports a restricted number of writes over the life of the device
- It uses two memories:
  - NAND-based flash memory: It retains memory even without power
  - Volatile RAM: It provides faster access

Disk Drive is a digital data storage device that uses different storage mechanisms such as mechanical, electronic, magnetic, and optical to store the data. It is addressable and rewritable to support changes and modification of data. Depending on the type of media and mechanism of reading and writing the data, the different types of disk drives are as follows:

- **Magnetic Storage Devices:** Magnetic storage devices store data using magnets to read and write the data by manipulating magnetic fields on the storage medium. These are mechanical devices with components moving to store or read the data. Few other examples include floppy disks, magnetic tapes, etc.

  In these types of hard disks, the disks inside the media rotate at high speed and heads in the disk drive read and write the data.

- **Optical Storage Devices:** Optical storage devices are electronic storage media that store and read the data in the form of binary values using a laser beam. The devices use lights of different densities to store and read the data. Examples of optical storage devices include Blue-Ray discs, CDs, and DVDs.

- **Flash Memory Devices:** Flash memory is a non-volatile electronically erasable and reprogrammable storage medium that is capable of retaining data even in the absence of power. It is a type of electronically erasable programmable read only memory (EEPROM). These devices are cheap and more efficient compared to other storage devices. Devices that use flash memory for data storage are USB flash drives, MP3 players, digital cameras, solid-state drives, etc. Few examples of flash memory are:

- BIOS chip in a computer
- Compact Flash (commonly found in digital cameras)
- Smart Media (commonly found in digital cameras)
- Memory Stick (commonly found in digital cameras)
- PCMCIA Type I and Type II memory cards found in laptops
- Memory cards for video game consoles

Hard Disk Drive (HDD)

Hard Disk Drive is a non-volatile, random access digital data storage device used in any computer system. The hard disk stores data in a method similar to that of a file cabinet. The user, when needed, can access the data and programs. When the computer needs the stored program or data, the system brings it to a temporary location from the permanent location. When the user or system makes changes to a file, the computer saves the file by replacing the older file with the new file. The HDD records data magnetically onto the hard disk.

The hard disks differ from each other considering various measurements such as:

- Capacity of the hard disk
- Interface used
- Speed in rotations per minute
- Seek time
- Access time
- Transfer time

Solid-State Drive (SSD)

A Solid-State Drive (SSD) is an electronic data storage device that implements solid-state memory technology to store data similar to a hard disk drive. Solid-state is an electrical term that refers to an electronic circuit entirely built with semiconductors.

It uses two memories:

- **NAND-based SSDs**: These SSDs use solid state memory NAND microchips to store the data. Data in these microchips is in a non-volatile state and does not need any moving parts. NAND memory is non-volatile in nature and retains memory even without power.

  NAND memory was developed primarily to reduce per bit cost of data storage. However, it is still more expensive than optical memory and HDDs. NAND-based memory is widely used today in mobile devices, digital cameras, MP3 players, etc. It has a finite number of writes over the life of the device.

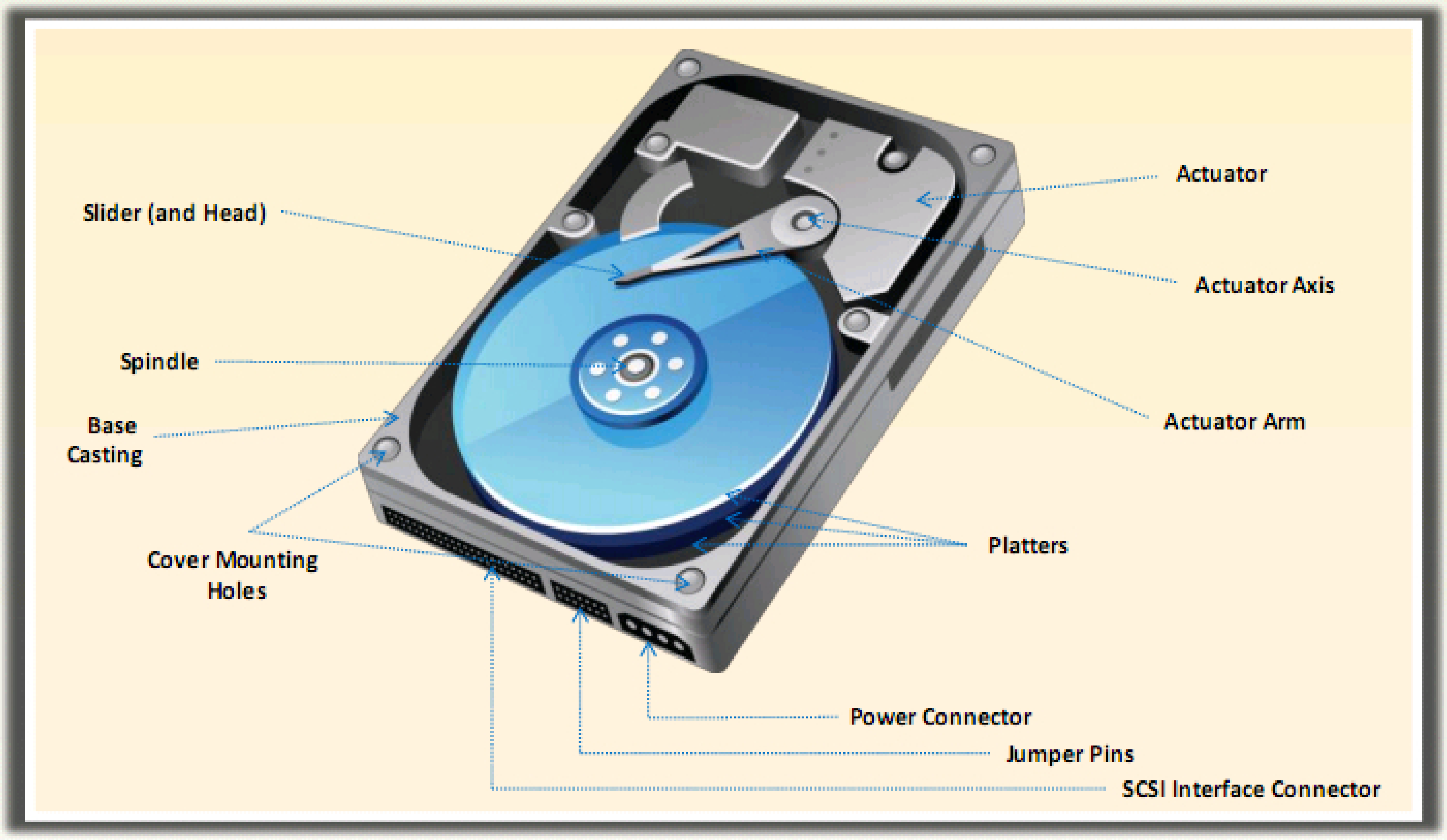
- **Volatile RAM-based SSDs**: SSDs, based on volatile RAM such as DRAM, are used when applications require faster data access. These SSDs include either an internal chargeable battery or an external AC/DC adapter, and a backup storage. Data resides in the DRAM during data access and is stored in the backup storage in case of a power failure.

Advantages of SSD

SSD has several advantages over magnetic hard drives. The three major advantages of SSD are:

- Faster data access
- Less power usage
- Higher reliability

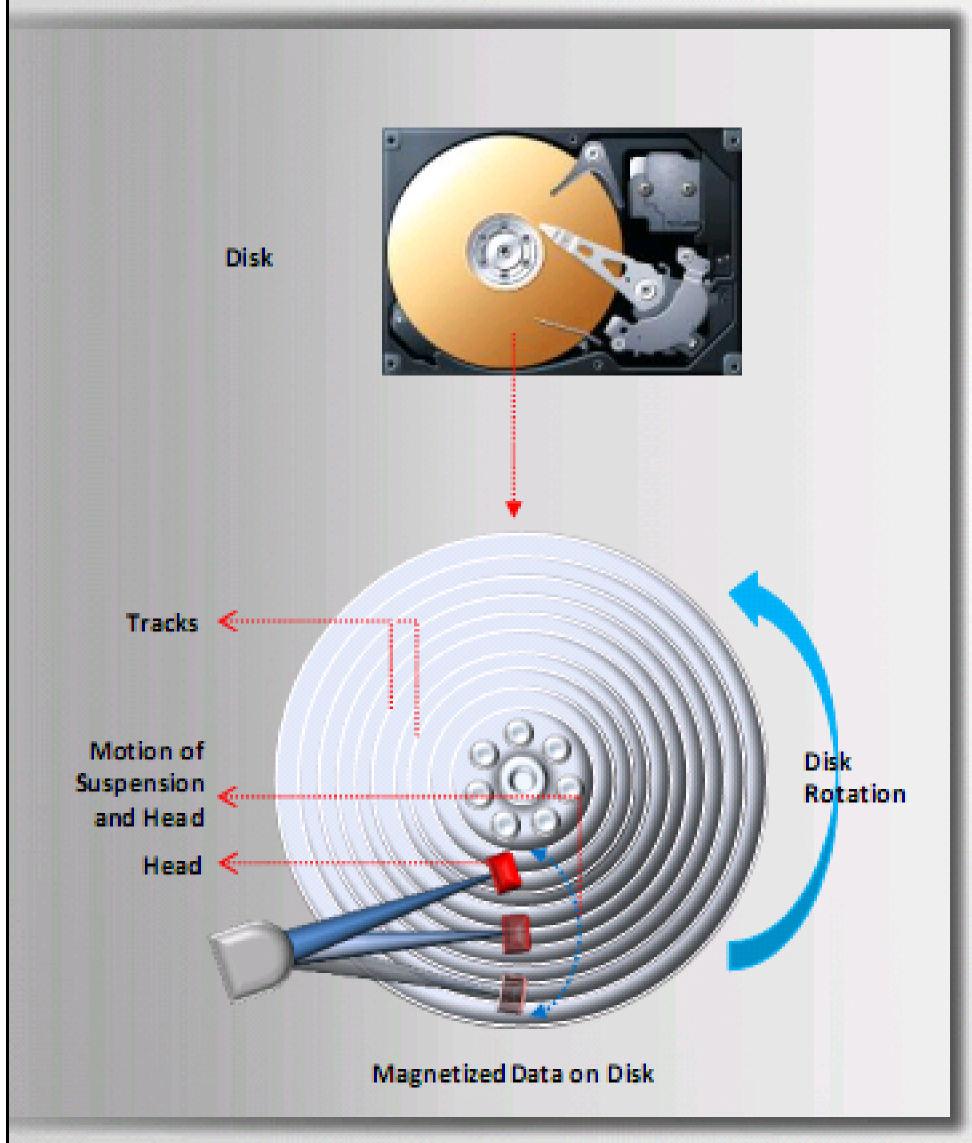
Physical Structure of a Hard Disk

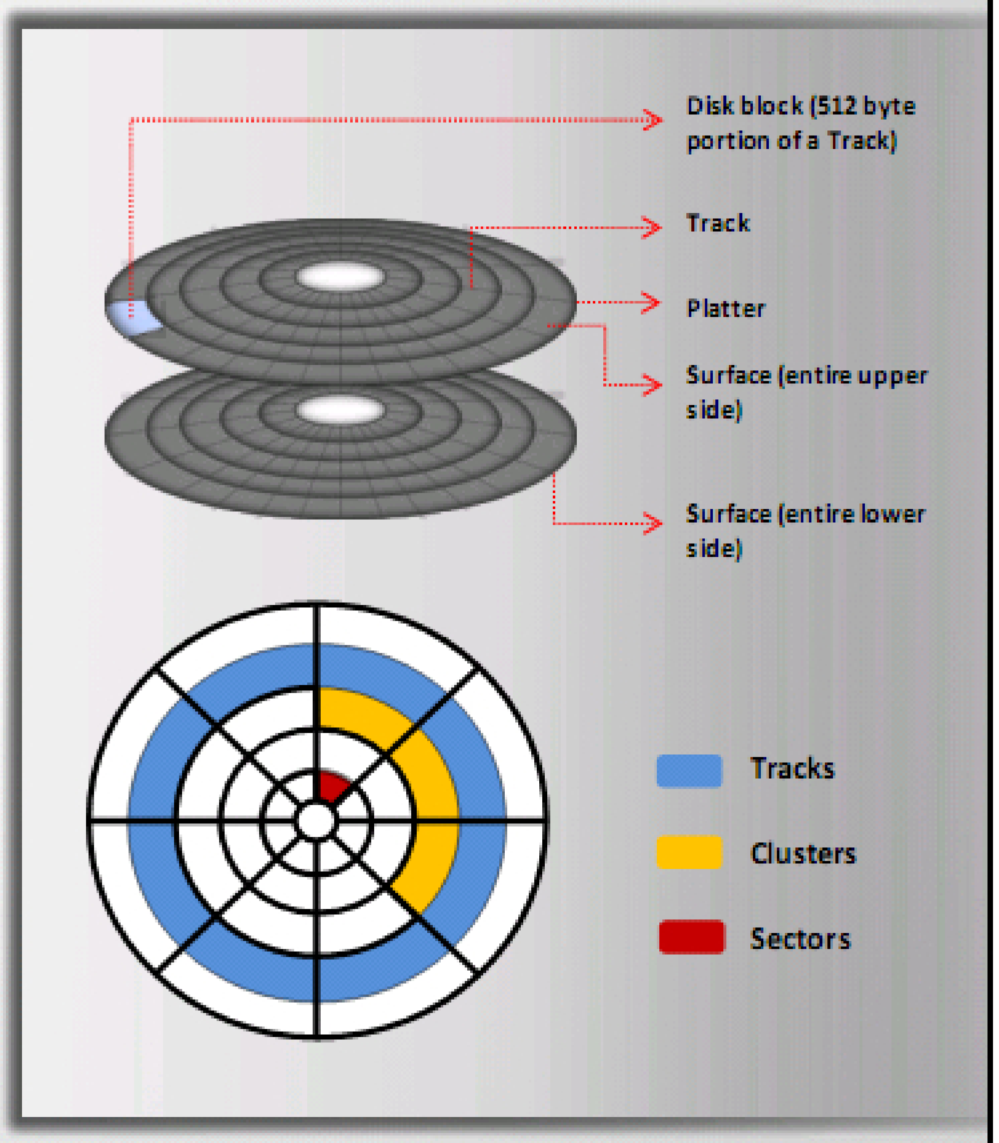


The main components of hard disk drive are:

- **Platters**: These are disk like structures present on the hard disk, stacked one above the other and store the data
- **Head**: It is a device present on the arm of the hard drive that reads or writes data on the magnetic platters, mounted on the surface of the drive
- **Spindle**: It is the spinning shaft on which holds the platters in a fixed position such that it is feasible for the read/write arms to get the data on the disks
- **Actuator**: It is a device, consisting of the read-write head that moves over the hard disk con to save or retrieve information
- **Cylinder** : These are the circular tracks present on the platters of the disk drive at equal distances from the center

Physical Structure of a Hard Disk (Cont'd)




Disk block (512 byte portion of a Track)
Track
Platter
Surface (entire upper side)
Surface (entire lower side)
Tracks
Clusters
Sectors

A hard disk contains a stack of platters, circular metal disks that are mounted inside the hard disk drive and coated with magnetic material, sealed in a metal case or unit. Fixed in a horizontal or vertical position, the hard disk has electromagnetic read or write heads above and below the platters. The surface of the disk consists of a number of concentric rings called as tracks; each of these tracks has smaller partitions called disk blocks. The size of each disk block is 512 bytes (0.5 KB). The track numbering starts with zero. When the platter rotates, the heads record data in tracks. A 3.5-inch hard disk can contain about thousand tracks.

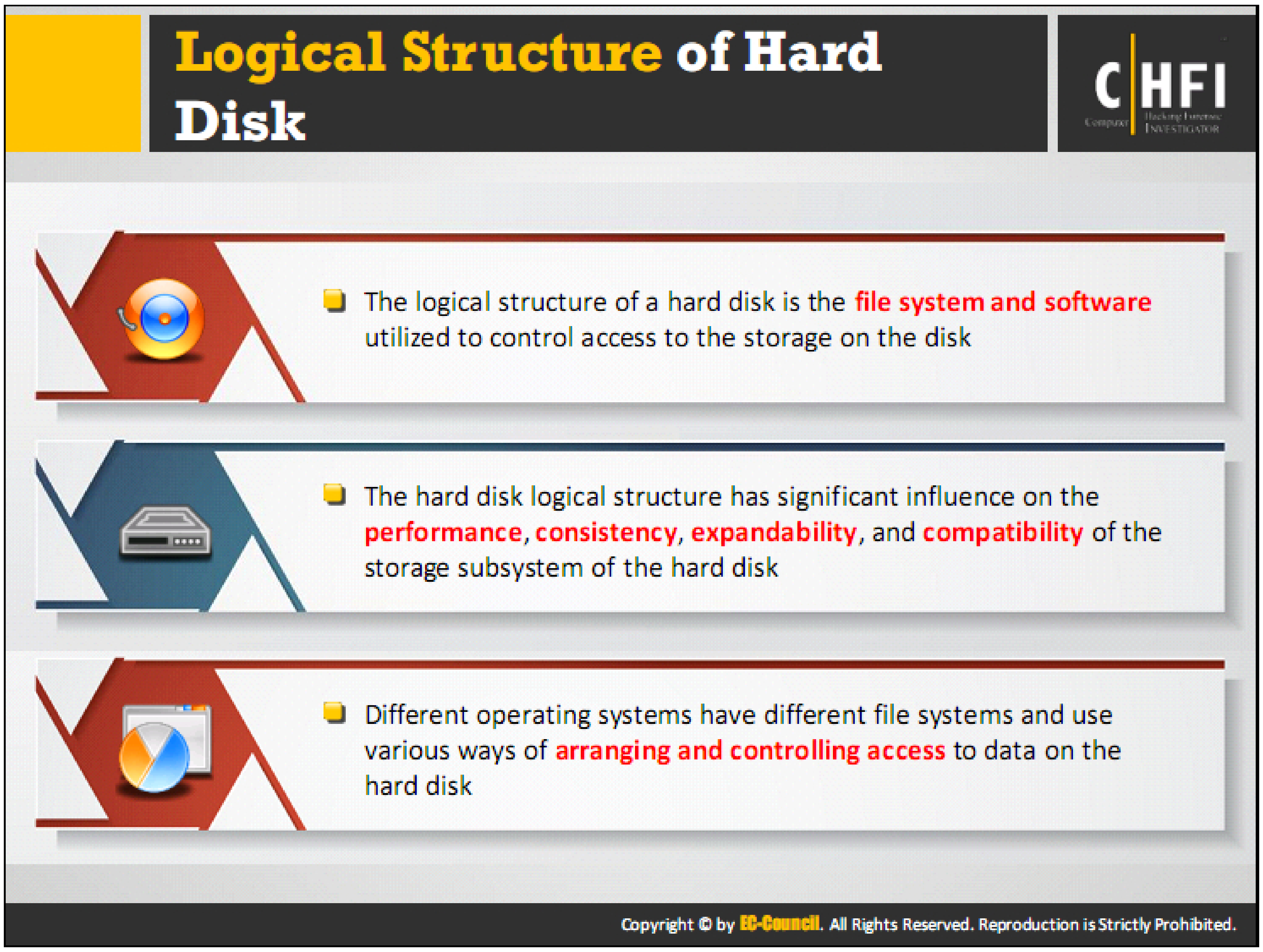
The spindle holds the platters in a fixed position such that it is feasible for the read/write arms to get the data on the disks. These platters rotate at a constant speed while the drive head, positioned close to the center of the disk, reads the data slowly from the surface of the disk compared to the outer edges of the disk. To maintain integrity of data, the head is reading at a particular period of time from any drive head position. The tracks at the outer edges of the disk have less densely populated sectors compared to the tracks close to the center of the disk.

The disk fills the space based on a standard plan. One side of the first platter contains space, reserved for hardware track-positioning information which is not available to the operating system. The disk controller uses the track-positioning information to place the drive heads in the correct sector position.

The hard disk records the data using the zoned bit recording technique, also known as multiple zone recording. This method combines the areas on the hard disk together as zones, depending on the distance from the center of the disk. A zone contains a certain number of sectors per track.

Calculation of data density of disk drives is done in the following terms:

- Track density: Refers to the number of tracks in a hard disk
- Area density: Area density is the platters' storage capacity in bits per square inch
- Bit density: It is bits per unit length of track



Logical Structure of Hard Disk

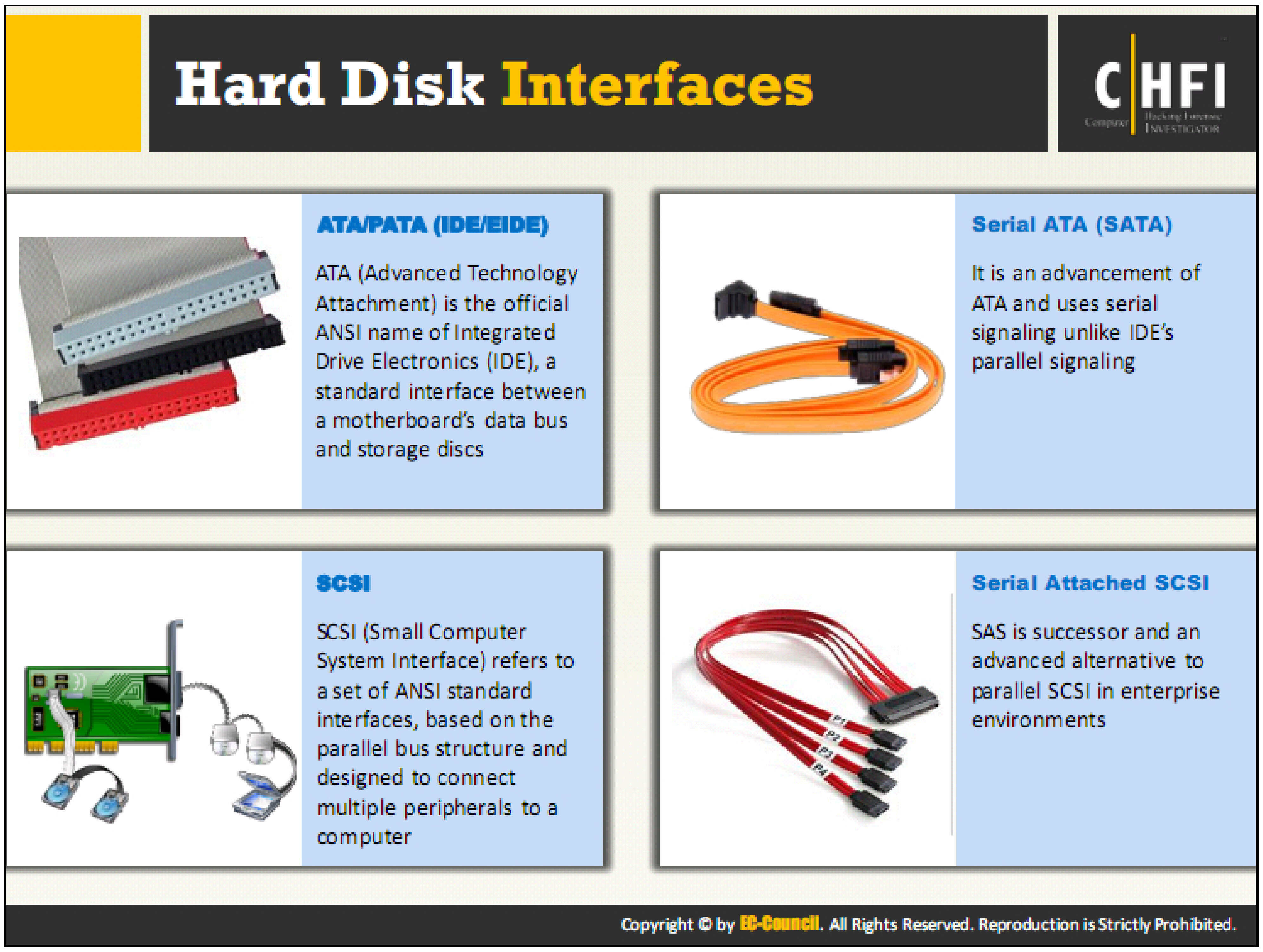
- The logical structure of a hard disk is the **file system and software** utilized to control access to the storage on the disk
- The hard disk logical structure has significant influence on the **performance**, **consistency**, **expandability**, and **compatibility** of the storage subsystem of the hard disk
- Different operating systems have different file systems and use various ways of **arranging and controlling access** to data on the hard disk

A hard disk's logical structure mainly depends on the file systems used and the software that defines the process of accessing data from the disk. Operating systems use different types of file systems, and those file systems use various other types of controlling and accessing mechanisms for data on the hard disk. Operating systems organize the same hard disk in many different ways.

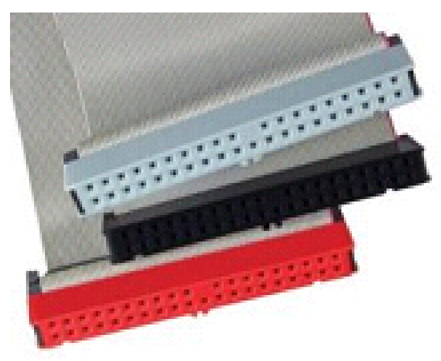
The logical structure of the hard disk directly influences the consistency, performance, compatibility, and expandability of the storage subsystems of the hard disk. The logical structure depends on the type of operating system and file system used, because these factors organize and control the data access on the hard disk.

The most common computer file systems are:

- FAT
- FAT32
- NTFS
- EXT
- EXT2 and 3
- EFS



Hard Disk Interfaces

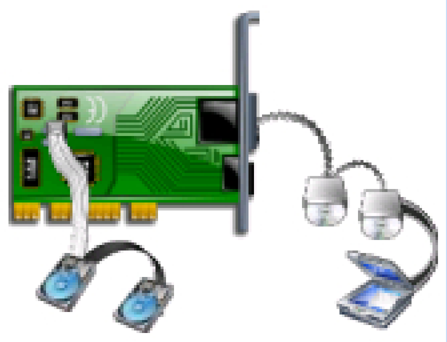


ATA/PATA (IDE/EIDE)

ATA (Advanced Technology Attachment) is the official ANSI name of Integrated Drive Electronics (IDE), a standard interface between a motherboard's data bus and storage discs

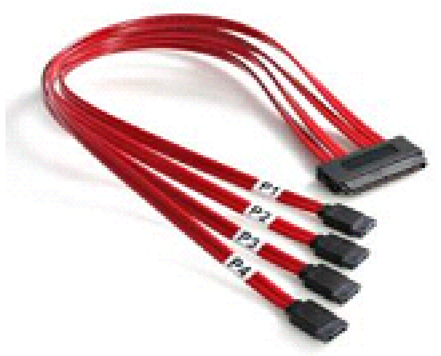
Serial ATA (SATA)

It is an advancement of ATA and uses serial signaling unlike IDE's parallel signaling



SCSI

SCSI (Small Computer System Interface) refers to a set of ANSI standard interfaces, based on the parallel bus structure and designed to connect multiple peripherals to a computer



Serial Attached SCSI

SAS is successor and an advanced alternative to parallel SCSI in enterprise environments

The hard disk drive connects to the PC using an interface. There are various types of interfaces: IDE, SATA, Fiber Channel, SCSI, etc.

ATA/PATA (IDE/EIDE)

IDE (Integrated Drive Electronics) is a standard electronic interface used between a computer motherboard's data paths or bus and the computer's disk storage devices, such as hard drives and CD-ROM/DVD drives. The IBM PC Industry Standard Architecture (ISA) 16-bit bus standard is base for the IDE interface, which offers connectivity in computers that use other bus standards. ATA (Advanced Technology Attachment) is the official American National Standards Institute's (ANSI) name of Integrated Drive Electronics (IDE).

Parallel ATA:

PATA, based on parallel signaling technology, offers a controller on the disk drive itself and thereby eliminates the need for a separate adaptor card. Parallel ATA standards only allow cable lengths up to 46 centimeters (18 inches).

Features of PATA:

- Relatively inexpensive
- Easy to configure
- Allows look-ahead caching

Enhanced Integrated Drive Electronics (EIDE)

Most computers sold today use an enhanced version of IDE called Enhanced Integrated Drive Electronics (EIDE). IDE drives connect with PCs, using an IDE host adapter card. The IDE controller in modern computers is a built-in feature on the motherboard itself. Enhanced IDE is an extension to the IDE interface that supports the ATA-2 and ATAPI standards.

Two types of Enhanced IDE sockets are present on the motherboard. A socket connects two drives, namely, 80 wire cables for fast hard drives and a 40-pin ribbon cable for CD-ROMs/DVD-ROMs.

Enhanced or Expanded IDE is a standard electronic interface, connecting a computer's motherboard to its storage drives. EIDE can address a hard disk bigger than 528 Mbytes and allows quick access to the hard drive as well as provides support for Direct Memory Access (DMA) and additional drives like tape devices, CD-ROM, etc. While updating the computer system with bigger hard drive, insert the EIDE controller in the system card slot.

The EIDE can access drives larger than 528 Mbytes using a 28-bit Logical Block Address (LBA) to indicate the actual head, sector, and cylinder locations of the disk data. The 28-bit Logical Block Address provides the information, which is enough to denote unique sectors for an 8.4 GB device.

Serial ATA

Serial ATA (SATA) offers a point-to-point channel between the motherboard and drive. The cables in SATA are shorter in length as compared to PATA. It uses four-wire shielded cable that can be maximum one meter in length. SATA cables are more flexible, thinner, and less massive than the ribbon cables, required for conventional PATA hard drives.

Features of SATA:

- Operates with great speed
- Easy to connect to storage devices
- Easy to configure
- Transfers data at a speed of 1.5 Gbps (SATA revision 1.0) and 6 Gbps (SATA revision 3)

Drive and motherboard connectivity through a SATA point-to-point channel is based on serial signaling technology. This technology enables data transfer of about 1.5 Gbps in a half-duplex channel mode.

SCSI

SCSI is a set of ANSI standard electronic interfaces that allow personal computers to communicate with peripheral hardware such as disk drives, tape drives, CD-ROM drives, printers, and scanners. Developed by Apple Computer and still used in the Macintosh, the present sets of SCSIs are parallel interfaces. SCSI ports continue to come as built-in feature in various personal computers today and gather supports from all major operating systems.

In addition to faster data rates, SCSI is more flexible than earlier parallel data transfer interfaces. SCSI allows up to 7 or 15 devices (depending on the bus width) to be connected to a single SCSI port in daisy-chain fashion. This allows one circuit board or card to accommodate all the peripherals, rather than having a separate card for each device, making it an ideal interface for use with portable and notebook computers. A single host adapter, in the form of a PC card, can serve as a SCSI interface for a laptop, freeing up the parallel and serial ports for use with an external modem and printer while allowing usage of other devices in addition.

Technology Name	Maximum Cable Length (meters)	Maximum Speed (MBps)	Maximum Number of Devices
SCSI-1	**6**	**5**	**8**
SCSI-2	**6**	**5-10**	**8 or 16**
Fast SCSI-2	**3**	**10-20**	**8**
Wide SCSI-2	**3**	**20**	**16**
Fast Wide SCSI-2	**3**	**20**	**16**
Ultra SCSI-3, 8-bit	**1.5**	**20**	**8**
Ultra SCSI-3, 16-bit	**1.5**	**40**	**16**
Ultra-2 SCSI	**12**	**40**	**8**
Wide Ultra-2 SCSI	**12**	**80**	**16**
Ultra-3 (Ultra160/m) SCSI	**12**	**160**	**16**

TABLE 3.1: SCSI

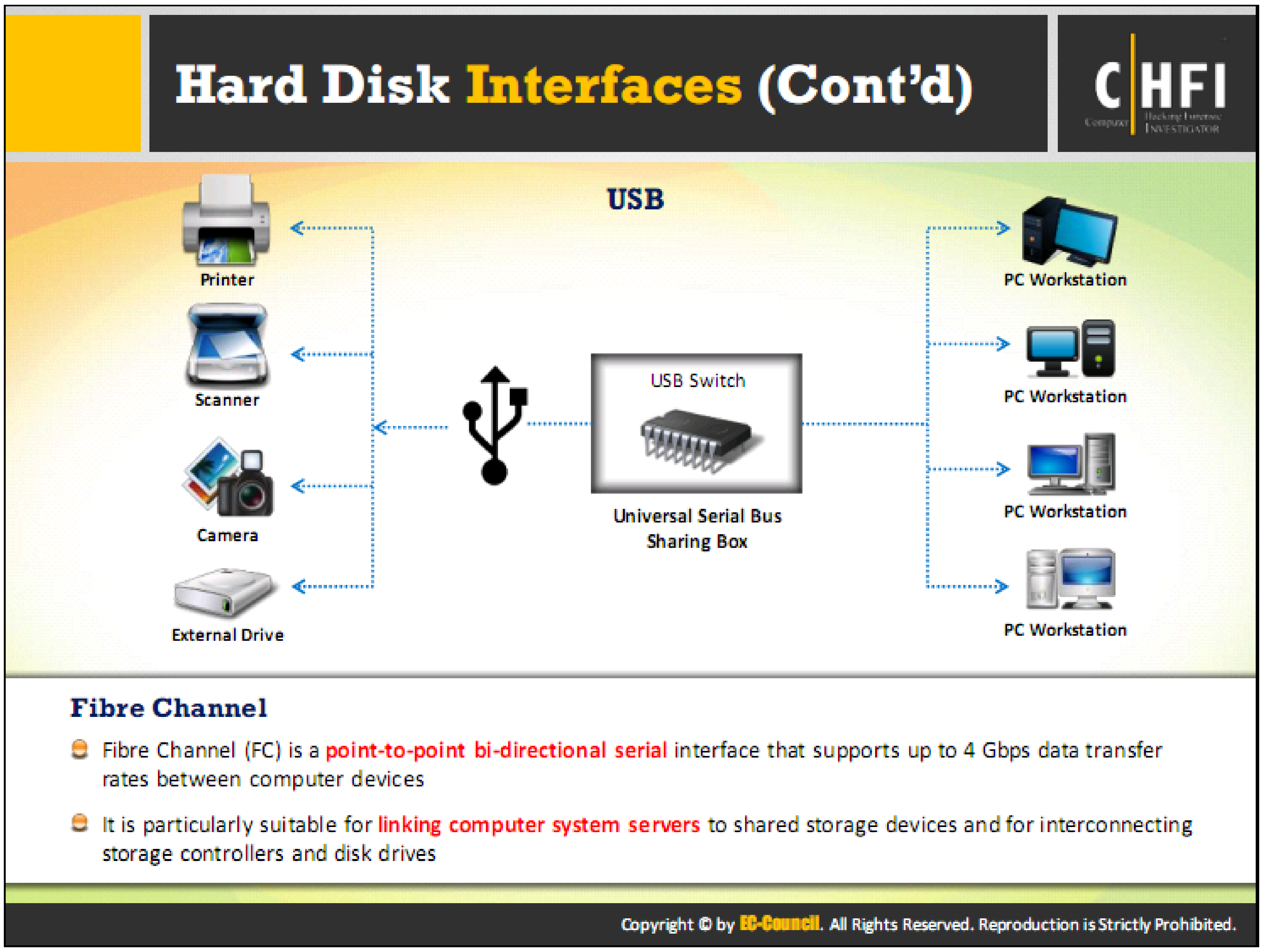
SCSI allows one circuit board or card to accommodate all the peripherals, rather than having a separate card for each device.

Serial Attached SCSI (SAS)

Serial Attached SCSI (SAS) is a point-to-point serial protocol that handles data flow among the computer storage devices such as hard drives and tape drives. It is the successor to Parallel SCSI and uses the standard SCSI command set. SAS is chosen over SCSI because of its flexibility and other beneficial features as given below:

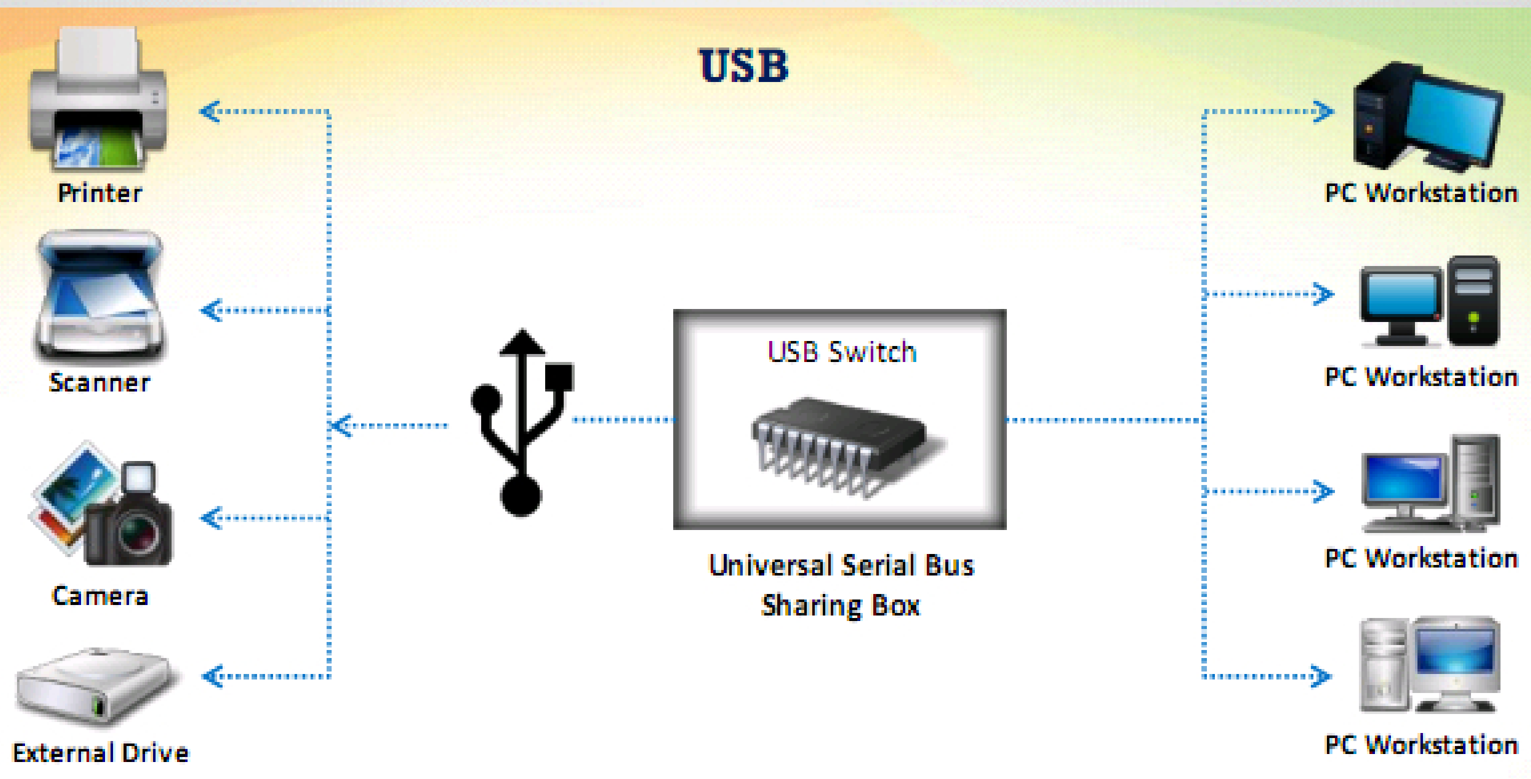
- While the latest parallel SCSI standard can support maximum of only 16 devices, SAS makes use of expanders and can support up to 65,535 devices.
- SAS is free from issues like termination and clock skew.
- SAS is a point-to-point technology, meaning the resource contention issues, which were common in parallel SCSI, do not affect it.
- SAS drives furnish better performance, scalability, and reliability in storage applications and can also operate in environments where SCSI cannot.

Source: *http://searchsecurity.techtarget.com*



Hard Disk Interfaces (Cont'd)

USB



Fibre Channel

- Fibre Channel (FC) is a point-to-point bi-directional serial interface that supports up to 4 Gbps data transfer rates between computer devices
- It is particularly suitable for linking computer system servers to shared storage devices and for interconnecting storage controllers and disk drives

USB is a "plug-and-play" interface, which allows users to add a device without an adapter card and without rebooting the computer. Universal Serial Bus (USB), developed by Intel, was first released in 1995 with a maximum speed support of 12 Mbps. Currently available USB supports data transfer speeds up to 5 Gbps. USB allows external peripheral devices such as disks, modems, printers, digitizers, and data gloves to connect to the computer.

The USB design architecture is asymmetrical that comprise a host, many USB ports, and many peripheral devices. Communication through USB device is mainly through pipes or logical channels, which are connections between the host controller and a logical entity called endpoint. USB cable length ranges from about 3 feet to over 16 feet. The maximum length being 16 feet 5 inches for high speed devices and 9 feet 10 inches for low speed devices.

Features of USB:

- Easy to use
- Provides expandability
- Provides speed for the end user
- Has high performance and ubiquity
- Allows easy connection of peripherals outside the PC
- Most operating systems configure USB-enabled devices automatically
- Useful in PC telephony and video conferencing

Hard Disk Interfaces: Fibre Channel

Fibre Channel is a point-to-point bi-directional high-speed network interface, which supports data transfer rates of up to 16-gigabit per second. It connects shared storage devices, computer system servers, disk drives, and storage controllers. Developed by the American National Standards Institute (ANSI), the fibre channel has three major topologies:

- **Point-to-point (FC-P2P):** In point to point topology, the fibre directly connects two devices with each other. This topology is simple and has limited connectivity.
- **Arbitrated loop (FC-AL):** In this topology, the connections between all devices form a loop or ring. Addition or removal of devices from the loop interrupts all the activities on the loop. Even if one device fails it causes a break in the topology. There are Fibre Channel hubs to connect many devices and can bypass the failed ports.
- **Switched fabric (FC-SW):** In this design, the fibre connects all the devices or loops of devices to fibre channel switches.

  Advantages:

  - The state of the fabric is handled by switches, which provide optimized interconnections
  - The traffic between two ports passes via only the switches and is not transmitted to any other port
  - Even if a port fails it will not affect the operation of other ports
  - Several pairs of ports can communicate at a time in a fabric.

The communication process by using fibre-optics has the following steps:

- Creates the optical signal by using a transmitter
- Relays the signal along the fibre
- Makes sure that the signal is not distorted or weak
- Receives the optical signal
- Finally converts it into an electrical signal

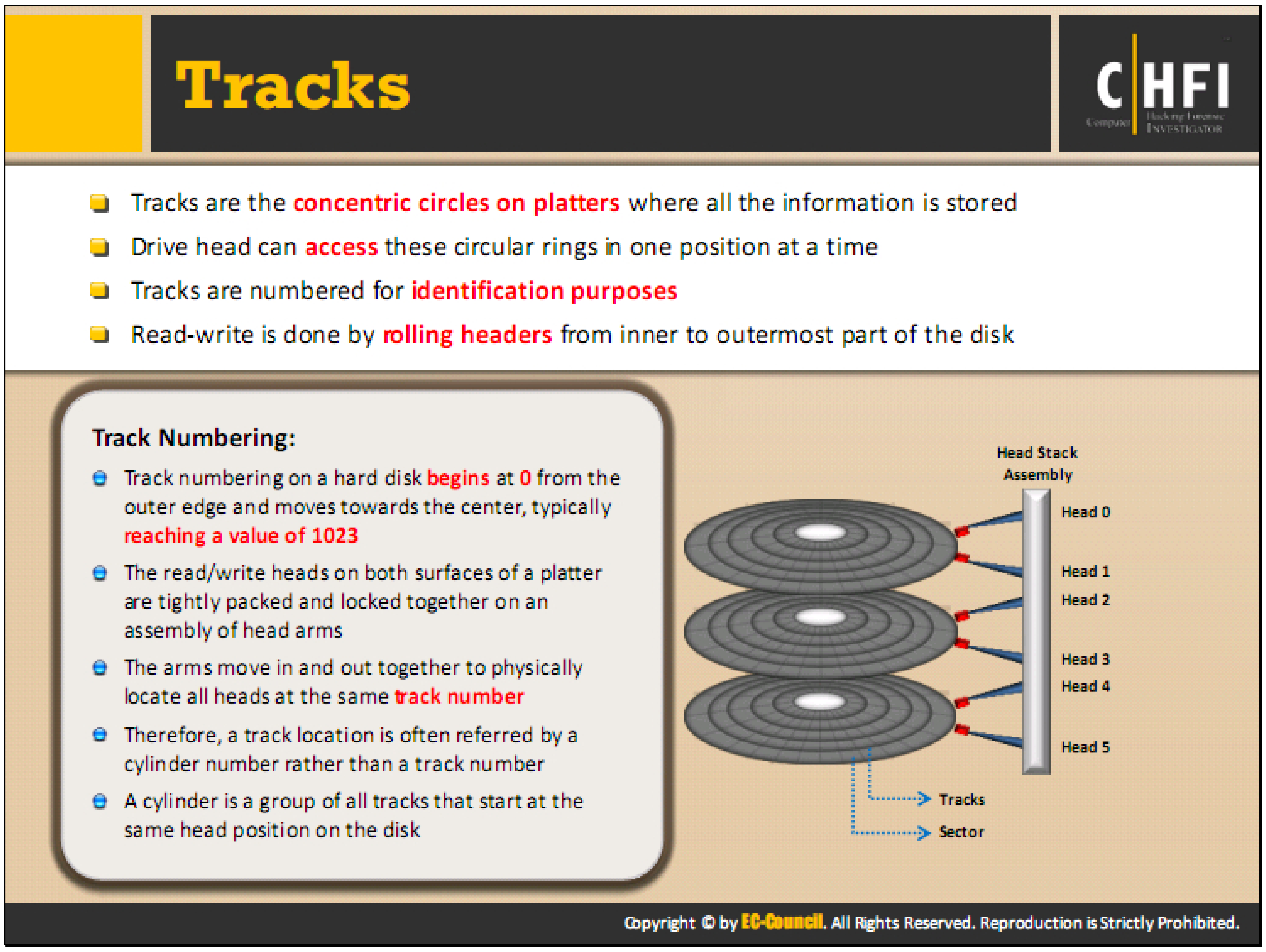
Many telecommunications companies make use of optical fibres to transmit telephone signals, cable television signals, and Internet communications.

Features of Fibre Channel:

- Inexpensive
- Supports higher data transfer rate between mainframes, workstations, desktop computers, supercomputers, displays, storage devices, etc.

Protocols supporting Fibre Channel:

- SCSI
- IP
- ATM
- HIPPI



Tracks

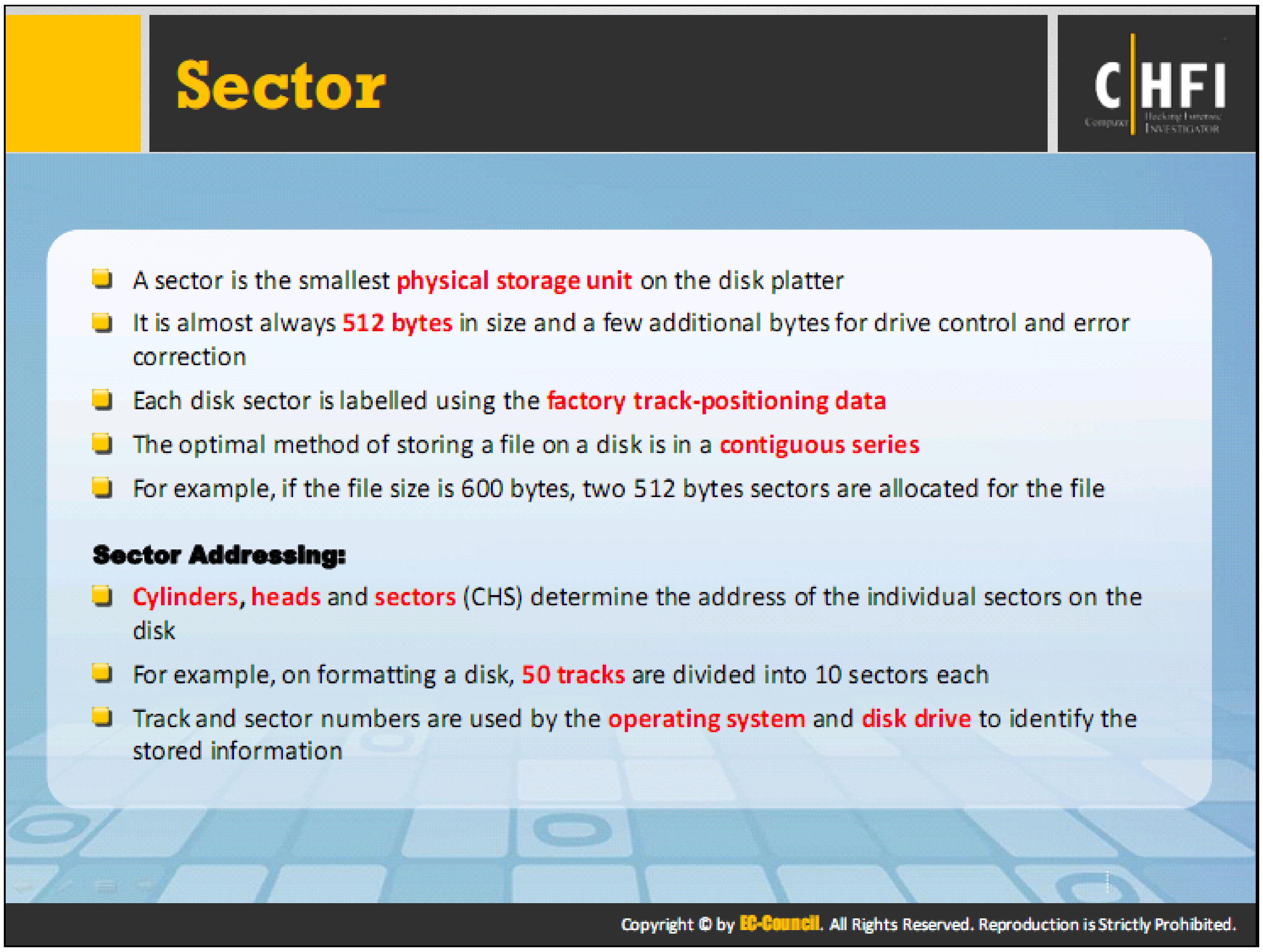
- Tracks are the concentric circles on platters where all the information is stored
- Drive head can access these circular rings in one position at a time
- Tracks are numbered for identification purposes
- Read-write is done by rolling headers from inner to outermost part of the disk

Track Numbering:

- Track numbering on a hard disk begins at 0 from the outer edge and moves towards the center, typically reaching a value of 1023
- The read/write heads on both surfaces of a platter are tightly packed and locked together on an assembly of head arms
- The arms move in and out together to physically locate all heads at the same track number
- Therefore, a track location is often referred by a cylinder number rather than a track number
- A cylinder is a group of all tracks that start at the same head position on the disk

Platters have two surfaces, and each surface divides into concentric circles called tracks. They store all the information on a hard disk. Tracks on the platter partition hold large chunks of data. A modern hard disk contains tens of thousands of tracks on each platter. The rolling heads read and write from the inner to outermost part of the disk. This kind of data arrangement enables easy access to any part of the disk; therefore, hard disks get the moniker as random access storage devices.

Each track contains a number of smaller units called sectors. Every platter has the same track density. The track density refers to the compactness of the track circles so that it can hold maximum number of bits within each unit area on the surface of the platter. It also determines the storage capacity of data on the hard disk. It is a component of area density in terms of capacity and performance.



Sector

- A sector is the smallest **physical storage unit** on the disk platter
- It is almost always **512 bytes** in size and a few additional bytes for drive control and error correction
- Each disk sector is labelled using the **factory track-positioning data**
- The optimal method of storing a file on a disk is in a **contiguous series**
- For example, if the file size is 600 bytes, two 512 bytes sectors are allocated for the file

Sector Addressing:

- **Cylinders**, **heads** and **sectors** (CHS) determine the address of the individual sectors on the disk
- For example, on formatting a disk, **50 tracks** are divided into 10 sectors each
- Track and sector numbers are used by the **operating system** and **disk drive** to identify the stored information

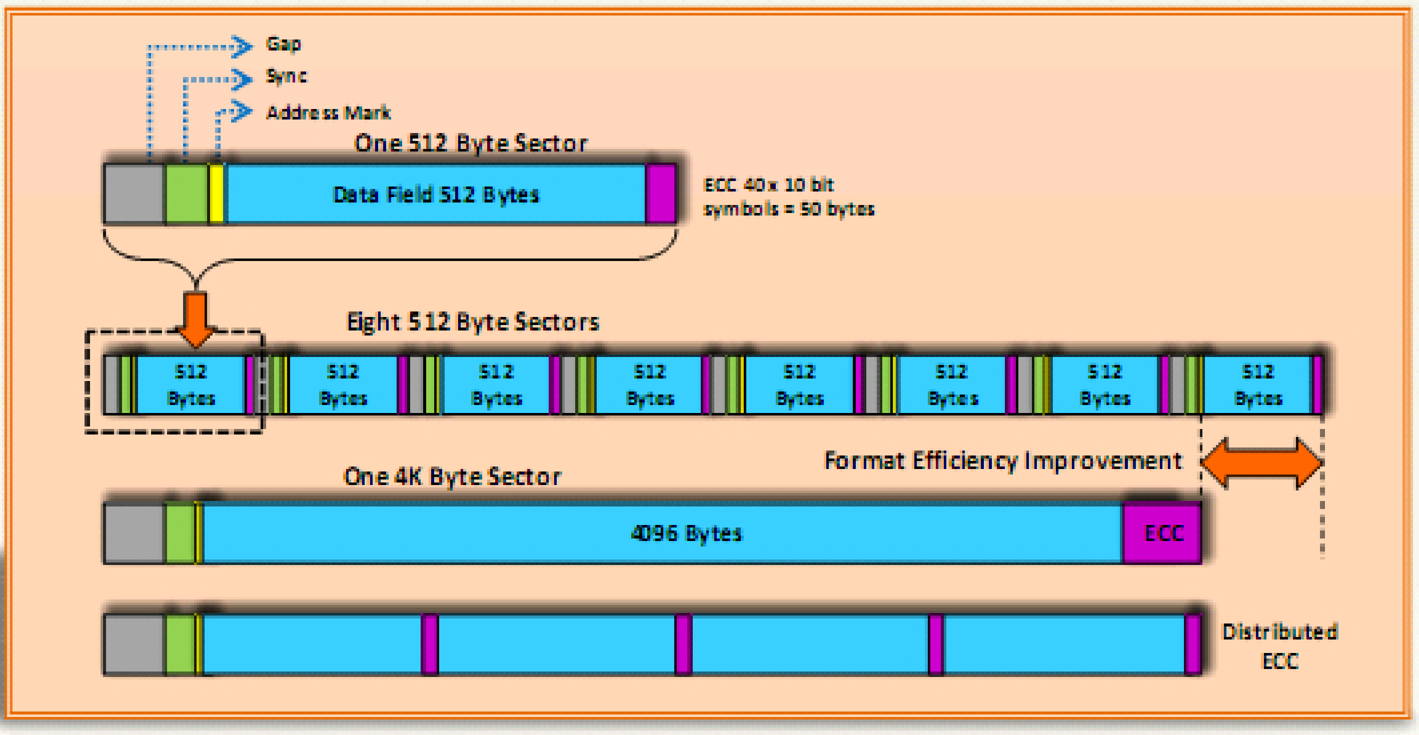
Tracks contain smaller divisions called sectors, and these sectors are the smallest physical storage units located on a hard disk platter. "Sector" is a mathematical term denoting the "pie-shaped" or angular part of the circle, surrounded by the perimeter of the circle between two radii. Each sector normally stores 512 bytes of data, with additional bytes utilized for internal drive control and for error correction and detection. This added information helps to control the drive, store the data, and perform error detection and correction. A group of sectors combines in a concentric circle to form a track. The group of tracks combines to form a surface of the disk platter. The contents of a sector are as follows:

- **ID information:** It contains the sector number and location that identify sectors on the disk. It also contains status information of the sectors
- **Synchronization fields:** The drive controller drives the read process using these fields
- **Data:** It is the information stored on the sector
- **ECC:** This code ensures integrity of the data
- **Gaps:** Spaces used to provide time for the controller to continue the read process

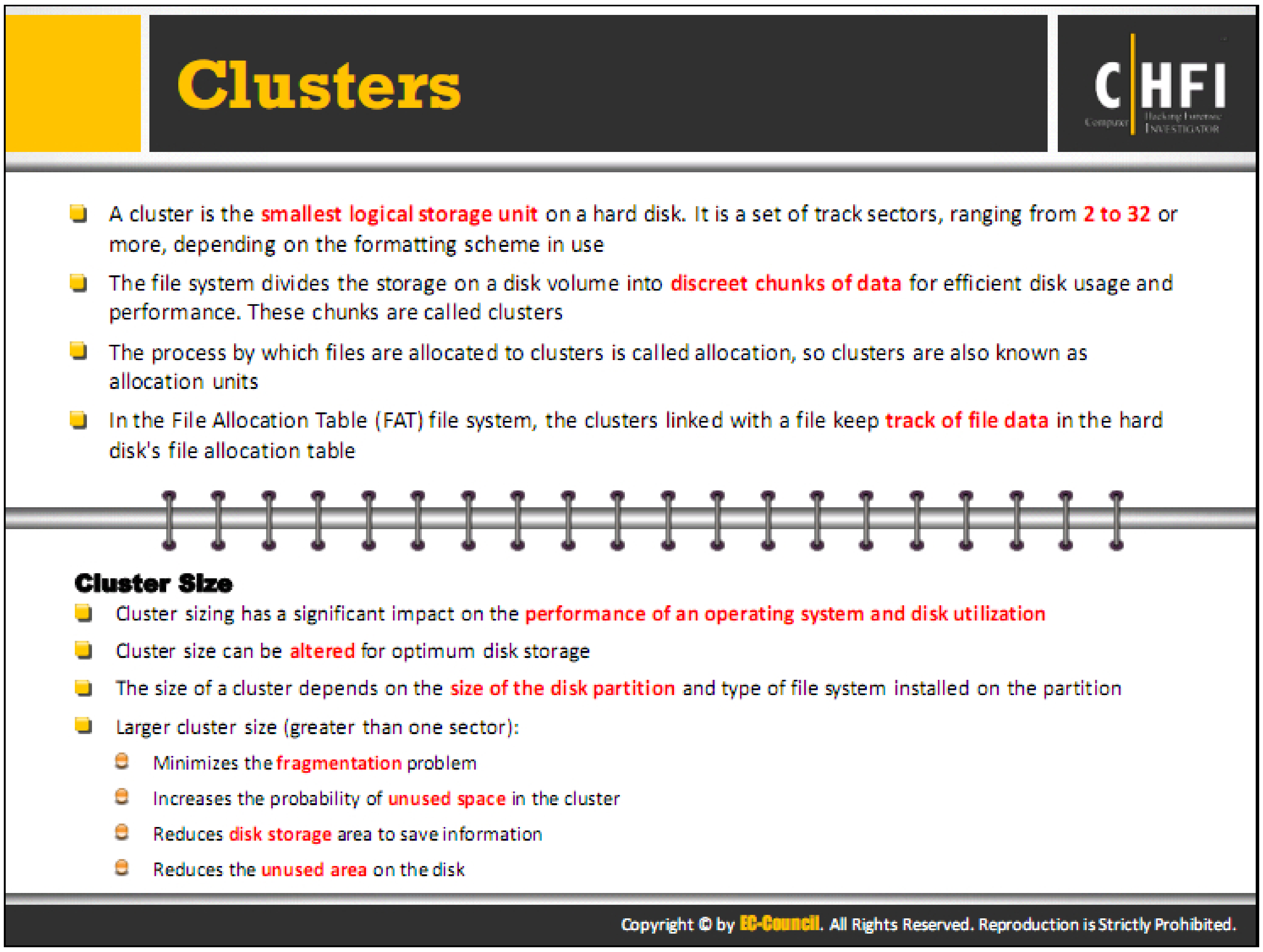
These elements constitute sector overhead. It is an important determinant in calculating time taken for accessing. As the hard disk uses bits for disk or data management, overhead size must be very less for higher efficiency. The file on a disk stores the data in a contiguous series for optimal space usage, while the system allocates sectors for the file according to the size of the file. If file size is 600 bytes, then it allocates two sectors, each of 512 bytes. The track number and the sector number refer to the address of any data on the hard disk.

Advanced Format: Sectors

- New hard drives use 4096 byte (4 KB or 4K) advanced format sectors
- Generation-one Advanced Format also called as 4K sector technology, efficiently uses the storage surface media of a disk efficiently by merging eight 512 byte sectors into one single sector (4096 bytes)
- After merging, the structure of 4K sector does not disturb the key design elements of the traditional 512-byte sector



New hard drives use 4096 byte (4 KB or 4 K) advanced format sectors. This format uses the storage surface media of a disk efficiently by merging eight 512-byte sectors into one single sector (4096 bytes). The structure of a 4K sector maintains the design elements of the 512-byte sector with representation of the beginning and the error correction coding (ECC) area with the identification and synchronization characters, respectively. The 4K sector technology removes redundant header areas, lying between the sectors.



Clusters

- A cluster is the smallest logical storage unit on a hard disk. It is a set of track sectors, ranging from 2 to 32 or more, depending on the formatting scheme in use
- The file system divides the storage on a disk volume into discreet chunks of data for efficient disk usage and performance. These chunks are called clusters
- The process by which files are allocated to clusters is called allocation, so clusters are also known as allocation units
- In the File Allocation Table (FAT) file system, the clusters linked with a file keep track of file data in the hard disk's file allocation table

Cluster Size

- Cluster sizing has a significant impact on the performance of an operating system and disk utilization
- Cluster size can be altered for optimum disk storage
- The size of a cluster depends on the size of the disk partition and type of file system installed on the partition
- Larger cluster size (greater than one sector):
  - Minimizes the fragmentation problem
  - Increases the probability of unused space in the cluster
  - Reduces disk storage area to save information
  - Reduces the unused area on the disk

Clusters are the smallest accessible storage units on the hard disk. The file systems divide the volume of data stored on the disk into discreet chunks of data for greater performance and efficient disk usage. Clusters form by combining sectors in order to ease the process of handling files. Also called allocation units, the clusters are sets of tracks and sectors ranging from 2 to 32, or more, depending on the formatting scheme. The file allocation systems must be flexible in order to allocate the required sectors to files. It can be the size of one sector per cluster. Any read or write will consume the minimum space of one cluster.

To store a file, the file system should assign the required number of clusters to them. The cluster size totally depends on the disk volume. For disk volumes, each cluster varies in size from four to 64 sectors. In some cases, a cluster size may be of 128 sectors. The sectors located in a cluster are continuous. Therefore, every cluster is a continuous chunk of space on the hard disk. In a cluster, when the file system stores a file relatively smaller than size of the cluster, extra space gets wasted and called as slack space.

Cluster Size:

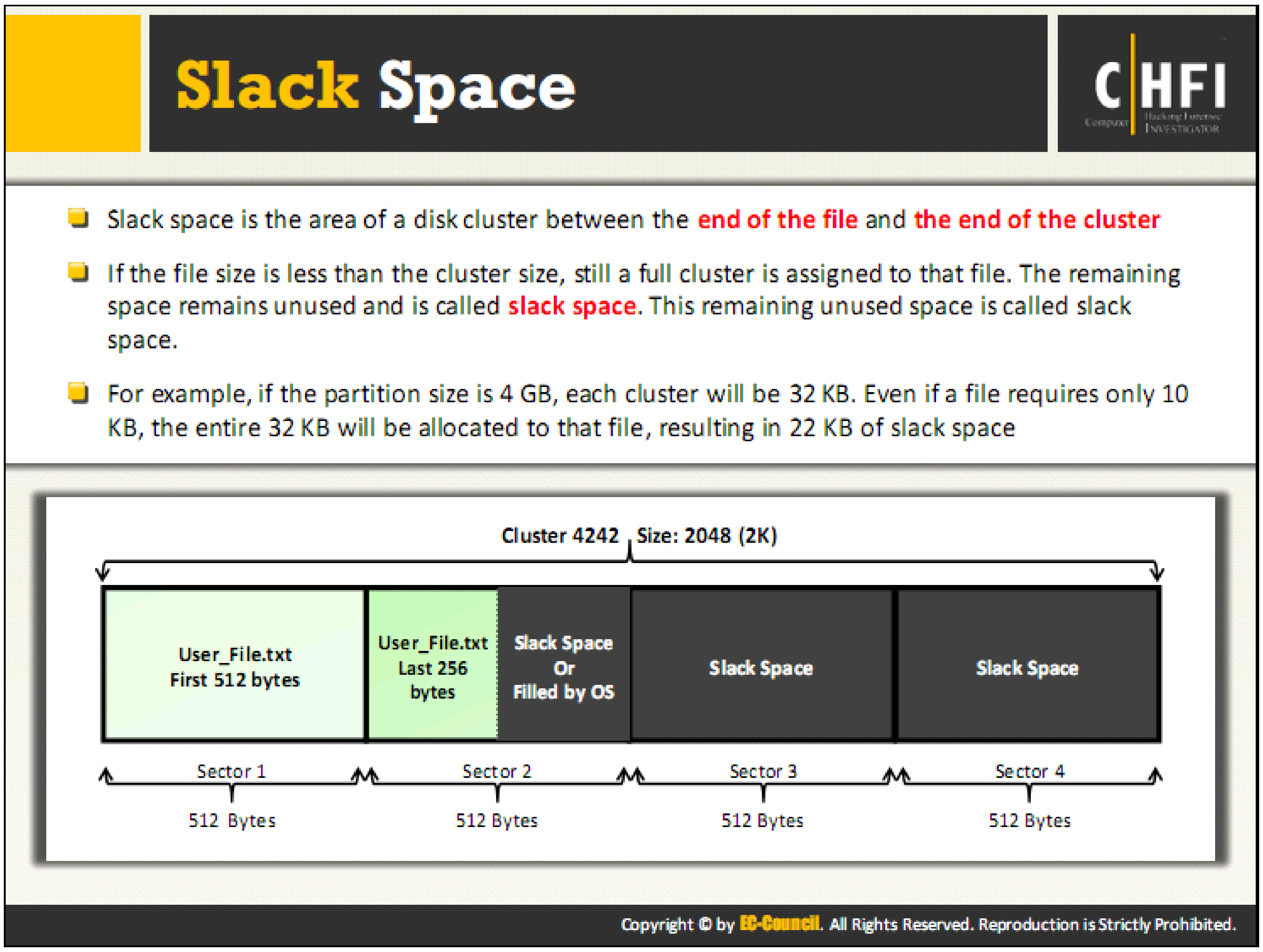
Cluster sizing has a significant impact on the performance of an operating system and disk utilization. Disk partitioning determines the size of a cluster and larger volumes use larger cluster sizes. The system can change the cluster size of an existing partition to enhance performance. If the cluster size is 8192 bytes, to store a file of 5000 bytes, the file system allocates whole cluster to the file and allocates two clusters of 16,384 bytes if the file size is 10,000 bytes. This is why cluster size plays a vital role in maximizing the efficient use of the disk.

By using a large cluster size, the fragmentation problem diminishes, but it will greatly increase the chances of unused space. The file system, running on the computer, maintains the cluster entries.

Clusters form chains on the disk using continuous numbers for which it is not required to store the entire file in one continuous block on the disk. The file system can store it in pieces located anywhere on the disk as well as move it anywhere after creating the file. This cluster chaining is invisible to the operating system.

Users can change the cluster size only when reformatting the drive. Following are the steps to change the cluster size:

- Right-click the drive that you want to format, and select Format
- In the Format dialog box, choose the allocation unit size that you wish the newly formatted drive to use. You can set the cluster size from 521 bytes to 4096 bytes



Slack Space

- Slack space is the area of a disk cluster between the end of the file and the end of the cluster
- If the file size is less than the cluster size, still a full cluster is assigned to that file. The remaining space remains unused and is called slack space. This remaining unused space is called slack space.
- For example, if the partition size is 4 GB, each cluster will be 32 KB. Even if a file requires only 10 KB, the entire 32 KB will be allocated to that file, resulting in 22 KB of slack space

Cluster 4242 , Size: 2048 (2K)
User_File.txt First 512 bytes
User_File.txt Last 256 bytes
Slack Space Or Filled by OS
Slack Space
Slack Space
Sector 1
512 Bytes
Sector 2
512 Bytes
Sector 3
512 Bytes
Sector 4
512 Bytes

Slack space is the wasted area of the disk cluster lying between end of the file and end of the cluster when the file system allocates a full cluster to a file, which is smaller than the cluster size.

More files with larger cluster sizes results in wasted disk space due to overhead attached to them. DOS and Windows file systems use fixed-size clusters. Size consumed is irrelevant of the data storage, but the file system reserves entire space for the file. The older versions of the Windows operating system and DOS used a 16-bit allocation table, which results in the large cluster size for large partitions. For example, if the size of each partition is 4 GB and the size of each cluster is 32 K, and a file requires only 10 K, the system will allocate whole 32 K cluster, resulting in 22 K of slack space.

To eliminate this inefficiency, the system uses partitioning. Another approach to reduce the slack space is to use NTFS, which allows much smaller clusters on large partitions. Archiving infrequently used files can also use compression to reduce slack. As the size of disks is increasing, this slack space problem is gaining much more importance.

File Slack Types

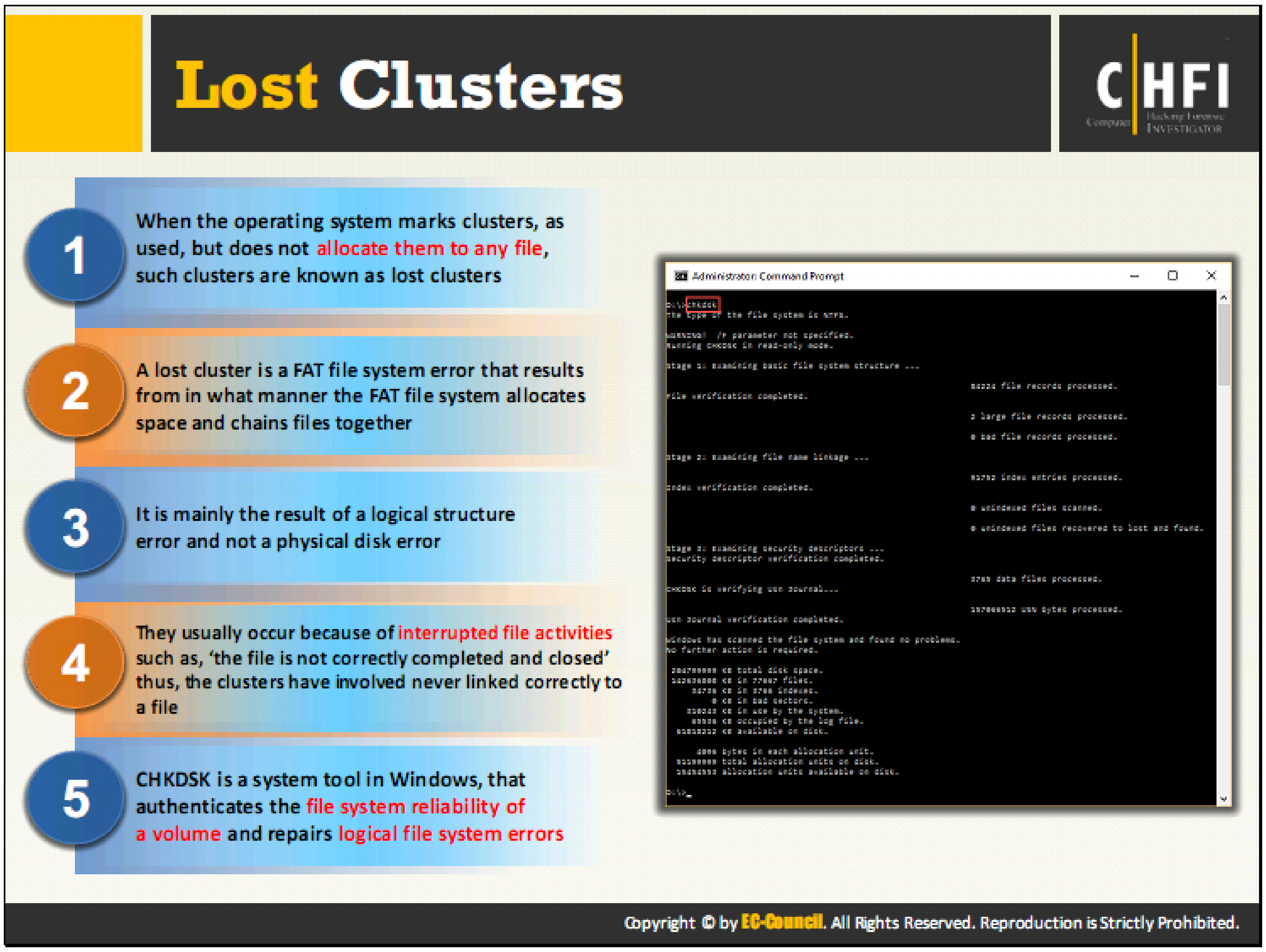
- **RAM Slack**

  RAM slack is the data storage space, which starts from the end of a file to the end of the last sector of the file.

- **Drive Slack**

  Drive Slack is the data storage space, which starts from the end of the last sector of a file to the end of the last cluster of file.

In the field of forensic investigation, slack space is an important form of evidence. Often, slack space can contain relevant suspect information, required by the prosecutor to present as evidence in the court. For example, if the suspect deleted files of the entire hard drive cluster and saved new files, which filled half of the cluster, the other half may not be empty. It can contain the information of the deleted files. Forensic examiners can collect this data by using computer forensic tools.



Lost Clusters

1. When the operating system marks clusters, as used, but does not allocate them to any file, such clusters are known as lost clusters
2. A lost cluster is a FAT file system error that results from in what manner the FAT file system allocates space and chains files together
3. It is mainly the result of a logical structure error and not a physical disk error
4. They usually occur because of interrupted file activities such as, 'the file is not correctly completed and closed' thus, the clusters have involved never linked correctly to a file
5. CHKDSK is a system tool in Windows, that authenticates the file system reliability of a volume and repairs logical file system errors

Administrator: Command Prompt

```
C:\>chkdsk
The type of the file system is NTFS.

WARNING!  /F parameter not specified.
Running CHKDSK in read-only mode.

Stage 1: Examining basic file system structure ...
                                                            [illegible] file records processed.
File verification completed.
                                                            [illegible] large file records processed.
                                                            0 bad file records processed.

Stage 2: Examining file name linkage ...
                                                            [illegible] index entries processed.
Index verification completed.
                                                            0 unindexed files scanned.
                                                            0 unindexed files recovered to lost and found.

Stage 3: Examining security descriptors ...
Security descriptor verification completed.
                                                            [illegible] data files processed.
CHKDSK is verifying Usn Journal...
                                                            [illegible] USN bytes processed.
Usn Journal verification completed.

Windows has scanned the file system and found no problems.
No further action is required.

 [illegible] KB total disk space.
 [illegible] KB in [illegible] files.
     [illegible] KB in [illegible] indexes.
         0 KB in bad sectors.
    [illegible] KB in use by the system.
     [illegible] KB occupied by the log file.
  [illegible] KB available on disk.

      [illegible] bytes in each allocation unit.
  [illegible] total allocation units on disk.
  [illegible] allocation units available on disk.

C:\>_
```

A lost cluster is a File allocation table (FAT) error that results when the operating system marks clusters as used but does not allocate them to any file. The error occurs from the process FAT file system, uses to assign spaces and group files together. It is mainly a logical structure error and not a physical disk error.

Lost clusters occur when the user does not close files properly or shuts down a computer without closing an application. These errors also occur due to disk corruption such as bad drivers, resource conflicts, etc.

Operating systems mark these clusters as in use, even though they have no files assigned or linked to them. Disk checking programs can examine a complete disk volume for lost clusters. To detect lost clusters, use the program that can save them as a file or clear them. The latter case will generate and link artificial files to these clusters. This method will damage newly formed file afterward, but orphaned data is visible and it is possible to recover some parts.

Disk checking programs can scan the computer system for lost clusters using the following procedure:

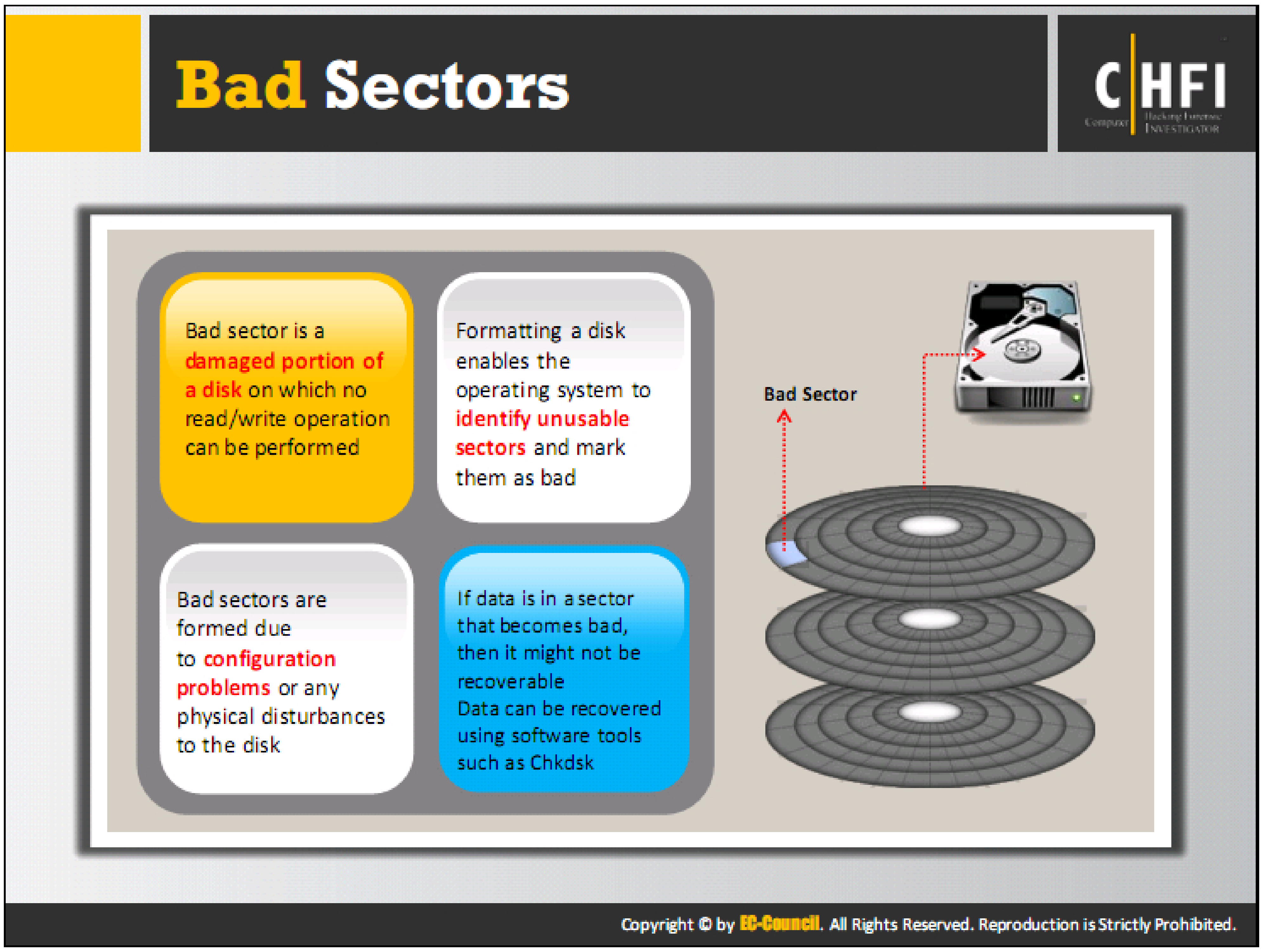
- Generate a duplicate copy in the memory of FAT, noting all of the clusters marked as "in use"
- Trace the clusters, beginning from the root directory, utilized by a file, and mark them as "accounted for", to connect them to a file. Then follow the same procedure for all the subdirectories

- Lost clusters or "orphan" clusters are the ones in use but have no account for

Chkdsk.exe or Check Disk is a built-in Windows utility that helps to detect errors in the file system and disk media. We can run the Check Disk utility If we face problems like, blue screens, difficulty to open or save files or folders. This utility also checks for bad sectors, lost clusters etc.

Steps to use the command line check disk version:

- Open Command Prompt by typing cmd in the Run utility
- Type chkdsk in the command prompt. It will run chkdsk in the Read-Only mode
- This will display the status of the current drive



Bad Sectors

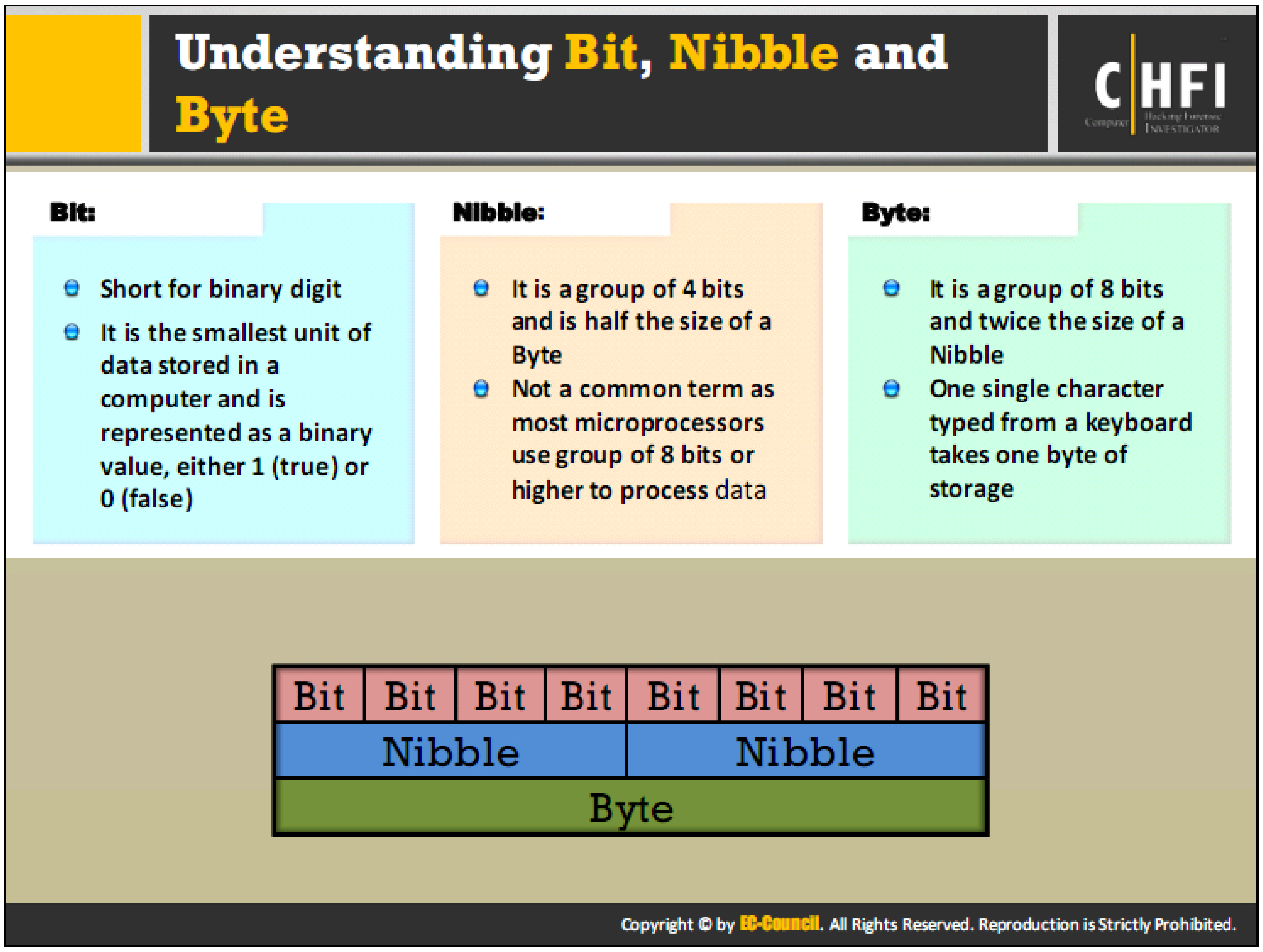
Bad sector is a **damaged portion of a disk** on which no read/write operation can be performed

Formatting a disk enables the operating system to **identify unusable sectors** and mark them as bad

Bad sectors are formed due to **configuration problems** or any physical disturbances to the disk

If data is in a sector that becomes bad, then it might not be recoverable
Data can be recovered using software tools such as Chkdsk

Bad sectors refer to the portions of a disk that are unusable due to some flaws in them and do not support the read or write operations. The data stored in bad sectors is not completely accessible. Bad sectors might be due to configuration problems or any physical disturbances to the disk. Logical errors or bad sectors are the corrupted files on the magnetic media created by problems such as unexpected voltage surges, read/write activities, changes in boot records, viruses, etc. To detect bad sectors on the drive, use a technique called re-mapping or spare sectoring to hide bad sectors. The operating system marks the bad sectors as unusable, while formatting the disk. Users can eliminate these problems to some extent by not putting the hard disk timing too high for the drive, not using an IDE cable that is too long, using correct BIOS settings, and eliminating configuration bottlenecks. If there is some data that becomes damaged, special software that checks for and repairs bad sectors can recover it. Microsoft provides <scandisk> and <chkdsk> utilities for checking and repairing the bad sectors.



Understanding Bit, Nibble and Byte

Bit:

- Short for binary digit
- It is the smallest unit of data stored in a computer and is represented as a binary value, either 1 (true) or 0 (false)

Nibble:

- It is a group of 4 bits and is half the size of a Byte
- Not a common term as most microprocessors use group of 8 bits or higher to process data

Byte:

- It is a group of 8 bits and twice the size of a Nibble
- One single character typed from a keyboard takes one byte of storage

Bit

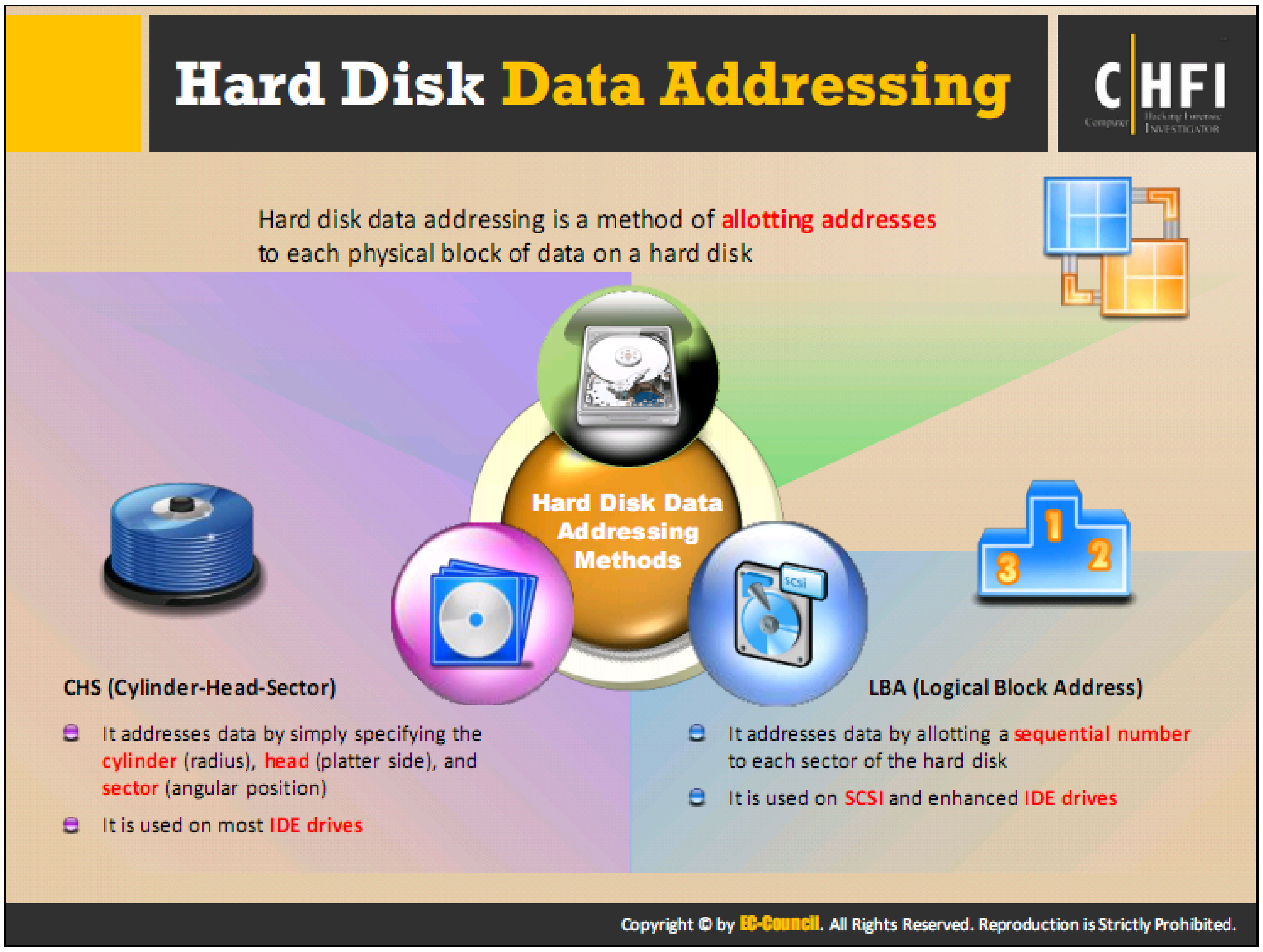
A bit, short for binary digit is the smallest unit of data or basic information unit in computing and digital communications. It can contain only one of the two values represented as 0 or 1. They also represent logical values such as true/false, yes/no, activation states (on/off), algebraic signs (+/−) or any other two-valued attribute.

Byte

A byte, short for binary term is a digital information unit of data that consists of eight bits. The byte is representation of the number of bits a system has used to encode one text character. Therefore, it is the smallest addressable memory unit in many computer architectures. Two hexadecimal digits represent a full byte or octet.

Nibble

A nibble, also known as half-byte or tetrade is a collection of four bits or half of an octet in computing. Common representation of a byte is two nibbles.



Hard Disk Data Addressing

Hard disk data addressing is a method of allotting addresses to each physical block of data on a hard disk

Hard Disk Data Addressing Methods

CHS (Cylinder-Head-Sector)

- It addresses data by simply specifying the cylinder (radius), head (platter side), and sector (angular position)
- It is used on most IDE drives

LBA (Logical Block Address)

- It addresses data by allotting a sequential number to each sector of the hard disk
- It is used on SCSI and enhanced IDE drives

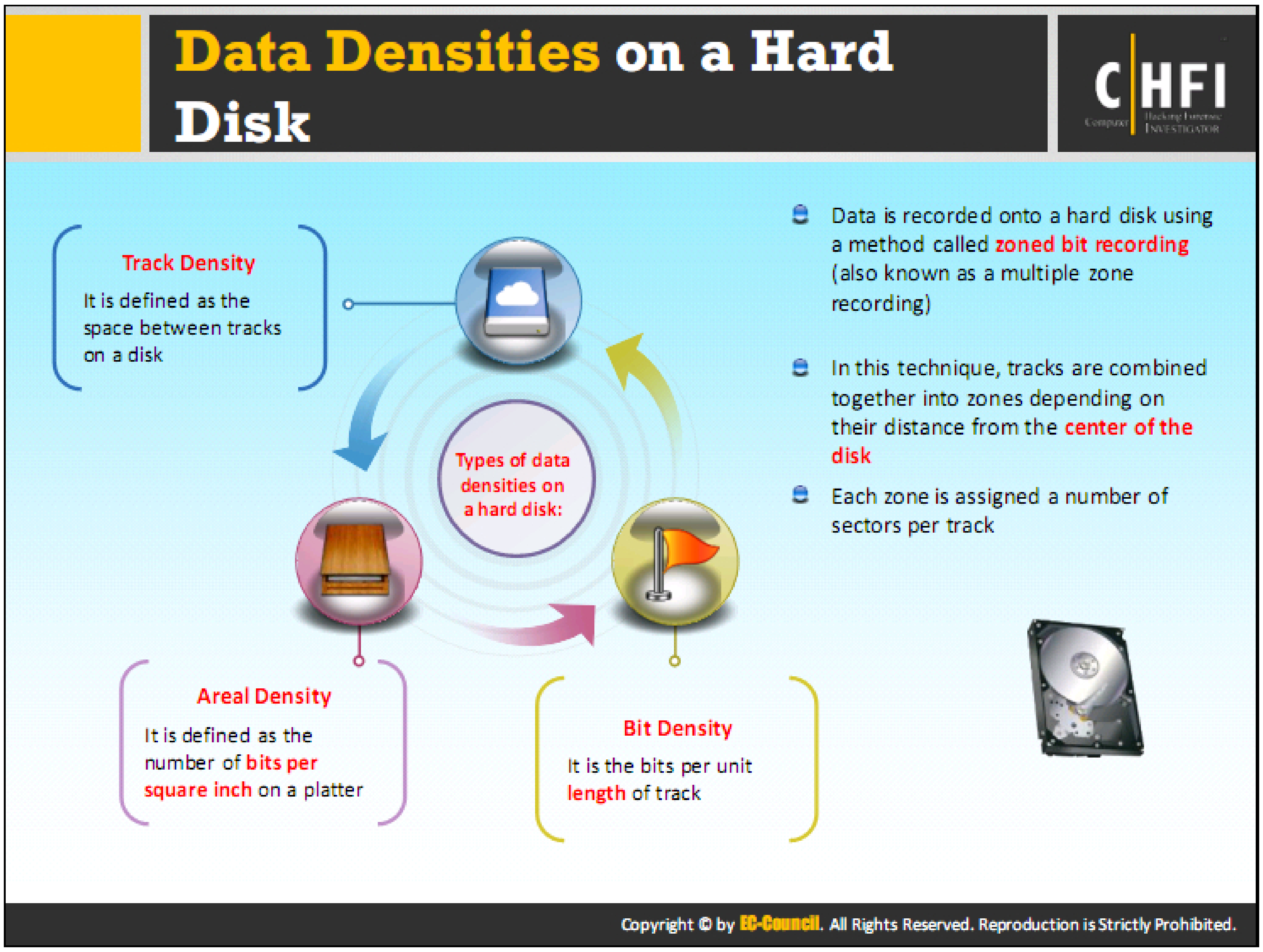
Hard disk data addressing is the technique of assigning addresses to physical blocks of data on the hard drives. There are two types of hard disk data addressing:

CHS (Cylinder-Head-Sector)

This process identifies individual sectors on a hard disk according to their positions in a track, and the head and cylinder numbers determine these tracks. It associates information on the hard drive by specifications such as head (platter side), cylinder (radius), and the sector (angular position).

LBA (Logical Block Address)

It addresses data by allotting a sequential number to each sector of the hard disk. The addressing mechanism specifies the location of blocks of data on computer storage devices and secondary storage systems such as hard disk drives, SCSI, and enhanced IDE drives. This method does not expose the physical details of the storage device to the operating system.



Data Densities on a Hard Disk

- Data is recorded onto a hard disk using a method called zoned bit recording (also known as a multiple zone recording)
- In this technique, tracks are combined together into zones depending on their distance from the center of the disk
- Each zone is assigned a number of sectors per track

Types of data densities on a hard disk:

Track Density
It is defined as the space between tracks on a disk

Areal Density
It is defined as the number of bits per square inch on a platter

Bit Density
It is the bits per unit length of track

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Hard disks store data using the zoned bit recording method, which is also known as multiple-zone recording. In this technique, tracks form a collection of zones depending on their distance from the center of the disk and the outer tracks have more sectors on them than the inner tracks. This allows the drive to store more bits in each outer track compared to the innermost zone and helps to achieve a higher total data capacity.

Track Density

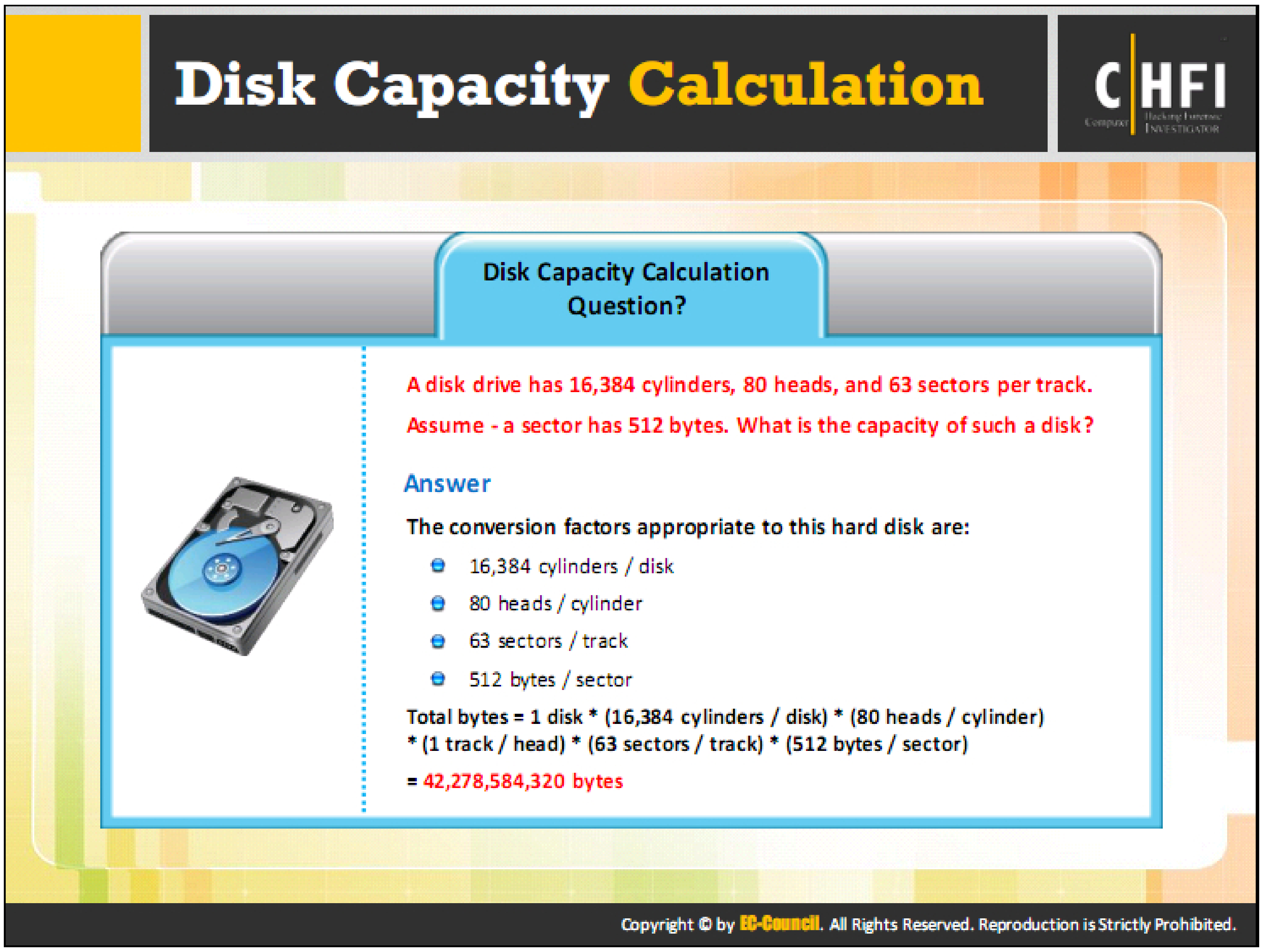
It refers to the space a particular number of tracks require on a disk. The disks with greater track density can store more information as well as offer better performance.

Areal Density

It refers to the number of bits per square inch on a platter and it represents the amount of data a hard disk can hold.

Bit Density

It is the number of bits a unit length of track can accommodate.



Disk Capacity Calculation

Disk Capacity Calculation Question?

A disk drive has 16,384 cylinders, 80 heads, and 63 sectors per track.

Assume - a sector has 512 bytes. What is the capacity of such a disk?

Answer

The conversion factors appropriate to this hard disk are:

- 16,384 cylinders / disk
- 80 heads / cylinder
- 63 sectors / track
- 512 bytes / sector

Total bytes = 1 disk * (16,384 cylinders / disk) * (80 heads / cylinder) * (1 track / head) * (63 sectors / track) * (512 bytes / sector)

= 42,278,584,320 bytes

Calculate

A disk drive that has:

- 16,384 cylinders
- 80 heads
- 63 sectors per track

Assume a sector has 512 bytes. What is the capacity of such a disk?

Answer

The conversion factors appropriate to this hard disk are:

- 16,384 cylinders / disk
- 80 heads / cylinder
- 63 sectors / track
- 512 bytes / sector

Solution

Total bytes = 1 disk * (16,384 cylinders / disk) * (80 heads / cylinder) * (1 track / head) * (63 sectors / track) * (512 bytes / sector) = **42,278,584,320 bytes**

1 Kilobyte (KB) = 2^10 bytes = 1,024 bytes

1 Megabyte (MB) = 2^20 bytes = 1,048,576 bytes = 1,024 KB

1 Gigabyte (GB) = 2^30 bytes = 1,073,741,824 bytes = 1,048,576 KB = 1,024 MB

1 Terabyte (TB) = 2^40 bytes = 1,099,511,627,776 bytes = 1,073,741,824 KB = 1,048,576

MB = 1,024 GB

Using these definitions, express the result in GB as:

42,278,584,320 bytes / (1,073,741,824 bytes / GB) = **39.375 GB**

Hard disk in a typical computer system has a storage capacity. Data is stored on the hard disk in the form of files. A file is nothing more than a collection of bytes. The bytes may be:

- ASCII code
- Instruction of a software program for the computer system to execute
- Record of a database
- Pixel colors for a GIF image

Measuring good hard disk drive performance includes calculation of it two characteristics including the access time and data transfer rate.

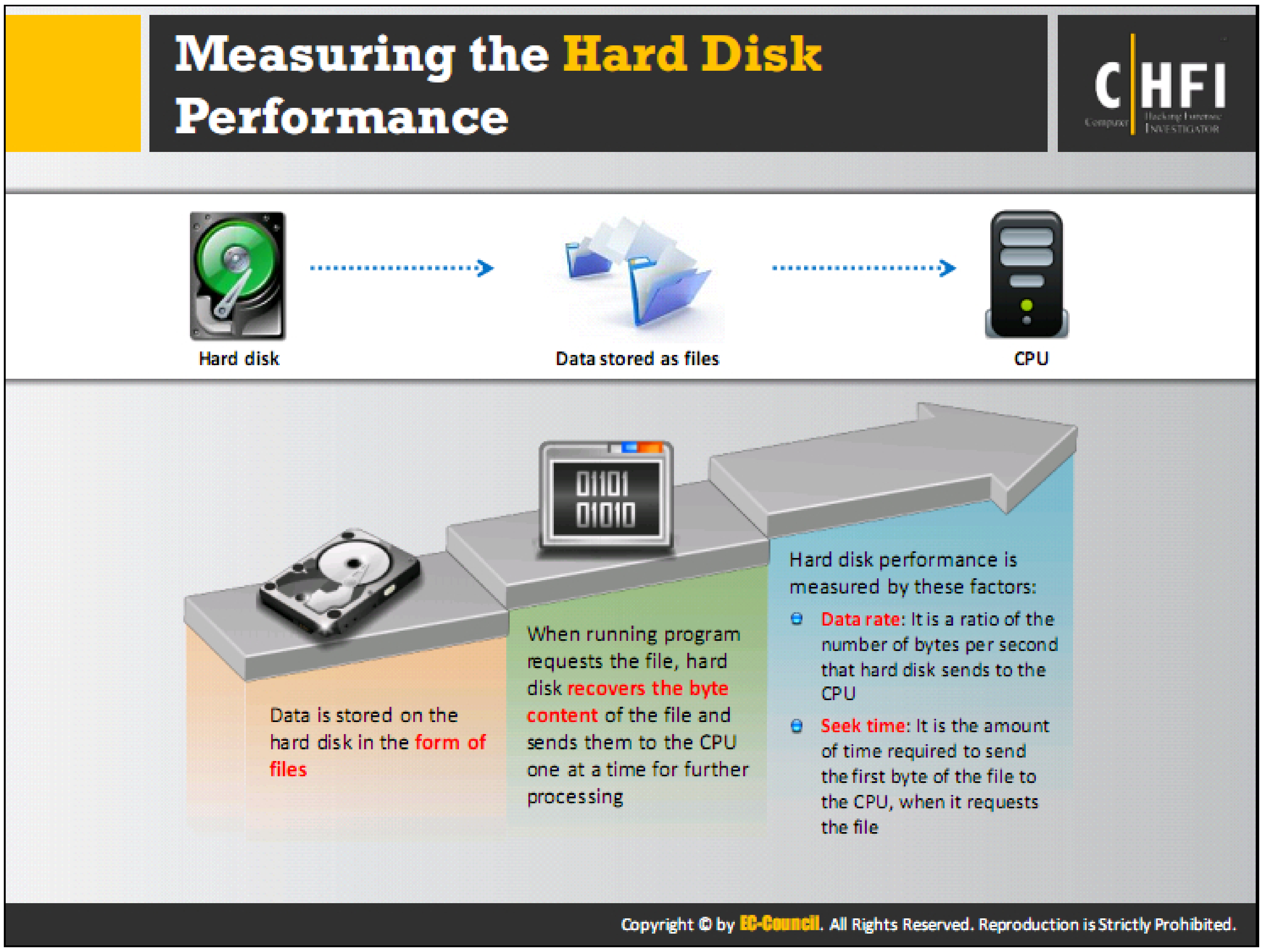
Access time

Access time refers to the time a drive takes to initiate the data transfer. The controlling factors of this time on a drive depend on the mechanical nature of rotating disks and moving heads.

The main components added to get the access time are:

- **Seek time:** The time a hard disk controller requires to find a particular data. When required to read or write data, the disc heads move to the correct position through the process of seeking. The time it takes to move read or write disc heads from one point to another of the disk is the seek time. Common seek time is between 10 to 20 milliseconds, with common desktop type normally being around 9 milliseconds.
- **Rotational latency:** It refers to the rotational delay the chosen disk sector takes to rotate under read or write disk drive heads. The average disk rotational latency is half of the time the disk takes to make one revolution. The term is applicable only to rotating storage devices like hard disk drives and floppy drives but not tape drives.
- **Data transfer rate:** The data transfer rate of a drive is expressed by internal rate, which is the data moving between the disk surface and the drive controller as well as the external rate, which is the data moving between the drive controller and the host system. The host transfer rate or data transfer rate is the speed at which host computer can transfer data from the IDE/EIDE or SCSI interface to CPU.
    - Inner zones' data transfer rates range from 44.2 MB/s to 74.5 MB/s
    - The transfer rate at the outer zone is 74.0 MB/s to 111.4 MB/s

Source: http://www.rwc.uc.edu



Measuring the Hard Disk Performance

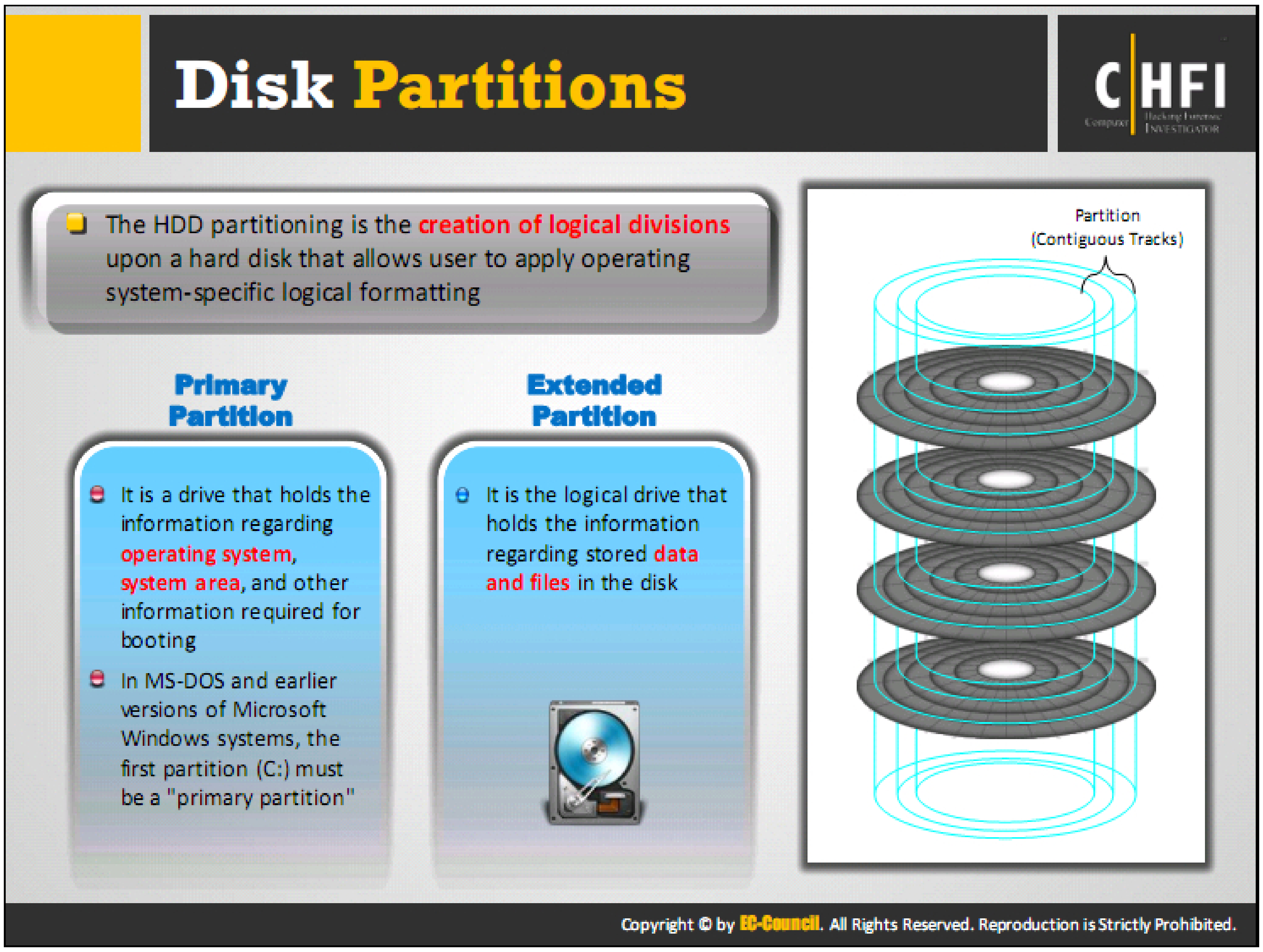
Hard disk → Data stored as files → CPU

Data is stored on the hard disk in the **form of files**

When running program requests the file, hard disk **recovers the byte content** of the file and sends them to the CPU one at a time for further processing

Hard disk performance is measured by these factors:

- **Data rate**: It is a ratio of the number of bytes per second that hard disk sends to the CPU
- **Seek time**: It is the amount of time required to send the first byte of the file to the CPU, when it requests the file



Disk Partitions

- The HDD partitioning is the creation of logical divisions upon a hard disk that allows user to apply operating system-specific logical formatting

Primary Partition

- It is a drive that holds the information regarding operating system, system area, and other information required for booting
- In MS-DOS and earlier versions of Microsoft Windows systems, the first partition (C:) must be a "primary partition"

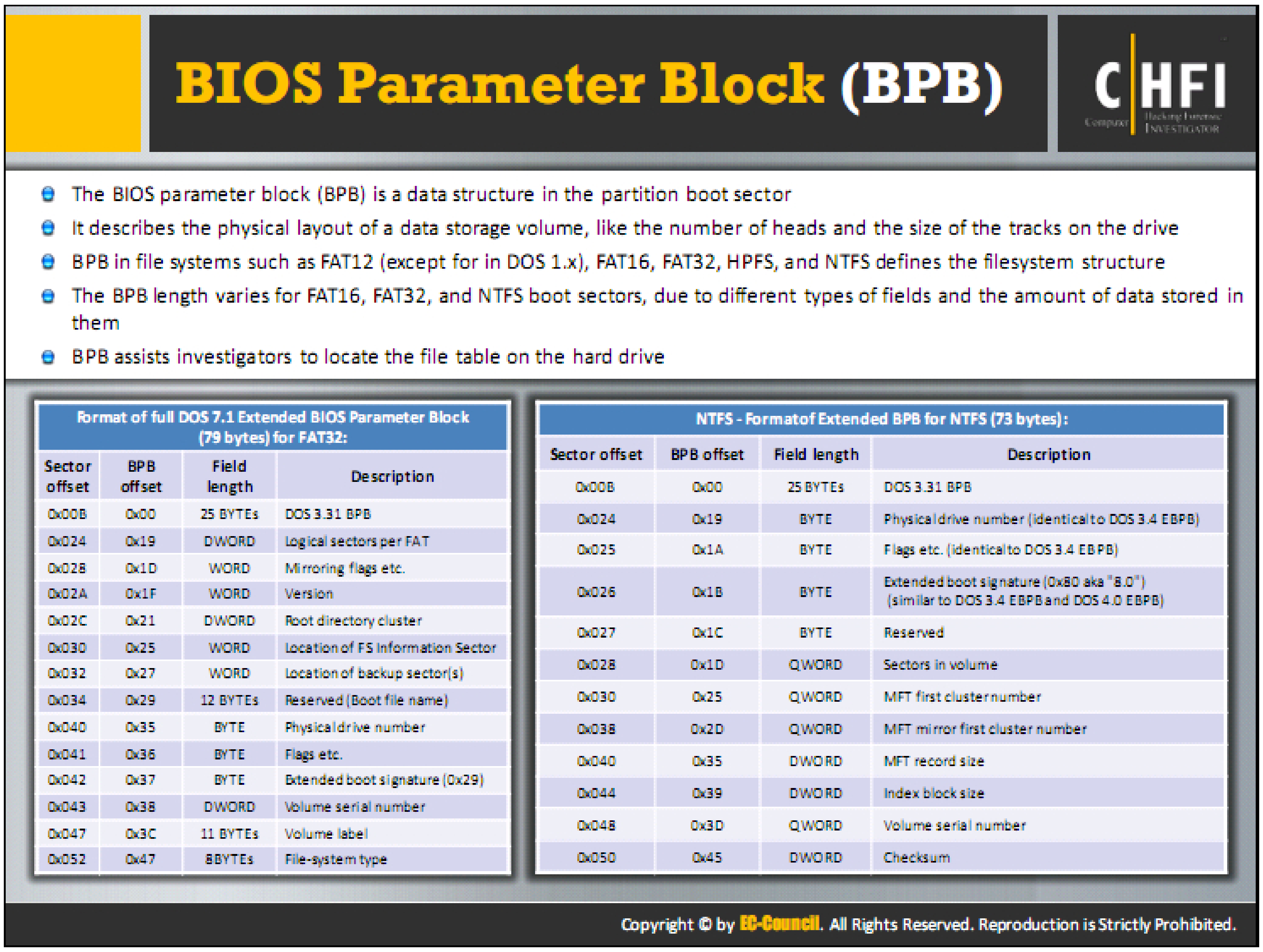
Extended Partition

- It is the logical drive that holds the information regarding stored data and files in the disk

Partitioning refers to the creation of logical drives for effective memory management and a partition is the logical drive for storing the data. Hidden partition created on a drive can hide the data. The inter-partition gap is the space between the primary partition and the secondary partition. If the inter-partition drive contains the hidden data, use disk editor utilities like Disk Editor to change the information in the partition table. Doing so will remove all the references to the hidden partition, which have been hiding it from the operating system. Another way of hiding the data is to place the digital evidence at the end of the disk by declaring a smaller number of bytes than the actual size of the drive. Disk Editor allows investigator to access these hidden or vacant areas of the disk.

The partitions are of two types:

- **Primary partition**: It is the drive that holds the information regarding the operating system, system area, and other information required for booting. In MS-DOS and earlier versions of Microsoft Windows systems, the first partition (C:) must be a "primary partition."
- **Extended partition**: It is the logical drive that holds the information regarding the data and files that are stored in the disk. Various tools are available for examining the disk partitions. A few of the disk editor tools are Disk Edit, WinHex, and Hex Workshop. These tools can help users to view the file headers and important information about the file. Both require analyzing the hexadecimal codes that an operating system identifies and uses to maintain the file system.



BIOS Parameter Block (BPB)

- The BIOS parameter block (BPB) is a data structure in the partition boot sector
- It describes the physical layout of a data storage volume, like the number of heads and the size of the tracks on the drive
- BPB in file systems such as FAT12 (except for in DOS 1.x), FAT16, FAT32, HPFS, and NTFS defines the filesystem structure
- The BPB length varies for FAT16, FAT32, and NTFS boot sectors, due to different types of fields and the amount of data stored in them
- BPB assists investigators to locate the file table on the hard drive

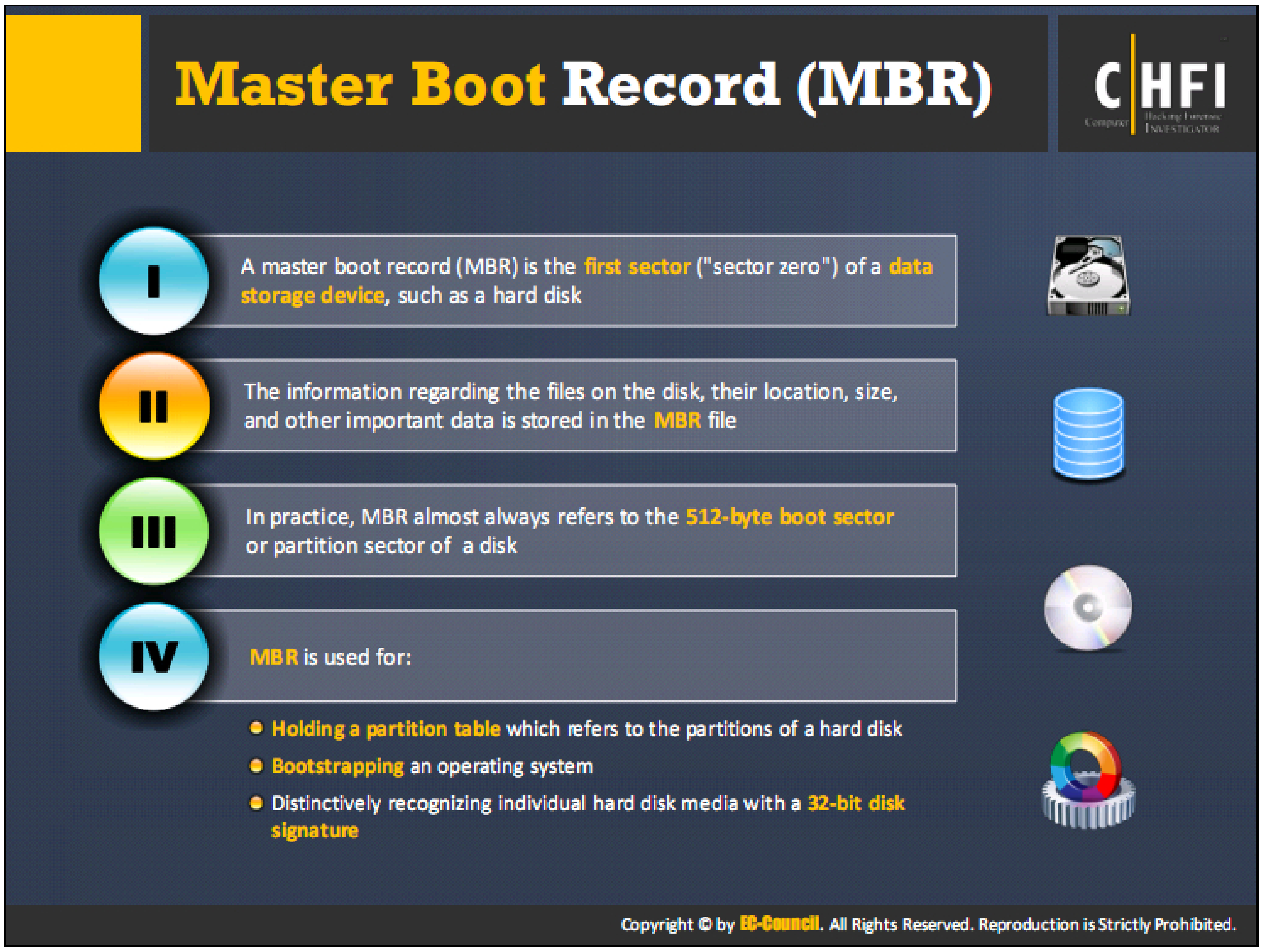
Format of full DOS 7.1 Extended BIOS Parameter Block (79 bytes) for FAT32:

Sector offset	BPB offset	Field length	Description
0x00B	0x00	25 BYTEs	DOS 3.31 BPB
0x024	0x19	DWORD	Logical sectors per FAT
0x028	0x1D	WORD	Mirroring flags etc.
0x02A	0x1F	WORD	Version
0x02C	0x21	DWORD	Root directory cluster
0x030	0x25	WORD	Location of FS Information Sector
0x032	0x27	WORD	Location of backup sector(s)
0x034	0x29	12 BYTEs	Reserved (Boot file name)
0x040	0x35	BYTE	Physical drive number
0x041	0x36	BYTE	Flags etc.
0x042	0x37	BYTE	Extended boot signature (0x29)
0x043	0x38	DWORD	Volume serial number
0x047	0x3C	11 BYTEs	Volume label
0x052	0x47	8BYTEs	File-system type

NTFS - Formatof Extended BPB for NTFS (73 bytes):

Sector offset	BPB offset	Field length	Description
0x00B	0x00	25 BYTEs	DOS 3.31 BPB
0x024	0x19	BYTE	Physical drive number (identical to DOS 3.4 EBPB)
0x025	0x1A	BYTE	Flags etc. (identical to DOS 3.4 EBPB)
0x026	0x1B	BYTE	Extended boot signature (0x80 aka "8.0") (similar to DOS 3.4 EBPB and DOS 4.0 EBPB)
0x027	0x1C	BYTE	Reserved
0x028	0x1D	QWORD	Sectors in volume
0x030	0x25	QWORD	MFT first cluster number
0x038	0x2D	QWORD	MFT mirror first cluster number
0x040	0x35	DWORD	MFT record size
0x044	0x39	DWORD	Index block size
0x048	0x3D	QWORD	Volume serial number
0x050	0x45	DWORD	Checksum

The BPB is data structure situated at sector 1 in the volume boot record of a hard disk and explains the physical layout of a disk volume. It describes the volume partition on partitioned devices such as hard disks, whereas on the un-partitioned devices it describes the entire medium. Any partition that includes the floppy disks can use BPB, which would also describe the basic file system architecture. The length of BPB varies across the listed file systems listed (i.e. FAT16, FAT32, and NTFS) due to the volume of the data it contains and also due to the types of fields present.



Master Boot Record (MBR)

I. A master boot record (MBR) is the first sector ("sector zero") of a data storage device, such as a hard disk

II. The information regarding the files on the disk, their location, size, and other important data is stored in the MBR file

III. In practice, MBR almost always refers to the 512-byte boot sector or partition sector of a disk

IV. MBR is used for:

- Holding a partition table which refers to the partitions of a hard disk
- Bootstrapping an operating system
- Distinctively recognizing individual hard disk media with a 32-bit disk signature

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Master Boot Record (MBR) refers to a hard disk's first sector or sector zero that specifies the location of an operating system for the system to load into the main storage. Also called as, partition sector or master partition table contains a table, which locates partitioned disk data. A program in the record loads the rest of the OS into the RAM.

Information about various files present on the disk, their location, and size is the Master Boot Record file. In practice, MBR almost always refers to the 512-byte boot sector or partition sector of a disk. FDISK/MBR commands help in creating MBR in Windows and DOS operating systems. When a computer starts and boots, the BIOS refers this first sector for the boot process instructions and information about how to load the operating system.

The master boot record consists of the structures as mentioned below:

Partition Table

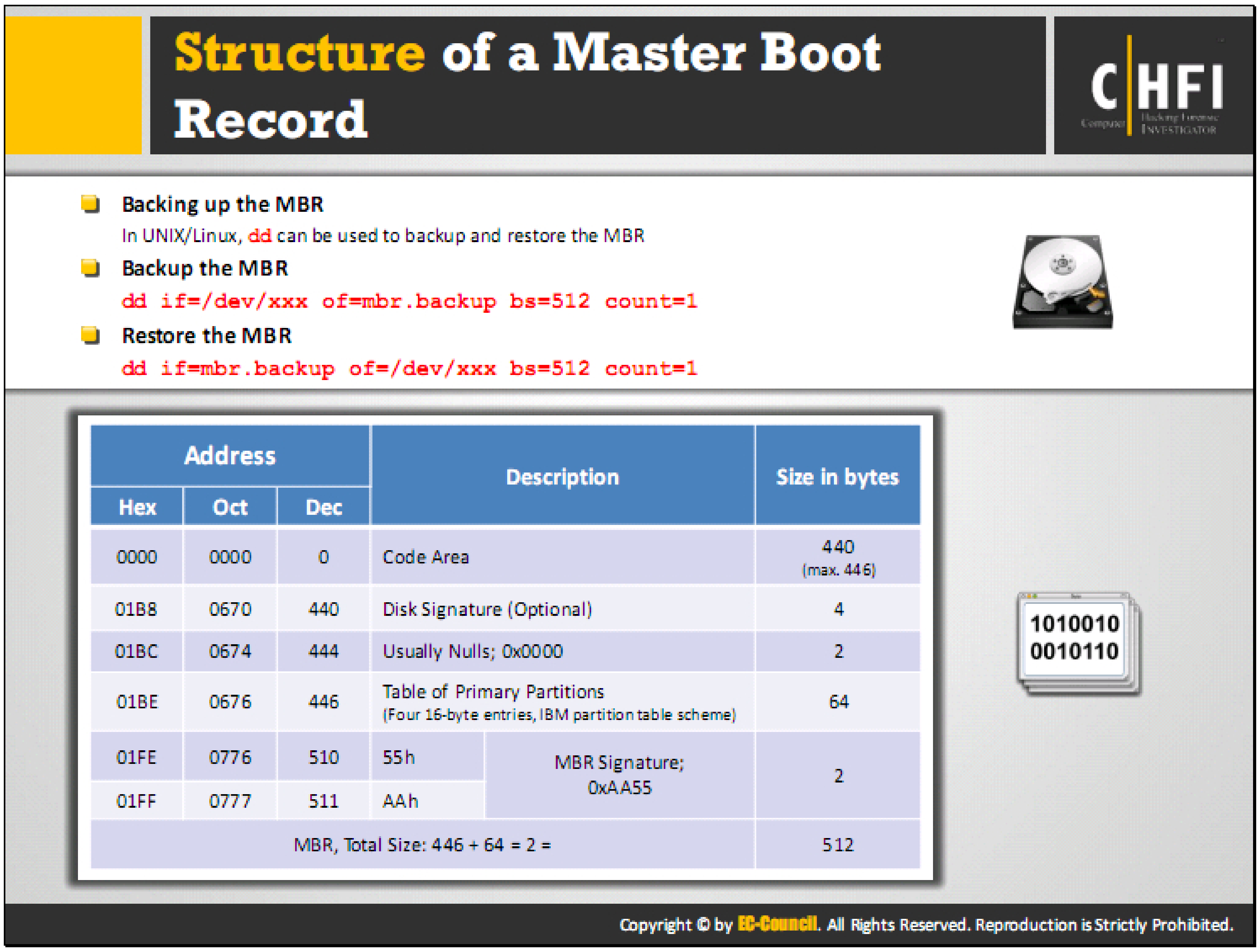
Partition table is a 64-byte data structure storing information about the type of partitions present on the hard disk and their location. This table has a standard layout that does not depend on the operating system. The table is capable of describing only four partitions, which are primary or physical partitions. All other partitions are logical partitions linked to one of the primary partitions.

Master Boot Code

A small part of the computer code, which the system loads into the BIOS and executes to initiate the system's boot process. After execution, the system transfers the controls to the boot program present on the active partition to load the operating system.

The master boot code implements the following functions:

- Examines the partition table to find the active partition
- Locates the first sector of the active partition
- Loads a boot sector copy from the active partition into memory
- Transfers control to the executable code in the boot sector



Structure of a Master Boot Record

- **Backing up the MBR**
  In UNIX/Linux, dd can be used to backup and restore the MBR
- **Backup the MBR**
  `dd if=/dev/xxx of=mbr.backup bs=512 count=1`
- **Restore the MBR**
  `dd if=mbr.backup of=/dev/xxx bs=512 count=1`

Address			Description		Size in bytes
Hex	Oct	Dec			
0000	0000	0	Code Area		440 (max. 446)
01B8	0670	440	Disk Signature (Optional)		4
01BC	0674	444	Usually Nulls; 0x0000		2
01BE	0676	446	Table of Primary Partitions (Four 16-byte entries, IBM partition table scheme)		64
01FE	0776	510	55h	MBR Signature; 0xAA55	2
01FF	0777	511	AAh		
MBR, Total Size: 446 + 64 = 2 =					512

The systems, working with Windows and DOS operating systems, use the MBR file to hold the information regarding the files on the disk. Many products replace the MBR file, provided by the Microsoft operating system. A few third-party utility tools help while installing two or more operating systems on the disk.

Investigators require many data acquisition tools for forensic investigation as one vendor product may not be reliable for computer forensic tasks.

Backing up the MBR

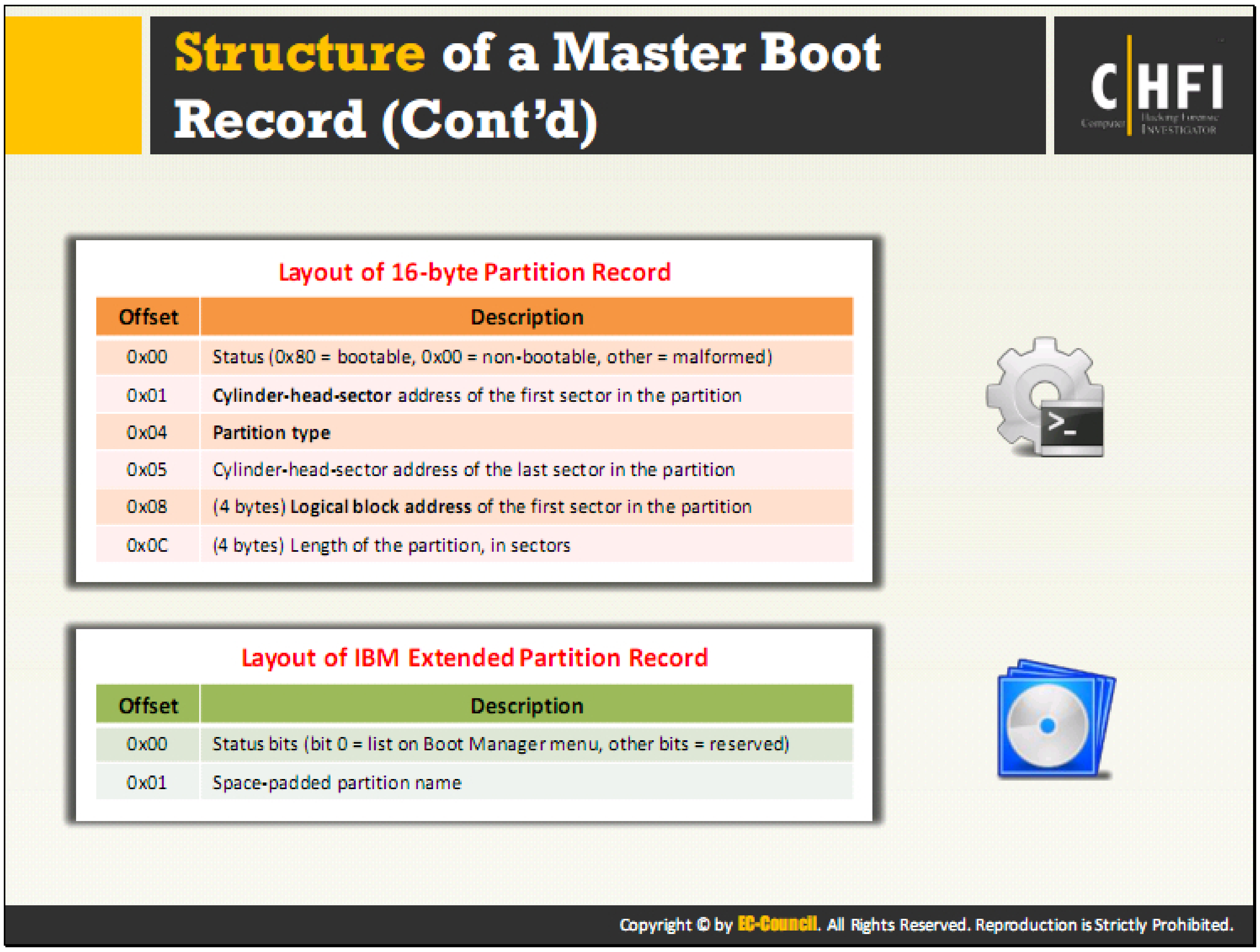
In UNIX/Linux, **dd** helps to create backup and restore the MBR.

Back up the MBR

```
dd if=/dev/xxx of=mbr.backupbs=512 count=1
```

Restore the MBR

```
dd if=mbr.backup of=/dev/xxx bs=512 count=1
```



Structure of a Master Boot Record (Cont'd)

Layout of 16-byte Partition Record

Offset	Description
0x00	Status (0x80 = bootable, 0x00 = non-bootable, other = malformed)
0x01	**Cylinder-head-sector** address of the first sector in the partition
0x04	**Partition type**
0x05	Cylinder-head-sector address of the last sector in the partition
0x08	(4 bytes) **Logical block address** of the first sector in the partition
0x0C	(4 bytes) Length of the partition, in sectors

Layout of IBM Extended Partition Record

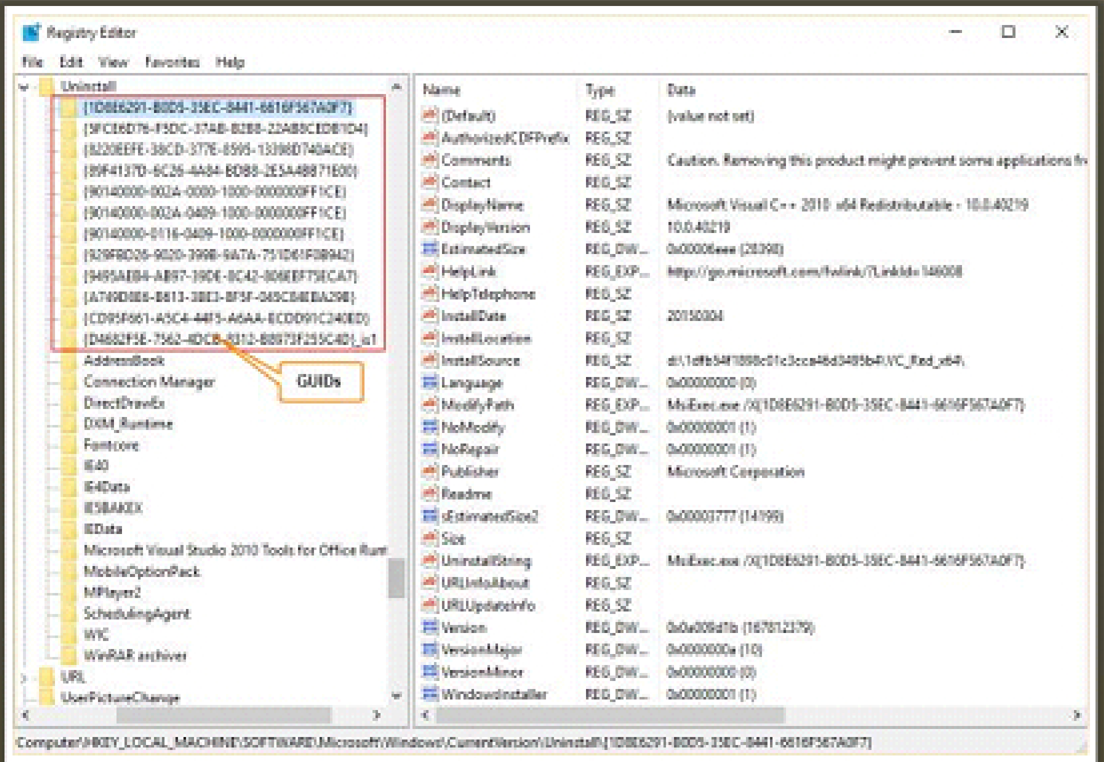
Offset	Description
0x00	Status bits (bit 0 = list on Boot Manager menu, other bits = reserved)
0x01	Space-padded partition name

Globally Unique Identifier (GUID)

- Global Unique Identifier (GUID) is a 128-bit **unique reference number** used as an identifier in computer software
- In general, GUIDs are displayed as **32 hexadecimal digits** with groups separated by hyphens

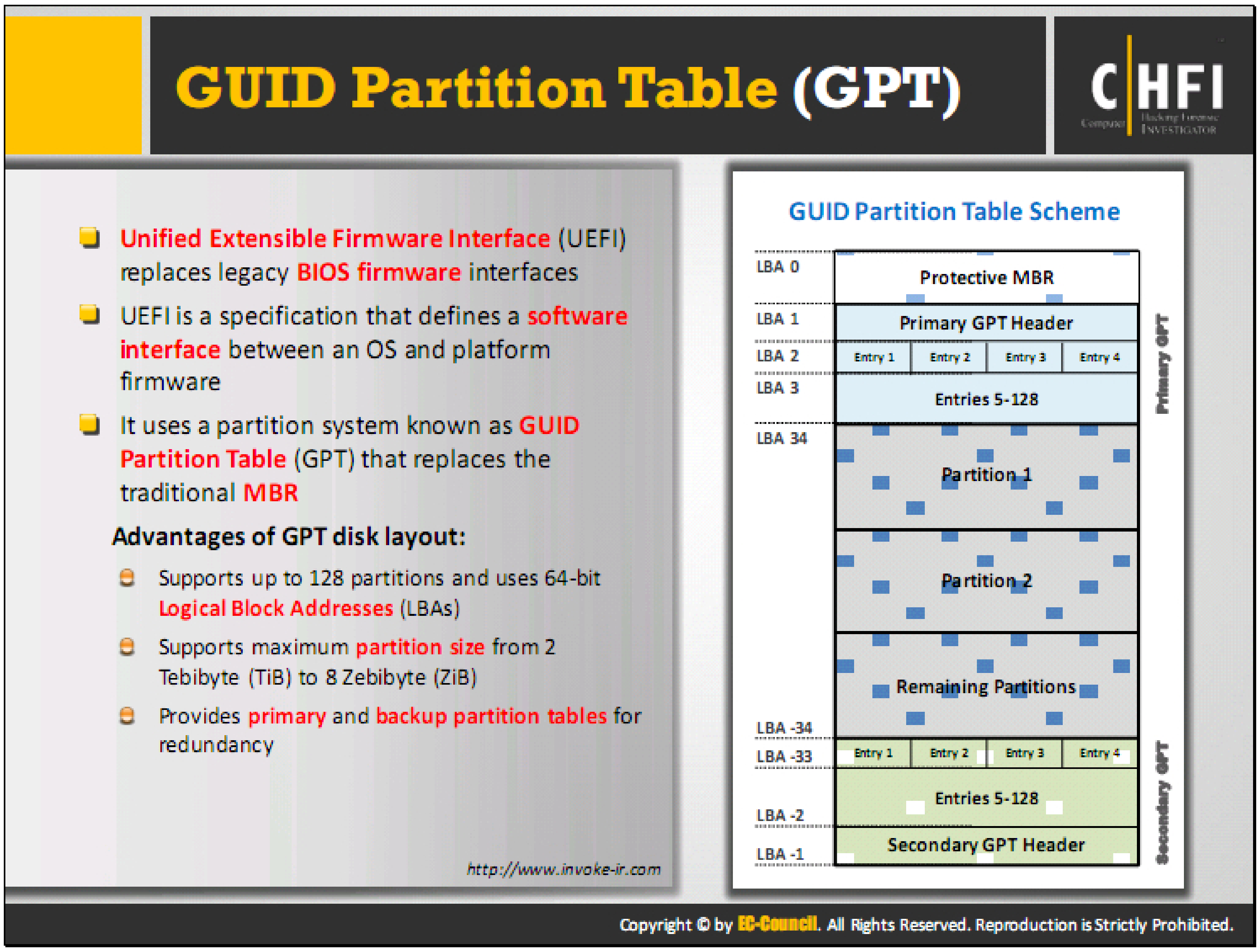
Common Uses:

- In Windows registry, GUIDs are used to identify COM DLLs
- In database tables, GUIDs are used as primary key values
- Website assigns GUID to a user's browser to record and track the session
- Windows assigns GUID to a username to identify user accounts



```
Uninstall
  {1D8E6291-B0D5-35EC-8441-6616F567A0F7}
  {5FCE6D76-F5DC-37AB-B2B8-22AB8CEDB1D4}
  {8220EEFE-38CD-377E-8595-13398D740ACE}
  {89F4137D-6C26-4A84-BDB8-2E5A4BB71E00}
  {90140000-002A-0000-1000-0000000FF1CE}
  {90140000-002A-0409-1000-0000000FF1CE}
  {90140000-0116-0409-1000-0000000FF1CE}
  {929FBD26-9020-399B-9A7A-751D61F0B942}
  {9495AEB4-AB97-39DE-8C42-806EEF75ECA7}
  {A749D8E6-B613-3BE3-8F5F-045C84EBA29B}
  {CD95F661-A5C4-44F5-A6AA-ECDD91C240ED}
  {D4682F5E-7562-4DCB-8312-B8973F255C4D}_is1
  AddressBook
  Connection Manager
  DirectDrawEx
  DXM_Runtime
  Fontcore
  IE40
  IE4Data
  IE5BAKEX
  IEData
  Microsoft Visual Studio 2010 Tools for Office Runt
  MobileOptionPack
  MPlayer2
  SchedulingAgent
  WIC
  WinRAR archiver
URL
UserPictureChange

Name                 Type        Data
(Default)            REG_SZ      (value not set)
AuthorizedCDFPrefix  REG_SZ
Comments             REG_SZ      Caution. Removing this product might prevent some applications fr
Contact              REG_SZ
DisplayName          REG_SZ      Microsoft Visual C++ 2010  x64 Redistributable - 10.0.40219
DisplayVersion       REG_SZ      10.0.40219
EstimatedSize        REG_DW...   0x00006eee (28398)
HelpLink             REG_EXP...  http://go.microsoft.com/fwlink/?LinkId=146008
HelpTelephone        REG_SZ
InstallDate          REG_SZ      20150304
InstallLocation      REG_SZ
InstallSource        REG_SZ      d:\1dfb54f1898c01c3cca46d3495b4\VC_Red_x64\
Language             REG_DW...   0x00000000 (0)
ModifyPath           REG_EXP...  MsiExec.exe /X{1D8E6291-B0D5-35EC-8441-6616F567A0F7}
NoModify             REG_DW...   0x00000001 (1)
NoRepair             REG_DW...   0x00000001 (1)
Publisher            REG_SZ      Microsoft Corporation
Readme               REG_SZ
sEstimatedSize2      REG_DW...   0x00003777 (14199)
Size                 REG_SZ
UninstallString      REG_EXP...  MsiExec.exe /X{1D8E6291-B0D5-35EC-8441-6616F567A0F7}
URLInfoAbout         REG_SZ
URLUpdateInfo        REG_SZ
Version              REG_DW...   0x0a009d1b (167812379)
VersionMajor         REG_DW...   0x0000000a (10)
VersionMinor         REG_DW...   0x00000000 (0)
WindowsInstaller     REG_DW...   0x00000001 (1)

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{1D8E6291-B0D5-35EC-8441-6616F567A0F7}
```

Globally Unique Identifier is a 128-bit unique number, generated by the Windows OS for identifying a specific device, document, a database entry, and/or the user. For example, while browsing a website generates a GUID and assigns to the browser, which will help in tracking and recording the user's browsing session. The Windows OS assigns a GUID to the registry in order to recognize COM DLLs (Dynamic Link Library) as well as to the user accounts by a username (domain).



GUID Partition Table (GPT)

- **Unified Extensible Firmware Interface** (UEFI) replaces legacy **BIOS firmware** interfaces
- UEFI is a specification that defines a **software interface** between an OS and platform firmware
- It uses a partition system known as **GUID Partition Table** (GPT) that replaces the traditional **MBR**

Advantages of GPT disk layout:

- Supports up to 128 partitions and uses 64-bit **Logical Block Addresses** (LBAs)
- Supports maximum **partition size** from 2 Tebibyte (TiB) to 8 Zebibyte (ZiB)
- Provides **primary** and **backup partition tables** for redundancy

http://www.invoke-ir.com

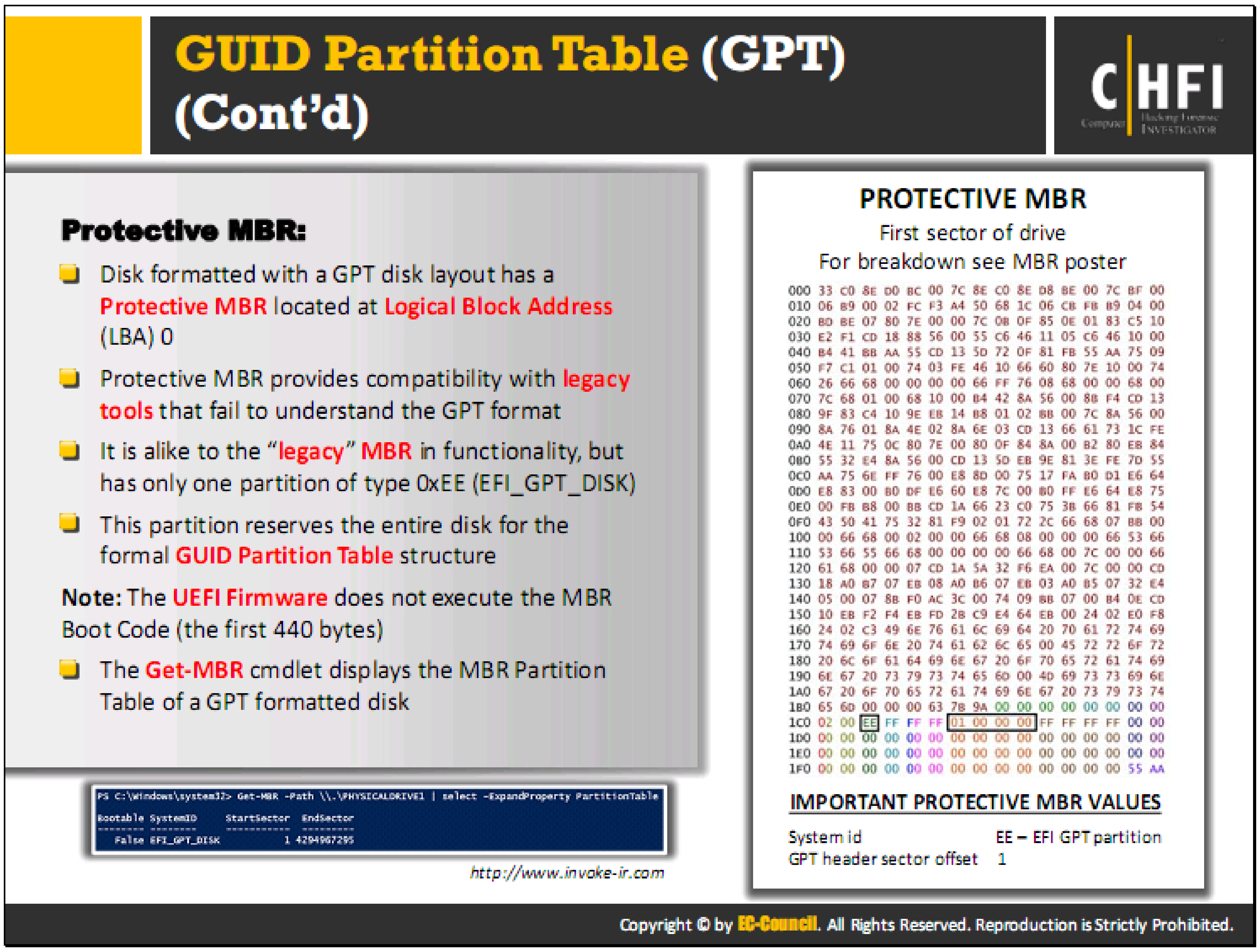
GUID is a standard partitioning scheme for hard disks and part of the Unified Extensible Firmware Interface (UEFI), which replaces legacy BIOS firmware interfaces. UEFI uses partition interfacing systems that overcome the limitations of the MBR partitioning scheme.

MBR partition scheme uses 32 bits for storing LBA (Logical Block Addresses) and the size information on 512-byte sector. In GPT, each logical block is 512 bytes and each partition entry is 128 bytes, and the negative addressing of the logical blocks starts from the end of the volume with -1 as the last addressable block. GPTs use logical block addressing (LBA) instead of the cylinder-head-sector (CHS) addressing similar to the modern MBRs. LBA 0 stores the protective MBR, LBA 1 contains the GPT header, and the GPT header comprises a pointer to the partition table or Partition Entry Array at LBA 2.

The UEFI assigns 16,384 bytes for the Partition Entry Array. Since the disk has 512-byte sectors with a partition entry array of 16,384 bytes and the minimum size of 128 bytes for each partition entry, LBA 34 will be the first usable sector.

Advantages of GPT disk layout:

- GPT allows users to partition disks larger than 2 terabytes
- It allows users to have 128 partitions in Windows using GPT partition layout
- GPT partition and boot data is more secure than MBR, as GPT stores data in multiple locations across the disk
- It uses Cyclic Redundancy Check (CRC) to ensure data integrity
- Uses CRC32 checksums that detect errors in the header and partition table



GUID Partition Table (GPT) (Cont'd)

Protective MBR:

- Disk formatted with a GPT disk layout has a **Protective MBR** located at **Logical Block Address** (LBA) 0
- Protective MBR provides compatibility with **legacy tools** that fail to understand the GPT format
- It is alike to the "**legacy**" **MBR** in functionality, but has only one partition of type 0xEE (EFI_GPT_DISK)
- This partition reserves the entire disk for the formal **GUID Partition Table** structure

Note: The **UEFI Firmware** does not execute the MBR Boot Code (the first 440 bytes)

- The **Get-MBR** cmdlet displays the MBR Partition Table of a GPT formatted disk

```
PS C:\Windows\system32> Get-MBR -Path \\.\PHYSICALDRIVE1 | select -ExpandProperty PartitionTable

Bootable SystemID     StartSector  EndSector
-------- --------     -----------  ---------
   False EFI_GPT_DISK           1 4294967295
```

http://www.invoke-ir.com

PROTECTIVE MBR

First sector of drive

For breakdown see MBR poster

```
000 33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00
010 06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00
020 BD BE 07 80 7E 00 00 7C 0B 0F 85 0E 01 83 C5 10
030 E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00
040 B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09
050 F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74
060 26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00
070 7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13
080 9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00
090 8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE
0A0 4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84
0B0 55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55
0C0 AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64
0D0 E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75
0E0 00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54
0F0 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00
100 00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66
110 53 66 55 66 68 00 00 00 00 66 68 00 7C 00 00 66
120 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD
130 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4
140 05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD
150 10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8
160 24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69
170 74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72
180 20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69
190 6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E
1A0 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74
1B0 65 6D 00 00 00 63 7B 9A 00 00 00 00 00 00 00 00
1C0 02 00 EE FF FF FF 01 00 00 00 FF FF FF FF 00 00
1D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
```

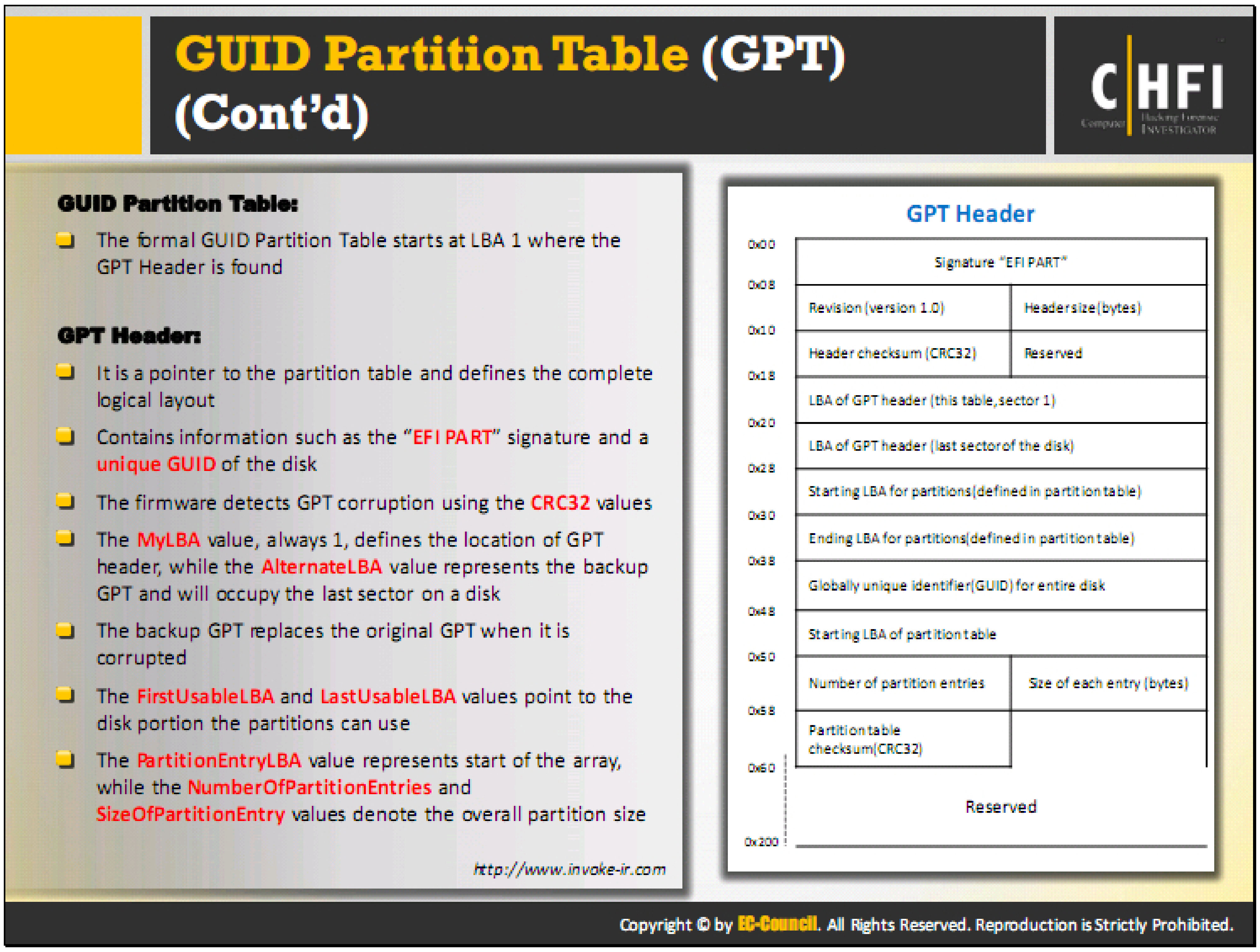
IMPORTANT PROTECTIVE MBR VALUES

System id	EE – EFI GPT partition
GPT header sector offset	1

Protective MBR

Protective MBR occupies the first position of the GPT at Logical Block Address (LBA) 0. It helps the legacy issues to solve compatibility issues when they fail to understand the GPT format. It stores the startup code for the operating systems that support GPT boot disk. It will make sure that the operating systems, which are unable to identify the GPT disk, will mark it as unknown, and cannot delete without user command. Additionally, the operating systems identifying the GPT partition table will also check the protective MBR before while starting the operations.

Being similar to the legacy MBR in functionality, the main difference is that the protective MBR has only one partition of type 0xEE (EFI_GPT_DISK). If the partition is not of 0xEE type or the MBR partition table consists of multiple entries, the MBR will not operate.



GUID Partition Table (GPT) (Cont'd)

GUID Partition Table:

- The formal GUID Partition Table starts at LBA 1 where the GPT Header is found

GPT Header:

- It is a pointer to the partition table and defines the complete logical layout
- Contains information such as the "EFI PART" signature and a unique GUID of the disk
- The firmware detects GPT corruption using the CRC32 values
- The MyLBA value, always 1, defines the location of GPT header, while the AlternateLBA value represents the backup GPT and will occupy the last sector on a disk
- The backup GPT replaces the original GPT when it is corrupted
- The FirstUsableLBA and LastUsableLBA values point to the disk portion the partitions can use
- The PartitionEntryLBA value represents start of the array, while the NumberOfPartitionEntries and SizeOfPartitionEntry values denote the overall partition size

http://www.invoke-ir.com

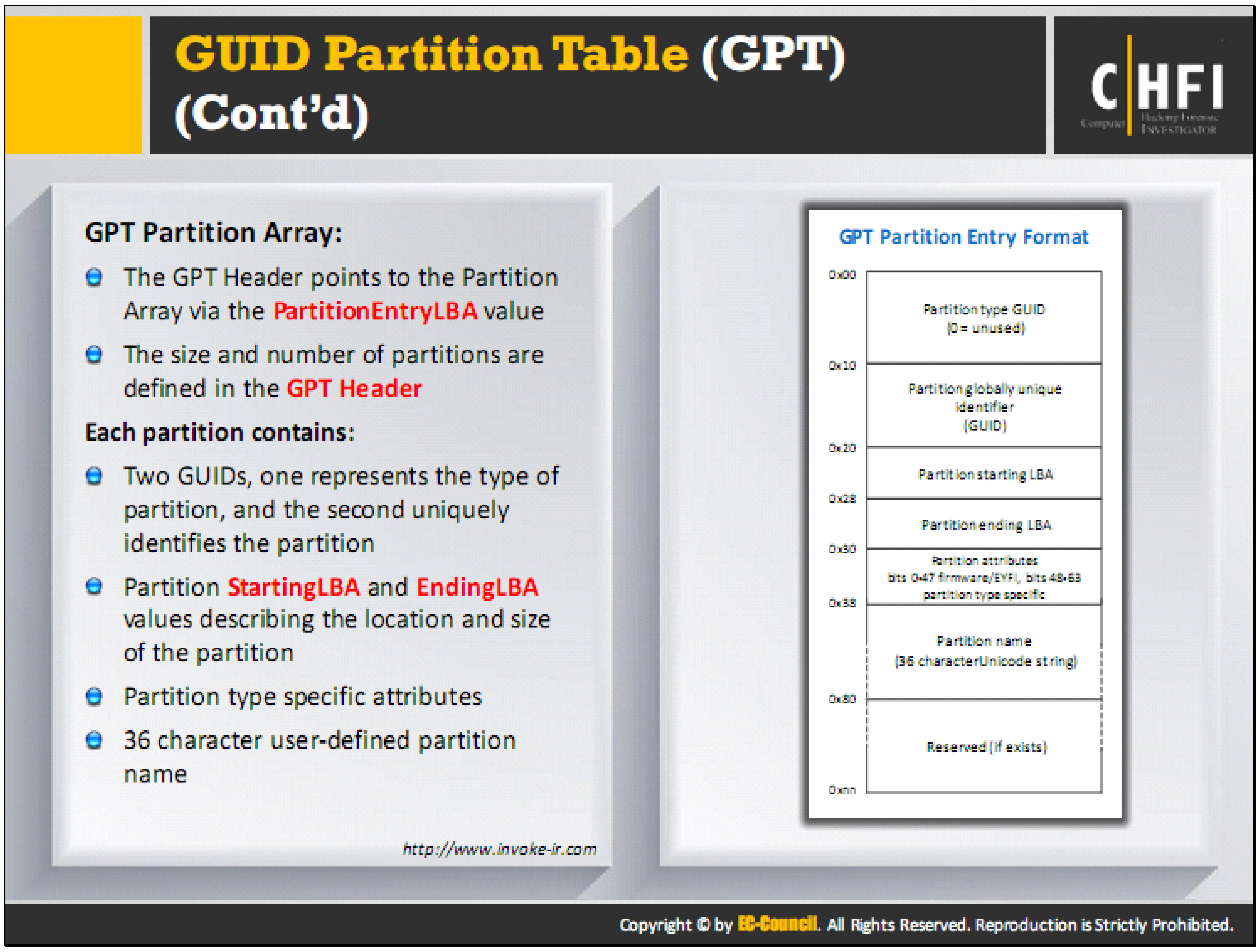
GPT Header

0x00	Signature "EFI PART"	
0x08	Revision (version 1.0)	Header size (bytes)
0x10	Header checksum (CRC32)	Reserved
0x18	LBA of GPT header (this table, sector 1)	
0x20	LBA of GPT header (last sector of the disk)	
0x28	Starting LBA for partitions (defined in partition table)	
0x30	Ending LBA for partitions (defined in partition table)	
0x38	Globally unique identifier (GUID) for entire disk	
0x48	Starting LBA of partition table	
0x50	Number of partition entries	Size of each entry (bytes)
0x58	Partition table checksum (CRC32)	
0x60	Reserved	
0x200		

GPT header represents the formal beginning of the partition table, the first accessible disk space, and structure of the partition table.

A hard disk running Windows 64 bit operating system can have up to 128 partitions with a size of 128 bytes each. Partition table header stores the disk details such as partition ID of GPT disk, the partition table header size and location, backup partition table header, position and size of the partition table.

Additionally, it stores the partition table CRC32, which is cyclic redundancy check or an error-detecting code used to detect accidental changes. The checking mechanism helps boot program, firmware, and the operating system to detect the error in a partition table. If any error occurs in the primary GPT, the users can recover the hard disk partitions from the secondary GPT, but the error in backup GPT parity can make the hard disk inaccessible.



GUID Partition Table (GPT) (Cont'd)

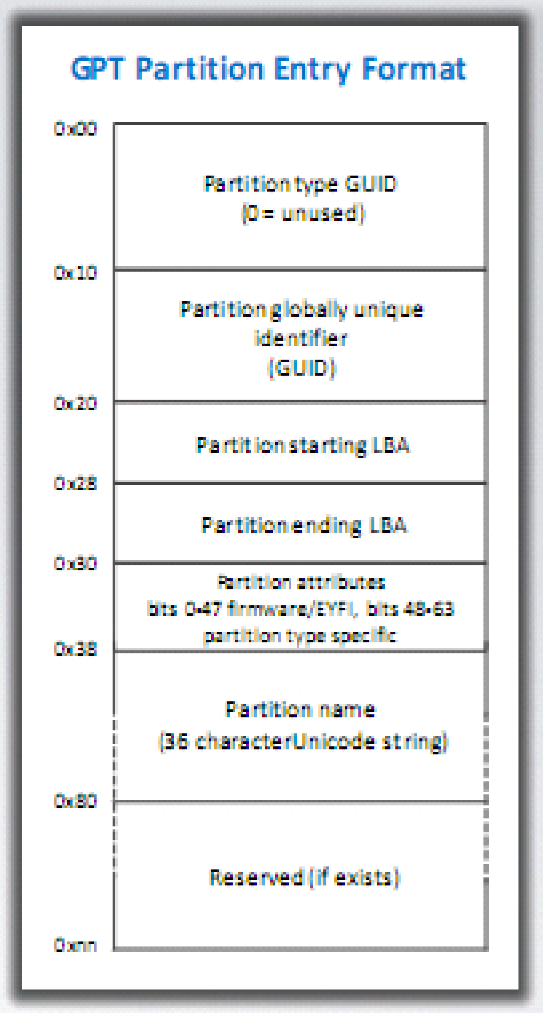
GPT Partition Array:

- The GPT Header points to the Partition Array via the **PartitionEntryLBA** value
- The size and number of partitions are defined in the **GPT Header**

Each partition contains:

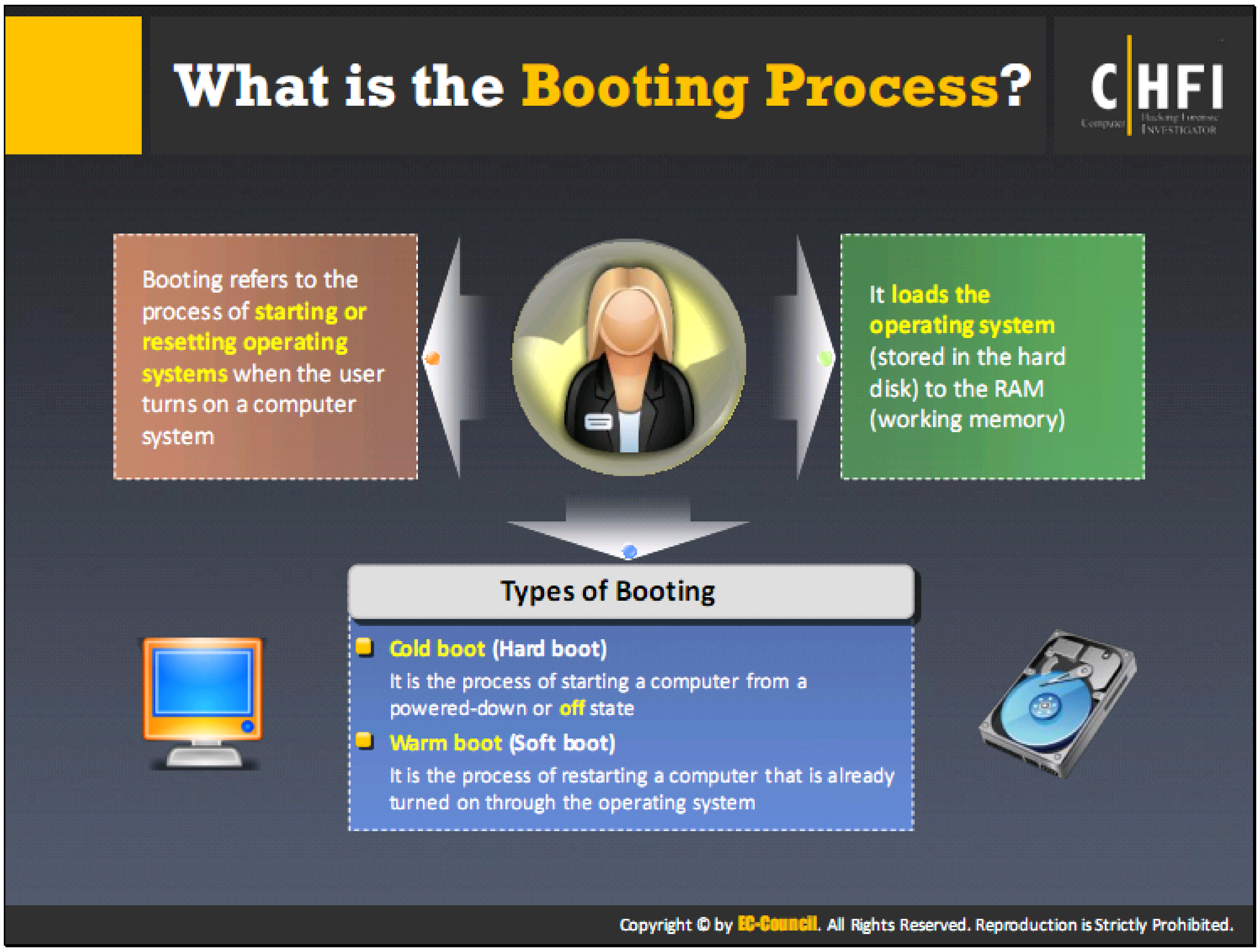
- Two GUIDs, one represents the type of partition, and the second uniquely identifies the partition
- Partition **StartingLBA** and **EndingLBA** values describing the location and size of the partition
- Partition type specific attributes
- 36 character user-defined partition name

http://www.invoke-ir.com


GPT Partition Entry Format
0x00
Partition type GUID (0 = unused)
0x10
Partition globally unique identifier (GUID)
0x20
Partition starting LBA
0x28
Partition ending LBA
0x30
Partition attributes bits 0-47 firmware/EYFI, bits 48-63 partition type specific
0x38
Partition name (36 characterUnicode string)
0x80
Reserved (if exists)
0xnn

Partition entry array is present next to the GPT header and uses 128 byte blocks per entry, of which the first 16 bytes of each block represent the partition type GUID. The next 16 bytes contain GUID specific to the partition block. Because of the unique nature of the GUID, there is no requirement for a central registry for the GUID partition type designator.

It is not necessary for each sector to be restricted to 512 bytes, (i.e. 3 primary partitions and 1 extended partition) it can have more than four partition entries in a single sector. The GPT specification describes the size and organization of the data structure on the whole (keeping aside LBA 0 and LBA 1), but not counts the number of sectors stored on the disk.



What is the Booting Process?

Booting refers to the process of starting or resetting operating systems when the user turns on a computer system

It loads the operating system (stored in the hard disk) to the RAM (working memory)

Types of Booting

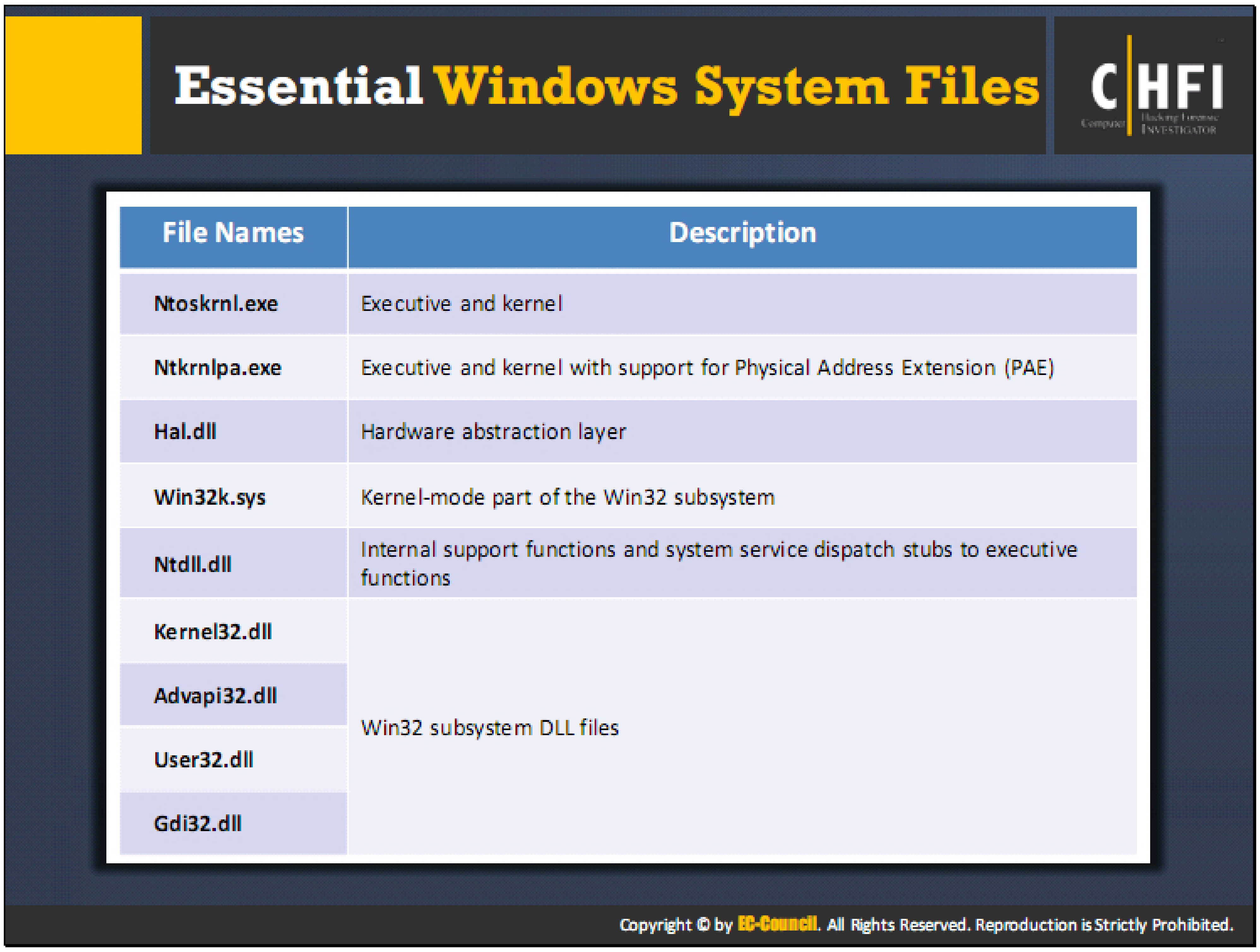
- **Cold boot (Hard boot)**
  It is the process of starting a computer from a powered-down or off state
- **Warm boot (Soft boot)**
  It is the process of restarting a computer that is already turned on through the operating system

Booting is the process of starting or resetting the computer when the user turns the system on. The process includes getting both the hardware and software ready and running. The booting process is of two types:

- **Cold booting:** The process happening when we first turn on the computer. Also called as hard boot, this happens when user completely cuts the power supply to the system.
- **Warm booting** is the process happening when we reset the computer. In this process, the user restarts the system via operating system.

During the process of booting, the computer loads the operating system to its memory or RAM and prepares it for use. During initialization, the system switches on the BIOS and loads it onto the ROM. BIOS stores the first instruction, which is the command to perform the power-on self-test (POST). Under POST, the system checks the BIOS chip and CMOS RAM.

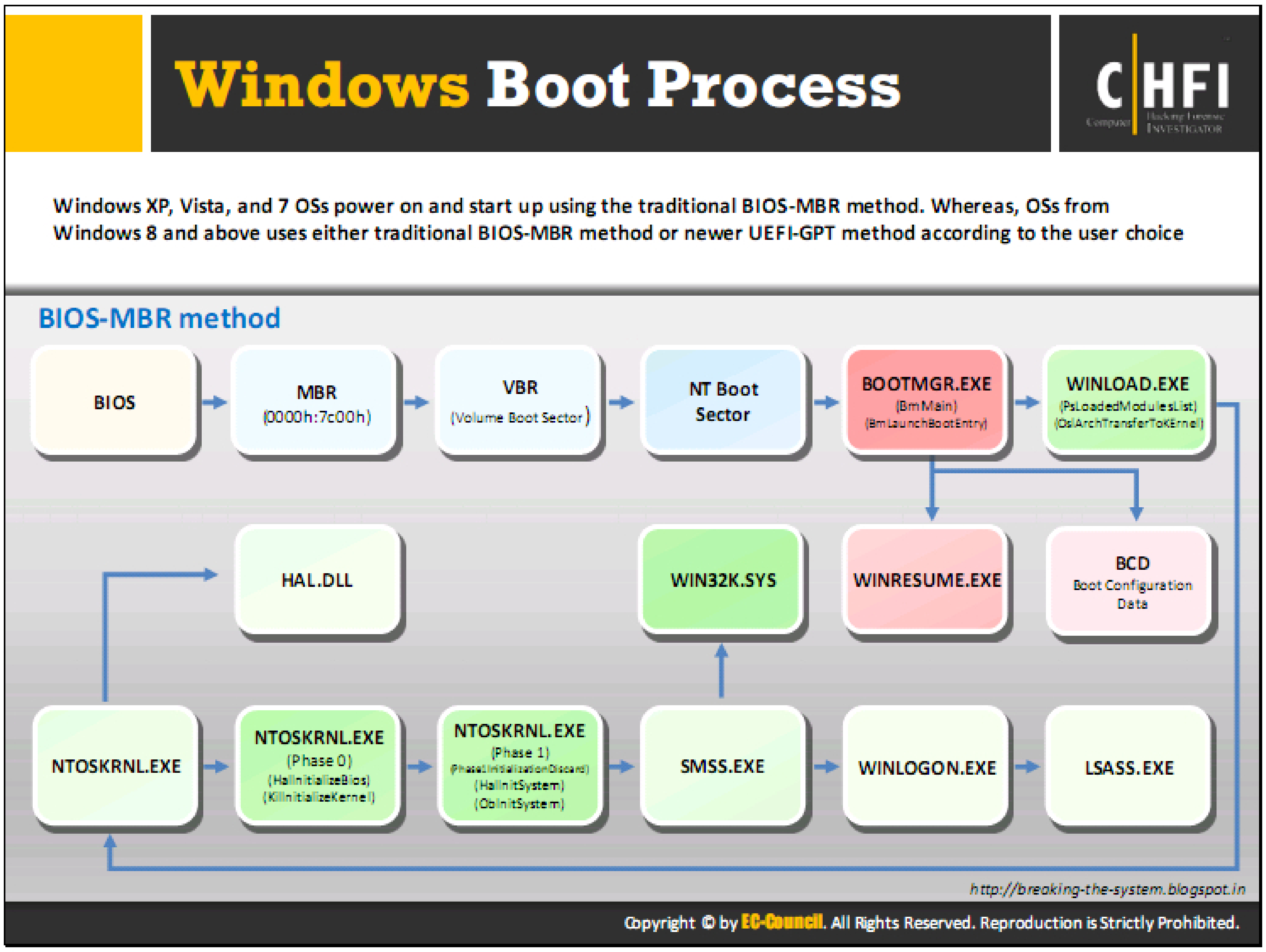
If the POST detects no battery failure, it continues to start other parts of the CPU by checking the hardware devices and secondary storage devices.



Essential Windows System Files

File Names	Description
Ntoskrnl.exe	Executive and kernel
Ntkrnlpa.exe	Executive and kernel with support for Physical Address Extension (PAE)
Hal.dll	Hardware abstraction layer
Win32k.sys	Kernel-mode part of the Win32 subsystem
Ntdll.dll	Internal support functions and system service dispatch stubs to executive functions
Kernel32.dll	Win32 subsystem DLL files
Advapi32.dll	
User32.dll	
Gdi32.dll	

After installation of an operating system, the setup program creates folders and required files on the system drive. The following are the essential Windows system files.



Windows Boot Process

Windows XP, Vista, and 7 OSs power on and start up using the traditional BIOS-MBR method. Whereas, OSs from Windows 8 and above uses either traditional BIOS-MBR method or newer UEFI-GPT method according to the user choice

BIOS-MBR method
BIOS
MBR (0000h:7c00h)
VBR (Volume Boot Sector)
NT Boot Sector
BOOTMGR.EXE (BmMain) (BmLaunchBootEntry)
WINLOAD.EXE (PsLoadedModulesList) (OslArchTransferToKernel)
HAL.DLL
WIN32K.SYS
WINRESUME.EXE
BCD Boot Configuration Data
NTOSKRNL.EXE
NTOSKRNL.EXE (Phase 0) (HalInitializeBios) (KiInitializeKernel)
NTOSKRNL.EXE (Phase 1) (Phase1InitializationDiscard) (HalInitSystem) (ObInitSystem)
SMSS.EXE
WINLOGON.EXE
LSASS.EXE
http://breaking-the-system.blogspot.in

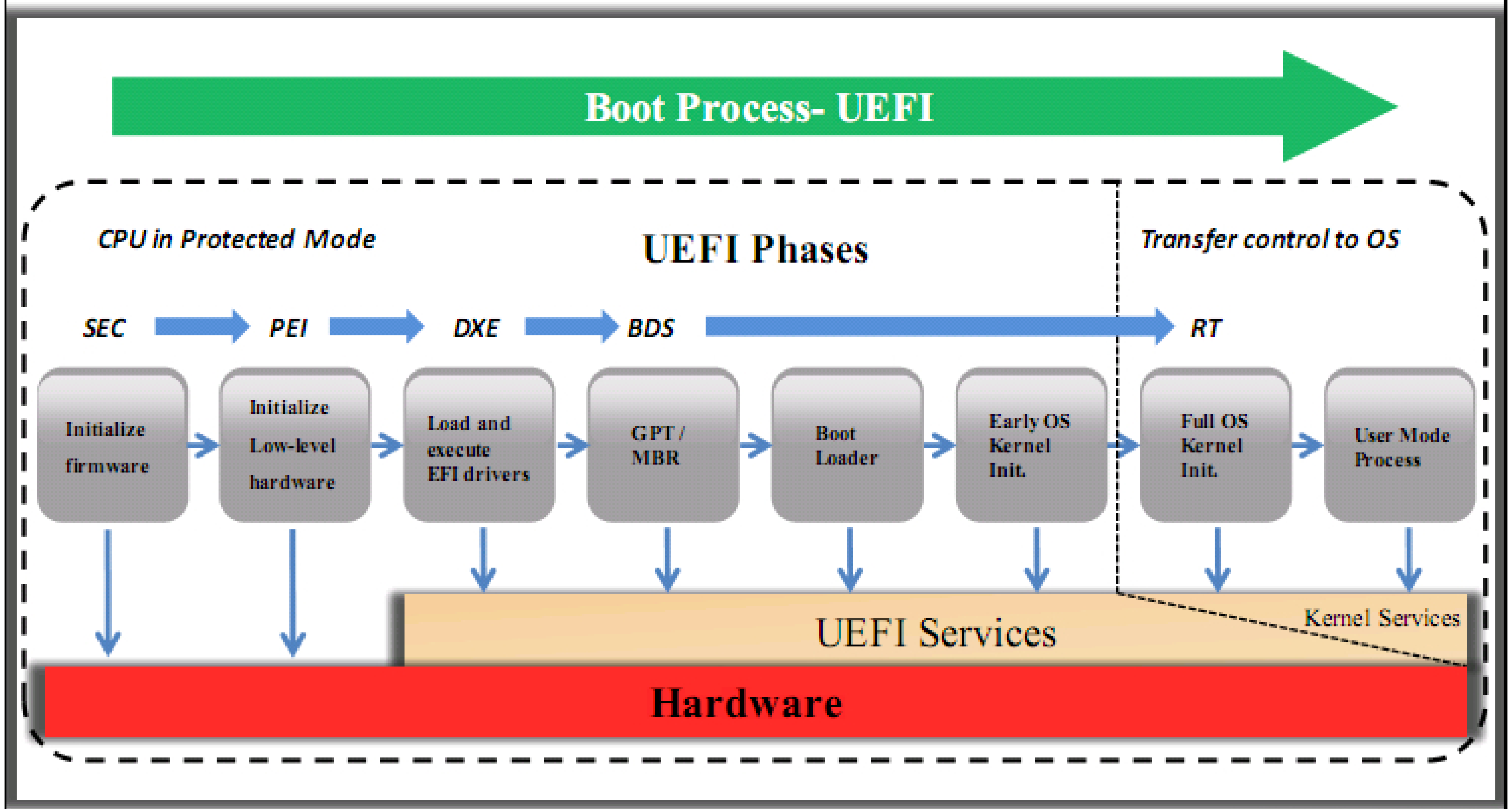
Windows XP, Vista, and 7 OSs power on and start up using the traditional BIOS-MBR method. Whereas, the Microsoft operating systems starting with Windows 8 and later versions will use either traditional BIOS-MBR method or newer UEFI-GPT method according to the user choice.

Below is process that occurs within the system when switched ON.

1. When the user switches the system **ON**, CPU sends a Power Good signal to motherboard and checks for **computer's BIOS firmware**.
2. BIOS starts a **Power-On Self-Test** (**POST**) which checks if all the hardware required for system boot are available and load all the firmware settings from nonvolatile memory on the motherboard.
3. If POST is successful, **add-on adapters** perform a self-test for integration with the system.
4. The **pre-boot process** will complete with POST, detecting a valid system boot disk.
5. After POST, the computer's firmware scans boot disk and loads the **master boot record** (**MBR**), which search for basic boot information in Boot Configuration Data (BCD).
6. MBR triggers **Bootmgr.exe,** which locates **Windows loader** (**Winload.exe**) on the Windows boot partition and triggers **Winload.exe**.
7. Windows loader loads the OS kernel **ntoskrnl.exe**.

8. Once the Kernel starts running, the Windows loader loads HAL.DLL, boot-class device drivers marked as **BOOT_START** and the **SYSTEM registry hive** into the memory.
9. Kernel passes the control of boot process to the **Session Manager Process (SMSS.exe),** which loads all other registry hives and drivers required to configure Win32 subsystem run environment.
10. **Session Manager Process** triggers **Winlogon.exe,** which presents the user logon screen for user authorization.
11. **Session Manager Process** initiates **Service control manager,** which starts all the services, rest of the non-essential device drivers, the security subsystem **LSASS.EX**E and Group policy scripts.
12. Once user **logs in**, Windows creates a session for the user.
13. **Service control manager** starts the **Explorer.exe** and initiates the **Desktop Window Manager** (**DMW**) process, which set the desktop for the user.

Windows Boot Process (Cont'd)



http://answers.microsoft.com

The EFI boot manager controls the UEFI boot process. It starts with platform firmware initialization; the boot manager loads UEFI drivers and UEFI applications (including UEFI OS boot loaders) to initialize platform functions. The system loads the OS loader at the final stage and then OS starts booting. Once the OS receives the controls, it halts the UEFI boot service.

The UEFI boot process has five phases and each phase has its own role. These five phases are:

- **SEC (Security) Phase**

  This phase of EFI consists of initialization code that the system executes after powering the EFI system on. It manages platform reset events and sets the system so that it can find, validate, install, and run the PEI.

- **PEI (Pre-EFI Initialization) Phase**

  The PEI phase initializes the CPU, temporary memory, and boot firmware volume (BFV). It locates and executes the Pre Initialization modules (PEIMs) present in the BFV so as to initialize all the found hardware in the system. Finally, it creates a Hand-Off Block List with all found resources interface descriptors and passes it to the next phase i.e. the DXE phase.

- **DXE (Driver Execution Environment) Phase**

  Most of the initialization happens in this phase. Using the Hand-Off Block List (HOBL), it initializes the entire system physical memory, I/O, and MIMO (Memory Mapped Input Output) resources and finally begins dispatching DXE Drivers present in the system

Firmware Volumes (given in the HOBL). The DXE core produces a set of EFI Boot Services and EFI Runtime Services. The EFI Boot Services provided are allocating memory and loading executable images. The EFI Runtime services provided are converting memory addresses from physical to virtual while handing over to the kernel, and resetting the CPU, to code running within the EFI environment or within the OS kernel once the CPU takes the control of the system.

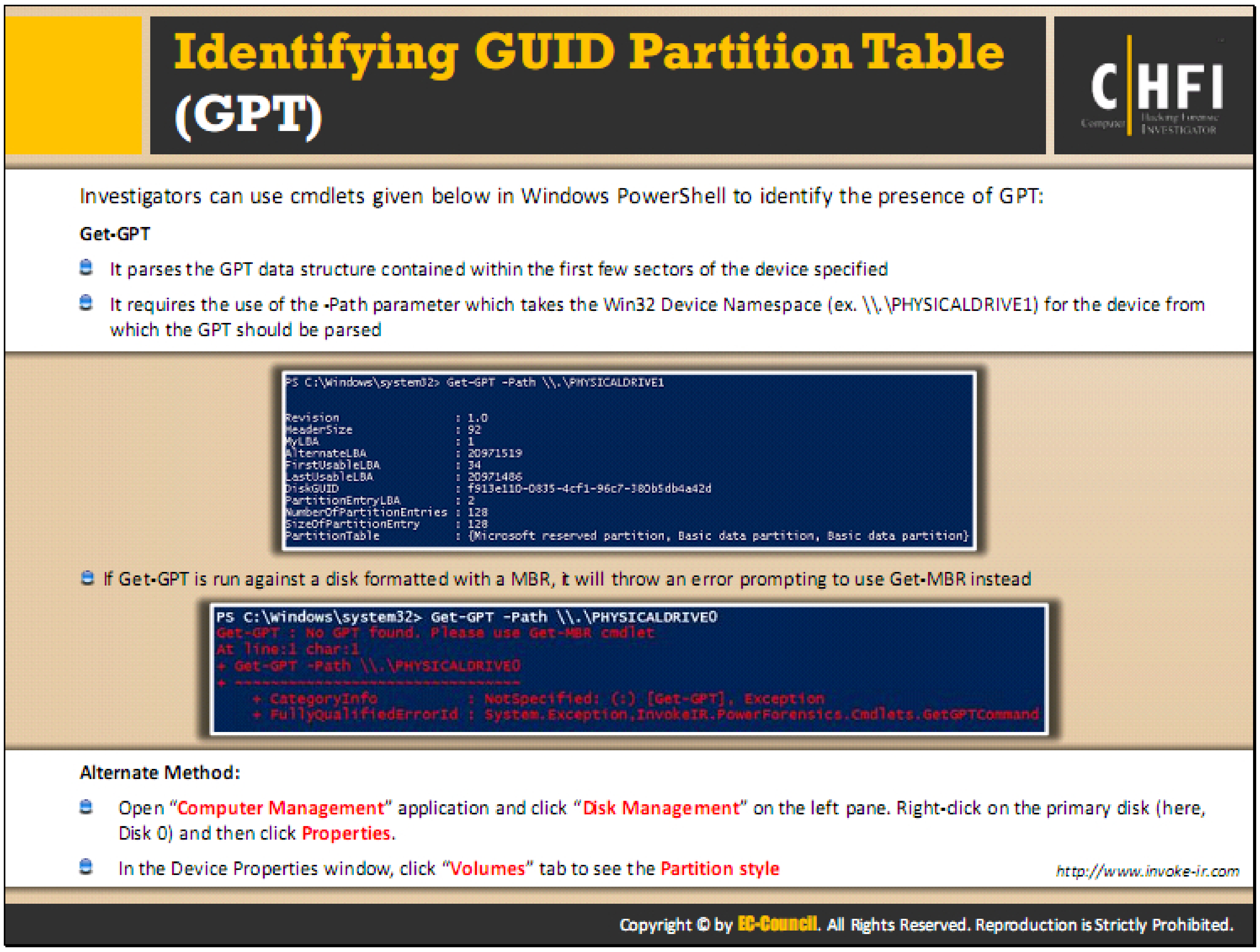
- **BDS (Boot Device Selection) Phase**

  In this phase, the BDS interprets the boot configuration data and selects the Boot Policy for later implementation. This phase works with the DXE to check if the device drivers require signature verification.

  In this phase, the system loads MBR boot code into memory for Legacy BIOS Boot or loads the Bootloader program from the EFI partition for UEFI Boot. It also provides an option for the user to choose EFI Shell or an UEFI application as the Boot Device from the Setup.

- **RT (Run Time) Phase**

  At this point, the system clears the UEFI program from memory and transfers it to the OS. During UEFI BIOS update, the OS calls the run time service using a small part of the memory.



Identifying GUID Partition Table (GPT)

Investigators can use cmdlets given below in Windows PowerShell to identify the presence of GPT:

Get-GPT

- It parses the GPT data structure contained within the first few sectors of the device specified
- It requires the use of the -Path parameter which takes the Win32 Device Namespace (ex. \\.\PHYSICALDRIVE1) for the device from which the GPT should be parsed

```
PS C:\Windows\system32> Get-GPT -Path \\.\PHYSICALDRIVE1


Revision                 : 1.0
HeaderSize               : 92
MyLBA                    : 1
AlternateLBA             : 20971519
FirstUsableLBA           : 34
LastUsableLBA            : 20971486
DiskGUID                 : f913e110-0835-4cf1-96c7-380b5db4a42d
PartitionEntryLBA        : 2
NumberOfPartitionEntries : 128
SizeOfPartitionEntry     : 128
PartitionTable           : {Microsoft reserved partition, Basic data partition, Basic data partition}
```

- If Get-GPT is run against a disk formatted with a MBR, it will throw an error prompting to use Get-MBR instead

```
PS C:\Windows\system32> Get-GPT -Path \\.\PHYSICALDRIVE0
Get-GPT : No GPT found. Please use Get-MBR cmdlet
At line:1 char:1
+ Get-GPT -Path \\.\PHYSICALDRIVE0
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Get-GPT], Exception
    + FullyQualifiedErrorId : System.Exception,InvokeIR.PowerForensics.Cmdlets.GetGPTCommand
```

Alternate Method:

- Open "Computer Management" application and click "Disk Management" on the left pane. Right-click on the primary disk (here, Disk 0) and then click Properties.
- In the Device Properties window, click "Volumes" tab to see the Partition style

http://www.invoke-ir.com

A GPT header will help an investigator analyze the layout of the disk including the locations of the partition table, partition area, and backup copies of the header and partition table. Investigators can use cmdlets given below in Windows PowerShell to identify the presence of GPT:

Get-GPT

Get-GPT command helps investigator to analyze the GUID Partition Table data structure of the hard disk. It requires the use of the -Path parameter which takes the Win32 Device Namespace (ex. \\.\PHYSICALDRIVE1) for the device from which it should parse the GPT.

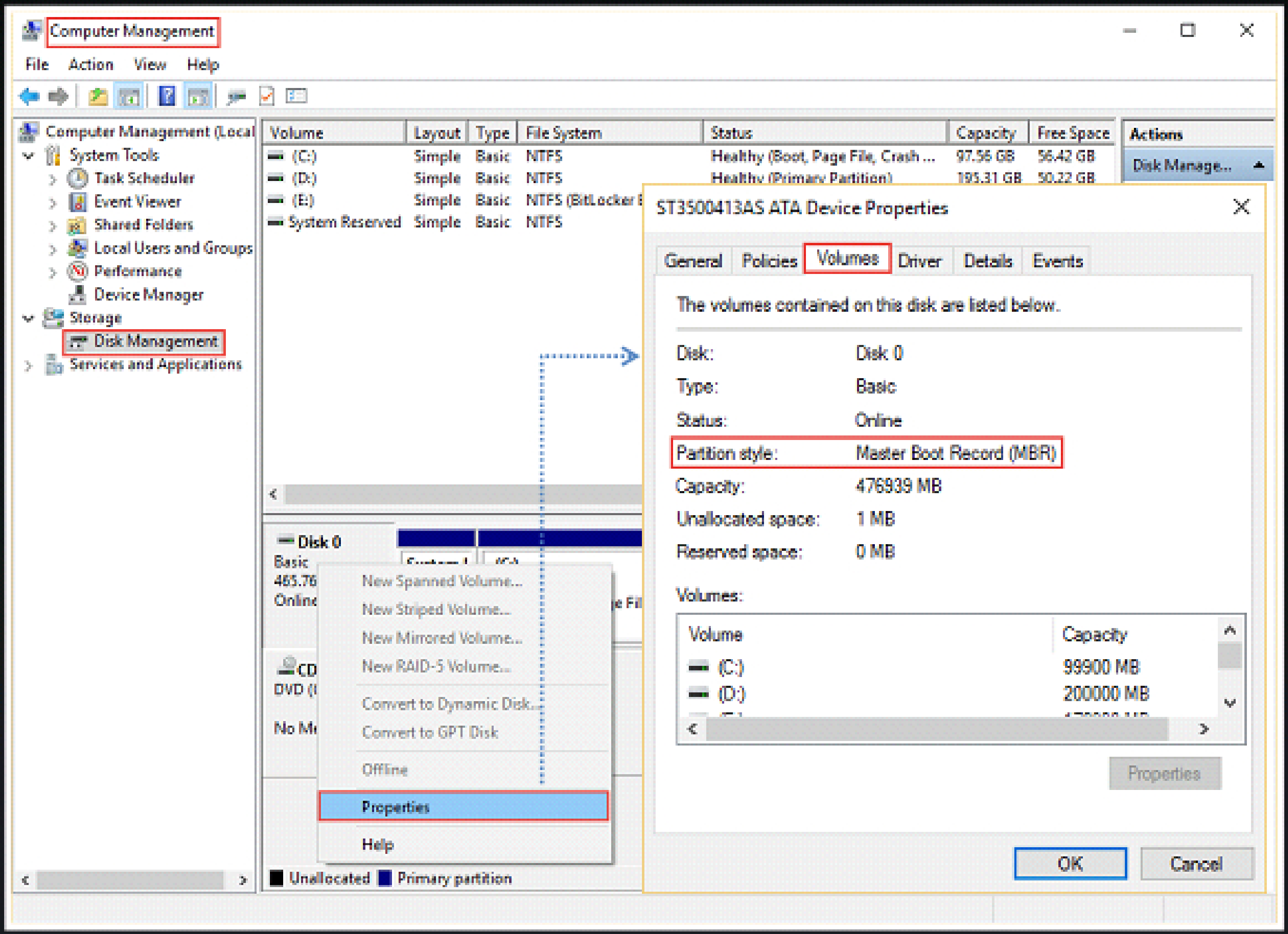
In case, the investigator uses the Get-GPT on a disk formatted with a Master Boot Record, it will display an error message prompting to use Get-MBR instead.

Alternate Method:

- Open "**Computer Management**" application and click "**Disk Management**" on the left pane. Right-click on the primary disk (here, Disk 0) and then click **Properties**
- In the Device Properties window, click "**Volumes**" tab to see the **Partition style**

Source: http://www.invoke-ir.com

Identifying GUID Partition Table (GPT) (Cont'd)



Volume	Layout	Type	File System	Status	Capacity	Free Space
(C:)	Simple	Basic	NTFS	Healthy (Boot, Page File, Crash ...	97.56 GB	56.42 GB
(D:)	Simple	Basic	NTFS	Healthy (Primary Partition)	195.31 GB	50.22 GB
(E:)	Simple	Basic	NTFS (BitLocker E			
System Reserved	Simple	Basic	NTFS			

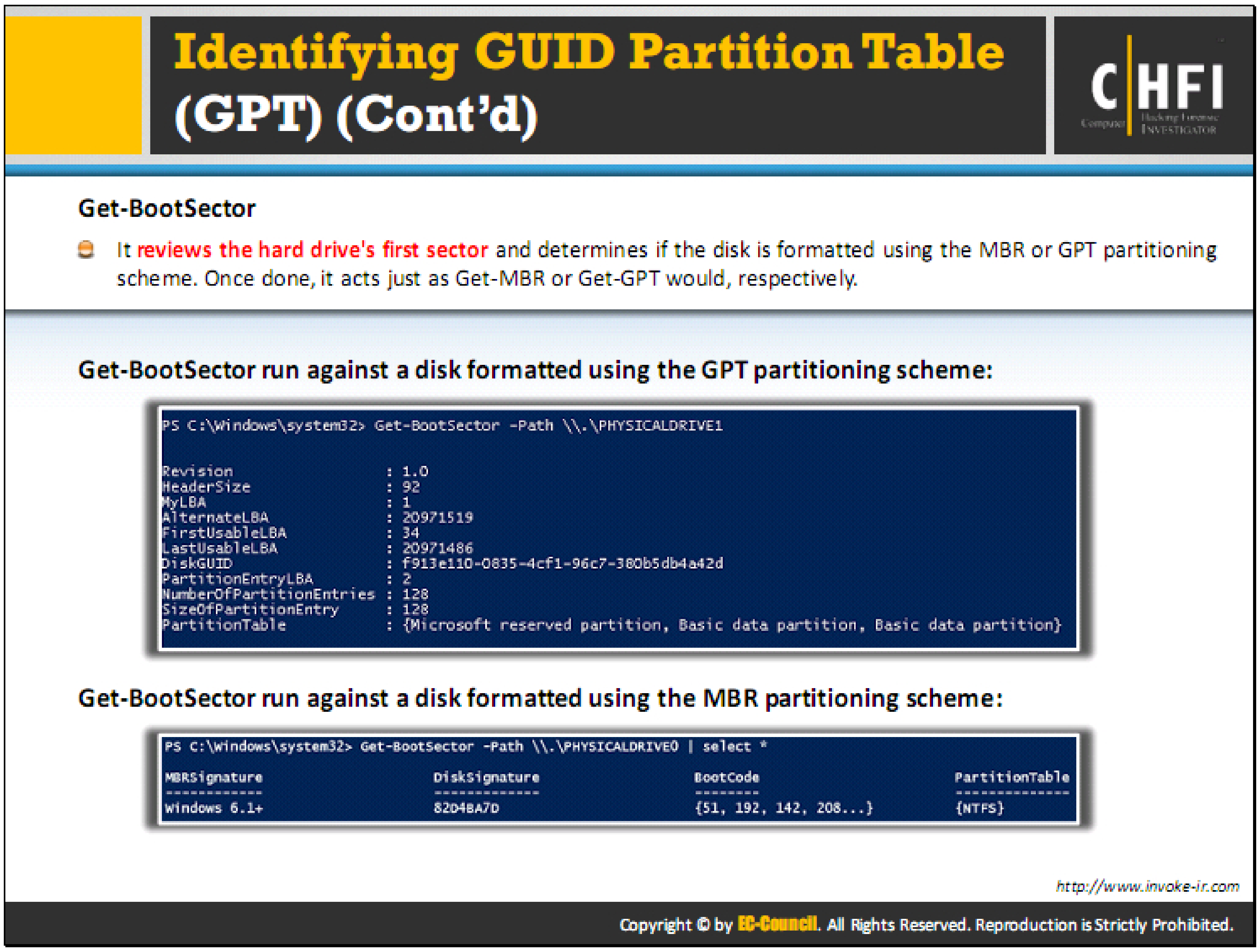
ST3500413AS ATA Device Properties

The volumes contained on this disk are listed below.

Disk:	Disk 0
Type:	Basic
Status:	Online
Partition style:	Master Boot Record (MBR)
Capacity:	476939 MB
Unallocated space:	1 MB
Reserved space:	0 MB

Volumes:

Volume	Capacity
(C:)	99900 MB
(D:)	200000 MB



Identifying GUID Partition Table (GPT) (Cont'd)

Get-BootSector

- It reviews the hard drive's first sector and determines if the disk is formatted using the MBR or GPT partitioning scheme. Once done, it acts just as Get-MBR or Get-GPT would, respectively.

Get-BootSector run against a disk formatted using the GPT partitioning scheme:

```
PS C:\Windows\system32> Get-BootSector -Path \\.\PHYSICALDRIVE1


Revision                 : 1.0
HeaderSize               : 92
MyLBA                    : 1
AlternateLBA             : 20971519
FirstUsableLBA           : 34
LastUsableLBA            : 20971486
DiskGUID                 : f913e110-0835-4cf1-96c7-380b5db4a42d
PartitionEntryLBA        : 2
NumberOfPartitionEntries : 128
SizeOfPartitionEntry     : 128
PartitionTable           : {Microsoft reserved partition, Basic data partition, Basic data partition}
```

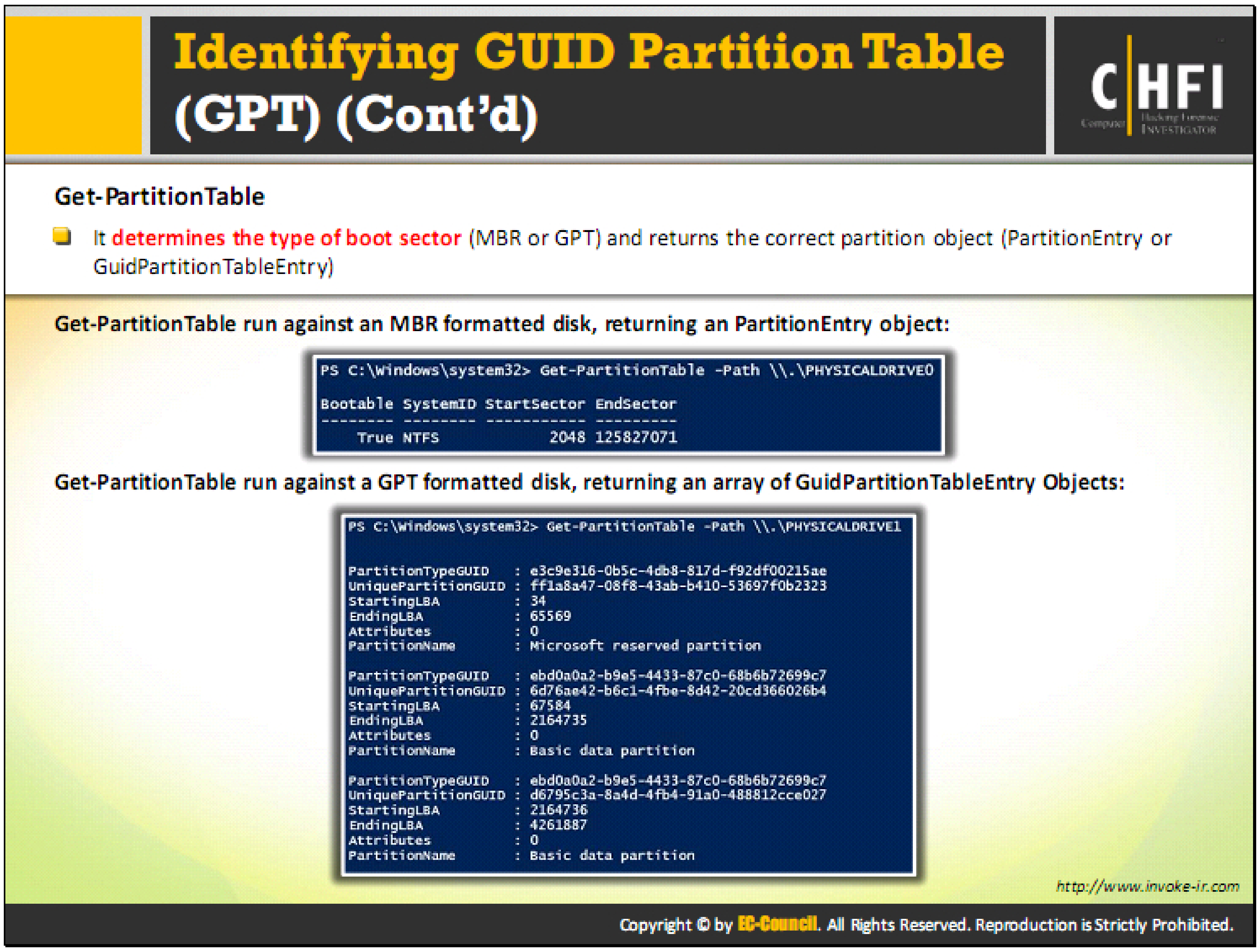
Get-BootSector run against a disk formatted using the MBR partitioning scheme:

```
PS C:\Windows\system32> Get-BootSector -Path \\.\PHYSICALDRIVE0 | select *

MBRSignature                 DiskSignature                 BootCode                      PartitionTable
------------                 -------------                 --------                      --------------
Windows 6.1+                 82D4BA7D                      {51, 192, 142, 208...}        {NTFS}
```

http://www.invoke-ir.com

Get-Boot Sector

The Get-BootSector is a command that can help the investigator parse GPTs of both types of hard disks including the ones formatted with either UEFI or MBR. This command acts as replacement for Get-MBR and Get-GPT cmdlets. Get-BootSector analyzes the first sector of hard drive and determines the formatting type used and then parses the hard drive GPT.



Identifying GUID Partition Table (GPT) (Cont'd)

Get-PartitionTable

- It determines the type of boot sector (MBR or GPT) and returns the correct partition object (PartitionEntry or GuidPartitionTableEntry)

Get-PartitionTable run against an MBR formatted disk, returning an PartitionEntry object:

```
PS C:\Windows\system32> Get-PartitionTable -Path \\.\PHYSICALDRIVE0

Bootable SystemID StartSector EndSector
-------- -------- ----------- ---------
    True NTFS            2048 125827071
```

Get-PartitionTable run against a GPT formatted disk, returning an array of GuidPartitionTableEntry Objects:

```
PS C:\Windows\system32> Get-PartitionTable -Path \\.\PHYSICALDRIVE1


PartitionTypeGUID   : e3c9e316-0b5c-4db8-817d-f92df00215ae
UniquePartitionGUID : ff1a8a47-08f8-43ab-b410-53697f0b2323
StartingLBA         : 34
EndingLBA           : 65569
Attributes          : 0
PartitionName       : Microsoft reserved partition

PartitionTypeGUID   : ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
UniquePartitionGUID : 6d76ae42-b6c1-4fbe-8d42-20cd366026b4
StartingLBA         : 67584
EndingLBA           : 2164735
Attributes          : 0
PartitionName       : Basic data partition

PartitionTypeGUID   : ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
UniquePartitionGUID : d6795c3a-8a4d-4fb4-91a0-488812cce027
StartingLBA         : 2164736
EndingLBA           : 4261887
Attributes          : 0
PartitionName       : Basic data partition
```

http://www.invoke-ir.com

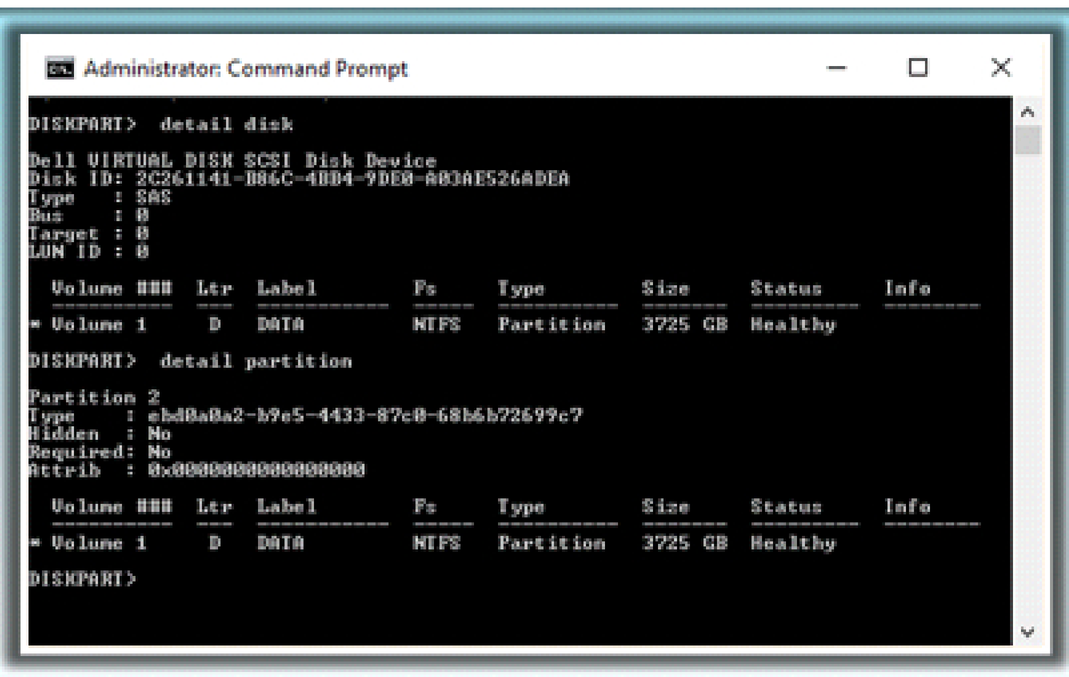
Get-PartitionTable

This command analyzes the GUID partition table to find the exact type of boot sector (Master Boot Record or GUID PartitionTable) and displays the partition object.

Analyzing the GPT Header and Entries

- Most of the operating systems that support GPT disk access come up with a basic partitioning tool, which **displays details about GPTs**

  E.g.: DiskPart tool (Windows), OS X Disk utility (Mac), GNU parted tool (Linux)

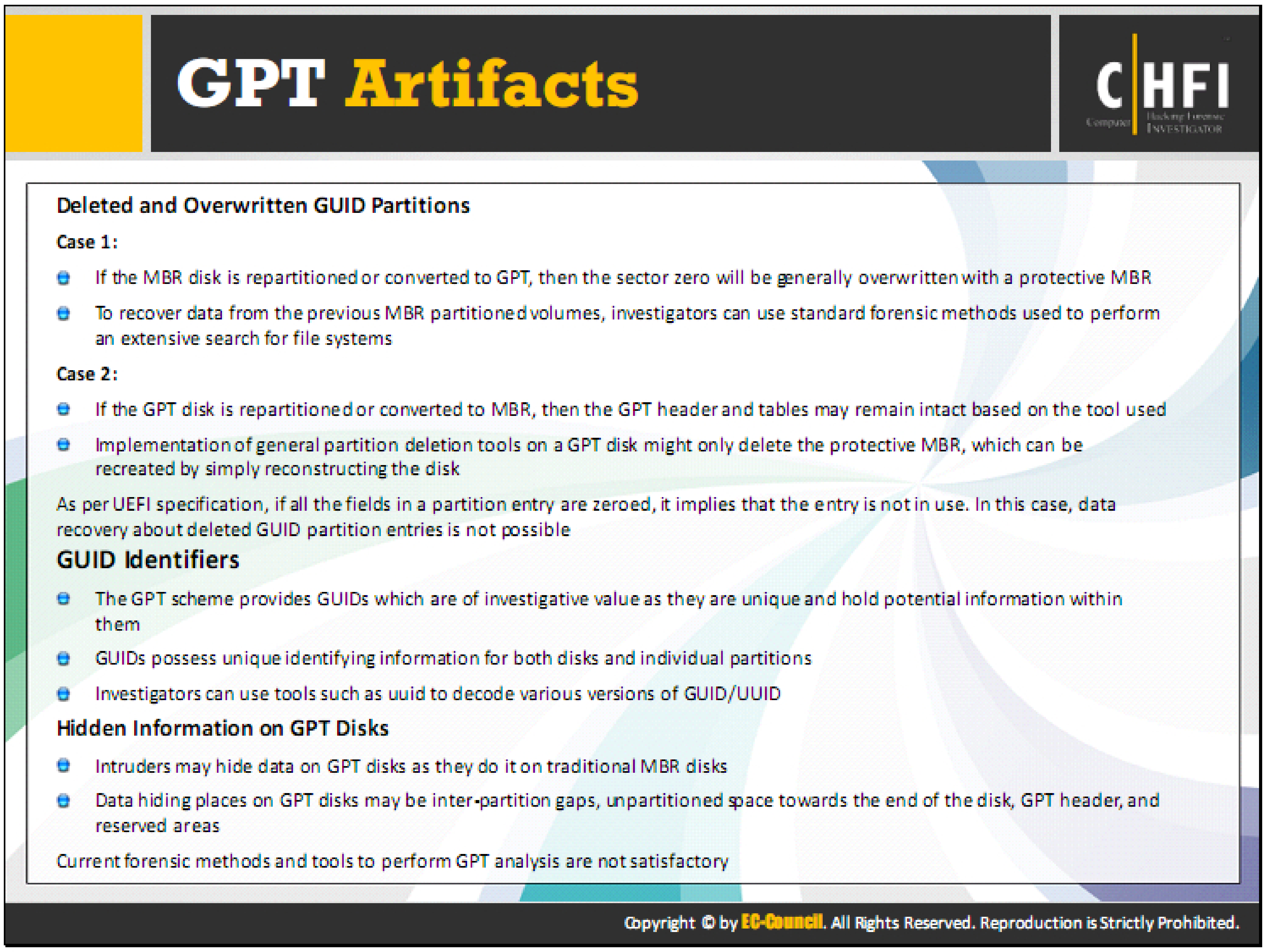

Administrator: Command Prompt

```
DISKPART>  detail disk

Dell VIRTUAL DISK SCSI Disk Device
Disk ID: 2C261141-D86C-4BD4-9DE0-A03AE526ADEA
Type    : SAS
Bus     : 0
Target  : 0
LUN ID  : 0

  Volume ###  Ltr  Label        Fs     Type        Size     Status     Info
  ----------  ---  -----------  -----  ----------  -------  ---------  --------
* Volume 1     D   DATA         NTFS   Partition   3725 GB  Healthy

DISKPART>  detail partition

Partition 2
Type     : ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
Hidden   : No
Required: No
Attrib   : 0x0000000000000000

  Volume ###  Ltr  Label        Fs     Type        Size     Status     Info
  ----------  ---  -----------  -----  ----------  -------  ---------  --------
* Volume 1     D   DATA         NTFS   Partition   3725 GB  Healthy

DISKPART>
```

- Sleuthkit (mmls command) can be used to view detailed partition layout for GPT disk
- Alternatively, details about GPT header and partition entries can be obtained via manual analysis using a hex editor

Most of the operating systems that support GPT disk access come up with a basic partitioning tool, which displays details about GPT partition tables. In windows tools such as DiskPart tool display the partition details, whereas MAC systems use the OS X Disk utility and Linux uses GNU parted tool.

Sleuthkit mmls command can help the investigators to view detailed partition layout for GPT disk along with the MBR details. Alternatively, investigators can gather details about GPT header and partition entries through manual analysis of disk drive using a hex calculation or editing tool called Hex editor.



GPT Artifacts

Deleted and Overwritten GUID Partitions

Case 1:

- If the MBR disk is repartitioned or converted to GPT, then the sector zero will be generally overwritten with a protective MBR
- To recover data from the previous MBR partitioned volumes, investigators can use standard forensic methods used to perform an extensive search for file systems

Case 2:

- If the GPT disk is repartitioned or converted to MBR, then the GPT header and tables may remain intact based on the tool used
- Implementation of general partition deletion tools on a GPT disk might only delete the protective MBR, which can be recreated by simply reconstructing the disk

As per UEFI specification, if all the fields in a partition entry are zeroed, it implies that the entry is not in use. In this case, data recovery about deleted GUID partition entries is not possible

GUID Identifiers

- The GPT scheme provides GUIDs which are of investigative value as they are unique and hold potential information within them
- GUIDs possess unique identifying information for both disks and individual partitions
- Investigators can use tools such as uuid to decode various versions of GUID/UUID

Hidden Information on GPT Disks

- Intruders may hide data on GPT disks as they do it on traditional MBR disks
- Data hiding places on GPT disks may be inter-partition gaps, unpartitioned space towards the end of the disk, GPT header, and reserved areas

Current forensic methods and tools to perform GPT analysis are not satisfactory

Deleted and Overwritten GUID Partitions

Case 1: In hard disks, the conversion or repartition of the MBR disk to GPT will generally overwrite the sector zero with a protective MBR, which will delete all the information about the old partition table. The investigators should follow the standard forensics methods of searching the filesystems to recover data about the previous MBR partitioned volumes.

Case 2: When conversion or repartition of the GPT to MBR disk takes place, then the GPT header and tables may remain intact based on the tool used. Investigators can easily recover or analyze data of such disk partitions.

Implementation of general partition deletion tools for deletion of partition on the GPT disk might will delete the protective MBR only, which investigators can easily recreate by simply reconstructing the disk.

As per UEFI specification, if all the fields in a partition entry have zeroed values, it implies that the entry is not in use. In this case, data recovery about deleted GUID partition entries is not possible.

GUID Identifiers

- GPT scheme provides GUIDs which are of investigative value as they are unique and hold potential information about entire disk and each partition within them
- GUIDs possess unique identifying information for both disks and individual partitions

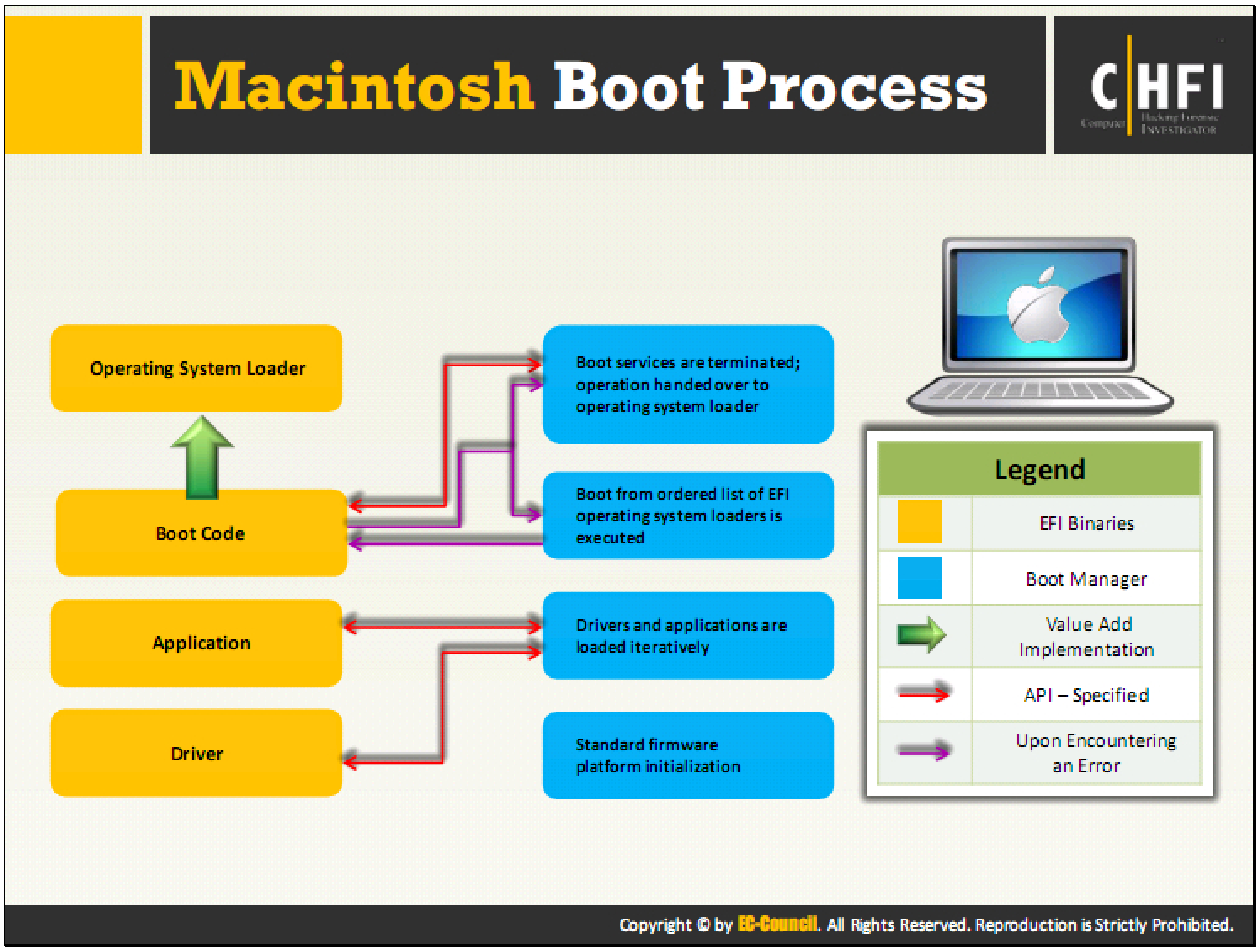
- Investigators can use tools such as UUID to decode various versions of GUID/UUID

Hidden Information on GPT Disks

Intruders may hide data on GPT disks as they do it on traditional MBR disks using a flexible and extensible disk partitioning schemes. Data hiding places on GPT disks may be inter-partition gaps, un-partitioned space towards the end of the disk, GPT header, and reserved areas. The other artifacts may include manipulated GPT headers that create place for hiding data, misplaced starting, and ending LBAs, as well as areas with reserved tag.

Current forensic methods and tools to perform GPT analysis are not satisfactory.

Macintosh Boot Process

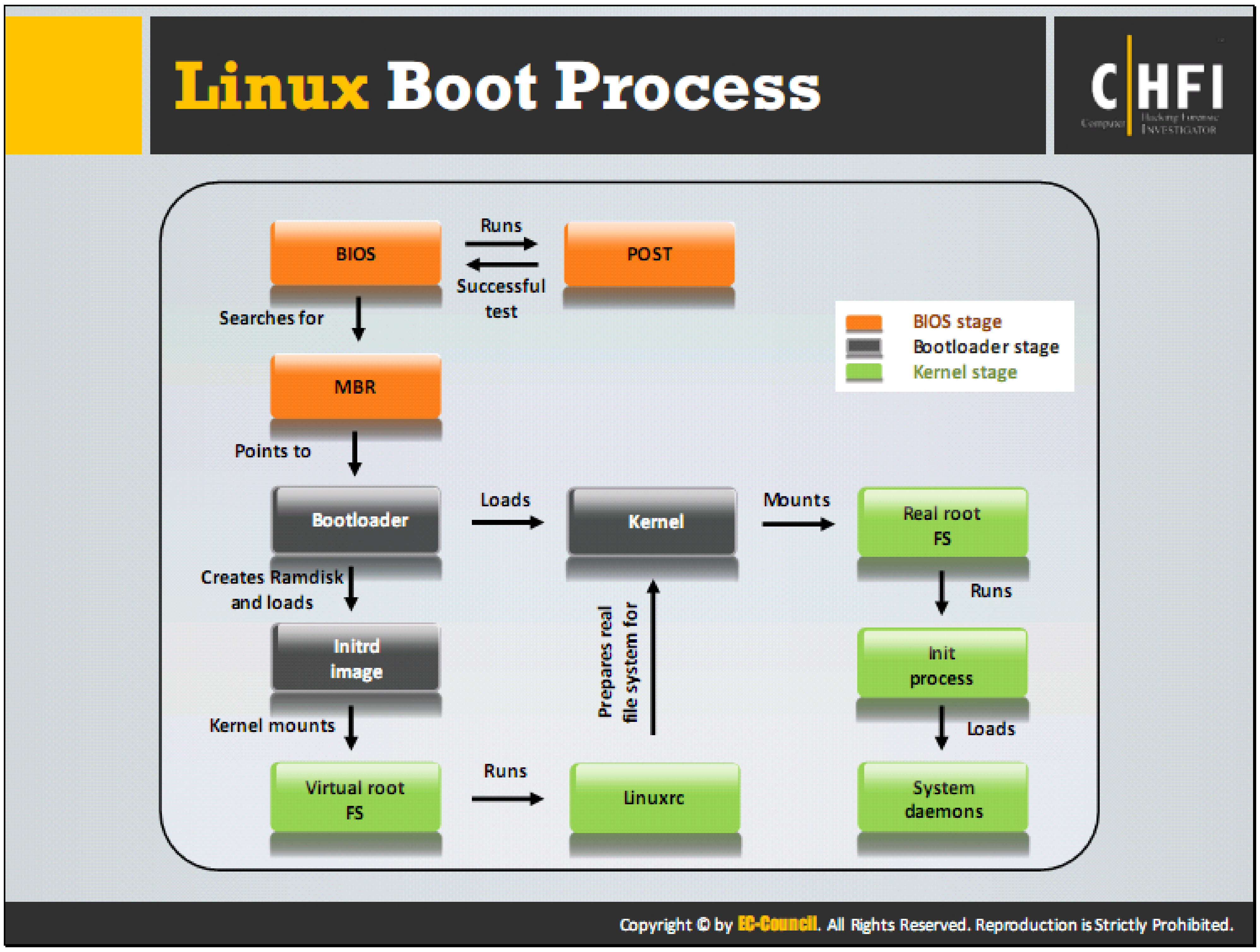

CHFI
Operating System Loader
Boot Code
Application
Driver
Boot services are terminated; operation handed over to operating system loader
Boot from ordered list of EFI operating system loaders is executed
Drivers and applications are loaded iteratively
Standard firmware platform initialization
Legend
EFI Binaries
Boot Manager
Value Add Implementation
API – Specified
Upon Encountering an Error

Following are the steps for the Macintosh boot process:

- The Macintosh boot process starts with the activation of **BootROM**, which initializes system hardware and selects an operating system to run
- Once you power on the Macintosh, BootROM performs **POST** (**Power-On Self-Test**) to test some hardware interfaces required for startup
- On PowerPC-based Macintosh computers, **Open Firmware** initializes the rest of the hardware interfaces
- On Intel-based Macintosh computers, **EFI** initializes the rest of the hardware interfaces
- After initializing the hardware interfaces, the system selects the **operating system**
- If the system contains multiple operating systems, then it allows the user to choose the **particular operating system** by holding down the Option key
- Once the BootROM operation is finished, the control passes to the **BootX** (**PowerPC**) **or boot.efi** (**Intel**) boot loader, which is located in the **/System/Library/CoreServices** directory
- The boot loader loads a pre-linked version of the kernel, which is located at **/System/Library/Caches/com.apple.kernelcaches**
- If the pre-linked kernel is missing, the boot loader attempts to load the **mkext cache file**, which contains a set of device drivers.

- If the mkext cache file is also missing, the boot loader searches for drivers in the **/System/Library/Extensions** directory
- Once the essential drivers are loaded, the boot loader starts initialization of the kernel, **Mach** and **BSD data structures**, as well as the I/O kit
- The I/O kit uses the device tree to link the loaded drivers to the kernel
- The launchd, which has replaced the mach_init process, runs startup items and prepares the system for the user

Linux Boot Process



In Linux boot process, the process flow starts with the BIOS, which searches for active and bootable devices. The system boots Linux from hard disk, in which the MBR contains the primary boot loader.

The Linux Boot Process consists of three stages. They are as follows:

- The BIOS Stage
- The Bootloader Stage
- Kernel Stage

The BIOS Stage

The first stage of the Linux boot process is the BIOS stage. It initializes the system hardware during the booting process. The BIOS retrieves the information, stored in the CMOS chip (Complementary Metal-Oxide Semiconductor) which is a battery operated memory chip on the motherboard that contains information about the system's hardware configuration.

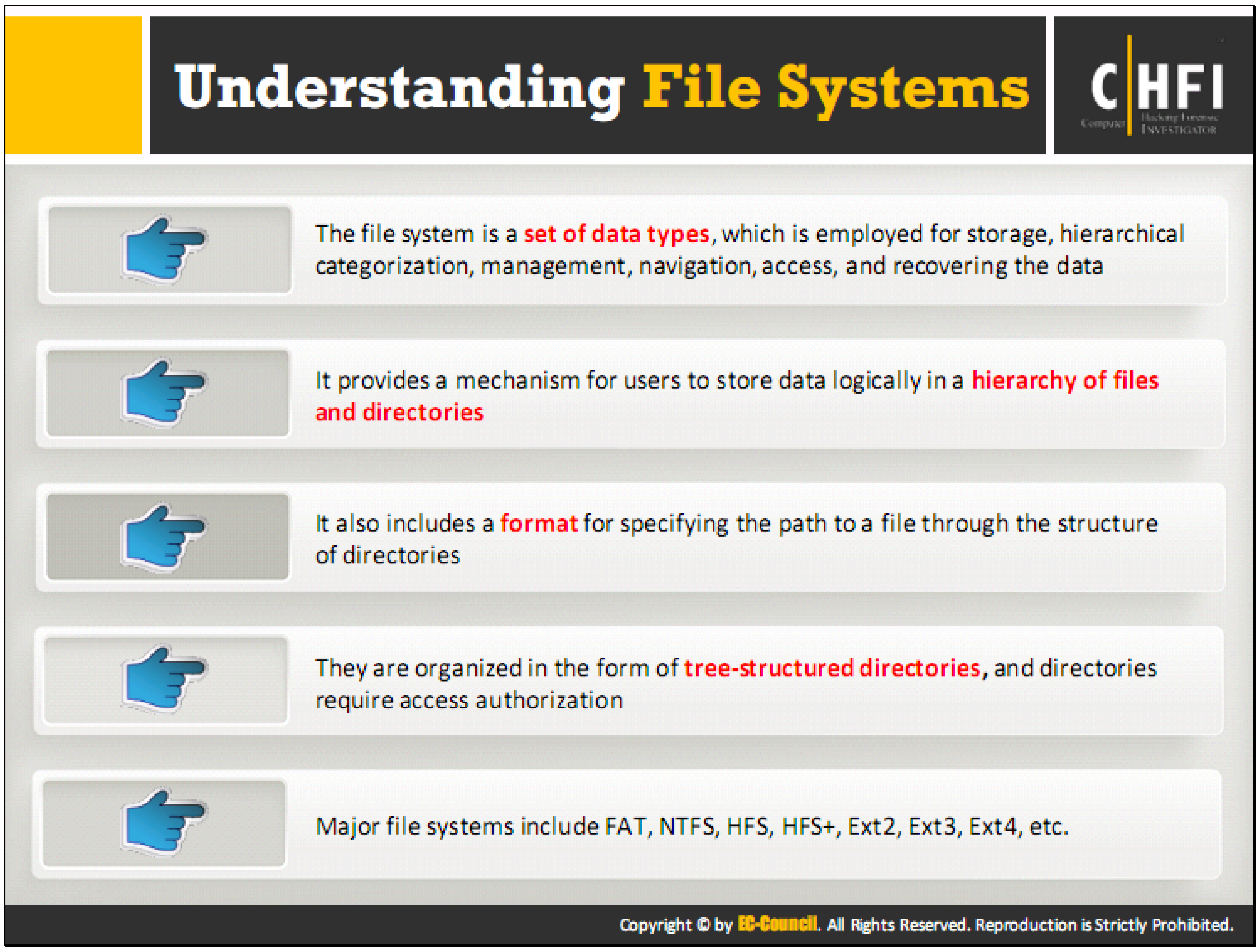
During the boot process, the BIOS performs a Power-On Self-Test (POST) to make sure that all the hardware components of the system are working. Once BIOS confirms that everything is fine, it starts searching for the drive or disk which contains the operating system in a standard sequence. If the first listed device is not available or not working, then it checks for the next one and so on. A drive can be bootable only if it has the Master Boot Record in its first sector known as the boot sector. The system’s hard disk acts as the primary boot disk and the optical drive works as the secondary boot disk for booting the operating system from the removable disk if in case the main hard disk fails.

The Bootloader Stage

The bootloader stage includes the task of loading the Linux kernel and optional initial RAM disk. The kernel will help enabling the CPU to access RAM and the disk. The second pre-cursor software is an image of a temporary virtual file system called the initrd image or initial RAMdisk. Now, the system prepares to deploy the actual root file system. It then detects the device that contains the file system and loads the necessary modules. The last step of the bootloader stage is to load the kernel into the memory.

Kernel Stage

Once the control shifts from the bootloader stage to the Kernel stage, the virtual root file system created by the initrd image executes the Linuxrc program. This program generates the real file system for the kernel and later removes the initrd image. The kernel then searches for new hardware and loads any suitable device drivers found. It then mounts the actual root file system and then performs the init process. The init reads the file "/etc/inittab" and uses this file to load the rest of the system daemons. This prepares the system and the user can log in and start using it. The typical bootloaders for Linux are LILO (Linux Loader) and GRUB (Grand Unified Bootloader). These bootloaders allow the user to select which OS kernel to load during boot time.



Understanding File Systems

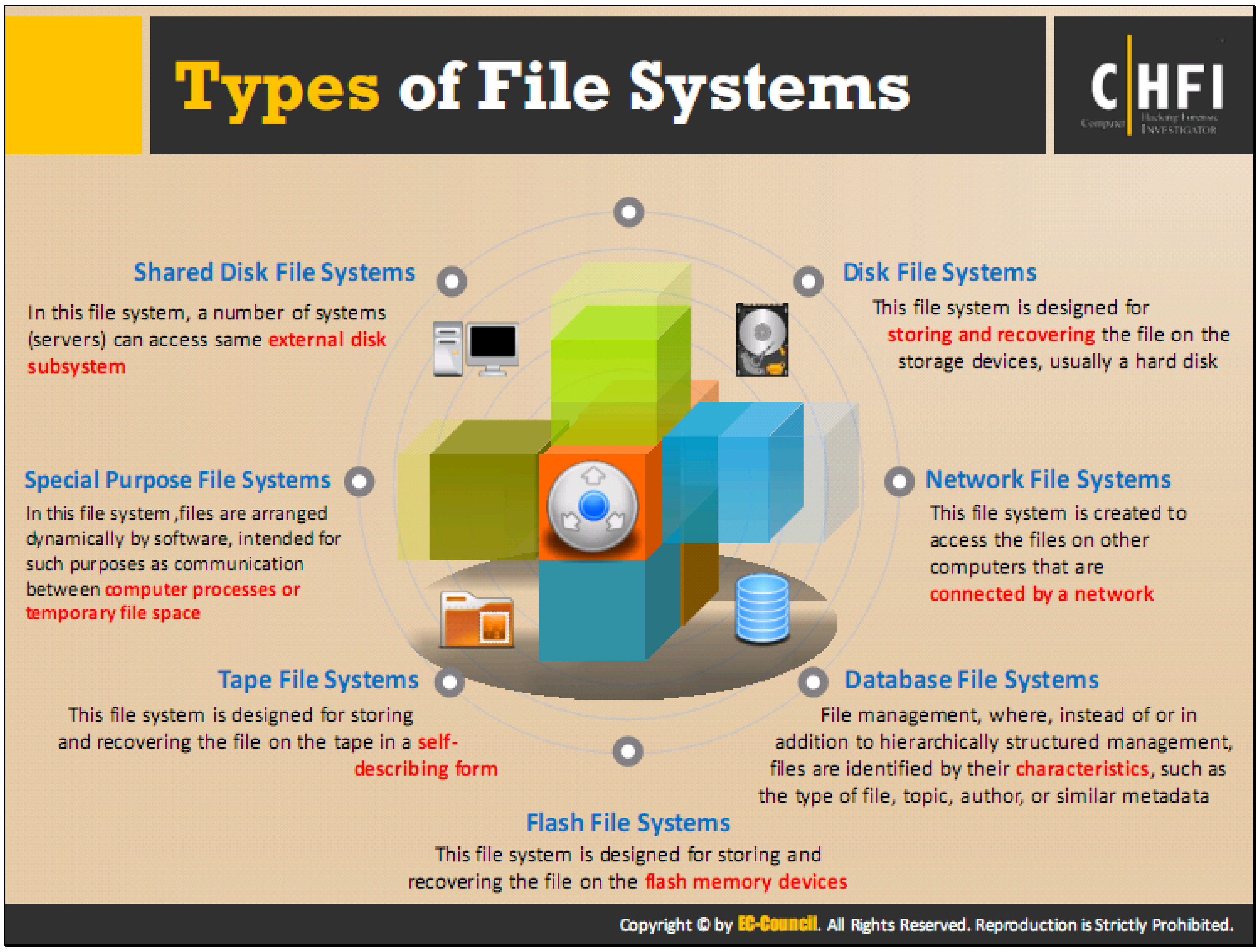
- The file system is a set of data types, which is employed for storage, hierarchical categorization, management, navigation, access, and recovering the data
- It provides a mechanism for users to store data logically in a hierarchy of files and directories
- It also includes a format for specifying the path to a file through the structure of directories
- They are organized in the form of tree-structured directories, and directories require access authorization
- Major file systems include FAT, NTFS, HFS, HFS+, Ext2, Ext3, Ext4, etc.

The computer not only computes data but also stores data. The issue of file structure and data storage is of prime concern. To solve this issue, manufacturers employ an effective storing and organization of the data on the computer called as a file system. The file system makes it easy to find and access the data. Data storage devices like hard disks or CD-ROMs can use the file system to store the data. The file system divides the file into smaller pieces and then stores them to hard disks or flash memory in clusters.

A file system is a set of data types employed for:

- Storage
- Hierarchical categorization
- Management
- Navigation
- Access
- Recovering the data

Major file systems include FAT, NTFS, HFS, Ext2, Ext3, etc. Users can access the files using the graphical user interfaces or command line user interfaces. File systems organize the data in the form of tree-structured directories. These are generally file cabinets and folders. Directories require authorized permission to access.



Types of File Systems

Shared Disk File Systems
In this file system, a number of systems (servers) can access same external disk subsystem

Disk File Systems
This file system is designed for storing and recovering the file on the storage devices, usually a hard disk

Special Purpose File Systems
In this file system, files are arranged dynamically by software, intended for such purposes as communication between computer processes or temporary file space

Network File Systems
This file system is created to access the files on other computers that are connected by a network

Tape File Systems
This file system is designed for storing and recovering the file on the tape in a self-describing form

Database File Systems
File management, where, instead of or in addition to hierarchically structured management, files are identified by their characteristics, such as the type of file, topic, author, or similar metadata

Flash File Systems
This file system is designed for storing and recovering the file on the flash memory devices

A file system refers to the structure a computer uses to organize data on media such as hard disks, CDs, DVDs, and many other storage devices or an index or database that contains the physical location of every piece of data on a hard drive or storage devices.

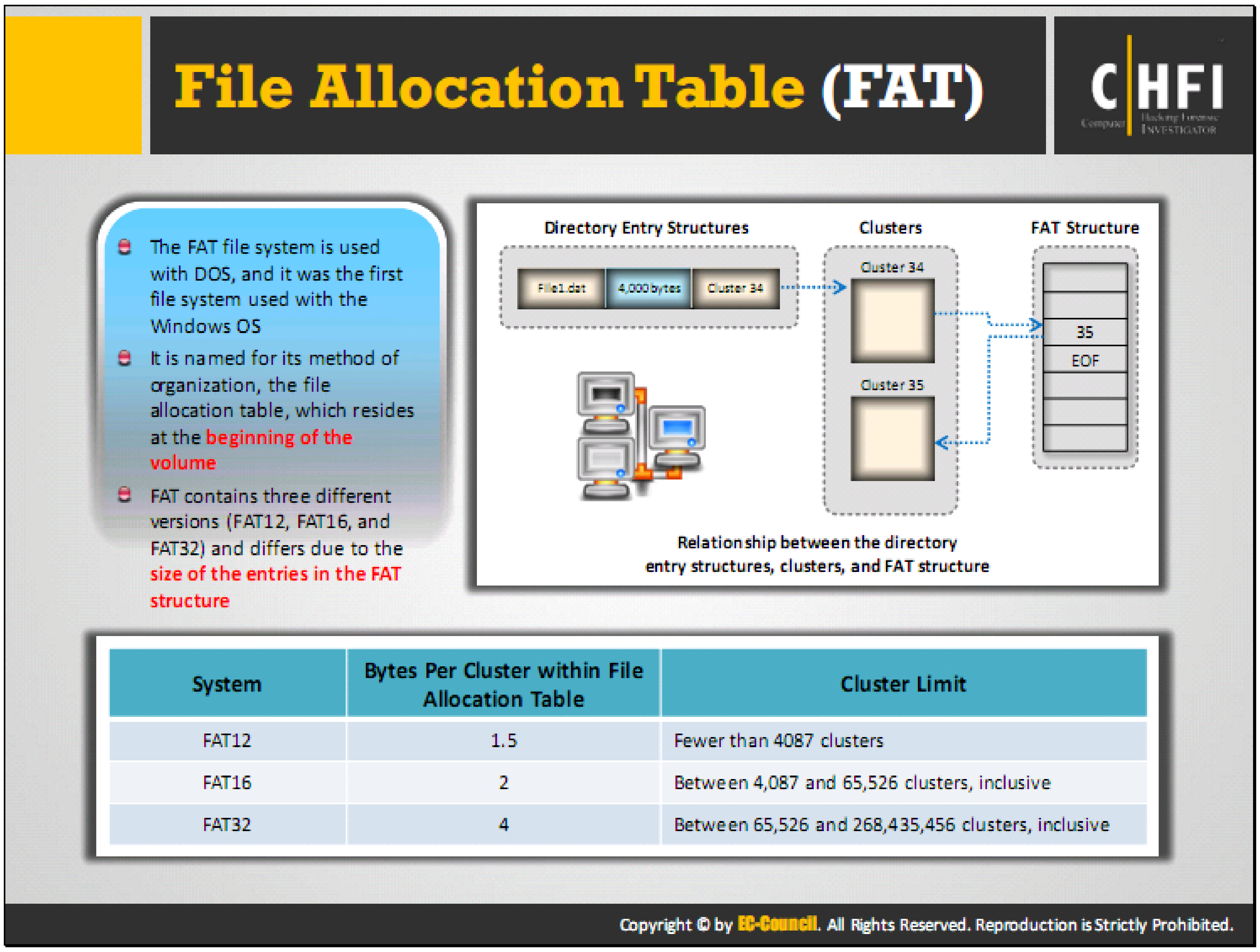
Following are the different types of file systems:

- **Disk file systems:** A disk file system is a technique designed for storing and recovering the file on a storage device, usually a hard disk, directly or indirectly connected to the computer. A few examples of the disk file system are FAT, NTFS, ext2, ISO 9660, ODS-5, and UDF.
- **Network file systems:** A network file system is a type of file system, which helps the users to access the files on other computers connected through a network. The file systems are transparent to the user. A few examples of network file systems are NFS, CIFS, and GFS.
- **Database file systems:** It is a new method of storing data on the computer and effectively managing the file system. Earlier file systems used hierarchical structured management, but the database file system identifies the files by their characteristics, such as the name of the file, type of the file, topic, author, or similar metadata. Therefore, a user can search for a file by formulating the SQL query or in natural speech. For example, if the user needs to find the documents written, then the query “documents written by ABC” will show the results.

- **Flash file systems:** This system stores the files or data in flash memory devices. In today's world, these file systems are becoming prevalent with the increasing number of mobile devices. With these file systems, the cost per memory size decreases, and the capacity of flash memory will increase.
- **Tape file systems**: It stores files on tape in a self-describing form. Magnetic tapes work as sequential storage media with significantly longer random data access time as compared to disks, posing challenges to the creation of a general-purpose file system with efficient management. Tape drives require a linear motion to unwind and wind potentially very long reels of media. This might take several seconds or minutes to move the read/write head.
- **Shared disk file systems:** A shared disk file system works on the principle of accessing an external disk subsystem (SAN) through a number of servers. The file system arbitrates access to that subsystem, to prevent write collisions.
- **Special-purpose file systems:** In a special-purpose file system, the software organizes files during the run time and uses them for tasks such as communication between computer processes or temporary file space. File-centric operating systems such as UNIX use this file system. Any file system that is not a disk file system or network file system is a special-purpose file system. For example, '/proc' in UNIX, can help to get information regarding processes and other operating system features.

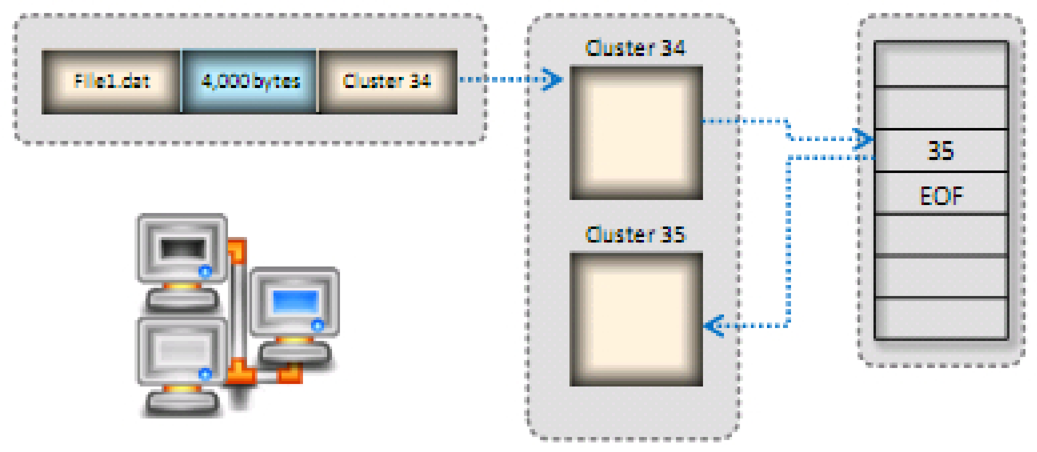
Windows File Systems

File systems are the basic storage units of any device and Windows operating systems power most of the computing devices across the globe. Therefore, the investigators are liable to come across various systems running on Windows while investigating a security incident and need to have fair knowledge on how the OS stores the files. This section will demonstrate the methods, Windows OS employs, to store the files in order to help the investigators extract and analyze them.



File Allocation Table (FAT)

- The FAT file system is used with DOS, and it was the first file system used with the Windows OS
- It is named for its method of organization, the file allocation table, which resides at the beginning of the volume
- FAT contains three different versions (FAT12, FAT16, and FAT32) and differs due to the size of the entries in the FAT structure



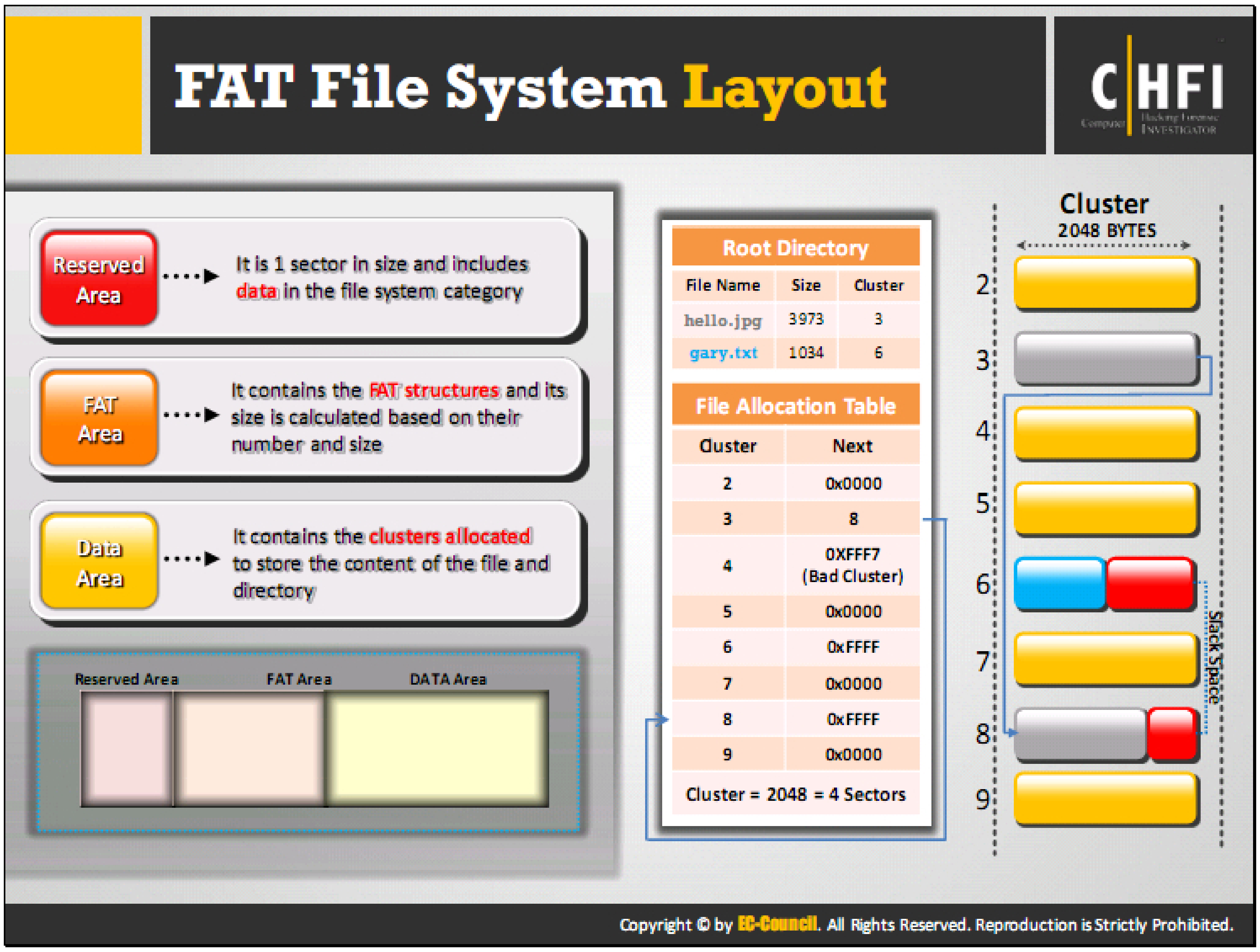
Relationship between the directory entry structures, clusters, and FAT structure

System	Bytes Per Cluster within File Allocation Table	Cluster Limit
FAT12	1.5	Fewer than 4087 clusters
FAT16	2	Between 4,087 and 65,526 clusters, inclusive
FAT32	4	Between 65,526 and 268,435,456 clusters, inclusive

FAT (File Allocation Table) is a file system, designed in 1976, for many operating systems such as DOS, Windows, OpenDOS, etc. Designed for small hard disks and a simple folder structure, the FAT file system got its name from the way it organizes folders and the file allocation table. The file allocation table stores all the files and resides at the beginning of the volume.

It creates two copies of the file allocation table to protect the volume from damage. The FAT file system stores the file allocation table and root folder in a permanent location. The volume formatted using the FAT file system form a cluster and size of the formatted volume determines the cluster size. The system fits the cluster number for the FAT file system in 16 bits and is in the power of two.

Few devices that implement FAT include flash memory, digital cameras, and other portable devices. Nearly almost all the operating systems installed on the personal computers implement FAT file system.



FAT File System Layout

Reserved Area → It is 1 sector in size and includes data in the file system category

FAT Area → It contains the FAT structures and its size is calculated based on their number and size

Data Area → It contains the clusters allocated to store the content of the file and directory

Root Directory

File Name	Size	Cluster
hello.jpg	3973	3
gary.txt	1034	6

File Allocation Table

Cluster	Next
2	0x0000
3	8
4	0XFFF7 (Bad Cluster)
5	0x0000
6	0xFFFF
7	0x0000
8	0xFFFF
9	0x0000

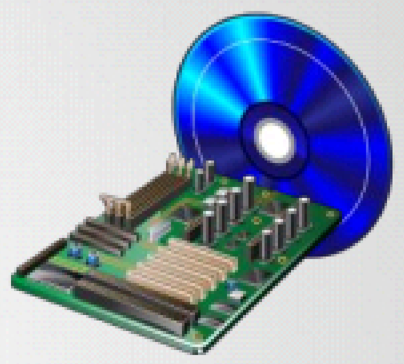
Cluster = 2048 = 4 Sectors

The typical FAT32 file system will comprise of the following components:

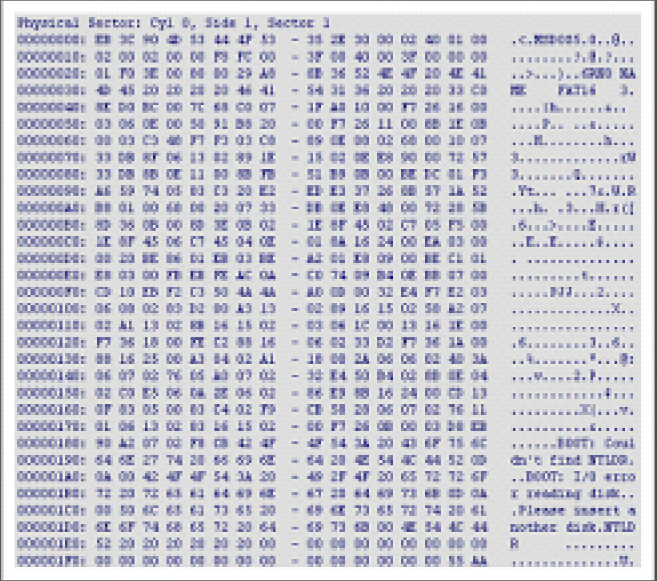
- **Reserved Area**: The first reserved sector is the Volume Boot Record or VBR, which comprises the BIOS Parameter Block (BPB) containing basic file system information, such as type of file system and pointers to the position of the other sections as well as the operating system's boot loader code.
- **FAT Area:** This area holds two duplicates (may change) of the File Allocation Table to help the system check for the empty or idle spaces. This area contains detailed information about clusters and their contents including files and directories. Extra copies contained in this file system are in perfect sync with writes and read, and will replace when the first or main FAT seems to include mistakes or damages.
- **Data Area:** This region, which occupies the largest part of a partition, stores the actual file and directory data. The FAT file system fills the unused parts or spaces with a filler estimation of 0xF6 based on the INT 1Eh's Disk Parameter Table (DPT). The FAT supports read-only, hidden, system, and archive attributes.

FAT Partition Boot Sector

- Boot Sector is the first sector (512 bytes) of a FAT file system
- FAT partition boot sector holds data that the file system uses to access the partition or volume
- MBR of x86-based computer systems uses this boot sector on the system partition to load the system kernel files



Byte Offset (in Hex)	Field Length	Sample Value	Meaning
00	3 bytes	EB 3C 90	Jump instruction
03	8 bytes	MSDOS5.0	OEM name in text
0B	25 bytes		BIOS Parameter Block (BPB)
24	26 bytes		Extended BIOS parameter block
3E	448 bytes		Bootstrap code
1FE	2 bytes	0x55AA	End of the sector marker



```
Physical Sector: Cyl 0, Side 1, Sector 1
[illegible]
```

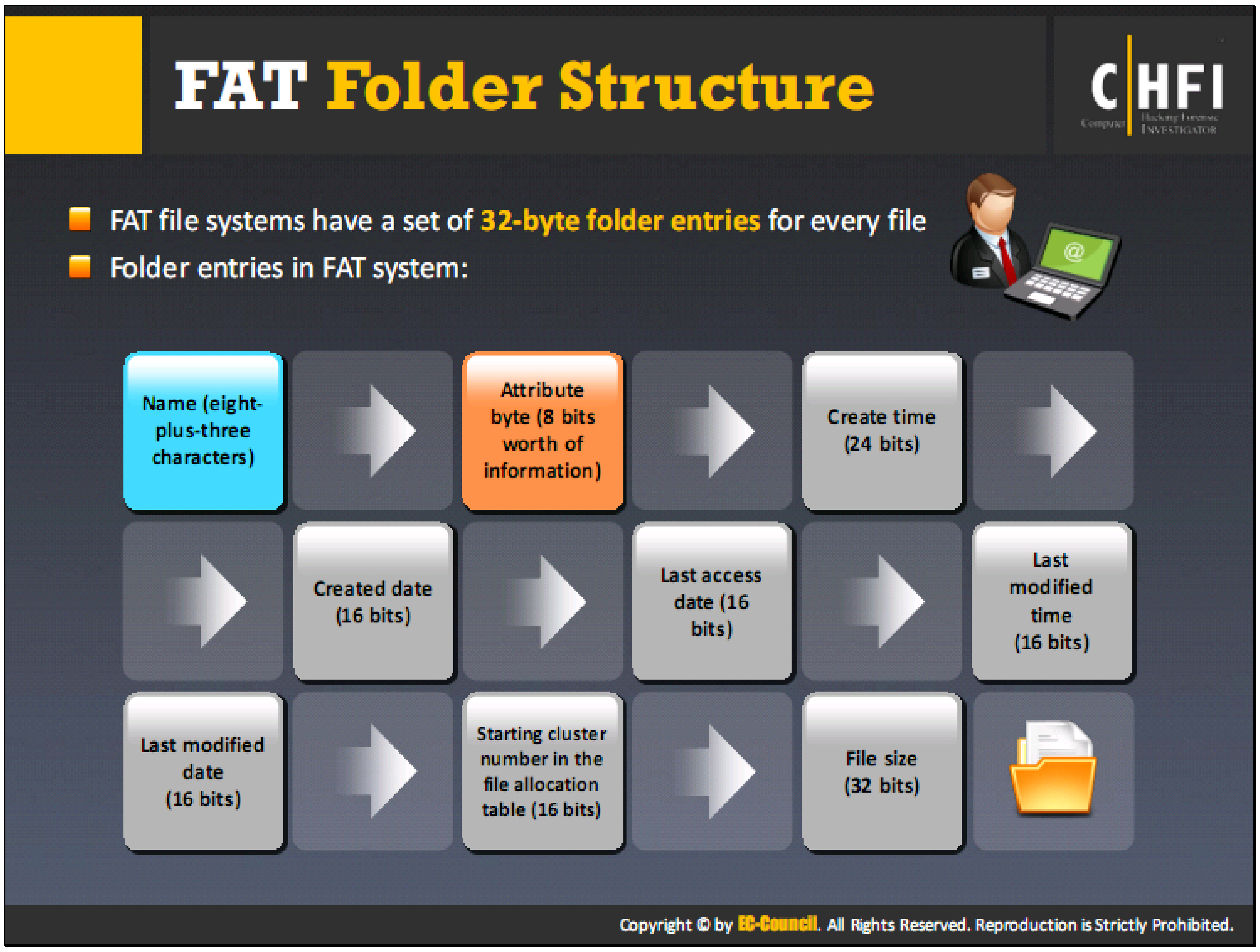
The Partition Boot Sector consists of data that the document framework uses to get to the volume. On x86-based PCs, the Master Boot Record utilizes the Partition Boot Sector on the framework parcel to stack the working framework portion documents.

In the UNIX operating system, this would be called a super block. It contains some general information.

The following is an example of the boot sector:

0000000 eb 3f 90 49 42 4d 20 20 33 2e 33 00 02 01 01 00 0000020 02 e0 00 40 0b f0 09 00 12 00 02 00 00 00 00 00 0000040 00 00 00 00 00 00 00 00 00 70 00 ffff 49 42 0000060 4d 42 49 4f 20 20 43 4f 4d 00 50 00 00 08 00 18...

The 2-byte numbers are stored in a little endian. The following table has the FAT12 version and is similar to FAT16 and FAT32 versions.



FAT Folder Structure

- FAT file systems have a set of **32-byte folder entries** for every file
- Folder entries in FAT system:

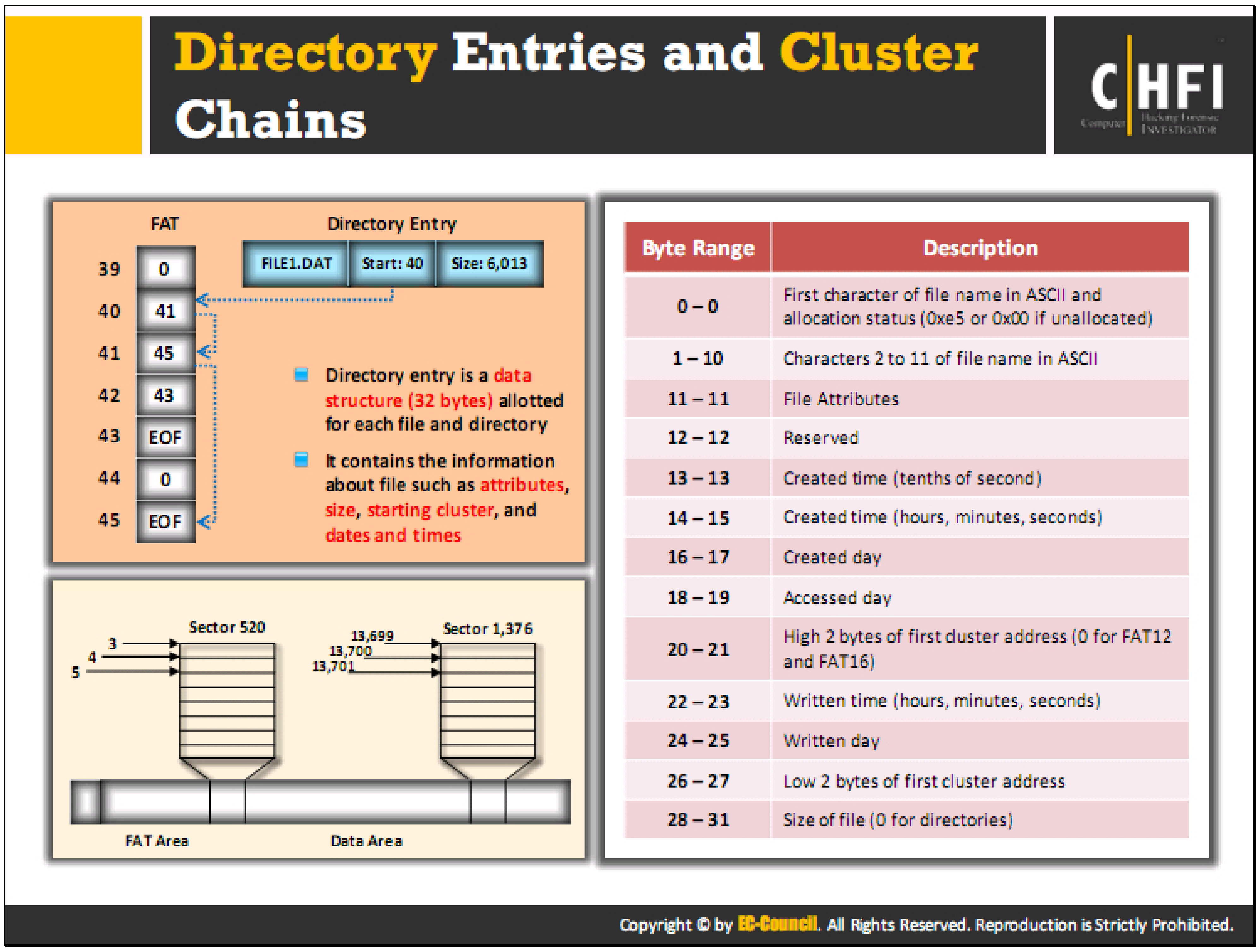
Name (eight-plus-three characters)
Attribute byte (8 bits worth of information)
Create time (24 bits)
Created date (16 bits)
Last access date (16 bits)
Last modified time (16 bits)
Last modified date (16 bits)
Starting cluster number in the file allocation table (16 bits)
File size (32 bits)
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

The FAT file system has a set of 32-byte folder entries for every folder.

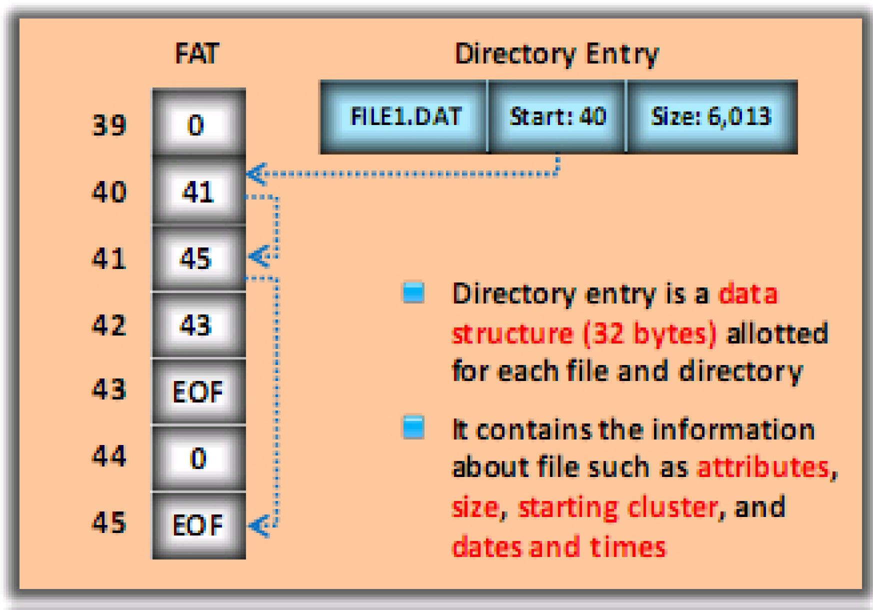
Folder entries in the FAT system are as follows:

- Name (eight-plus-three characters)
- Attribute byte (8 bits worth of information, described later in this section)
- Create time (24 bits)
- Create date (16 bits)
- Last access date (16 bits)
- Last modified time (16 bits)
- Last modified date (16 bits)
- Starting cluster number in the file allocation table (16 bits)
- File size (32 bits)

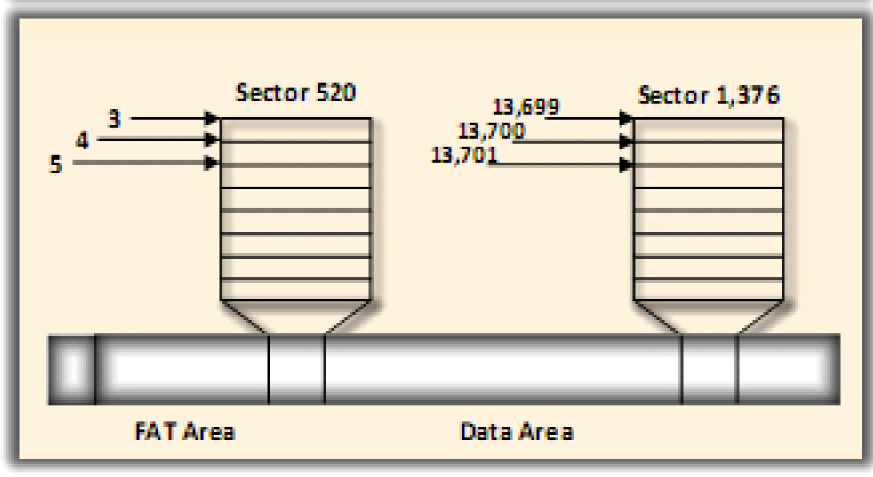
All operating systems that support FAT's file system use the information present in the FAT folder.



Directory Entries and Cluster Chains



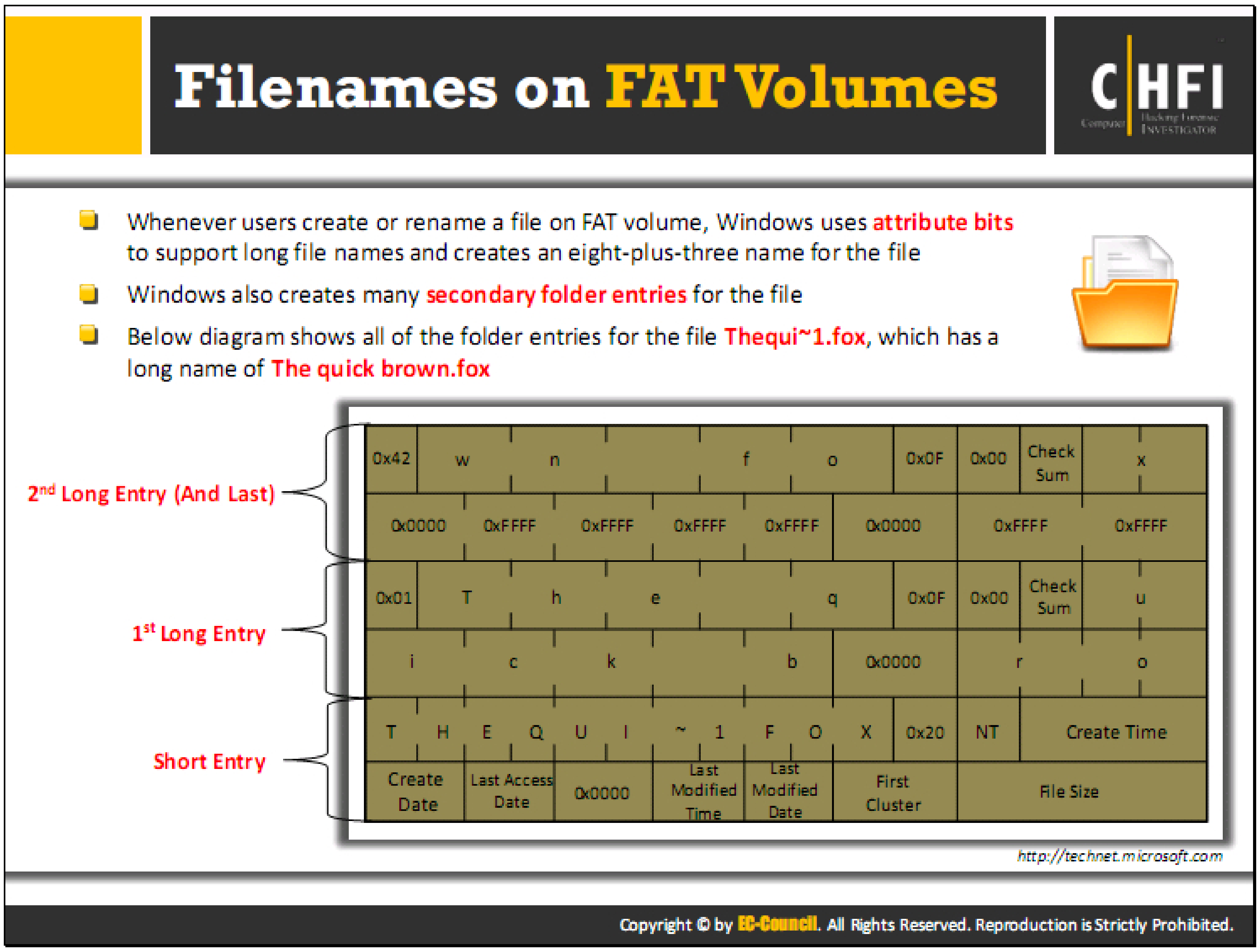
- Directory entry is a data structure (32 bytes) allotted for each file and directory
- It contains the information about file such as attributes, size, starting cluster, and dates and times



Byte Range	Description
0 – 0	First character of file name in ASCII and allocation status (0xe5 or 0x00 if unallocated)
1 – 10	Characters 2 to 11 of file name in ASCII
11 – 11	File Attributes
12 – 12	Reserved
13 – 13	Created time (tenths of second)
14 – 15	Created time (hours, minutes, seconds)
16 – 17	Created day
18 – 19	Accessed day
20 – 21	High 2 bytes of first cluster address (0 for FAT12 and FAT16)
22 – 23	Written time (hours, minutes, seconds)
24 – 25	Written day
26 – 27	Low 2 bytes of first cluster address
28 – 31	Size of file (0 for directories)

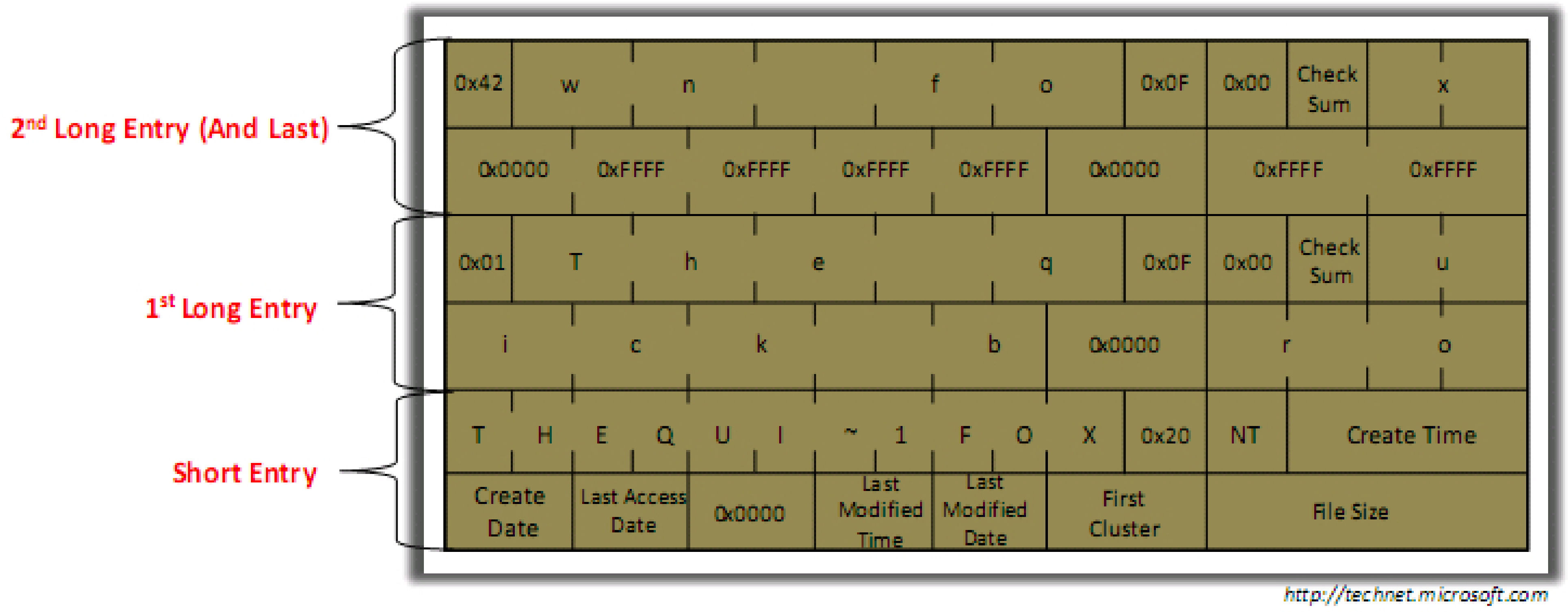
Directory entry is a data structure (32 bytes) allotted for each file and directory. Operating systems use directory entries to store additional metadata such as file passwords, access rights, owner IDs, file deletion data as well as attributes, size, starting cluster, date and time.

A file system divides the volume's data area into identically sized clusters of different sizes depending on the type of FAT file system used and the size of the partition. When users store data, each file may occupy more than one of such clusters depending on its size. Thus, a chain of these clusters represent a file.



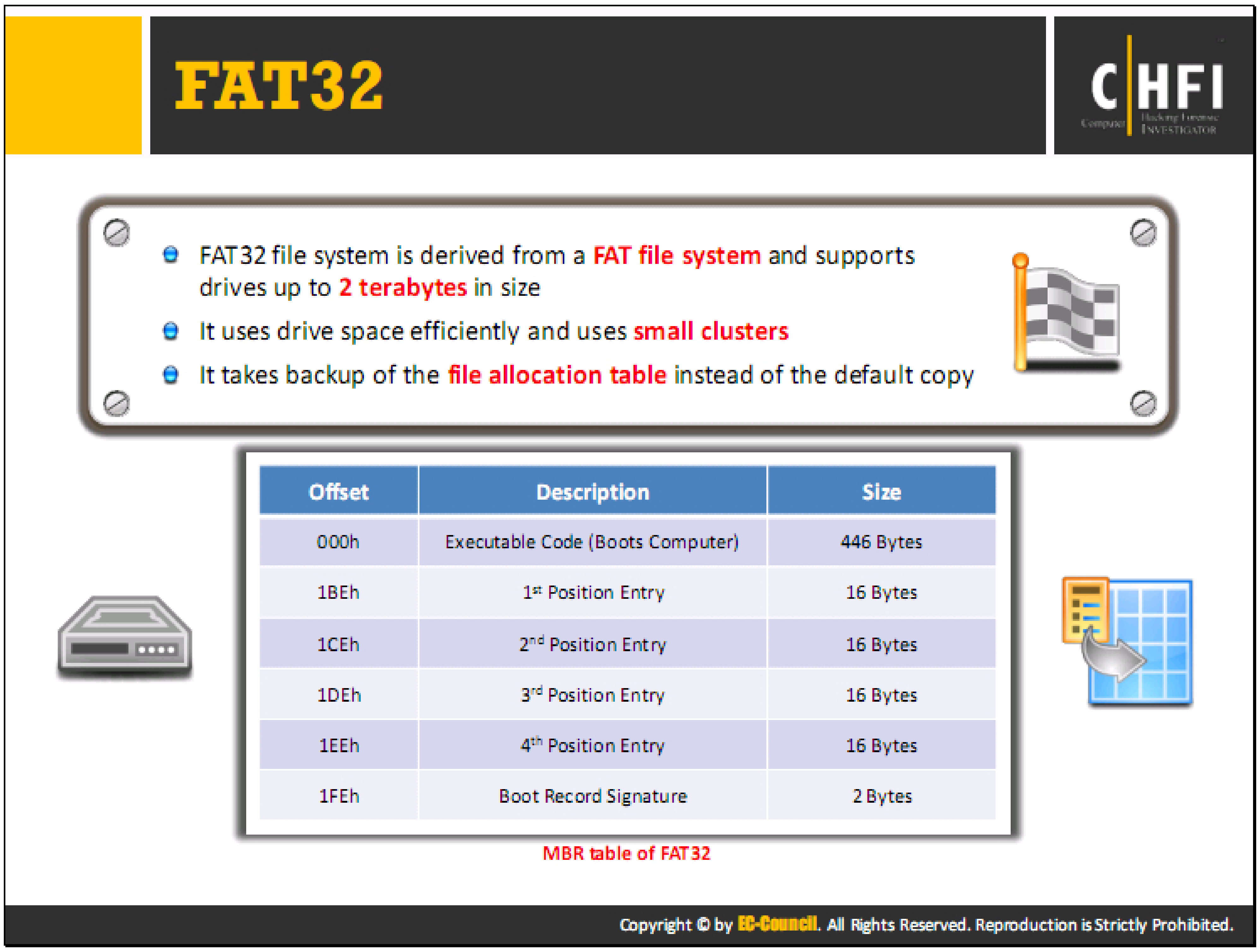
Filenames on FAT Volumes

- Whenever users create or rename a file on FAT volume, Windows uses **attribute bits** to support long file names and creates an eight-plus-three name for the file
- Windows also creates many **secondary folder entries** for the file
- Below diagram shows all of the folder entries for the file **Thequi~1.fox**, which has a long name of **The quick brown.fox**



http://technet.microsoft.com

When a user generates a file with a long filename, the Windows based system allocates an eight-plus-three name for the file and creates one or more secondary folder entries. These folder entries store a corresponding part of the long filename in Unicode. Windows assembles attribute bits such as the volume, read-only, system, and hidden file of the entry to represent parts of the filename.



FAT32

- FAT32 file system is derived from a FAT file system and supports drives up to 2 terabytes in size
- It uses drive space efficiently and uses small clusters
- It takes backup of the file allocation table instead of the default copy

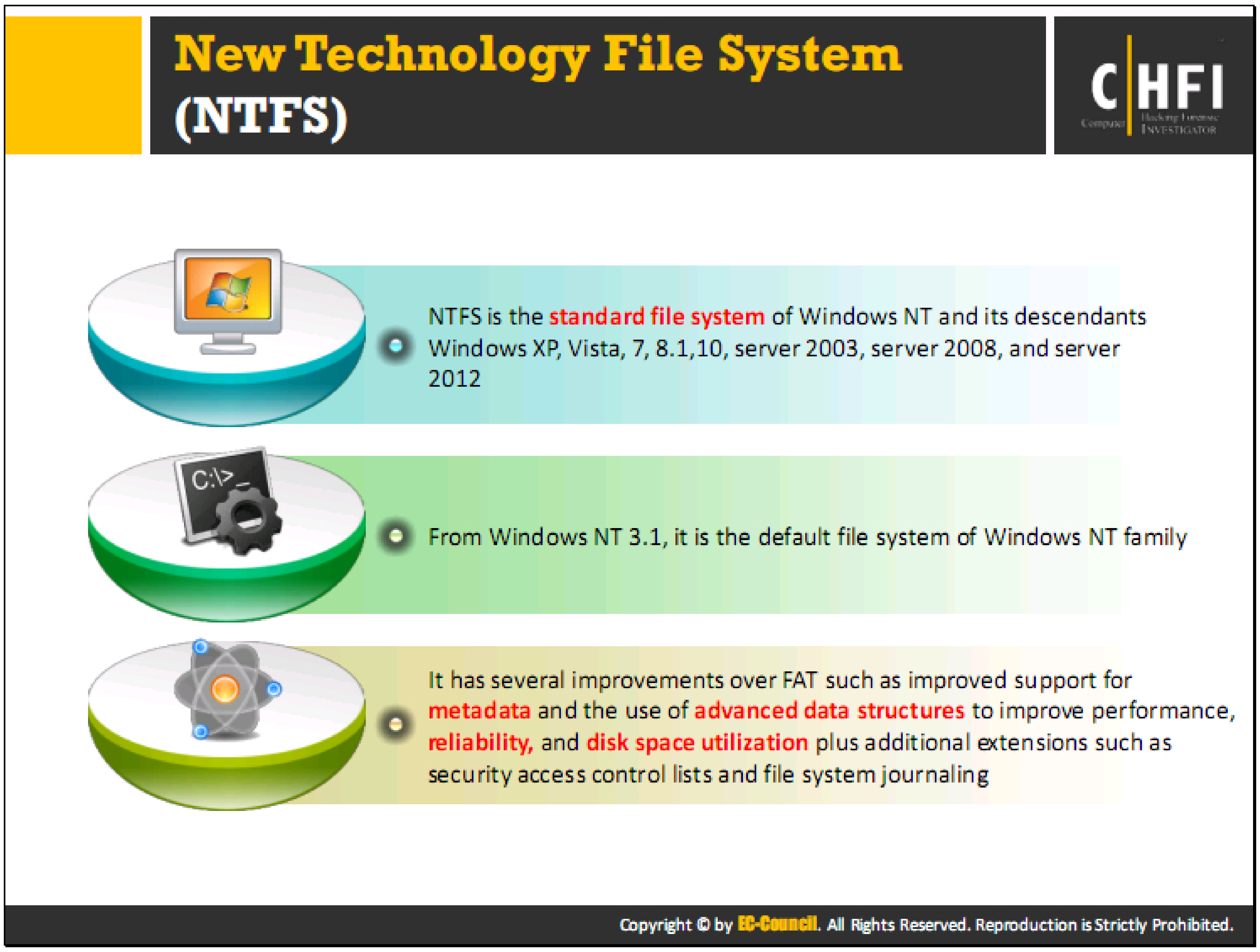
Offset	Description	Size
000h	Executable Code (Boots Computer)	446 Bytes
1BEh	1st Position Entry	16 Bytes
1CEh	2nd Position Entry	16 Bytes
1DEh	3rd Position Entry	16 Bytes
1EEh	4th Position Entry	16 Bytes
1FEh	Boot Record Signature	2 Bytes

MBR table of FAT32

FAT32 is a version of the file allocation table (FAT) that replaces the FAT16 file system and is available in Windows 95 OSR 2 and Windows 98. FAT32 uses smaller clusters with more address bits to support larger disks as well as offer better storage. It always creates a backup of the file allocation table instead of the default copy.

FAT32 Features

- Utilizes space more effectively, about 10 to 15 percent, due to usage of smaller clusters
- Highly robust as it can change the destination of the root folder and utilize the backup copy of the file allocation table
- Consists of extended boot record to incorporate a backup copy of basic information structures
- Have lesser failure rate compared to the FAT16 drives
- More adaptable
- Available anywhere on the drive as the root organizer on a FAT32 drive is a standard cluster chain
- Does not have restrictions on the quantity of root folder entries
- Allows users to disable repetitions of the file allocation table



New Technology File System (NTFS)

- NTFS is the standard file system of Windows NT and its descendants Windows XP, Vista, 7, 8.1,10, server 2003, server 2008, and server 2012
- From Windows NT 3.1, it is the default file system of Windows NT family
- It has several improvements over FAT such as improved support for metadata and the use of advanced data structures to improve performance, reliability, and disk space utilization plus additional extensions such as security access control lists and file system journaling

New Technology File System (NTFS) is one of the latest file systems supported by Windows. It is a high-performance file system, which repairs itself; it supports several advanced features such as file-level security, compression, and auditing. It also supports large and powerful volume storage solutions such as self-recovering disks.

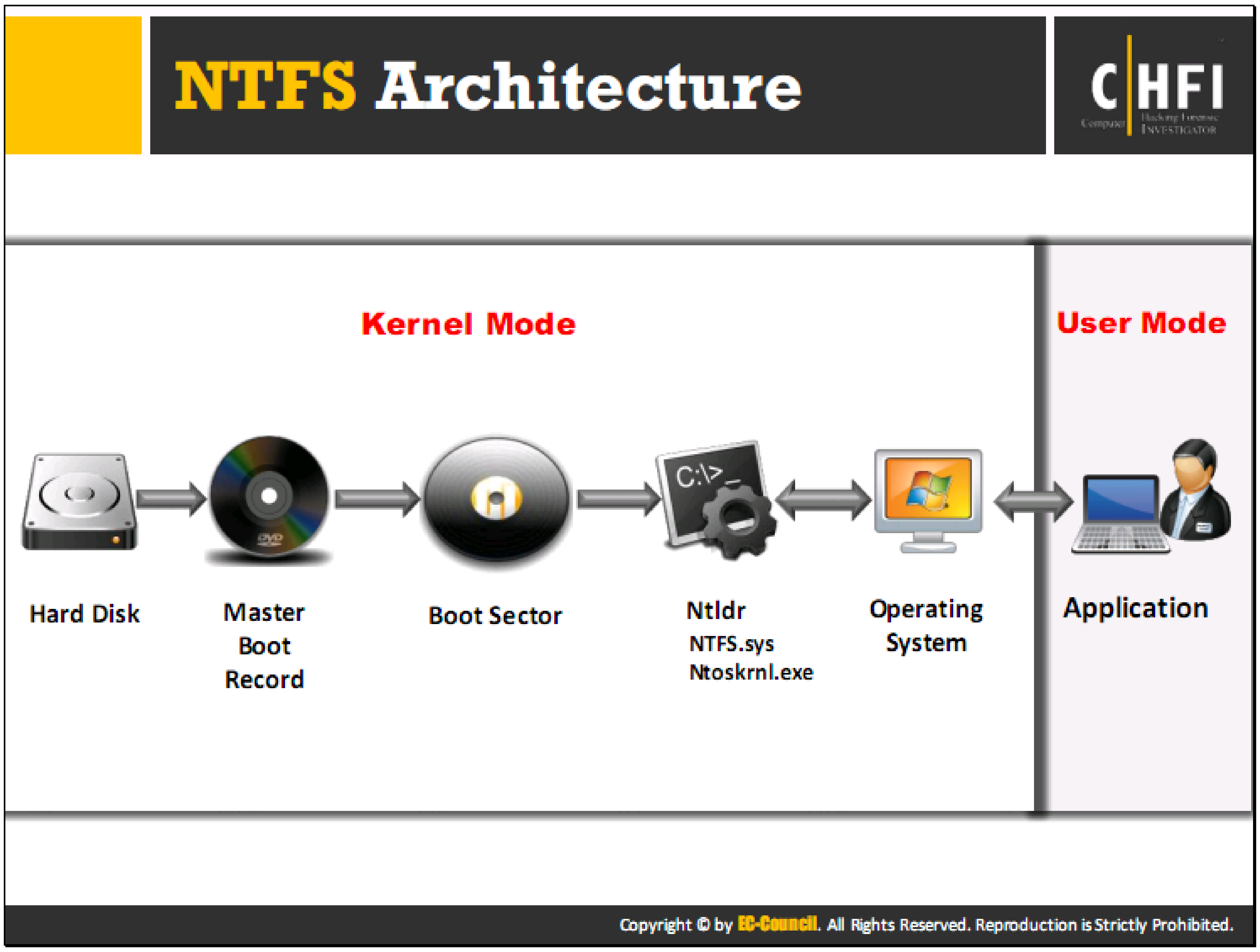
NTFS provides data security as it has the capability to encrypt or decrypt data, files, or folders. NTFS uses a 16-bit Unicode method to character set naming of files and folders. This attribute of NTFS allows users around the world to manage their files in their native languages. It has fault tolerance for the file system. If the user makes any modifications or changes to the files, NTFS makes a note of all changes in specific log files. If the system crashes, NTFS uses these log files to restore the hard disk to a reliable condition with minimal data loss. NTFS also provides the concept of metadata and master file tables. Metadata contains the information about the data stored in the computer. A master file table also contains the same information in a tabular form, but its capacity to store data in its table is comparatively less.

NTFS uses the Unicode data format. NTFS has many versions and they are as follows:

- v1.0 (found in Windows NT 3.1), v1.1 (Windows NT 3.5), and v1.2 (Windows NT 3.51 and Windows NT 4)
- v3.0, found in Windows 2000
- v3.1, found in Windows XP, Windows Server 2003, Windows Vista, and Windows 7
- These final three versions are sometimes referred to as v4.0, v5.0, and v5.1

Features of NTFS include

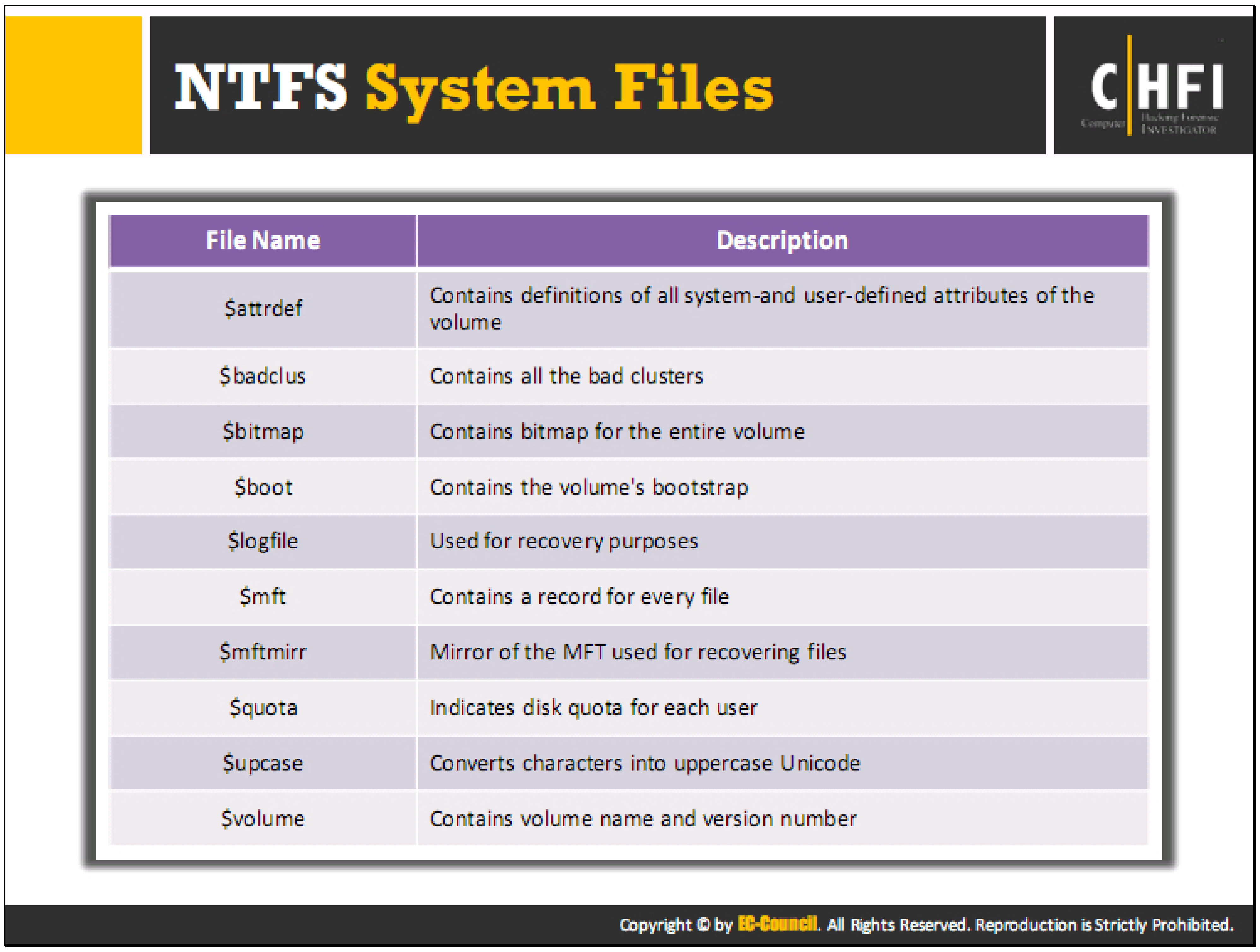
- Uses b-tree directory scheme to store information about file clusters
- Stores the information about a file's clusters and other data within the cluster
- Supports files up to 16 billion bytes in size approximately
- An access control list (ACL) allows the server administrator to access specific files
- Integrated file compression
- Data security on both removable and fixed disks



NTFS Architecture

At the time of formatting the volume of the file system, the system creates Master Boot Record. It contains some executable code called a master boot code and information about the partition table for the hard disk. When a new volume is mounted, the Master Boot Record runs the executable master boot code. It also transfers control to the boot sector on the hard disk, which allows the server to boot the operating system on the file system of that particular volume. Components of the NTFS architecture are as follows:

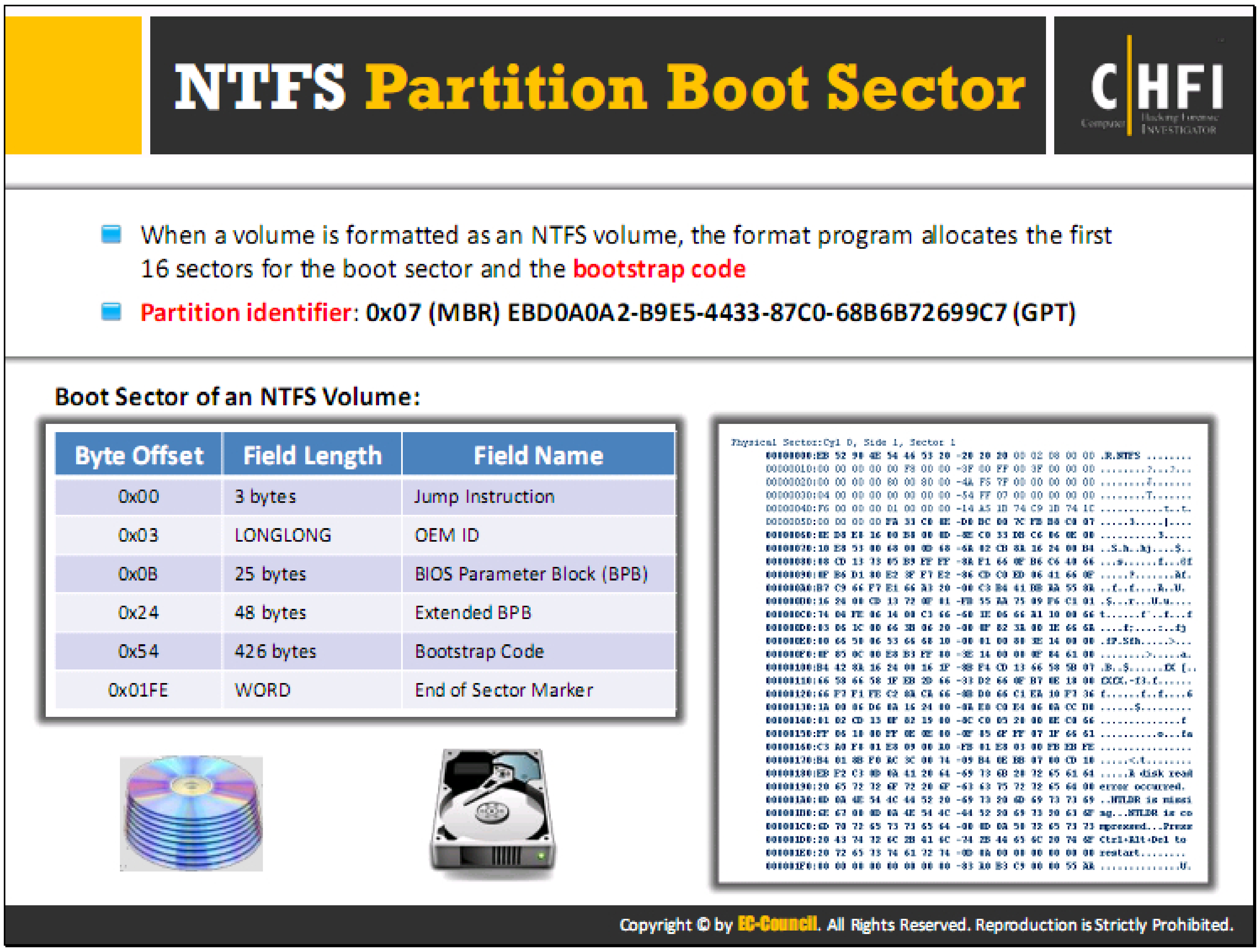
- **Hard disk:** It contains one or more partitions
- **Master Boot Record:** It contains executable master boot code that the computer system BIOS loads into memory; this code is used to scan the Master Boot Record to locate the partition table to find out which partition is active/bootable
- **Boot sector:** It is a bootable partition that stores data related to the layout of the volume and the file system structures
- **Ntldlr.dll:** It reads the contents of the Boot.ini file
- **Ntfs.sys:** It is a computer system file driver for NTFS
- **Kernel mode:** It is the processing mode that permits the executable code to have direct access to all the system components
- **User mode:** It is the processing mode in which an executable program or code runs



NTFS System Files

File Name	Description
$attrdef	Contains definitions of all system-and user-defined attributes of the volume
$badclus	Contains all the bad clusters
$bitmap	Contains bitmap for the entire volume
$boot	Contains the volume's bootstrap
$logfile	Used for recovery purposes
$mft	Contains a record for every file
$mftmirr	Mirror of the MFT used for recovering files
$quota	Indicates disk quota for each user
$upcase	Converts characters into uppercase Unicode
$volume	Contains volume name and version number

NTFS has many system files stored in root directory of the NTFS volume that store file system metadata.

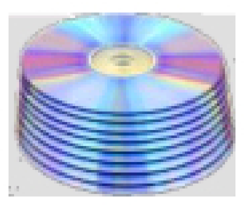


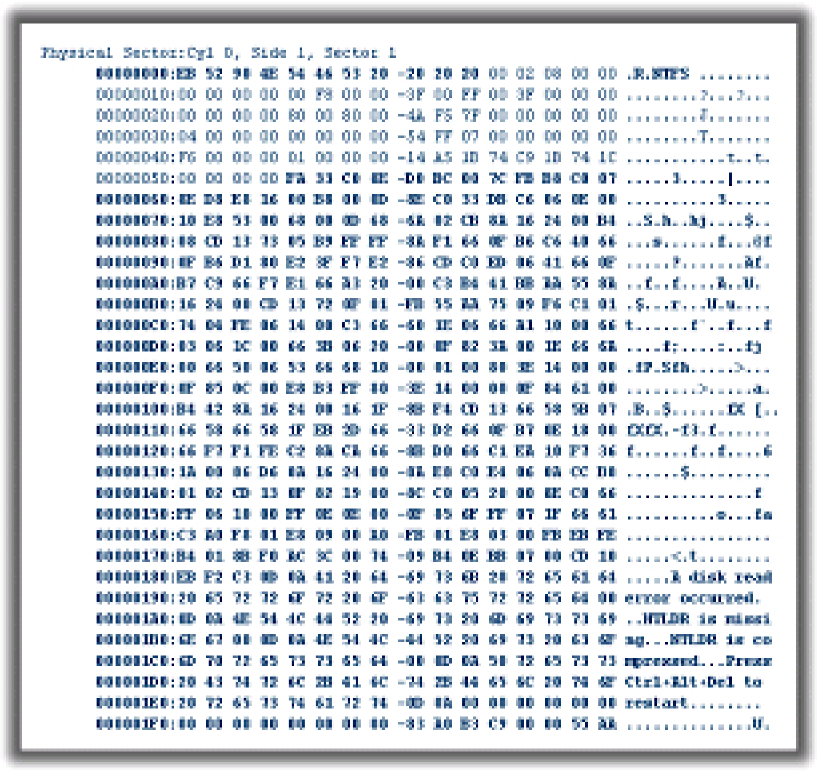
NTFS Partition Boot Sector

- When a volume is formatted as an NTFS volume, the format program allocates the first 16 sectors for the boot sector and the bootstrap code
- Partition identifier: **0x07 (MBR) EBD0A0A2-B9E5-4433-87C0-68B6B72699C7 (GPT)**

Boot Sector of an NTFS Volume:

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	LONGLONG	OEM ID
0x0B	25 bytes	BIOS Parameter Block (BPB)
0x24	48 bytes	Extended BPB
0x54	426 bytes	Bootstrap Code
0x01FE	WORD	End of Sector Marker



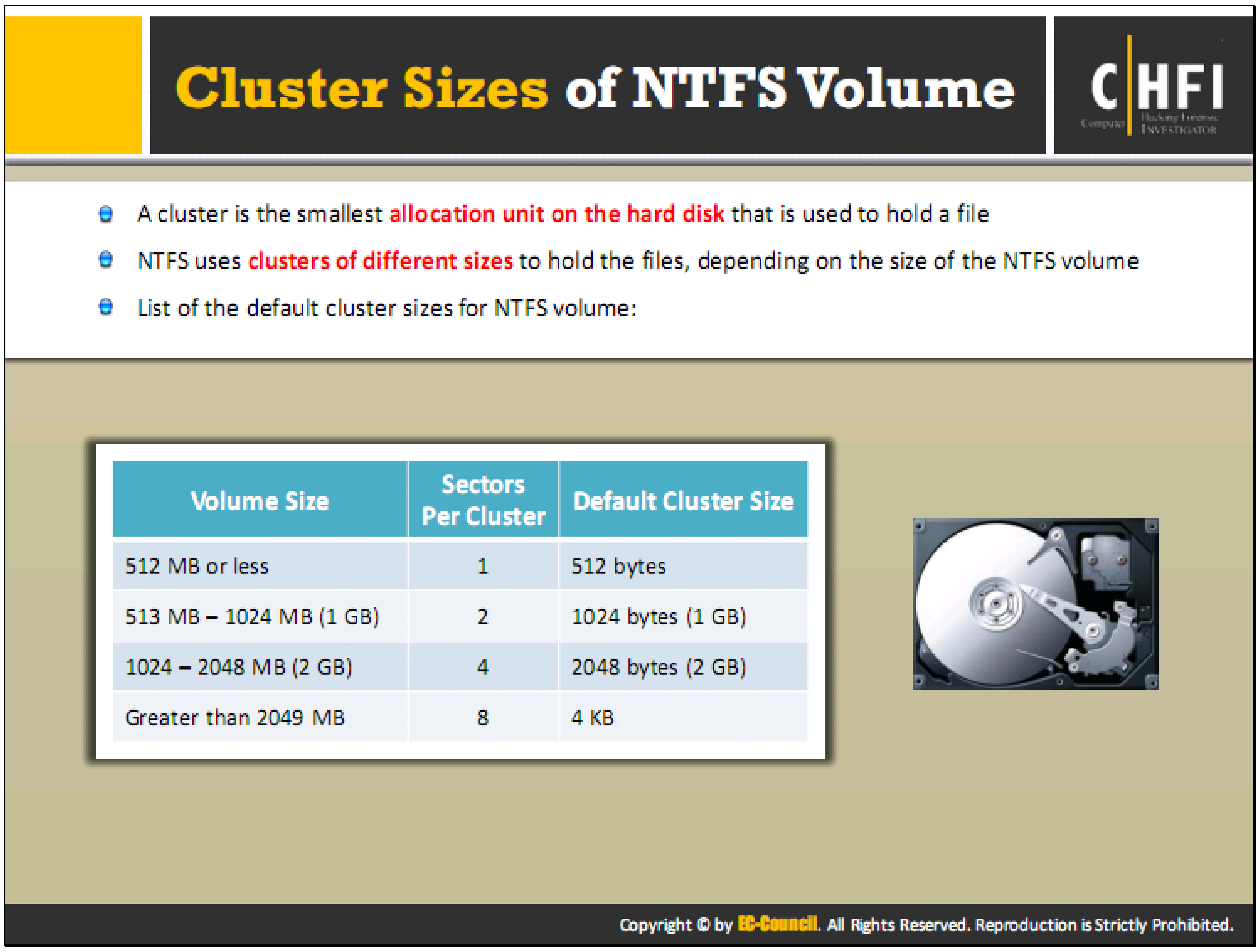


```
Physical Sector:Cyl 0, Side 1, Sector 1
        00000000:EB 52 90 4E 54 46 53 20 -20 20 20 00 02 08 00 00 .R.NTFS ........
        00000010:00 00 00 00 00 F8 00 00 -3F 00 FF 00 3F 00 00 00 ........?...?...
        00000020:00 00 00 00 80 00 80 00 -4A F5 7F 00 00 00 00 00 ........J.......
        00000030:04 00 00 00 00 00 00 00 -54 FF 07 00 00 00 00 00 ........T.......
        00000040:F6 00 00 00 01 00 00 00 -14 A5 1B 74 C9 1B 74 1C ...........t..t.
        00000050:00 00 00 00 FA 33 C0 8E -D0 BC 00 7C FB B8 C0 07 .....3.....|....
        00000060:8E D8 E8 16 00 B8 00 0D -8E C0 33 DB C6 06 0E 00 ..........3.....
        00000070:10 E8 53 00 68 00 0D 68 -6A 02 CB 8A 16 24 00 B4 ..S.h..hj....$..
        00000080:08 CD 13 73 05 B9 FF FF -8A F1 66 0F B6 C6 40 66 ...s......f...@f
        00000090:0F B6 D1 80 E2 3F F7 E2 -86 CD C0 ED 06 41 66 0F .....?.......Af.
        000000A0:B7 C9 66 F7 E1 66 A3 20 -00 C3 B4 41 BB AA 55 8A ..f..f. ...A..U.
        000000B0:16 24 00 CD 13 72 0F 81 -FB 55 AA 75 09 F6 C1 01 .$...r...U.u....
        000000C0:74 04 FE 06 14 00 C3 66 -60 1E 06 66 A1 10 00 66 t......f`..f...f
        000000D0:03 06 1C 00 66 3B 06 20 -00 0F 82 3A 00 1E 66 6A ....f;. ...:..fj
        000000E0:00 66 50 06 53 66 68 10 -00 01 00 80 3E 14 00 00 .fP.Sfh.....>...
        000000F0:0F 85 0C 00 E8 B3 FF 80 -3E 14 00 00 0F 84 61 00 ........>.....a.
        00000100:B4 42 8A 16 24 00 16 1F -8B F4 CD 13 66 58 5B 07 .B..$.......fX[.
        00000110:66 58 66 58 1F EB 2D 66 -33 D2 66 0F B7 0E 18 00 fXfX..-f3.f.....
        00000120:66 F7 F1 FE C2 8A CA 66 -8B D0 66 C1 EA 10 F7 36 f......f..f....6
        00000130:1A 00 86 D6 8A 16 24 00 -8A E8 C0 E4 06 0A CC B8 ......$.........
        00000140:01 02 CD 13 0F 82 19 00 -8C C0 05 20 00 8E C0 66 ...............f
        00000150:FF 06 10 00 FF 0E 0E 00 -0F 85 6F FF 07 1F 66 61 ..........o...fa
        00000160:C3 A0 F8 01 E8 09 00 A0 -FB 01 E8 03 00 FB EB FE ................
        00000170:B4 01 8B F0 AC 3C 00 74 -09 B4 0E BB 07 00 CD 10 .....<.t........
        00000180:EB F2 C3 0D 0A 41 20 64 -69 73 6B 20 72 65 61 64 .....A disk read
        00000190:20 65 72 72 6F 72 20 6F -63 63 75 72 72 65 64 00 error occurred.
        000001A0:0D 0A 4E 54 4C 44 52 20 -69 73 20 6D 69 73 73 69 ..NTLDR is missi
        000001B0:6E 67 00 0D 0A 4E 54 4C -44 52 20 69 73 20 63 6F ng...NTLDR is co
        000001C0:6D 70 72 65 73 73 65 64 -00 0D 0A 50 72 65 73 73 mpressed...Press
        000001D0:20 43 74 72 6C 2B 41 6C -74 2B 44 65 6C 20 74 6F Ctrl+Alt+Del to
        000001E0:20 72 65 73 74 61 72 74 -0D 0A 00 00 00 00 00 00 restart........
        000001F0:00 00 00 00 00 00 00 00 -83 A0 B3 C9 00 00 55 AA ..............U.
```

In an NTFS volume, system allocates the first 16 sectors to the boot metadata file and the next 15 sectors to the boot sector's initial program loader (IPL). The first sector, which is a boot sector, contains the bootstrap including the file system type, size, and location of NTFS data. The last sector contains an extra copy of the boot sector in order to increase file system reliability.

The following instance demonstrates the boot sector of the NTFS volume, formatted on Windows 2000. The layout has three parts, and they are as follows:

- Bytes **0x00-0x0A** constitute the jump instruction and the **OEM ID**
- Bytes **0x0B-0x53** are the **BIOS parameter block** (**BPB**) and the extended BPB
- The remaining code is the **bootstrap code** and the end of the sector marker

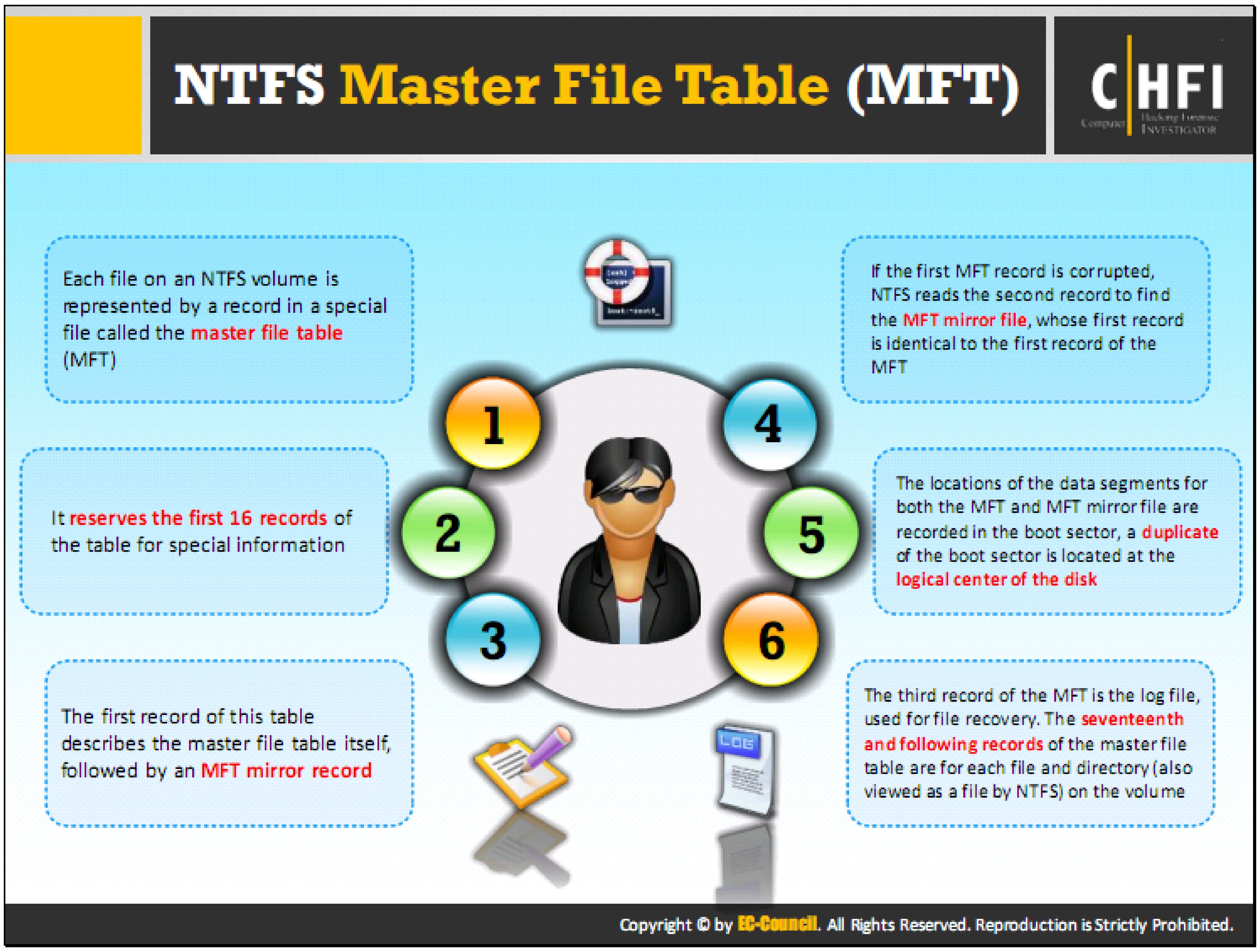


Cluster Sizes of NTFS Volume

- A cluster is the smallest **allocation unit on the hard disk** that is used to hold a file
- NTFS uses **clusters of different sizes** to hold the files, depending on the size of the NTFS volume
- List of the default cluster sizes for NTFS volume:

Volume Size	Sectors Per Cluster	Default Cluster Size
512 MB or less	1	512 bytes
513 MB – 1024 MB (1 GB)	2	1024 bytes (1 GB)
1024 – 2048 MB (2 GB)	4	2048 bytes (2 GB)
Greater than 2049 MB	8	4 KB

A cluster is the smallest allocation unit on the hard disk used to hold a file. NTFS uses clusters of different sizes to hold files depending on the size of the NTFS volume. The NTFS file system has a maximum number of clusters it can support. If the size of the cluster is small, the hard disk can efficiently store information because the other files cannot use the empty space within a cluster. The NTFS file system is an efficient file organization structure because it uses clusters of smaller sizes.



NTFS Master File Table (MFT)

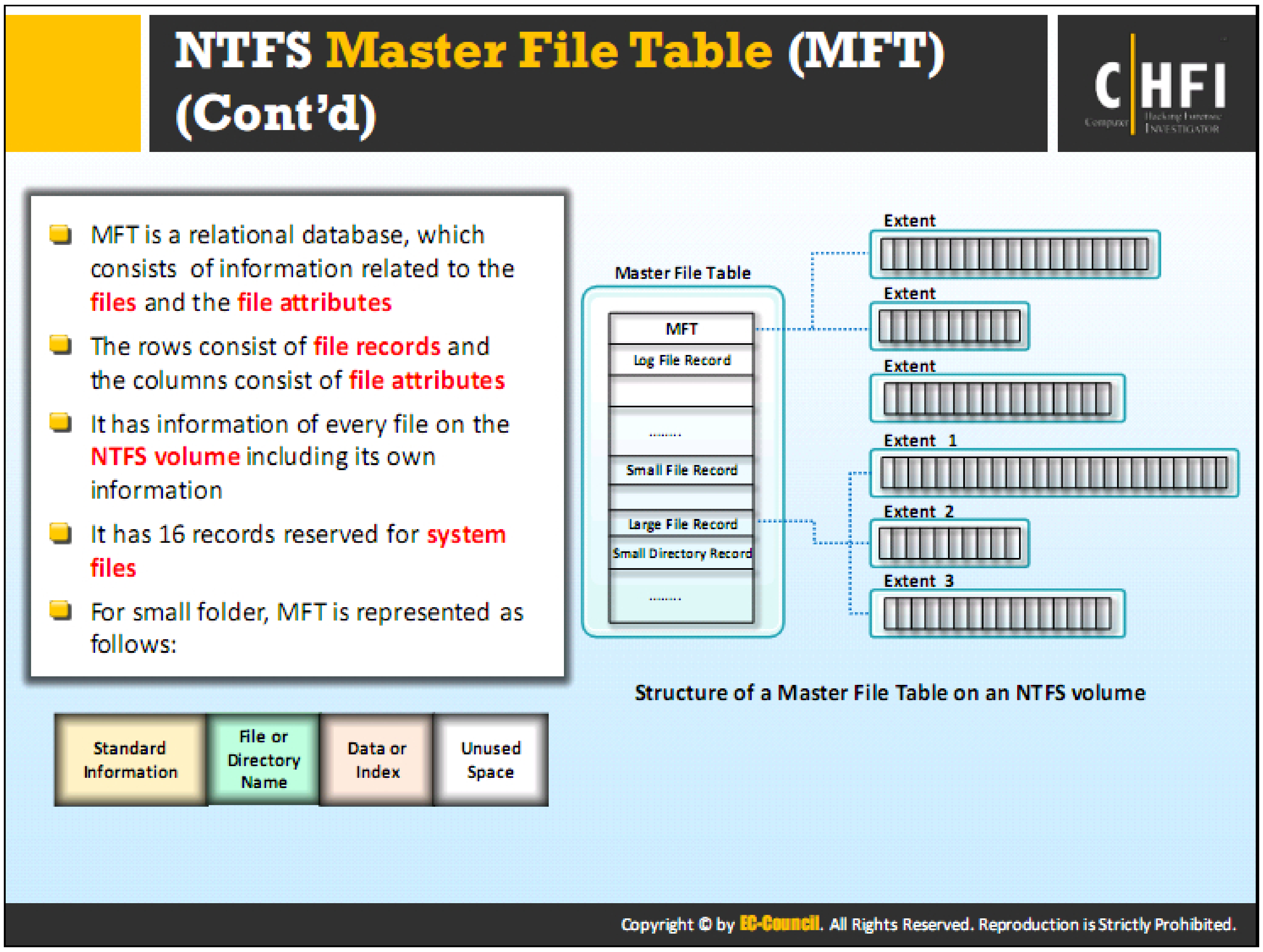
1. Each file on an NTFS volume is represented by a record in a special file called the **master file table** (MFT)
2. It **reserves the first 16 records** of the table for special information
3. The first record of this table describes the master file table itself, followed by an **MFT mirror record**
4. If the first MFT record is corrupted, NTFS reads the second record to find the **MFT mirror file**, whose first record is identical to the first record of the MFT
5. The locations of the data segments for both the MFT and MFT mirror file are recorded in the boot sector, a **duplicate** of the boot sector is located at the **logical center of the disk**
6. The third record of the MFT is the log file, used for file recovery. The **seventeenth and following records** of the master file table are for each file and directory (also viewed as a file by NTFS) on the volume

The NTFS file system consists of a unique file called the master file table (MFT). NTFS volumes have at least one entry that is stored in MFT.

MFT entries or memory spaces MFT entries describe outside themselves store information regarding the file attributes such as size, time and date stamps, permissions, and data contents. With the increase in the number of files added to the NTFS volume and the entries added to the MFT, the size of the MFT increases. When the user deletes a file from the NTFS, the file system marks the values in MFT as free and makes that place reusable.

The utilities that defrag NTFS volumes on Windows 2000-based systems cannot move MFT entries, and as unnecessary fragmentation of the MFT breaks down the performance of the file system, the NTFS saves space for the MFT to maintain it as close as possible as it expands. NTFS reserves some space for the MFT in each volume, called as the MFT zone. The allocation of space follows certain simple rules, like allocation of the volume space exterior to the MFT zone in the first place, as well as allocation of the memory to the files and directories.

NTS considers average file size and other variables while allocating memory to the reserved MFT zone or the unreserved memory on the disk as the disk fills to its capacity. Volumes having less number of relatively large files will allocate the unreserved space first, whereas, volumes with a large number of relatively small files allocate the MFT zone first.



NTFS Master File Table (MFT) (Cont'd)

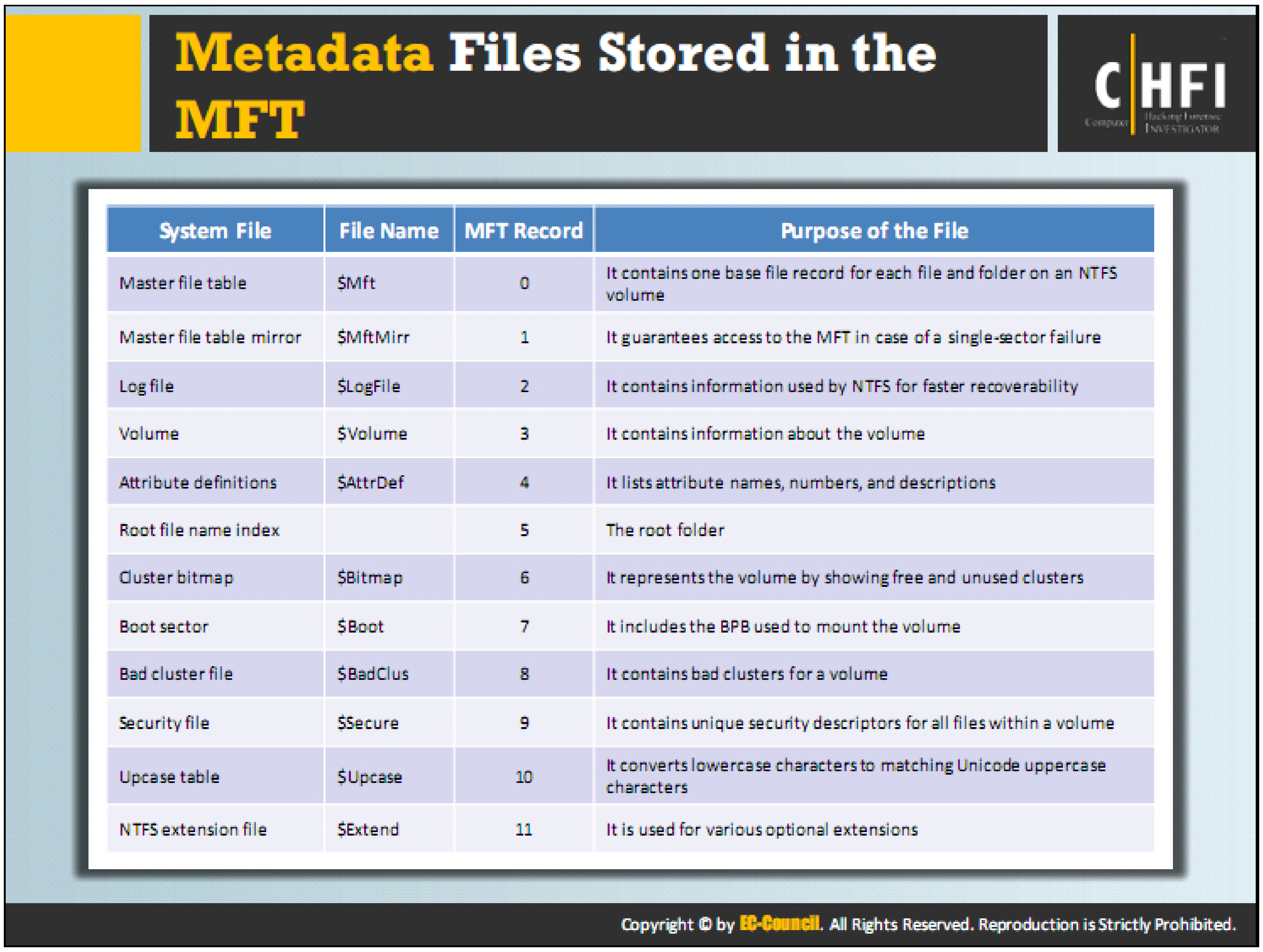
- MFT is a relational database, which consists of information related to the files and the file attributes
- The rows consist of file records and the columns consist of file attributes
- It has information of every file on the NTFS volume including its own information
- It has 16 records reserved for system files
- For small folder, MFT is represented as follows:

Structure of a Master File Table on an NTFS volume

MFT is a relational database, which consists of information regarding the files and file attributes and also defines an NTFS volume and retrieves the information about every file and directory present on it. It has a "starting point" and a sort of "table of contents" for the NTFS volume. MFT maintains a record for all the new files or directories created on the NTFS where each record's size is almost equivalent to the cluster size of the NTFS volume.

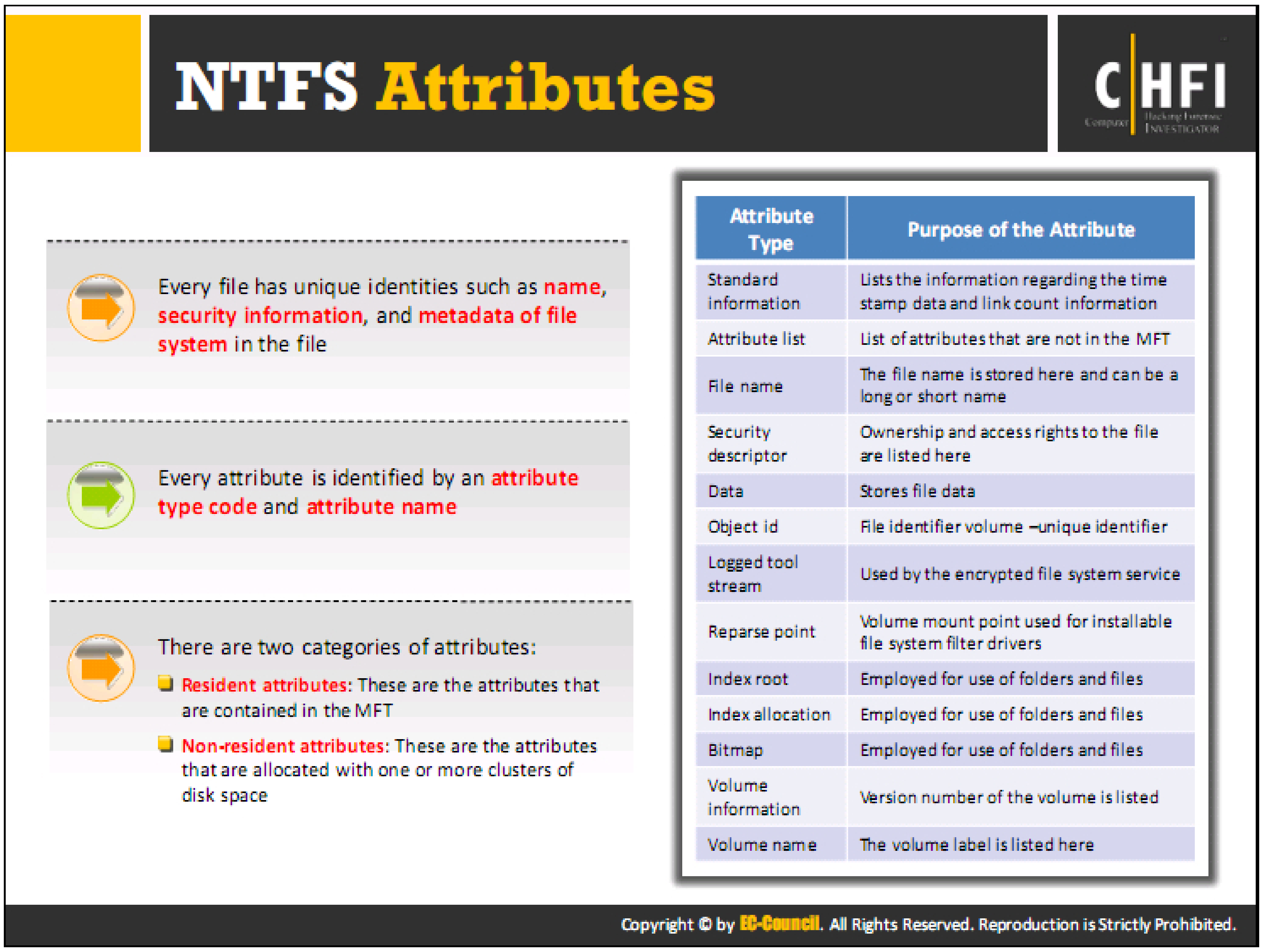
MFT stores the information regarding the files in the form of "attributes." The rows consist of file records and the columns consist of file attributes. It has 16 records reserved for system files. The figure in the above slide illustrates the small folder of MFT.

The file attributes stored within its record are resident attributes, and those that lie outside MFT are non-resident attributes. If the data attributes are small in size, then the MFT record can be stored within the record without the need of additional storage space on the NTFS volume. But it is critical that, for larger files, the additional attributes that do not fit in the MFT record moved out of the MFT record as non-resident attributes and store them as external attributes.



Metadata Files Stored in the MFT

System File	File Name	MFT Record	Purpose of the File
Master file table	$Mft	0	It contains one base file record for each file and folder on an NTFS volume
Master file table mirror	$MftMirr	1	It guarantees access to the MFT in case of a single-sector failure
Log file	$LogFile	2	It contains information used by NTFS for faster recoverability
Volume	$Volume	3	It contains information about the volume
Attribute definitions	$AttrDef	4	It lists attribute names, numbers, and descriptions
Root file name index		5	The root folder
Cluster bitmap	$Bitmap	6	It represents the volume by showing free and unused clusters
Boot sector	$Boot	7	It includes the BPB used to mount the volume
Bad cluster file	$BadClus	8	It contains bad clusters for a volume
Security file	$Secure	9	It contains unique security descriptors for all files within a volume
Upcase table	$Upcase	10	It converts lowercase characters to matching Unicode uppercase characters
NTFS extension file	$Extend	11	It is used for various optional extensions



NTFS Attributes

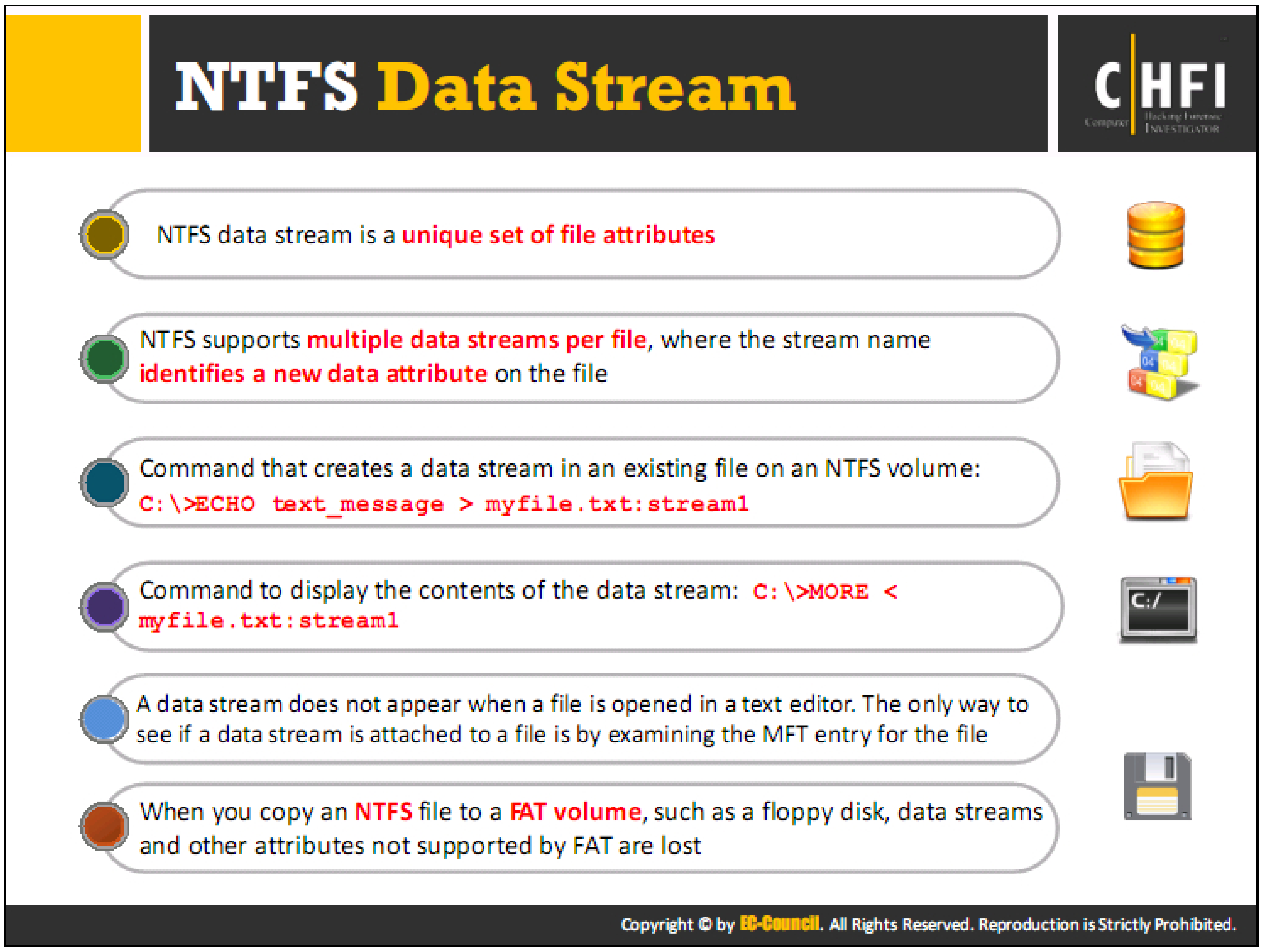
- Every file has unique identities such as **name**, **security information**, and **metadata of file system** in the file
- Every attribute is identified by an **attribute type code** and **attribute name**
- There are two categories of attributes:
  - **Resident attributes**: These are the attributes that are contained in the MFT
  - **Non-resident attributes**: These are the attributes that are allocated with one or more clusters of disk space

Attribute Type	Purpose of the Attribute
Standard information	Lists the information regarding the time stamp data and link count information
Attribute list	List of attributes that are not in the MFT
File name	The file name is stored here and can be a long or short name
Security descriptor	Ownership and access rights to the file are listed here
Data	Stores file data
Object id	File identifier volume –unique identifier
Logged tool stream	Used by the encrypted file system service
Reparse point	Volume mount point used for installable file system filter drivers
Index root	Employed for use of folders and files
Index allocation	Employed for use of folders and files
Bitmap	Employed for use of folders and files
Volume information	Version number of the volume is listed
Volume name	The volume label is listed here

The NTFS regards every file as a set of attributes. Every file has unique identities such as name, security information, and metadata of file system in the file. An attribute is an entity that has a name, property, and functions. The file system identifies every attribute with the help of an attribute type code and attribute name that assist in defining the file. The system stores every file and directory in two different ways. They are as follows:

- **Resident Attributes:** Resident attributes refer to the information, stored in the small amount of storage space directly in the MFT record. The MFT file stores common file attributes as resident attributes. For proper operation, the NTFS requires the attributes, saved in the MFT.
- **Non-Resident Attributes**: If there is no sufficient space for the attributes or if the attributes require more space than what exists in the MFT record, then the system stores such attributes in a different location and places a reference in the MFT to refer to the location of the file. These attributes stored out of MFT are non-resident attributes.
  - **STANDARD_INFORMATION:** This attribute provides general information about the file, such as CreationTime, LastChangeTime, LastModificationTime, LastAccessTime, etc.
  - **ATTRIBUTE_LIST**: The attribute list maintains a list of all the file attribute types. This attribute is present only if at least one of the attribute types is non-resident.

- **$FILE_NAME**: The file name is stored in this attribute. It also consists of fields for CreationTime, LastChangeTime, LastModificationTime and LastAccessTime.
- **OBJECT_ID**: This attribute holds an ID that the Distributed Link Tracking Service uses.
- **SECURITY_DESCRIPTOR**: This attribute has the security information of the file. In the latest NTFS versions, all security information is stored in one file called $SECURE. With this attribute, the files that have same security level need not store that information in every single file.
- **INDEX_ROOT**: A directory consists of an index, which provides information about the files related to that directory. If the index has only few entries, they are stored in the $INDEX_ROOT attribute. If there are many entries, they are stored in the $INDEX_ALLOCATION attribute. The entries in the index form a b-tree.



NTFS Data Stream

- NTFS data stream is a **unique set of file attributes**
- NTFS supports **multiple data streams per file**, where the stream name **identifies a new data attribute** on the file
- Command that creates a data stream in an existing file on an NTFS volume: `C:\>ECHO text_message > myfile.txt:stream1`
- Command to display the contents of the data stream: `C:\>MORE < myfile.txt:stream1`
- A data stream does not appear when a file is opened in a text editor. The only way to see if a data stream is attached to a file is by examining the MFT entry for the file
- When you copy an **NTFS** file to a **FAT volume**, such as a floppy disk, data streams and other attributes not supported by FAT are lost

A data stream refers to a sequence of bytes. Data addition or modification is possible to the existing files during the investigation of the disk. The data stream can be meaningless or valuable data useful for evidence either intentionally or by coincidence. It is an additional attribute of files in NTFS.

It is mandatory to use a colon (:) between the file extension and the data stream because it is the same as the data stream in the MFT.

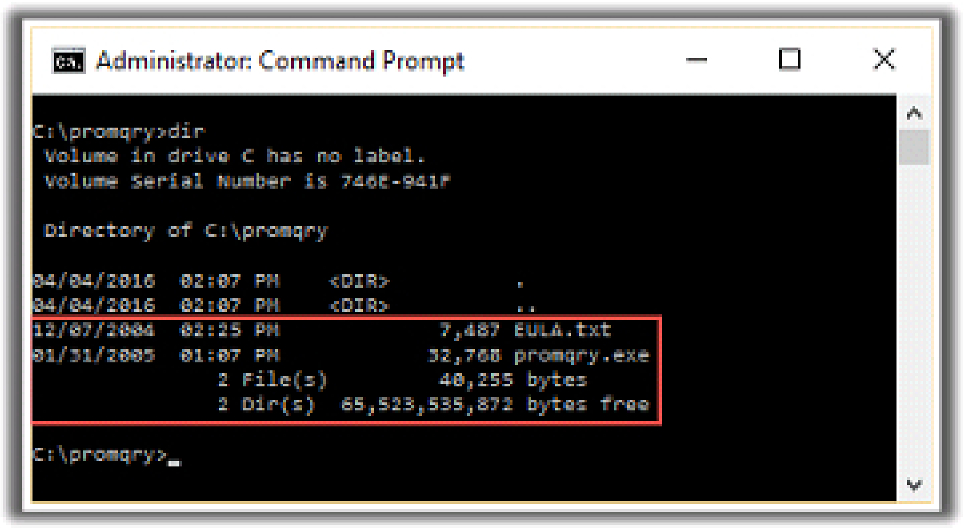
```
C:\ECHO text_message> myfile.txt: stream1
```

To display the content of the data stream, use the following command:

```
C:\ MORE < myfile.txt.stream1
```

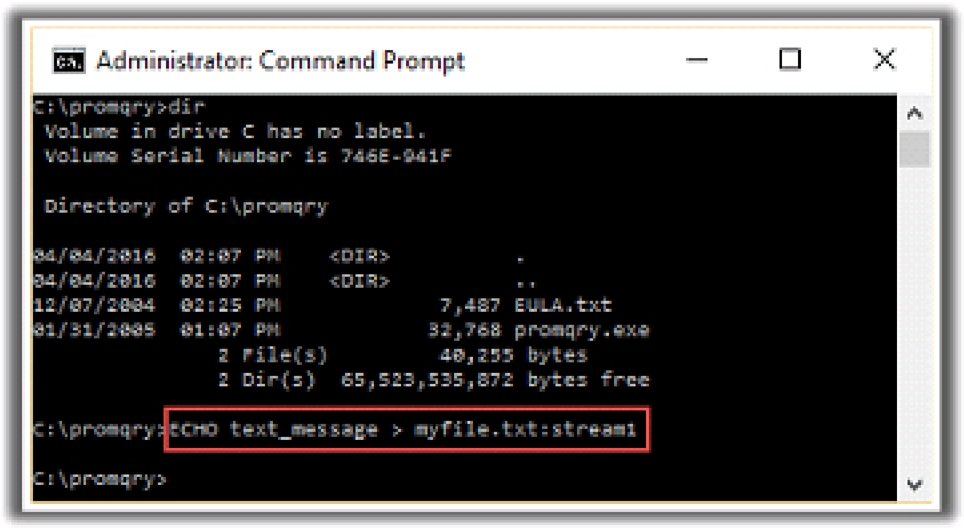
A data stream does not appear when the user opens a file stream in a text editor. The only way one can investigate whether the data stream is present in the file or not is by examining the MFT entry for the file.

NTFS Data Stream (Cont'd)



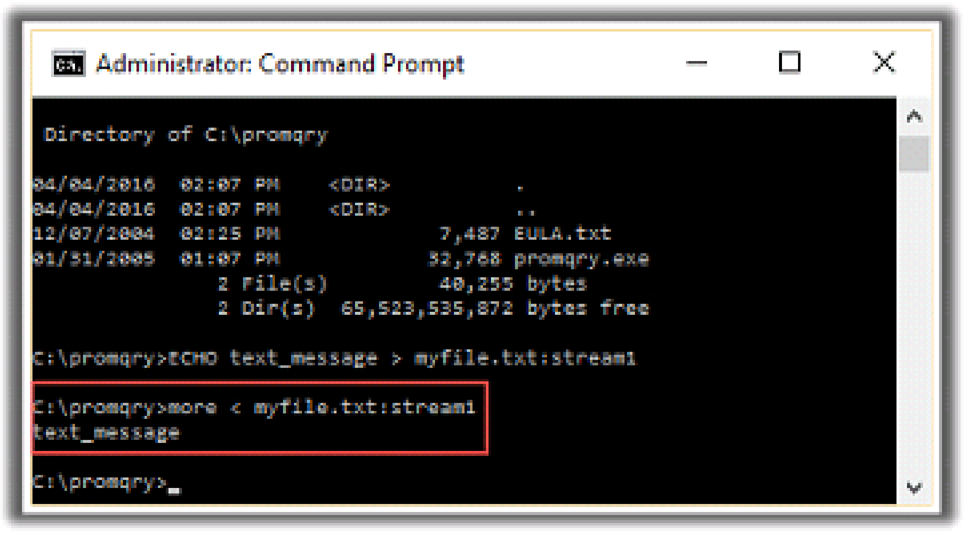
Administrator: Command Prompt

```
C:\promqry>dir
 Volume in drive C has no label.
 Volume Serial Number is 746E-941F

 Directory of C:\promqry

04/04/2016  02:07 PM    <DIR>          .
04/04/2016  02:07 PM    <DIR>          ..
12/07/2004  02:25 PM             7,487 EULA.txt
01/31/2005  01:07 PM            32,768 promqry.exe
               2 File(s)         40,255 bytes
               2 Dir(s)  65,523,535,872 bytes free

C:\promqry>_
```



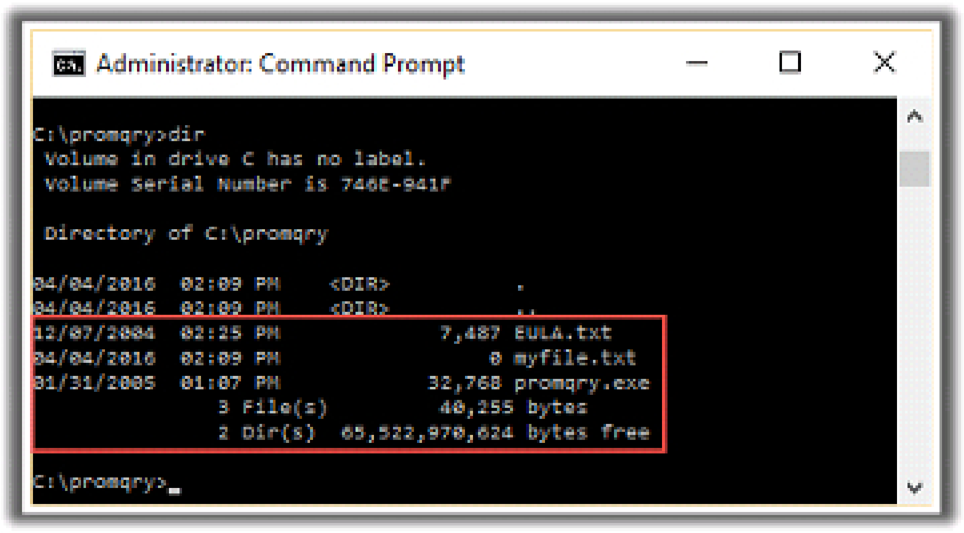
Administrator: Command Prompt

```
C:\promqry>dir
 Volume in drive C has no label.
 Volume Serial Number is 746E-941F

 Directory of C:\promqry

04/04/2016  02:07 PM    <DIR>          .
04/04/2016  02:07 PM    <DIR>          ..
12/07/2004  02:25 PM             7,487 EULA.txt
01/31/2005  01:07 PM            32,768 promqry.exe
               2 File(s)         40,255 bytes
               2 Dir(s)  65,523,535,872 bytes free

C:\promqry>ECHO text_message > myfile.txt:stream1

C:\promqry>
```



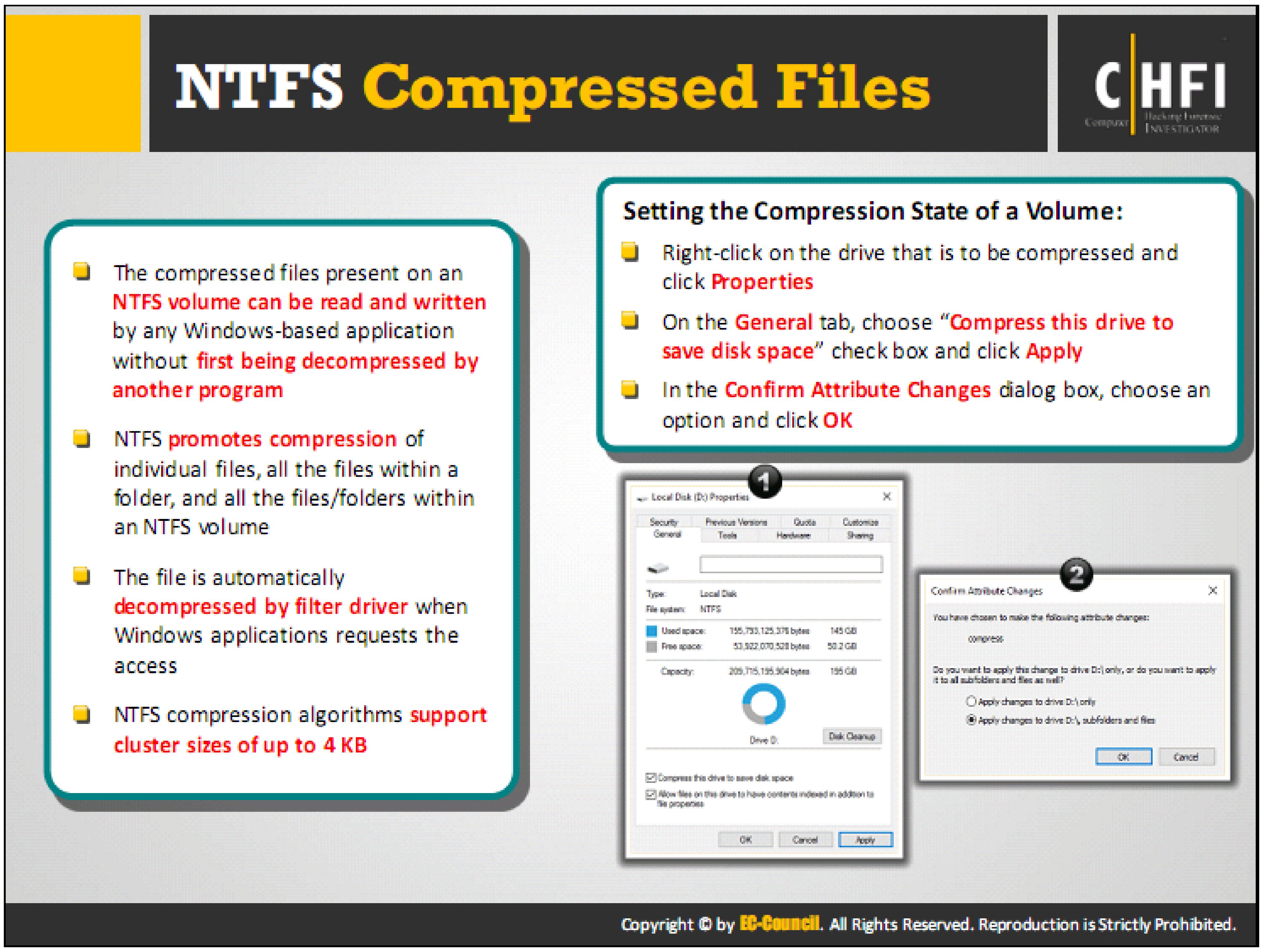
Administrator: Command Prompt

```
 Directory of C:\promqry

04/04/2016  02:07 PM    <DIR>          .
04/04/2016  02:07 PM    <DIR>          ..
12/07/2004  02:25 PM             7,487 EULA.txt
01/31/2005  01:07 PM            32,768 promqry.exe
               2 File(s)         40,255 bytes
               2 Dir(s)  65,523,535,872 bytes free

C:\promqry>ECHO text_message > myfile.txt:stream1

C:\promqry>more < myfile.txt:stream1
text_message

C:\promqry>_
```



Administrator: Command Prompt

```
C:\promqry>dir
 Volume in drive C has no label.
 Volume Serial Number is 746E-941F

 Directory of C:\promqry

04/04/2016  02:09 PM    <DIR>          .
04/04/2016  02:09 PM    <DIR>          ..
12/07/2004  02:25 PM             7,487 EULA.txt
04/04/2016  02:09 PM                 0 myfile.txt
01/31/2005  01:07 PM            32,768 promqry.exe
               3 File(s)         40,255 bytes
               2 Dir(s)  65,522,970,624 bytes free

C:\promqry>_
```



NTFS Compressed Files

- The compressed files present on an NTFS volume can be read and written by any Windows-based application without first being decompressed by another program
- NTFS promotes compression of individual files, all the files within a folder, and all the files/folders within an NTFS volume
- The file is automatically decompressed by filter driver when Windows applications requests the access
- NTFS compression algorithms support cluster sizes of up to 4 KB

Setting the Compression State of a Volume:

- Right-click on the drive that is to be compressed and click Properties
- On the General tab, choose "Compress this drive to save disk space" check box and click Apply
- In the Confirm Attribute Changes dialog box, choose an option and click OK

1

Local Disk (D:) Properties

Security | Previous Versions | Quota | Customize
General | Tools | Hardware | Sharing

Type: Local Disk
File system: NTFS
Used space: 155,793,125,376 bytes 145 GB
Free space: 53,922,070,528 bytes 50.2 GB
Capacity: 209,715,195,904 bytes 195 GB
Drive D:

[✓] Compress this drive to save disk space
[✓] Allow files on this drive to have contents indexed in addition to file properties

2

Confirm Attribute Changes

You have chosen to make the following attribute changes:

compress

Do you want to apply this change to drive D:\ only, or do you want to apply it to all subfolders and files as well?

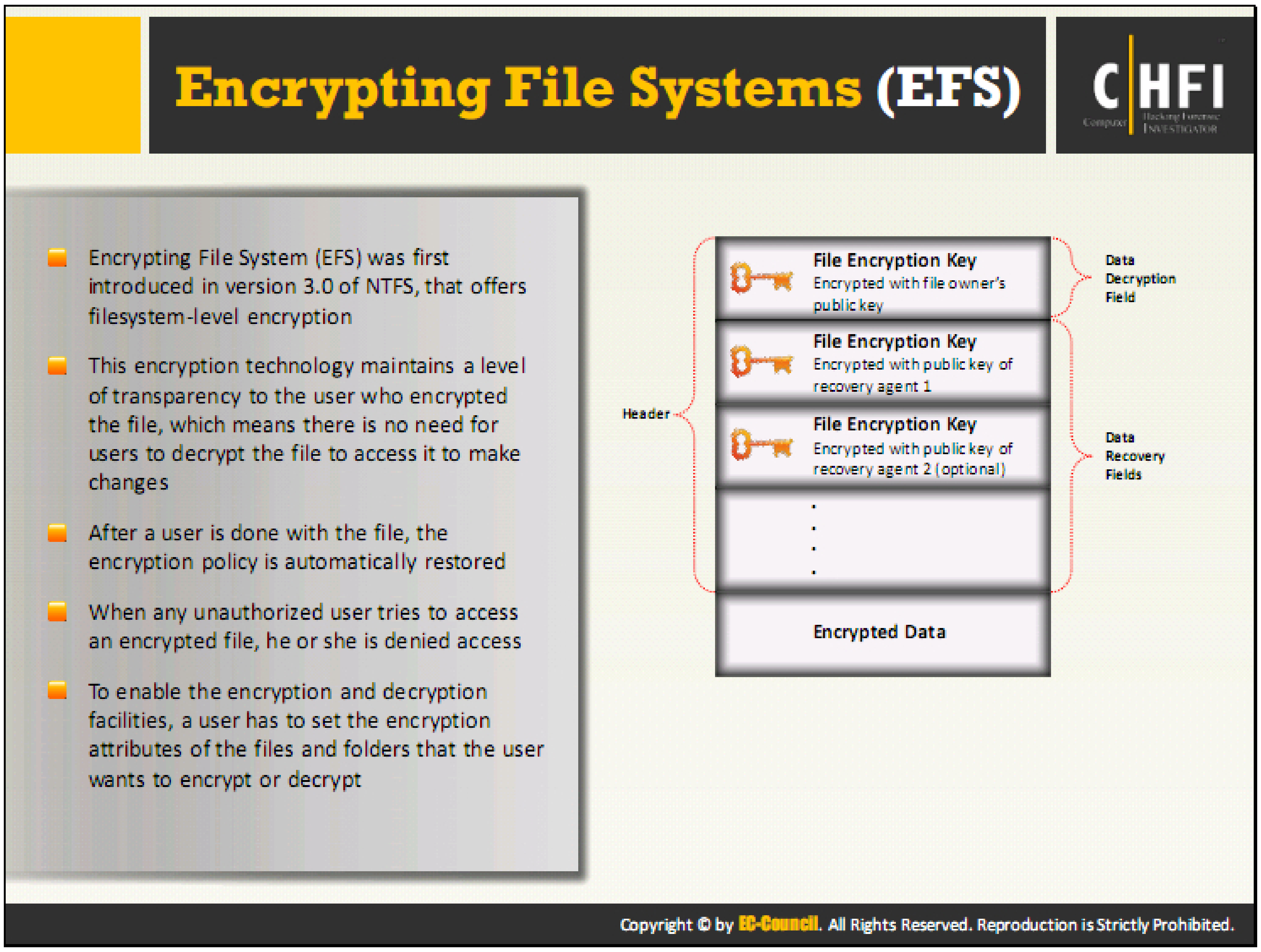
( ) Apply changes to drive D:\ only
(●) Apply changes to drive D:\, subfolders and files

Windows NT/2000 supports compression of files, folders, and the NTFS volumes. All the any Windows-based application can read and write the files, compressed on an NTFS volume without decompressing them. The decompression occurs automatically when the system or application tries to read file and compression takes place when it closes or saves the file.

NTFS contains compression algorithms that support cluster sizes of about 4 KB. When the cluster size is greater than 4 KB on an NTFS volume, none of the NTFS compression functions are available.

Setting the Compression State of a Volume

- Right-click on the drive that is to be compressed and click **Properties**
- On the **General** tab, choose "**Compress this drive to save disk space**" check box and click **Apply**
- In the **Confirm Attribute Changes** dialog box, choose an option and click **OK**



Encrypting File Systems (EFS)

- Encrypting File System (EFS) was first introduced in version 3.0 of NTFS, that offers filesystem-level encryption
- This encryption technology maintains a level of transparency to the user who encrypted the file, which means there is no need for users to decrypt the file to access it to make changes
- After a user is done with the file, the encryption policy is automatically restored
- When any unauthorized user tries to access an encrypted file, he or she is denied access
- To enable the encryption and decryption facilities, a user has to set the encryption attributes of the files and folders that the user wants to encrypt or decrypt

Header
File Encryption Key
Encrypted with file owner's public key
File Encryption Key
Encrypted with public key of recovery agent 1
File Encryption Key
Encrypted with public key of recovery agent 2 (optional)
Encrypted Data
Data Decryption Field
Data Recovery Fields

To protect files from mishandling and to ensure their security, the system should encrypt them. NTFS has Encrypting File System (EFS) as built-in feature. Encryption in file systems uses symmetric key encryption technology with public key technology for encryption. The user gets a digital certificate with a public key and private key pair. A private key is not applicable for the users logged on to the local systems; instead the system uses an EFS key to set the key for local users.

This encryption technology maintains a level of transparency to the users, who have encrypted the file. There is no need for users to decrypt the file when they access it to make changes. Again, after the user has completed working on a file, the systems will save the changes and restore the encryption policy automatically. When any unauthorized user tries to access the encrypted file, he or she receives an "Access denied" message.

To enable encryption and decryption facilities in a Windows NT–based operating system, the user has to set the encryption attributes to files and folders that he or she wants to encrypt or decrypt.

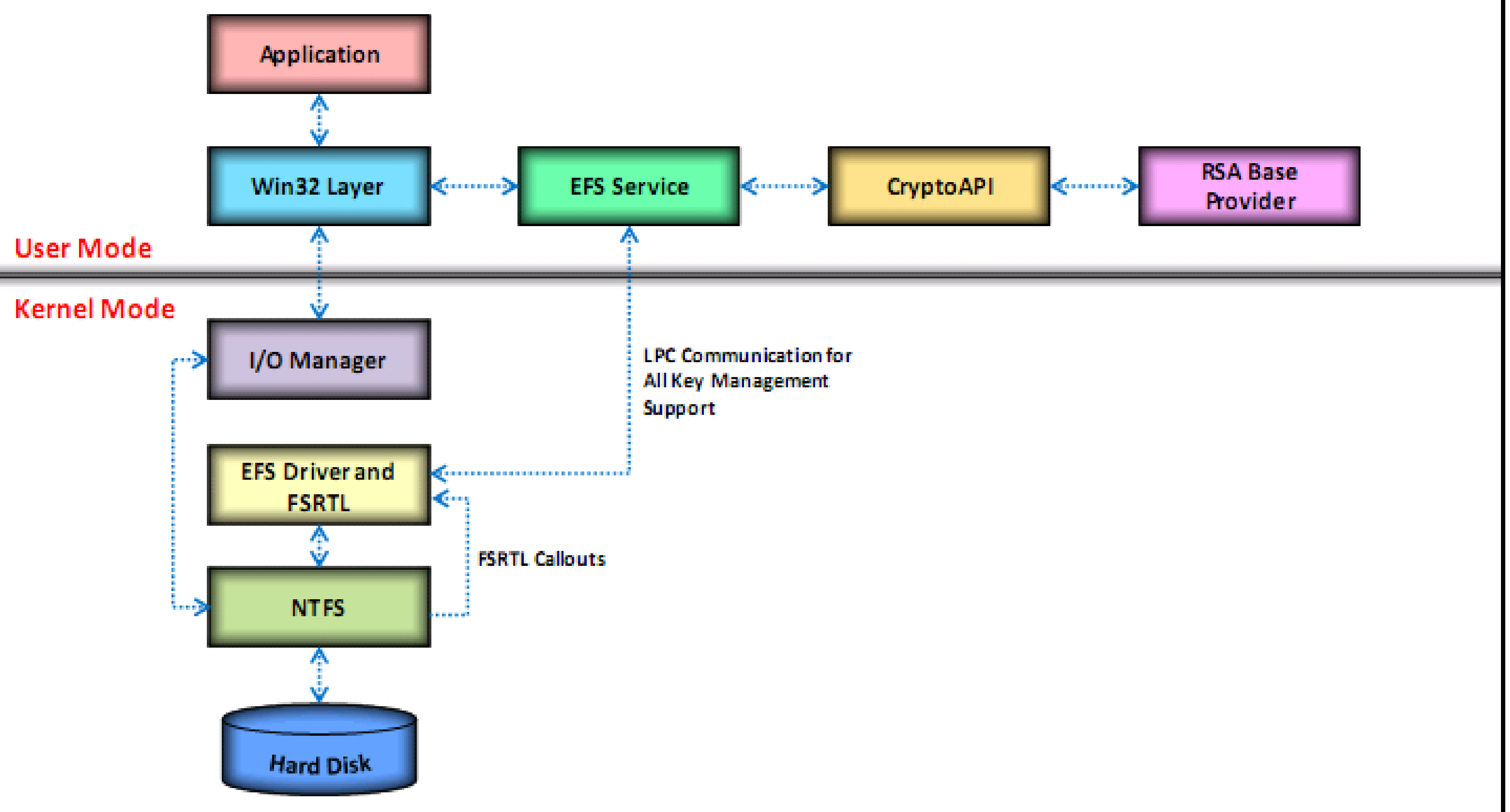
The system automatically encrypts all the files and subfolders present in a folder. To take the best advantage of the encryption capability, experts recommend that the system should have encryption at the folder level. That means a folder should not contain encrypted files along with unencrypted files.

The users can manually encrypt the files using the graphical user interface (GUI) in Windows, or by the use of a command line tool like Cipher to encrypt a file or folder or using Windows Explorer and selecting proper options available in the menu.

Encrypting a file, as NTFS protects files from unauthorized access and ensures a high level of security, is important to the files present in the system. The system issues a file encryption certificate whenever a user encrypts a file. If the person loses that certificate and related private key (through a disk or any other reason), he or she can perform data recovery through the recovery key agent.

In a Windows 2000 server–based network, which maintains Active Directory, the domain administrator is the recovery agent by default. There is an advance preparation of recovery for the files even before the user or system encrypts them. The recovery agent holds a special certificate and related private key, which helps in data recovery, giving a scope of influence of the recovery policy supported by new versions of Windows.

Components of EFS



EFS Service

EFS service, which is part of the security subsystem, acts as an interface with the EFS driver by using local procedure call (LPC) communication port between the Local Security Authority (LSA) and the kernel-mode security reference monitor. It also acts as interface with CryptoAPI in user mode in order to derive file encryption keys to generate data decryption fields (DDFs) and data recovery fields (DRFs). This service also supports Win32 APIs.

The EFS service uses CryptoAPI to extract the file encryption key (FEK) for a data file, uses it to encode the FEK and produce the DDF.

EFS Driver

The EFS driver is a file system filter driver stacked on top of NTFS. It connects with the EFS service to obtain file encryption keys, DDFs, DRFs, and other key management services. It sends this information to the EFS FSRTL to perform file system functions, such as open, read, write, and append.

CryptoAPI

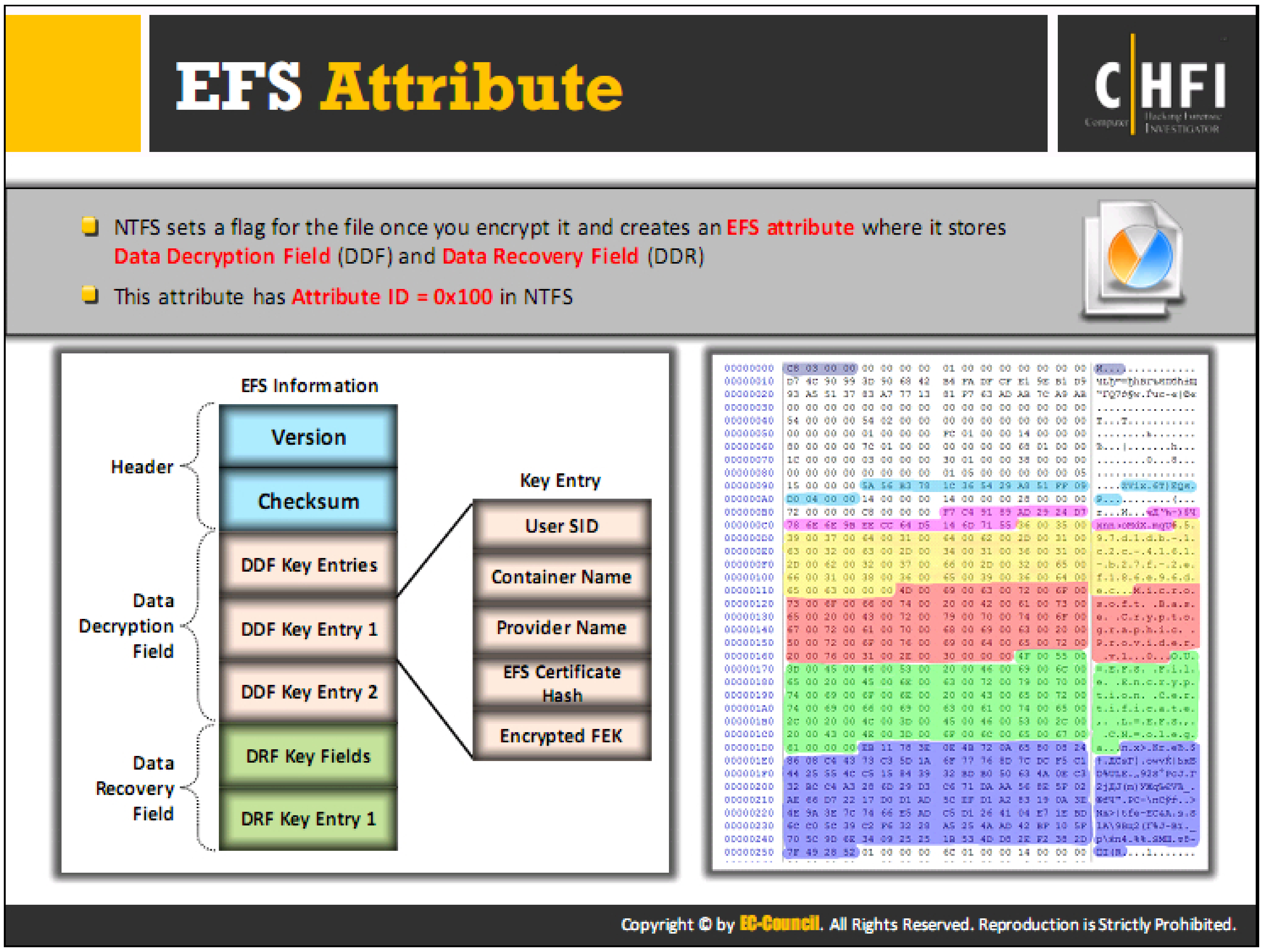
CryptoAPI contains a set of functions that allow application developers to encrypt their Win32 as the functions allow applications to encrypt or digitally sign data and also offer security for private key data. It supports public key and symmetric-key operations such as generation, management and secure storage, exchange, encryption, decryption, hashing, digital signatures, and verification of signatures.

EFS FSRTL

The EFS FSRTL is part of EFS driver that implements NTFS callouts to handle various file system operations such as reads, writes, and opens on encrypted files and directories, and operations to encrypt, decrypt, and recover file data when the system writes it to or reads it from disk. The EFS driver and FSRTL act as single component, but never communicate directly. They communicate by using the NTFS file control callout mechanism for sending messages to each other.

Win32 API

EFS provides an API set to expose its features that also provides a programming interface for operations such as encrypting plaintext files, decrypting or recovering cipher text files, and importing and exporting encrypted files without decrypting them.

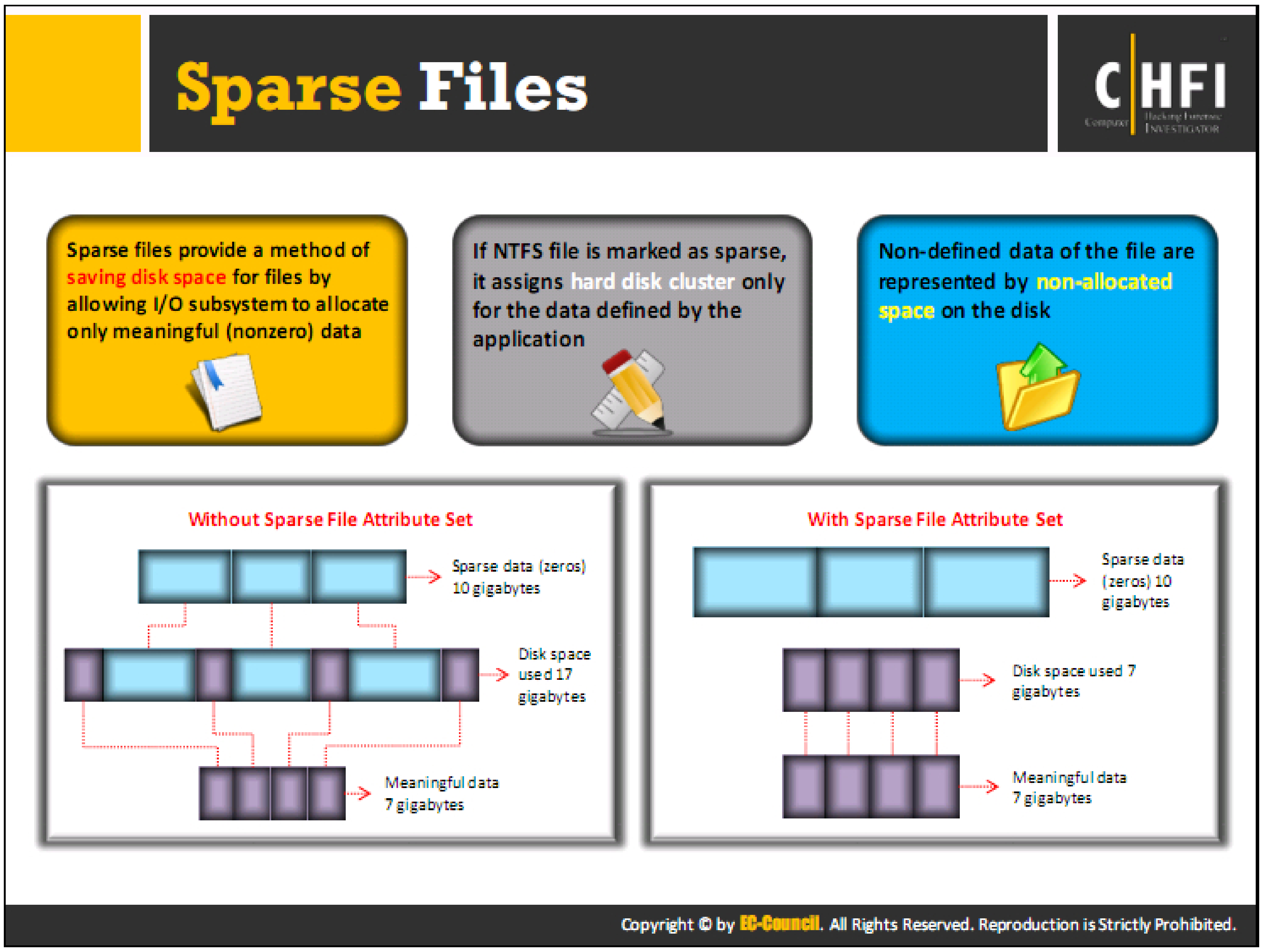


EFS Attribute

- NTFS sets a flag for the file once you encrypt it and creates an **EFS attribute** where it stores **Data Decryption Field** (DDF) and **Data Recovery Field** (DDR)
- This attribute has **Attribute ID = 0x100** in NTFS

```
00000000  C8 03 00 00 00 00 00 00  01 00 00 00 00 00 00 00  È...............
00000010  D7 4C 90 99 3D 90 68 42  84 FA DF CF E1 5E B1 D9  ×L..=.hB„úßÏá^±Ù
00000020  93 A5 51 37 83 A7 77 13  81 F7 63 AD A8 7C A9 A8  "¥Q7ƒ§w..÷c.¨|©¨
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040  54 00 00 00 54 02 00 00  00 00 00 00 00 00 00 00  T...T...........
00000050  00 00 00 00 01 00 00 00  FC 01 00 00 14 00 00 00  ........ü.......
00000060  80 00 00 00 7C 01 00 00  00 00 00 00 68 01 00 00  €...|.......h...
00000070  1C 00 00 00 03 00 00 00  30 01 00 00 38 00 00 00  ........0...8...
00000080  00 00 00 00 00 00 00 00  01 05 00 00 00 00 00 05  ................
00000090  15 00 00 00 5A 56 83 78  1C 36 54 29 A8 51 FF 09  ....ZVƒx.6T)¨Qÿ.
000000A0  D0 04 00 00 14 00 00 00  14 00 00 00 28 00 00 00  Ð...........(...
000000B0  72 00 00 00 C8 00 00 00  F7 C4 91 89 AD 29 24 D7  r...È...÷Ä'‰.)$×
000000C0  78 6E 6E 9B EE CC 64 D5  14 6D 71 55 36 00 35 00  xnn›îÌdÕ.mqU6.5.
000000D0  39 00 37 00 64 00 31 00  64 00 62 00 2D 00 31 00  9.7.d.1.d.b.-.1.
000000E0  63 00 32 00 63 00 2D 00  34 00 31 00 36 00 31 00  c.2.c.-.4.1.6.1.
000000F0  2D 00 62 00 32 00 37 00  66 00 2D 00 32 00 65 00  -.b.2.7.f.-.2.e.
00000100  66 00 31 00 38 00 36 00  65 00 39 00 36 00 64 00  f.1.8.6.e.9.6.d.
00000110  65 00 63 00 00 00 4D 00  69 00 63 00 72 00 6F 00  e.c...M.i.c.r.o.
00000120  73 00 6F 00 66 00 74 00  20 00 42 00 61 00 73 00  s.o.f.t. .B.a.s.
00000130  65 00 20 00 43 00 72 00  79 00 70 00 74 00 6F 00  e. .C.r.y.p.t.o.
00000140  67 00 72 00 61 00 70 00  68 00 69 00 63 00 20 00  g.r.a.p.h.i.c. .
00000150  50 00 72 00 6F 00 76 00  69 00 64 00 65 00 72 00  P.r.o.v.i.d.e.r.
00000160  20 00 76 00 31 00 2E 00  30 00 00 00 4F 00 55 00   .v.1...0...O.U.
00000170  3D 00 45 00 46 00 53 00  20 00 46 00 69 00 6C 00  =.E.F.S. .F.i.l.
00000180  65 00 20 00 45 00 6E 00  63 00 72 00 79 00 70 00  e. .E.n.c.r.y.p.
00000190  74 00 69 00 6F 00 6E 00  20 00 43 00 65 00 72 00  t.i.o.n. .C.e.r.
000001A0  74 00 69 00 66 00 69 00  63 00 61 00 74 00 65 00  t.i.f.i.c.a.t.e.
000001B0  2C 00 20 00 4C 00 3D 00  45 00 46 00 53 00 2C 00  ,. .L.=.E.F.S.,.
000001C0  20 00 43 00 4E 00 3D 00  6F 00 6C 00 65 00 67 00   .C.N.=.o.l.e.g.
000001D0  61 00 00 00 EB 11 78 3E  0E 4B 72 0A 65 80 08 24  a...ë.x>.Kr.e€.$
000001E0  86 08 C4 43 73 C3 5D 1A  6F 77 76 8D 7C DC F5 C1  †.ÄCsÃ].owv.|ÜõÁ
000001F0  44 25 55 4C C5 15 84 39  32 8D B0 50 63 4A 0E C3  D%ULÅ.„92.°PcJ.Ã
00000200  32 8C C4 A3 28 6D 29 D3  C6 71 DA AA 56 8E 5F 02  2ŒÄ£(m)ÓÆqÚªVŽ_.
00000210  AE 66 D7 22 17 D0 D1 AD  5C EF D1 A2 83 19 0A 3E  ®f×".ÐÑ.\ïÑ¢ƒ..>
00000220  4E 9A 3E 7C 74 66 E5 AD  C5 D1 26 41 04 E7 1E BD  Nš>|tfå.ÅÑ&A.ç.½
00000230  6C C0 5C 39 C2 F6 32 28  A5 25 4A AD 42 BF 10 5F  lÀ\9Âö2(¥%J.B¿._
00000240  70 5C 9D 6E 34 09 25 25  1B 53 4D D8 2E F2 38 2D  p\.n4.%%.SMØ.ò8-
00000250  7F 49 28 52 01 00 00 00  6C 01 00 00 14 00 00 00  .I(R....l.......
```

NTFS sets a flag for the file after encrypting it and creates an EFS attribute where it stores the Data Decryption Field (DDF) and Data Recovery Field (DRF). This attribute has Attribute ID = 0x100 in NTFS.



Sparse Files

Sparse files provide a method of saving disk space for files by allowing I/O subsystem to allocate only meaningful (nonzero) data

If NTFS file is marked as sparse, it assigns hard disk cluster only for the data defined by the application

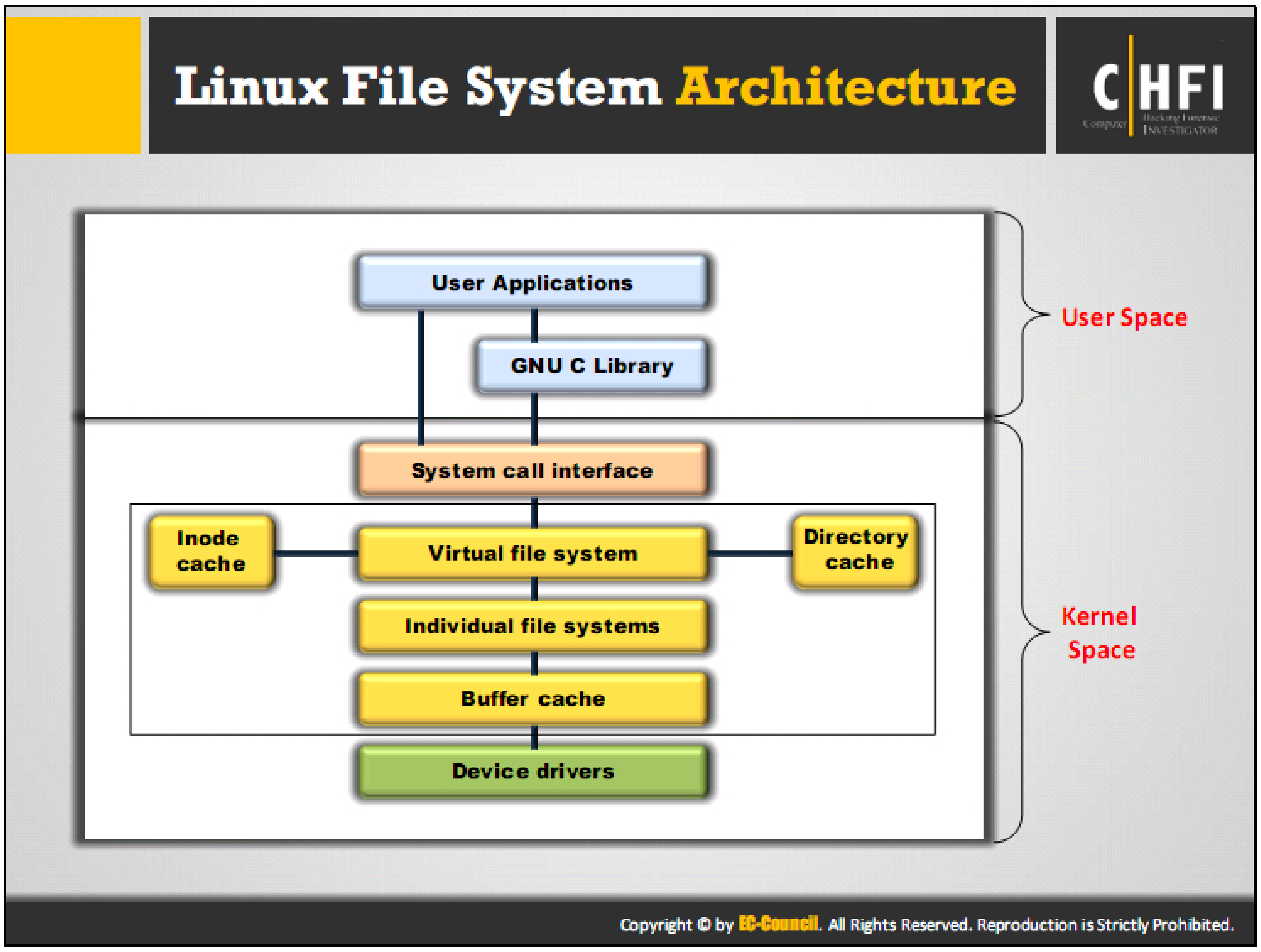
Non-defined data of the file are represented by non-allocated space on the disk

Without Sparse File Attribute Set
Sparse data (zeros) 10 gigabytes
Disk space used 17 gigabytes
Meaningful data 7 gigabytes
With Sparse File Attribute Set
Sparse data (zeros) 10 gigabytes
Disk space used 7 gigabytes
Meaningful data 7 gigabytes

A sparse file is a type of computer file that attempts to use file system space more efficiently when blocks allocated to the file are mostly empty. For better efficiency, the file system writes brief information (metadata) about the file to the empty blocks to make up the block, using less disk space. For files, they offer a technique of saving disk space by allowing the I/O subsystem to allocate only meaningful (nonzero) data. In a sparse NFTS file, clusters assigned for the data that an application defines, and the file system marks the space as non-allocated in the case of non-defined data.

Linux File Systems

Linux OS uses different file systems to store the data. As the investigators may encounter the attack source or victim systems to be running on Linux, they should have comprehensive knowledge regarding the storage methods it employs. The following section will provide you a deep insight about the various Linux file systems and their storage mechanisms.



Linux File System Architecture

The Linux file system architecture consists of two parts namely:

- User Space: The protected memory area where the user processes run and this area contains the available memory.
- Kernel Space: The memory space where the system supplies all kernel services through kernel processes. The users can access this space through the system call only. A user process turns into kernel process only when it executes a system call.

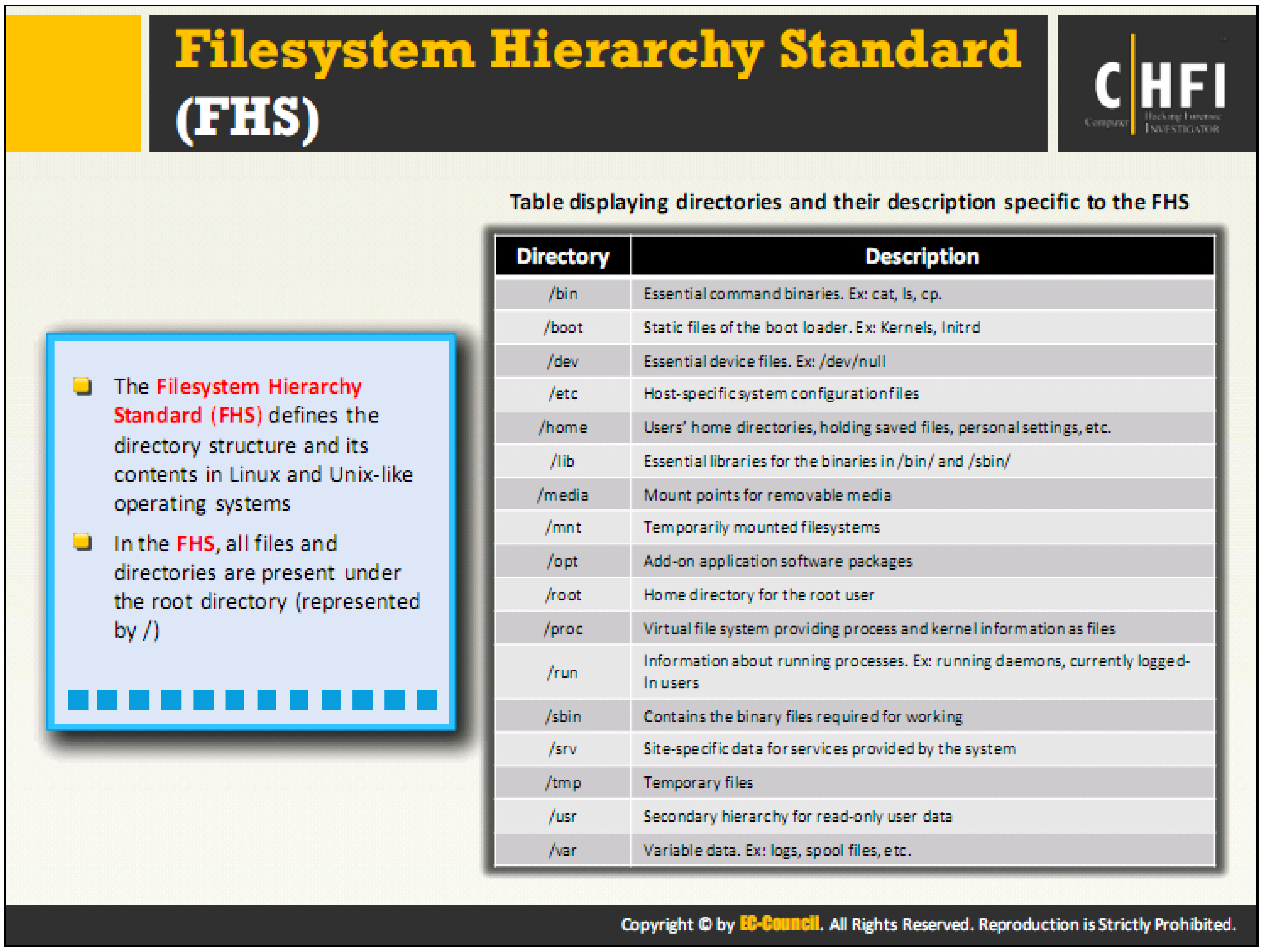
The GNUC Library (glibc) sits between the User Space and Kernel Space and provides the system call interface that connects the kernel to the user-space applications.

The Virtual file system (VFS) is an abstract layer, residing on top of a complete file system. It allows client applications to access various file systems. Its internal architecture consists of a dispatching layer which provides file system abstraction and numerous caches to enhance the file system operations performance.

The main objects managed dynamically in the VFS are the dentry and inode objects in cached manner to enhance file system access speed. Once a user opens a file, the dentry cache fills with entries that represent the directory levels which in turn represent the path. The system also creates an inode for the object which represents the file. The system develops a dentry cache using a hash table and allocates the dentry cache entries from the dentry_cache slab allocator. The system uses a least-recently-used (LRU) algorithm to prune the entries when the memory is scarce.

The inode cache acts as two lists and a hash table for quick look up. The first list defines the used inodes and the unused ones are positioned in the second list. The hash table also stores the used inodes.

Device drivers are pieces of code, linked with every physical or virtual device and help the OS in managing the device hardware. Functions of the device drivers include setting up hardware, getting the related devices in and out of services, getting data from hardware and giving it to the kernel, transferring data from the kernel to the device, and identifying and handling device errors.



Filesystem Hierarchy Standard (FHS)

- The Filesystem Hierarchy Standard (FHS) defines the directory structure and its contents in Linux and Unix-like operating systems
- In the FHS, all files and directories are present under the root directory (represented by /)

Table displaying directories and their description specific to the FHS

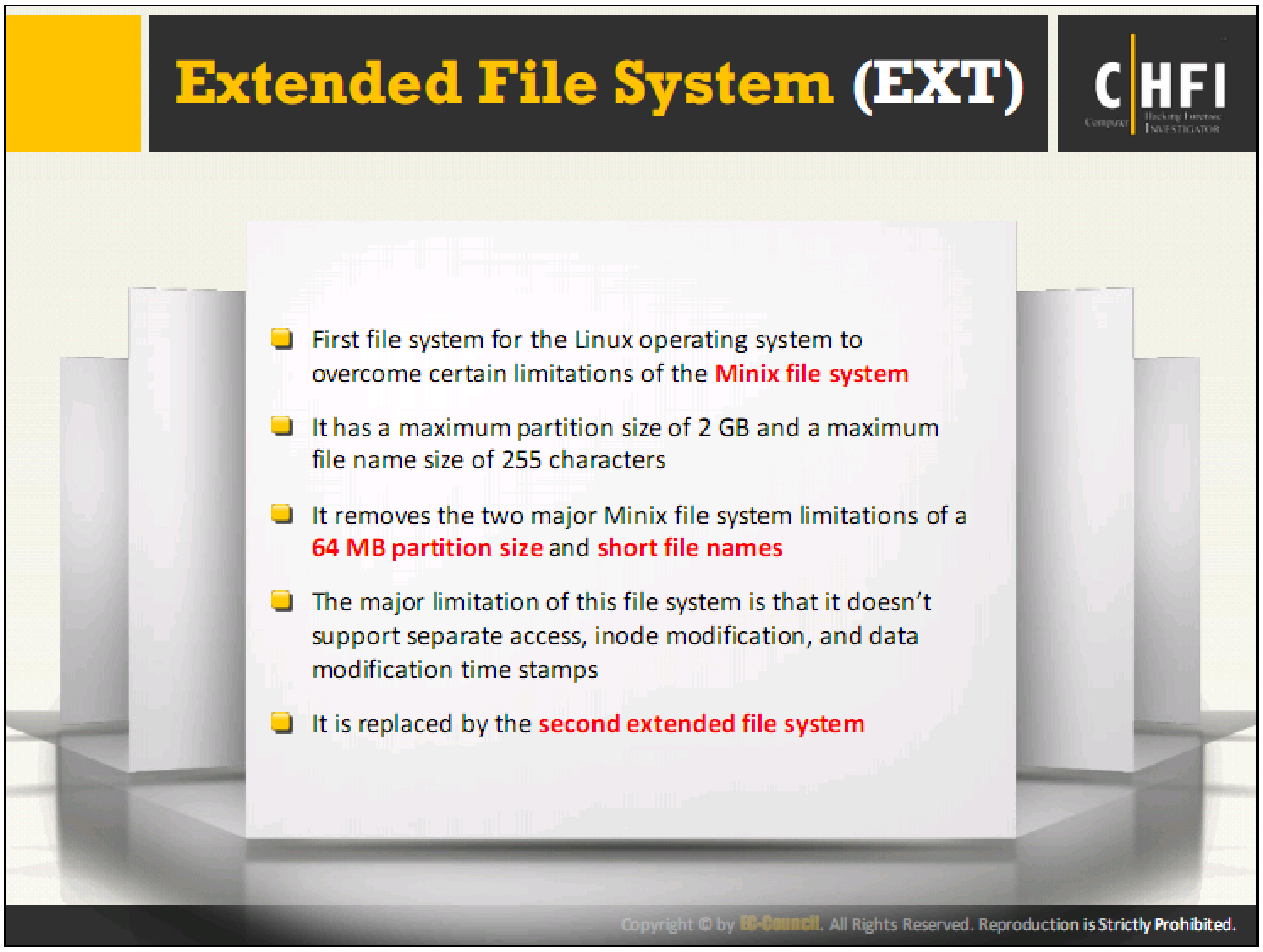
Directory	Description
/bin	Essential command binaries. Ex: cat, ls, cp.
/boot	Static files of the boot loader. Ex: Kernels, Initrd
/dev	Essential device files. Ex: /dev/null
/etc	Host-specific system configuration files
/home	Users' home directories, holding saved files, personal settings, etc.
/lib	Essential libraries for the binaries in /bin/ and /sbin/
/media	Mount points for removable media
/mnt	Temporarily mounted filesystems
/opt	Add-on application software packages
/root	Home directory for the root user
/proc	Virtual file system providing process and kernel information as files
/run	Information about running processes. Ex: running daemons, currently logged-in users
/sbin	Contains the binary files required for working
/srv	Site-specific data for services provided by the system
/tmp	Temporary files
/usr	Secondary hierarchy for read-only user data
/var	Variable data. Ex: logs, spool files, etc.

Linux is a single hierarchical tree structure, representing the file system as one single entity. It supports many different file systems. It implements a basic set of common concepts, developed for UNIX. Some of the Linux file system types are minix, Filesystem Hierarchy Standard (FHS), ext, ext2, ext3, xia, msdos, umsdos, vfat, /proc, nfs, iso 9660, hpfs, sysv, smb, and ncpfs. Minix was Linux's first file system.

The following are some of the most popular file systems:

Filesystem Hierarchy Standard (FHS)

The Filesystem Hierarchy Standard (FHS) defines the directory structure and its contents in Linux and Unix-like operating systems. In the FHS, all files and directories are present under the root directory (represented by /).

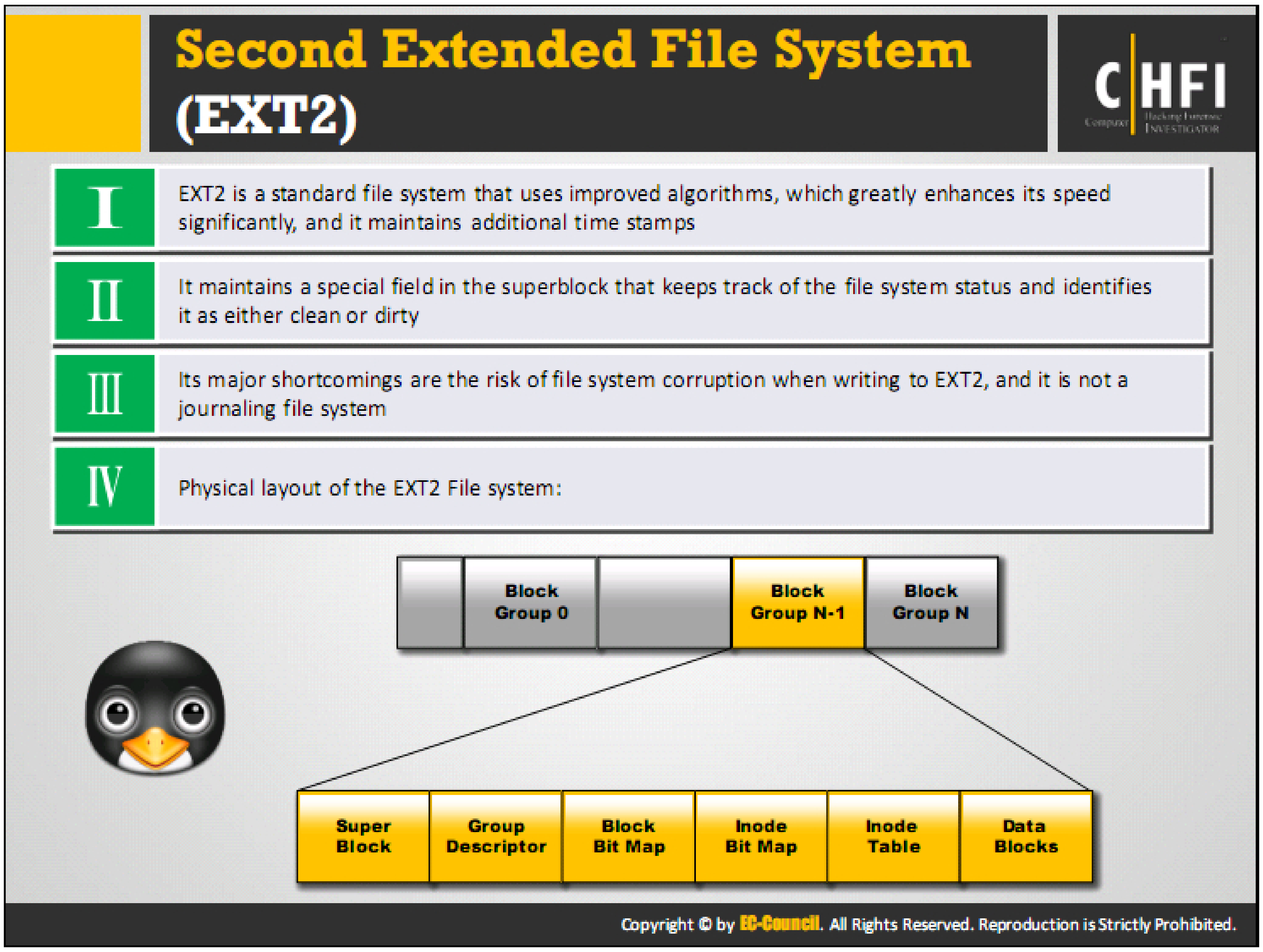


Extended File System (EXT)

- First file system for the Linux operating system to overcome certain limitations of the **Minix file system**
- It has a maximum partition size of 2 GB and a maximum file name size of 255 characters
- It removes the two major Minix file system limitations of a **64 MB partition size** and **short file names**
- The major limitation of this file system is that it doesn't support separate access, inode modification, and data modification time stamps
- It is replaced by the **second extended file system**

The Ext file system, released in April 1992, is the first file system developed for Linux. It came as an extension of the Minix file system and to overcome some of its limitations such as 64 MB partition size and short file names. The Ext file system provides a maximum partition size of 2 GB and a maximum file name size of 255 characters. The major limitation of this file system was that it did not offer support for separate access, inode modification, and data modification timestamps. It kept an unsorted list of free blocks and inodes, and fragmented the file system.

This has a metadata structure inspired by Unix File System (UFS). Other drawbacks of this file system include only one timestamp and linked lists for free space, which resulted in fragmentation and poor performance. The second extended file system (Ext2) replaced it.



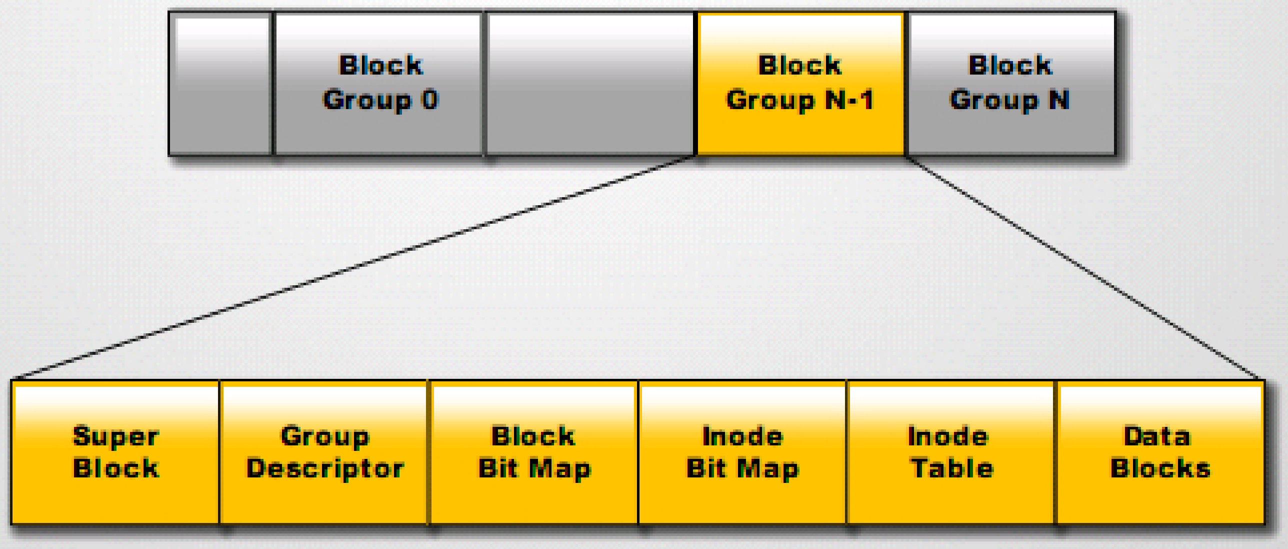
Second Extended File System (EXT2)

I. EXT2 is a standard file system that uses improved algorithms, which greatly enhances its speed significantly, and it maintains additional time stamps

II. It maintains a special field in the superblock that keeps track of the file system status and identifies it as either clean or dirty

III. Its major shortcomings are the risk of file system corruption when writing to EXT2, and it is not a journaling file system

IV. Physical layout of the EXT2 File system:



Remy Card developed the second extended file system (ext2) as an extensible and powerful file system for Linux. Being the most successful file system so far in the Linux community, Ext2 is the basis for all of the currently shipping Linux distributions.

The development of ext2 file system depends on the principle that the data storage is in the form of data blocks of the same length and. Although the length can vary between different ext2 file systems, the block size of an ext2 file system sets during its creation. The system rounds up every file size to an integral number of blocks. If the block size is 1024 bytes, then a file of 1025 bytes will occupy two 1024 byte blocks. Not all of the blocks in the file system hold data; some must contain the information that describes the structure of the file system. Ext2 defines the file system topology by describing each file in the system with an inode data structure. An inode describes the blocks, which the data within a file occupies, as well as the access rights of the file, the file modification times, and the type of the file. A single inode describes every file in the ext2 file system and each inode has a single unique number identifying it. Inode tables store all the inodes for the file system. Ext2 directories are simply special files (themselves described by inodes) that contain pointers to the inodes of their directory entries.

Superblock

This block stores information about the size and shape of the Ext2 file system. This information enables the file system manager to use and manage the file system. Generally, the system reads only the Superblock in Block Group 0 when the user mounts the file system. But every Block Group has a duplicate copy if the file system gets corrupted.

Superblock holds the following information:

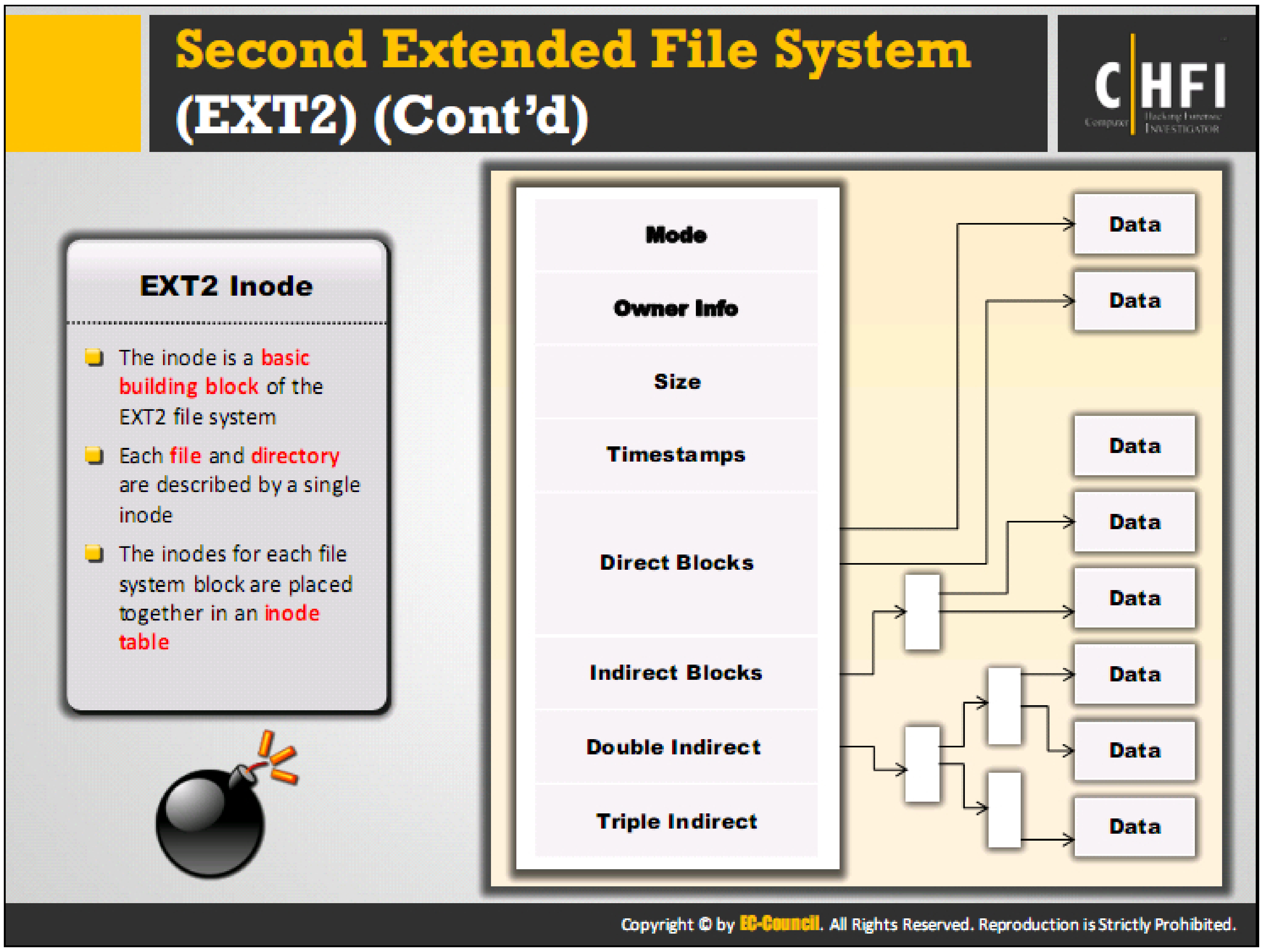
- **Magic Number**: It allows the mounting software to verify the Superblock for the EXT2 file system. For the present EXT2 version, it is 0xEF53.
- **Revision Level**: The major and minor revision levels allow the mounting code to determine whether or not this file system supports features that are only available in particular revisions of the file system. There are also feature compatibility fields that help the mounting code to determine which new features can safely be used on this file system.
- **Mount Count and Maximum Mount Count**: Together these allow the system to determine if it needs to fully check the file system. The mount count increments each time the system mounts the file system and displays the warning message of "maximal mount count reached, running e2fsck is recommended'' when it equals the maximum mount count.
- **Block Group Number**: It is the Block Group number containing the Superblock copy.
- **Block Size**: It informs about the size of the block for the file system in bytes.
- **Blocks per Group**: A fixed number that mentions the number of blocks in a group.
- **Free Blocks**: It mentions about the number of free blocks in the file system.
- **Free Inodes**: It mentions about the number of free Inodes in the file system.
- **First Inode**: It is an inode number of the first inode of the file system.

Group Descriptor

Every Group Descriptor has the following data:

- **Blocks Bitmap**: It is the block number of the block allocation bitmap for the Block Group. It is used in block allocation and deallocation.
- **Inode Bitmap**: It is the block number of the inode allocation bitmap for the Block Group. It is used in inode allocation and deallocation.
- **Inode Table**: It is the block number of the starting block for the inode table for the Block Group.
- **Free blocks count, Free Inodes count, and Used directory count**: All the group descriptors together make the group descriptor table. Every Blocks Group has the whole group descriptors table.

Source: *http://tldp.org*



Second Extended File System (EXT2) (Cont'd)

EXT2 Inode

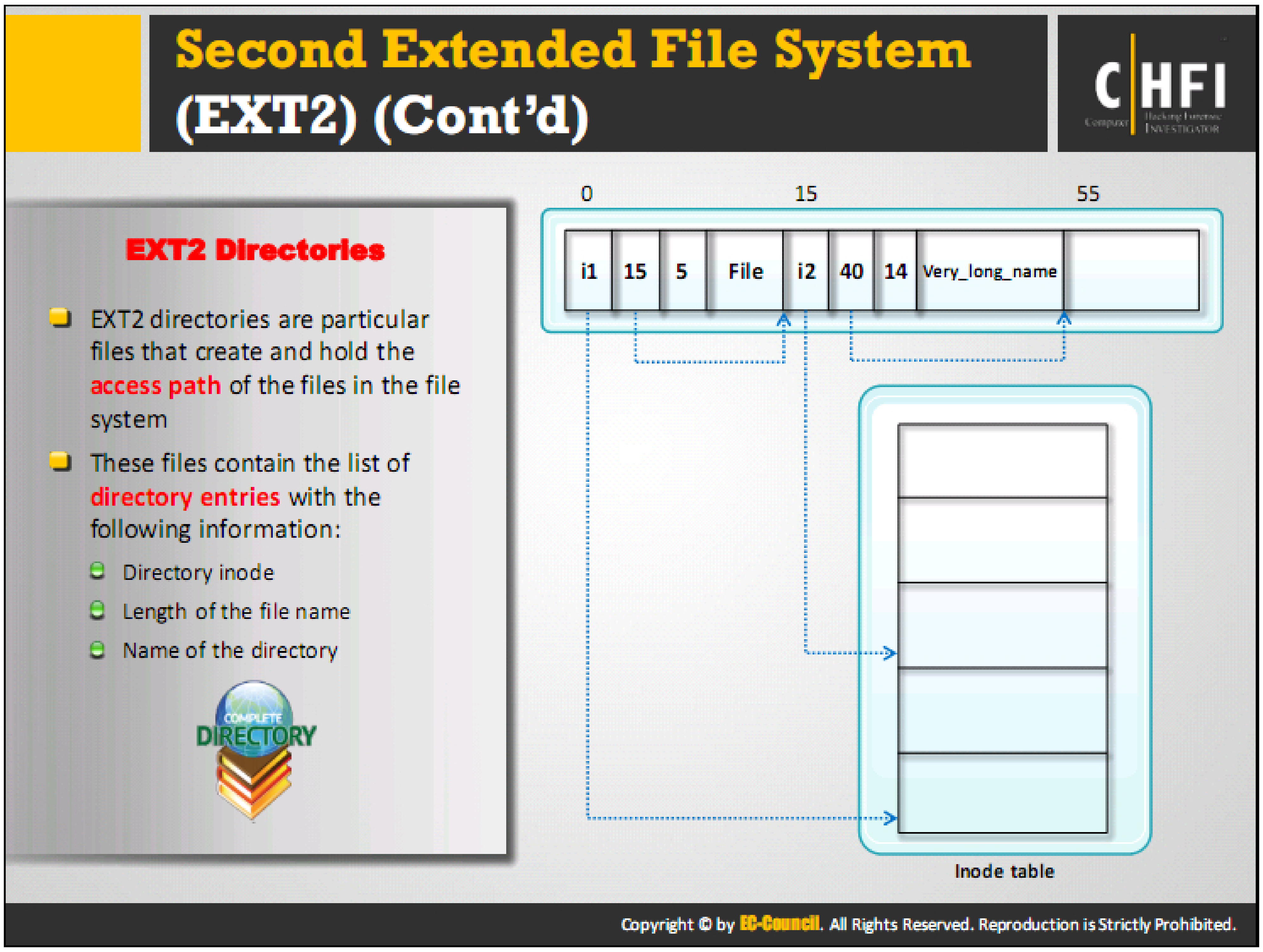
- The inode is a basic building block of the EXT2 file system
- Each file and directory are described by a single inode
- The inodes for each file system block are placed together in an inode table

Ext2 Inode

In the ext2 file system, the inode is the basic building block. One and only one inode describes every file and directory in the file system. The file system stores the ext2 inodes for each block group in the inode table together with a bitmap that allows the system to keep track of the allocated and unallocated inodes.

It contains the following fields:

- **Mode:** This holds two pieces of information: what this inode describes and the permissions that users have to it. For ext2, an inode can describe one file, directory, symbolic link, block device, character device, or FIFO.
- **Owner Information:** This is information about the users and group identifiers of the owners of a file or directory. It allows the file system to correctly allow the right sort of accesses.
- **Size:** This field holds the size of the file in bytes.
- **Timestamps:** This shows the inode creation time and the last modification time.
- **Data blocks:** Data blocks are the pointers to the blocks containing the data that this inode is describing. The first 12 are pointers to the physical blocks containing the data described by this inode, and the last 3 pointers contain more and more levels of indirection. For example, the double indirect blocks pointer points at a block of pointers to blocks of pointers to data blocks.



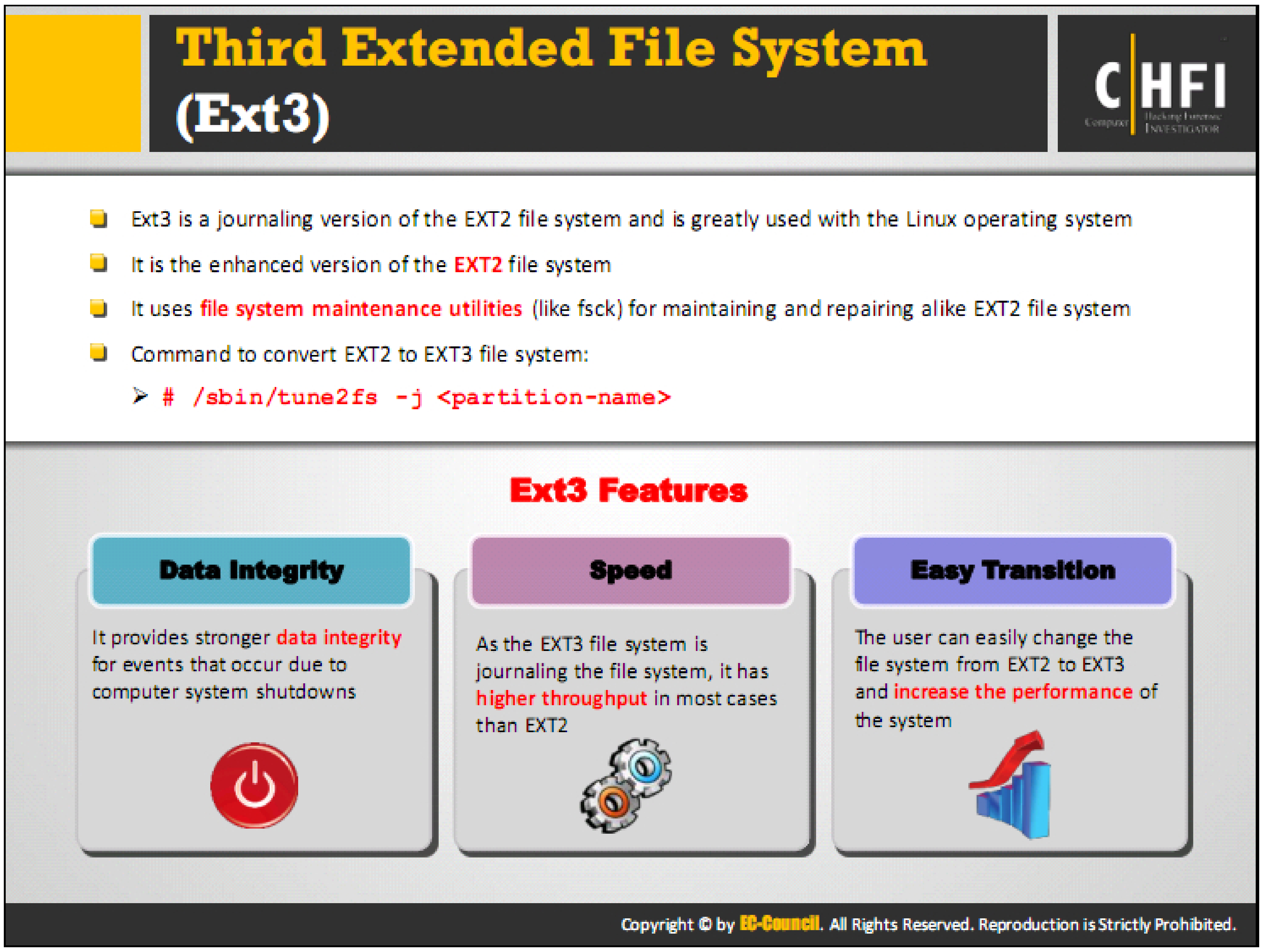
Second Extended File System (EXT2) (Cont'd)

EXT2 Directories

- EXT2 directories are particular files that create and hold the access path of the files in the file system
- These files contain the list of directory entries with the following information:
  - Directory inode
  - Length of the file name
  - Name of the directory

In the ext2 file system, directories are special files used to create and hold access paths to the files in the file system. A directory file is a list of directory entries, each one containing the following information:

- **Inode:** The inode for this directory entry. This is an index into the array of inodes held in the inode table of the block group.
- **Name length:** The length of this directory entry in bytes.
- **Name:** The name of this directory entry.



Third Extended File System (Ext3)

- Ext3 is a journaling version of the EXT2 file system and is greatly used with the Linux operating system
- It is the enhanced version of the EXT2 file system
- It uses file system maintenance utilities (like fsck) for maintaining and repairing alike EXT2 file system
- Command to convert EXT2 to EXT3 file system:
  - # /sbin/tune2fs -j <partition-name>

Ext3 Features

Data Integrity: It provides stronger data integrity for events that occur due to computer system shutdowns

Speed: As the EXT3 file system is journaling the file system, it has higher throughput in most cases than EXT2

Easy Transition: The user can easily change the file system from EXT2 to EXT3 and increase the performance of the system

Developed by Stephen Tweedie in the year 2001, the third extended file system (ext3) is a journaling file system used in the GNU/Linux operating system. It is the enhanced version of the ext2 file system. The main advantage of this file system is journaling that improves reliability of the computer system. It can be mounted and used as an ext2 file system. It can make use of all of the previous programs developed in the ext2 file system.

A maximum single Ext3 file size ranges between 16 GB to 2 TB, and the whole ext3 file system size ranges between 2 TB to 32 TB. Ext3 also offers a better data integrity. It makes sure that the data is consistent with the file system state. Ext3 is faster than ext2 because the journaling feature optimizes hard disk drive (HDD) head motion. It also gives a choice of three journaling modes, which provide trade-offs between maximizing data integrity and optimizing speed.

Ext3 is also highly reliable. Ext3 also has the ability to convert ext2 partitions to ext3 and vice-versa without the need for repartitioning and data backup.

Command to convert ext2 to ext3 file system:

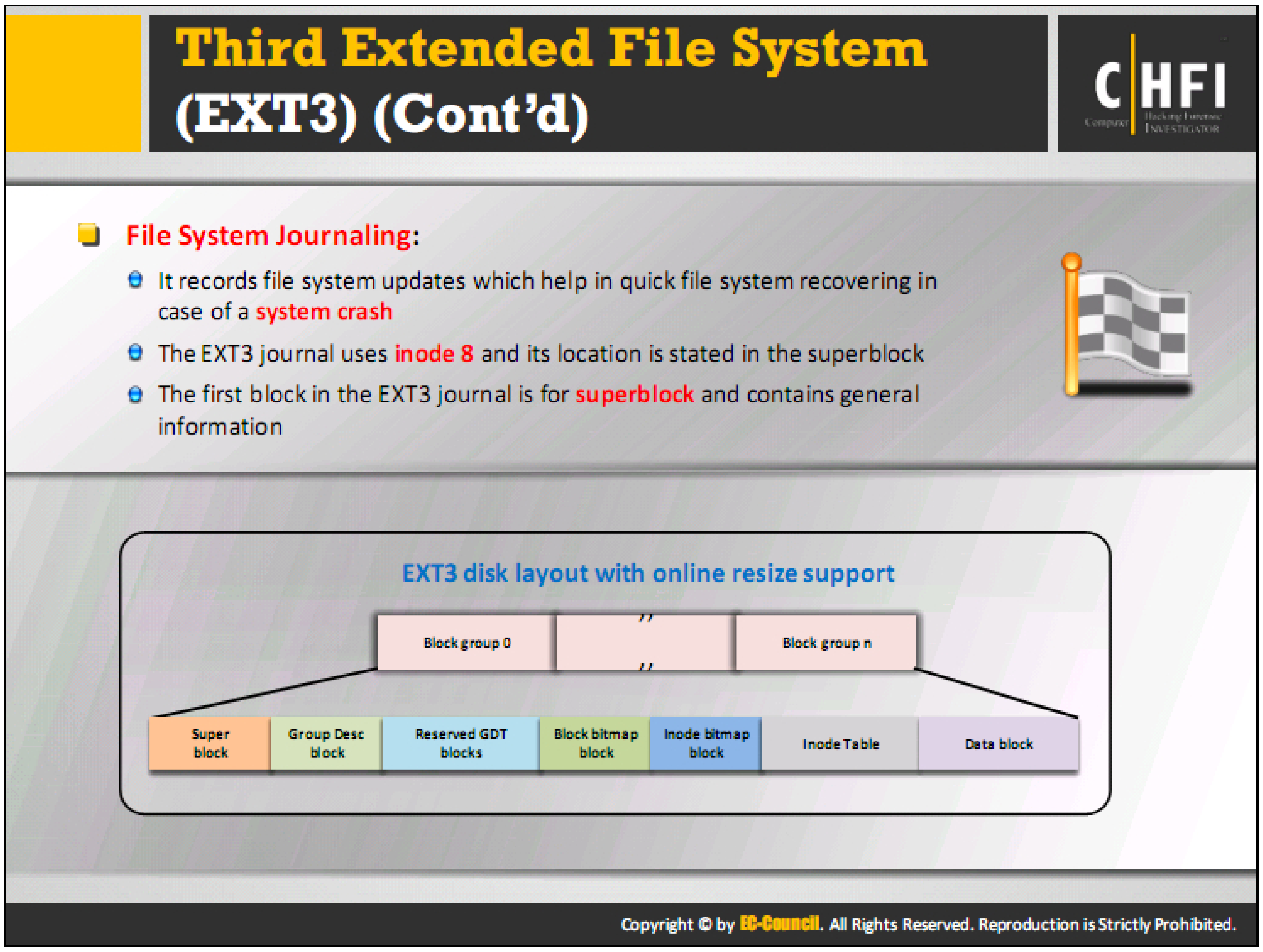
```
# /sbin/tune2fs -j <partition-name>
```

For example, if the user needs to convert an ext2 file system located on the partition /dev/hda5 to an ext3 file system, he or she should use the following command:

```
# /sbin/tune2fs -j /dev/hda5
```

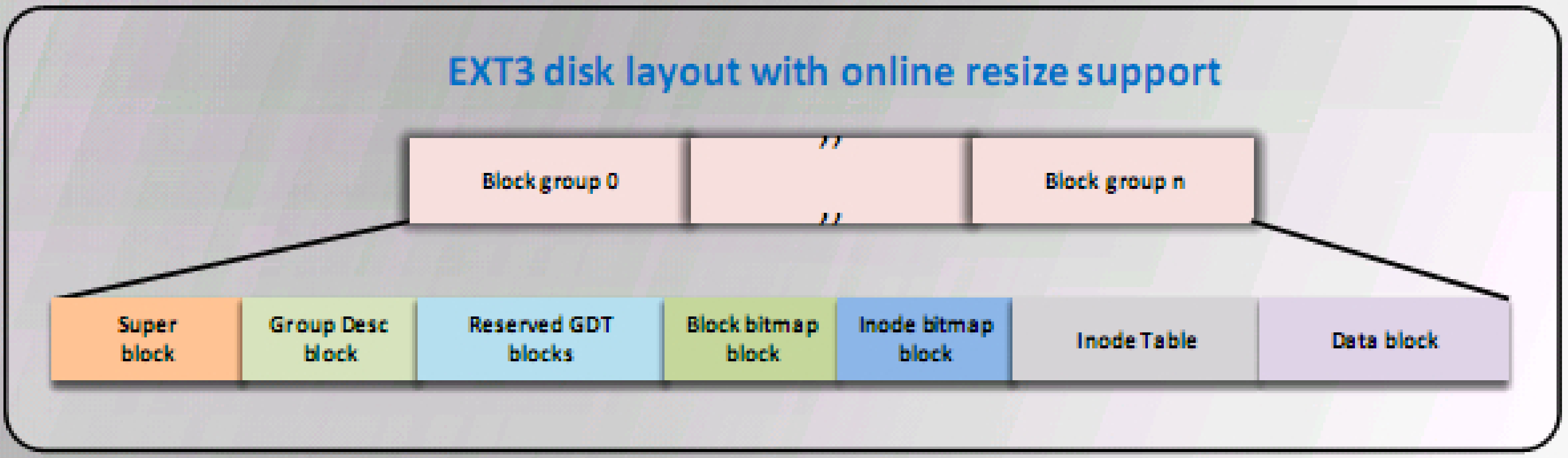
Features of Ext3

- **Data integrity:** It provides stronger data integrity for events that occur due to computer system shutdowns. It allows the user to choose the type and level of protection for the received data.
- **Speed:** As the ext3 file system is journaling the file system, it has higher throughput in most cases than ext2. The user can choose the optimized speed from three different journaling modes.
- **Easy transition:** The user can easily change the file system from ext2 to ext3 and increase the performance of the system by using the journaling file system without reformatting.



Third Extended File System (EXT3) (Cont'd)

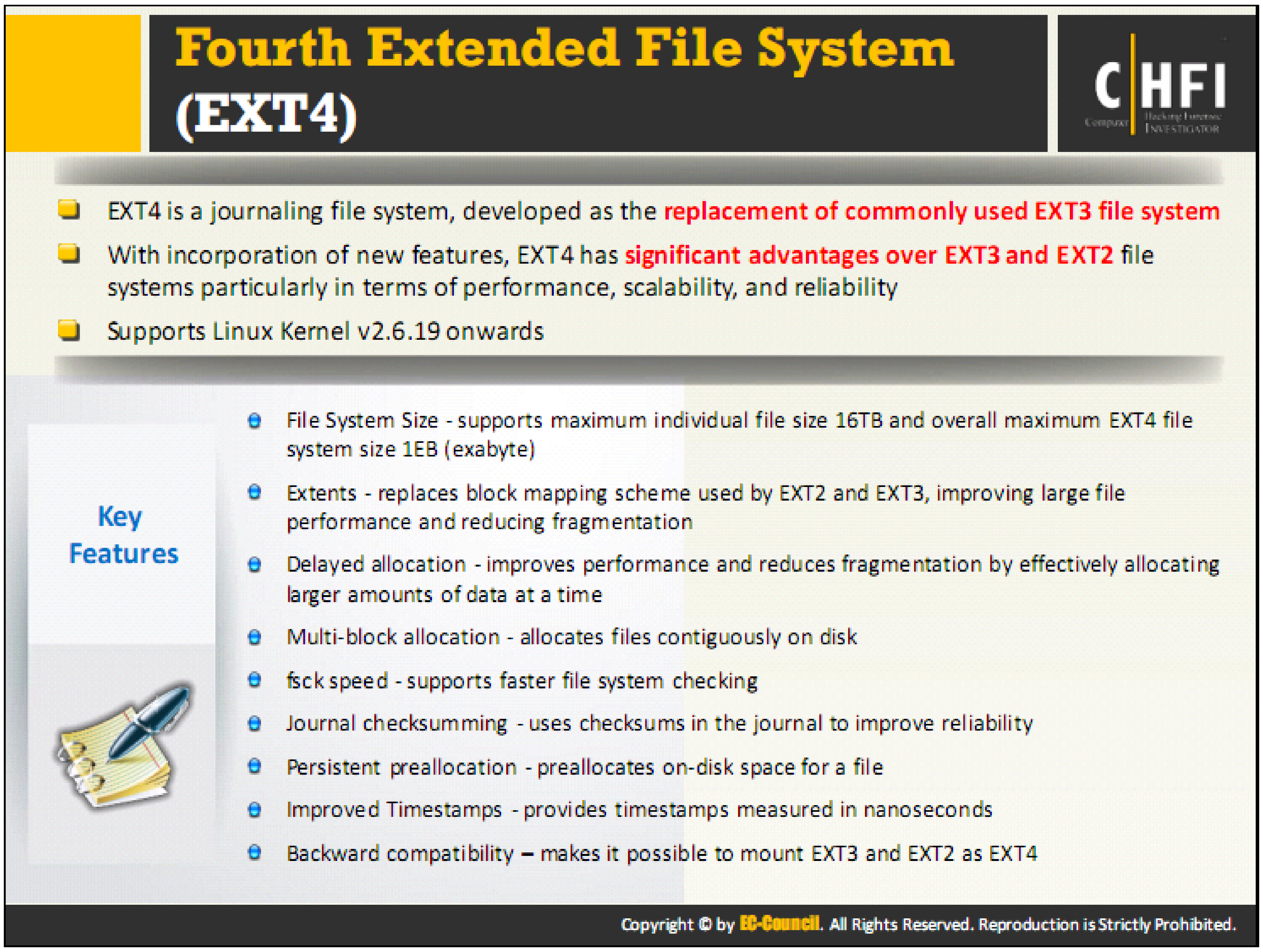
- **File System Journaling**:
  - It records file system updates which help in quick file system recovering in case of a **system crash**
  - The EXT3 journal uses **inode 8** and its location is stated in the superblock
  - The first block in the EXT3 journal is for **superblock** and contains general information



File System Journaling

File system journaling works as follows:

- It records updates to the file system, which helps in recovering the file system quickly in case of a system crash
- The ext3 journal uses inode 8, and the super block contains its location
- The first block in the ext3 journal is for the super block and contains general information



Fourth Extended File System (EXT4)

- EXT4 is a journaling file system, developed as the **replacement of commonly used EXT3 file system**
- With incorporation of new features, EXT4 has **significant advantages over EXT3 and EXT2** file systems particularly in terms of performance, scalability, and reliability
- Supports Linux Kernel v2.6.19 onwards

Key Features

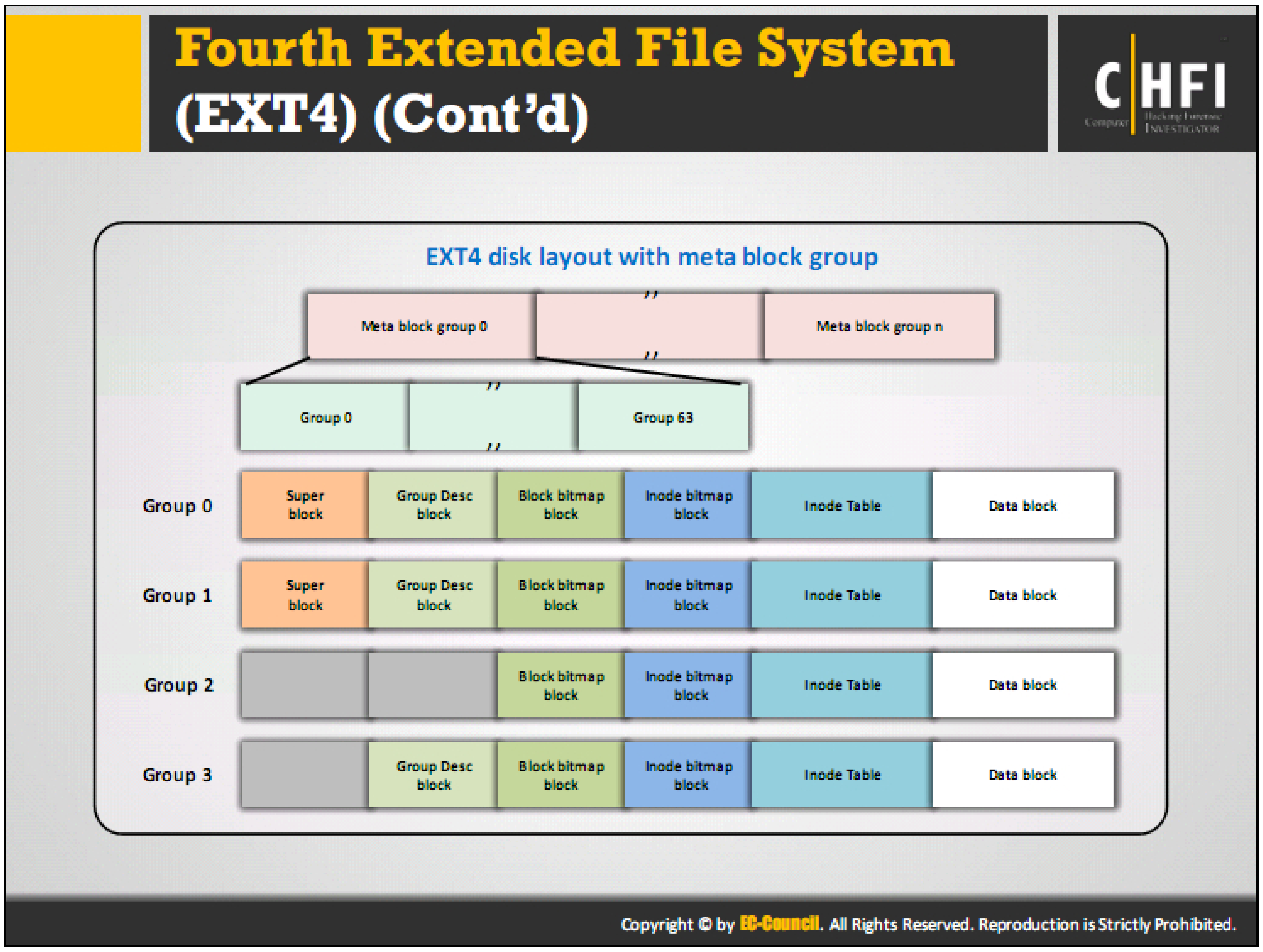
- File System Size - supports maximum individual file size 16TB and overall maximum EXT4 file system size 1EB (exabyte)
- Extents - replaces block mapping scheme used by EXT2 and EXT3, improving large file performance and reducing fragmentation
- Delayed allocation - improves performance and reduces fragmentation by effectively allocating larger amounts of data at a time
- Multi-block allocation - allocates files contiguously on disk
- fsck speed - supports faster file system checking
- Journal checksumming - uses checksums in the journal to improve reliability
- Persistent preallocation - preallocates on-disk space for a file
- Improved Timestamps - provides timestamps measured in nanoseconds
- Backward compatibility – makes it possible to mount EXT3 and EXT2 as EXT4

Ext4 is a journaling file system, developed as the replacement of commonly used ext3 file system, offers better scalability and reliability for supporting large file systems of 64 bit machines in order to meet the increasing disk capacity demands. Ext4 enables write barriers by default and allows users to mount an ext3 file system as an ext4 file system. The file system supports Linux Kernel v2.6.19 onwards.

Key Features

- File System Size: Supports maximum individual file sizes up to 16 TB and overall volumes of about size 1 EiB (exbibyte).
- Extents: Replaces block mapping scheme found in ext2 and ext3 in order to increase performance and reduces fragmentation.
- Delayed allocation: Improves performance and reduces fragmentation by effectively allocating larger amounts of data at a time by delaying allocation till the system flushes data to the disk.
- Multiblock allocation: Allocates multiple files contiguously on disk, thus reducing the work of calling the block allocator and optimizing the allocation of memory.
- Increased file system checking (fsck) speed: Marks unallocated block groups and sections and skips the marked elements while performing checks. Thus, supports faster file system checking.
- Journal check summing: Uses checksums in the journal to improve reliability.

- Persistent pre-allocation: The file system can pre-allocate the on-disk space for a file, by writing zeroes to it during creation.
- Improved Timestamps: Provides timestamps measured in nanoseconds and comes with support for date-created timestamps.
- Backward compatibility: The file system is backward compatible and allows the user to mount ext3 and ext2 as ext4.



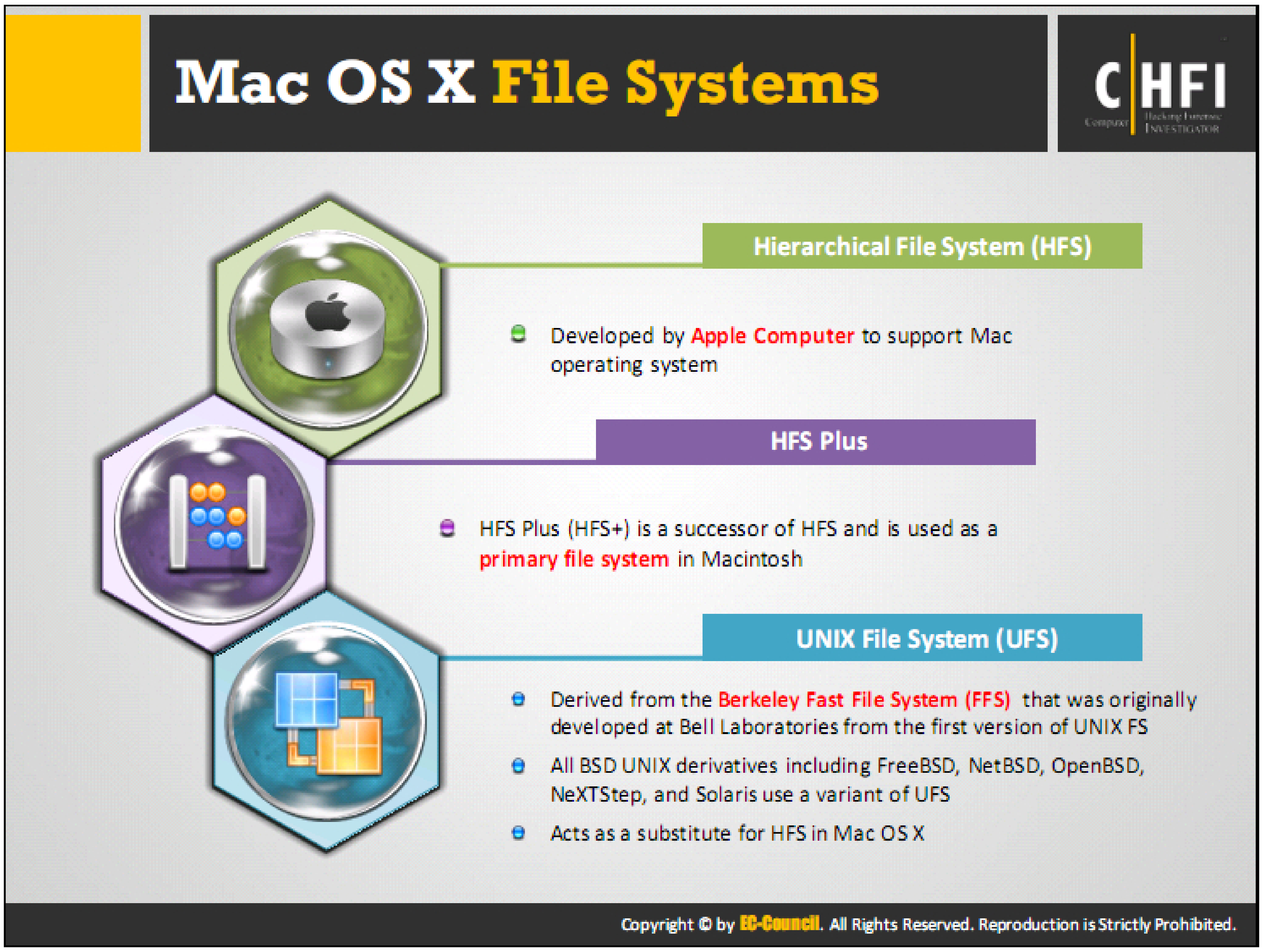
Fourth Extended File System (EXT4) (Cont'd)

EXT4 disk layout with meta block group

CHFI
Computer Hacking Forensic INVESTIGATOR

Mac OS X File Systems

Apple's Mac OS X uses a different approach in storing the data, when compared to the Windows and Linux. This section will make investigators aware of the file systems that different versions of Mac operating systems use.



Mac OS X File Systems

Hierarchical File System (HFS)

- Developed by Apple Computer to support Mac operating system

HFS Plus

- HFS Plus (HFS+) is a successor of HFS and is used as a primary file system in Macintosh

UNIX File System (UFS)

- Derived from the Berkeley Fast File System (FFS) that was originally developed at Bell Laboratories from the first version of UNIX FS
- All BSD UNIX derivatives including FreeBSD, NetBSD, OpenBSD, NeXTStep, and Solaris use a variant of UFS
- Acts as a substitute for HFS in Mac OS X

Hierarchical File System (HFS)

Apple had developed the Hierarchical File System (HFS) in September 1985 to support the MAC OS in its proprietary Macintosh system and as a replacement for the Macintosh File System (MFS). HFS divides the volume into logical blocks of 512 bytes each and groups these logical blocks into allocation blocks. Each allocation block can store one or more logical blocks depending on the total size of the volume. The file system uses a 16-bit value to address allocation blocks, which restricts the number of allocation blocks to 65,535.

Five structures make up an HFS volume:

Logical blocks 0 and 1 of the volume are the boot blocks, which include system startup information, for example, the names of the system and shell files, which are loaded at startup.

Logical block 2 contains the Master Directory Block (MDB). This defines a wide variety of data about the volume itself, for instance, date and time stamps of creation of the volume; the location of the other volume structures, such as the volume bitmap' or the size of logical structures, such as allocation blocks. There is also a duplicate of the MDB called the Alternate Master Directory Block (Alternate MDB) located at the opposite end of the volume in the second to last logical block. This is intended mainly for use by disk utilities and is only updated when either the catalog file or extents overflow file grow in size.

Logical block 3 is the starting block of the volume bitmap, which keeps track of the allocation blocks in use and those which are free. A bit in the map represents each allocation block on the volume; if the bit is set then the block is in use, else if it is clear then the block is free.

The extents overflow file is a B*-tree including extra extents that store information about the files and the allocation blocks allocated to them, after the system uses the initial three extents in the catalog file. Later versions also added the ability for the extents overflow file to store extents that record bad blocks, to prevent a machine from trying to write to them.

The catalog file is another B*-tree that holds records for all the files and directories stored in the volume. It stores four types of records. Each file consists of a file thread record and a file record, while each directory contains a directory thread record and a directory record. Unique catalog node ID helps in finding the files and directories in the catalog file.

HFS Plus (HFS+)

HFS Plus (HFS+) is a successor of HFS and is a primary file system in Macintosh.

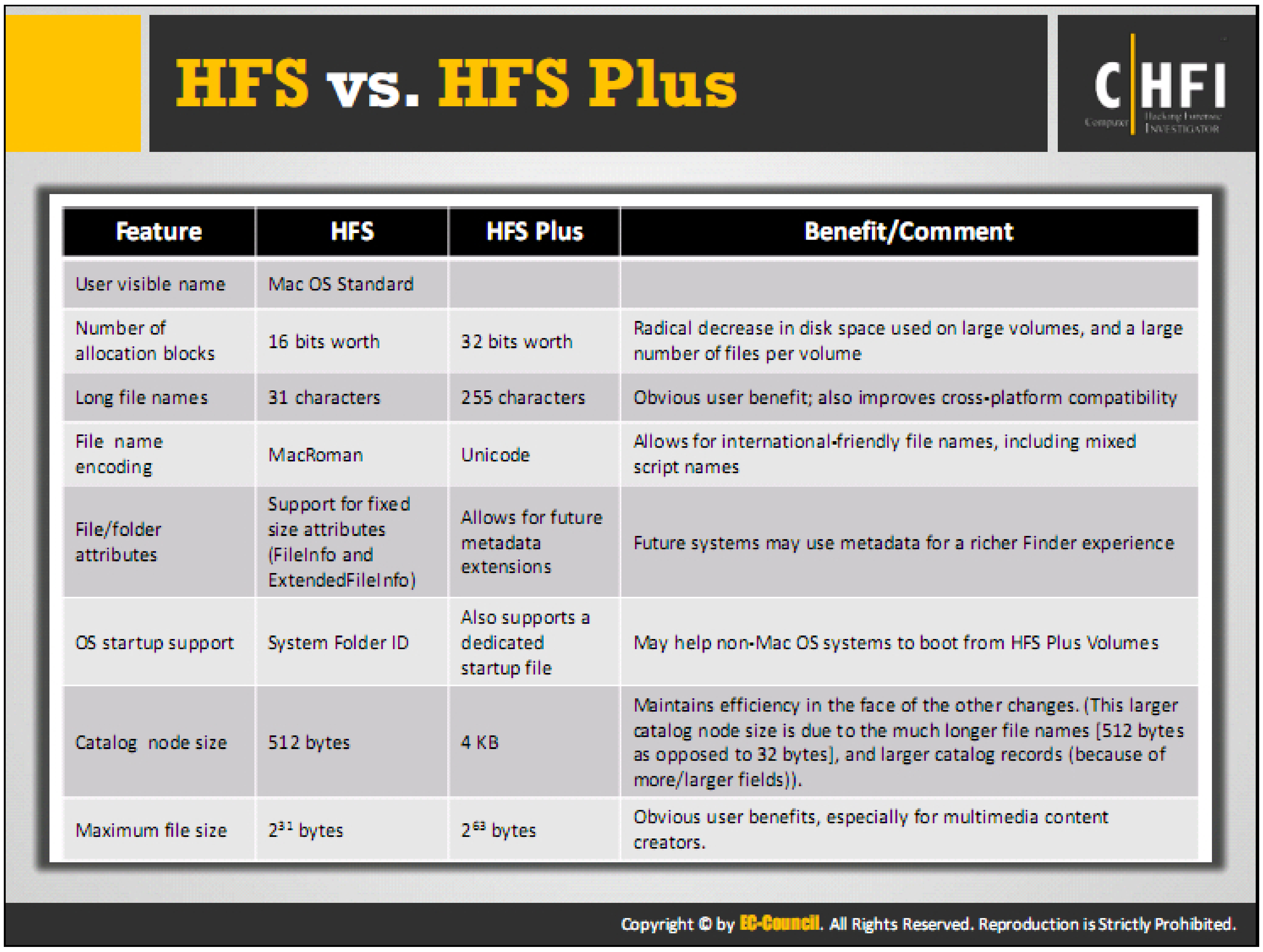
UFS (UNIX File System)

UFS is a file system utilized by many UNIX and UNIX-like operating systems. Derived from the Berkeley Fast File System, it is the first version of UNIX developed at Bell Labs.

Design:

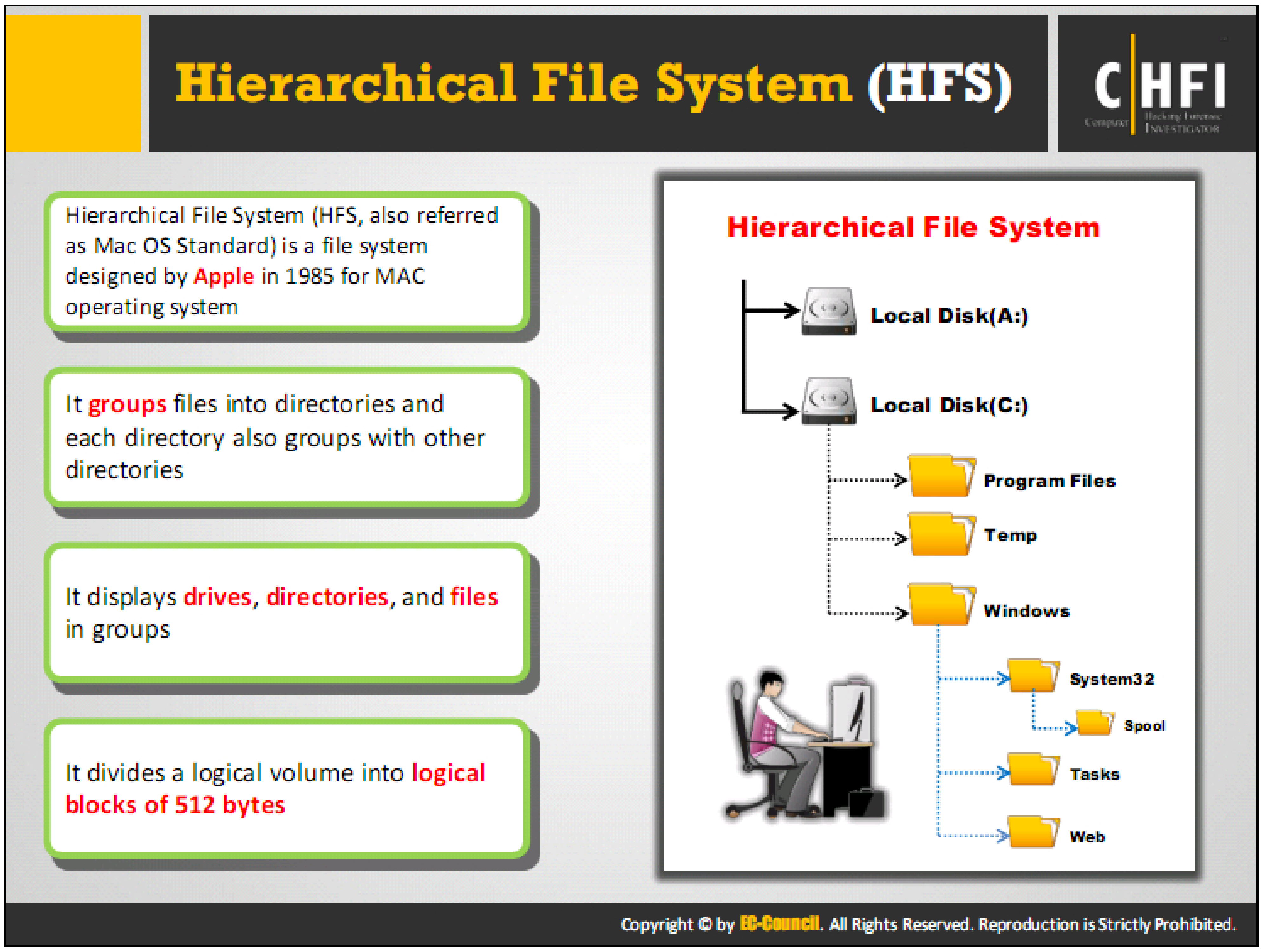
A UFS file system is composed of the following parts:

- A few blocks at the beginning of the partition reserved for boot blocks (which must be initialized separately from the file system)
- A super block, including a magic number identifying this as a UFS file system, and some other vital numbers describing this file system's geometry and statistics and behavioral tuning parameters
- A collection of cylinder groups, of which each cylinder group has the following components:
    - A backup copy of the super block
    - A cylinder group header, with statistics, free lists, etc., and which is similar to those in the super block
    - A number of inodes, each containing file attributes
    - A number of data blocks



HFS vs. HFS Plus

Feature	HFS	HFS Plus	Benefit/Comment
User visible name	Mac OS Standard		
Number of allocation blocks	16 bits worth	32 bits worth	Radical decrease in disk space used on large volumes, and a large number of files per volume
Long file names	31 characters	255 characters	Obvious user benefit; also improves cross-platform compatibility
File name encoding	MacRoman	Unicode	Allows for international-friendly file names, including mixed script names
File/folder attributes	Support for fixed size attributes (FileInfo and ExtendedFileInfo)	Allows for future metadata extensions	Future systems may use metadata for a richer Finder experience
OS startup support	System Folder ID	Also supports a dedicated startup file	May help non-Mac OS systems to boot from HFS Plus Volumes
Catalog node size	512 bytes	4 KB	Maintains efficiency in the face of the other changes. (This larger catalog node size is due to the much longer file names [512 bytes as opposed to 32 bytes], and larger catalog records (because of more/larger fields)).
Maximum file size	2^{31} bytes	2^{63} bytes	Obvious user benefits, especially for multimedia content creators.



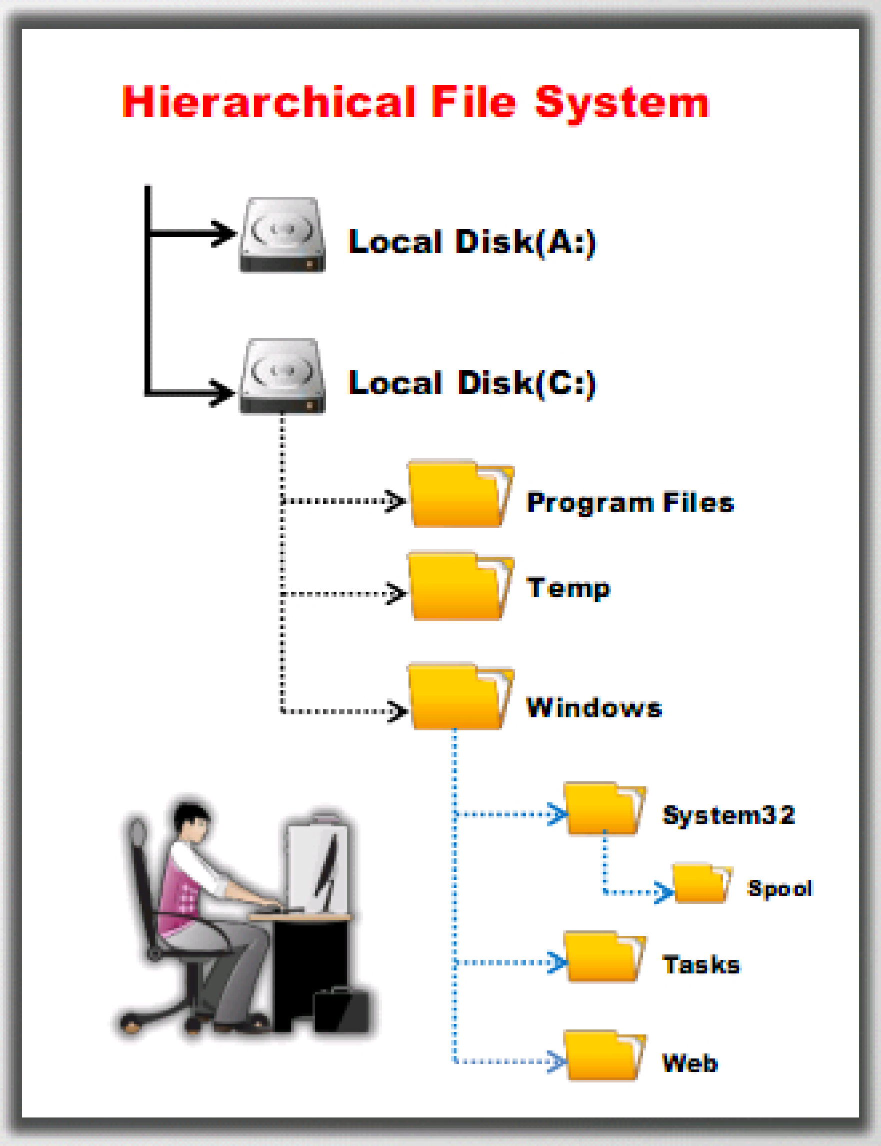
Hierarchical File System (HFS)

Hierarchical File System (HFS, also referred as Mac OS Standard) is a file system designed by **Apple** in 1985 for MAC operating system

It **groups** files into directories and each directory also groups with other directories

It displays **drives**, **directories**, and **files** in groups

It divides a logical volume into **logical blocks of 512 bytes**



Apple designed the Hierarchical File System (HFS) in 1985 for the Mac operating system. It allows the user to store files in a hierarchical manner. It groups files into directories and each directory also groups with other directories. It displays drives, directories, and files in groups. It divides a logical volume into logical blocks of 512 bytes. The file system then groups these blocks into an allocation block. HFS uses a 16-bit value to address allocation blocks.

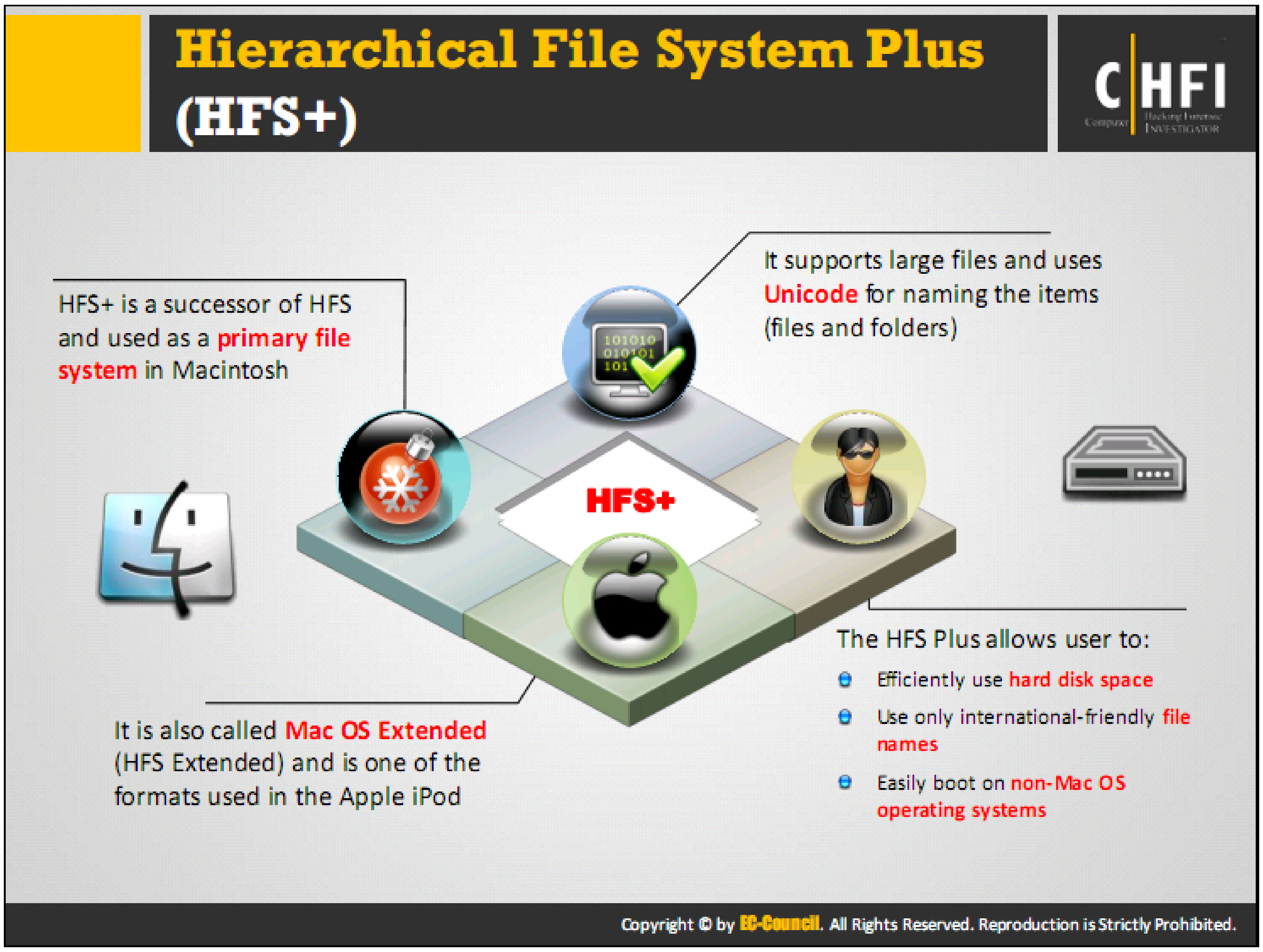
Logical volume blocks 0 and 1 are the boot blocks, which contain system startup information. For example, system names and shell files loaded at startup.

The Logical block 2 has the Master Directory Block (MDB) which defines data about the volume such as date and time stamps of volume creation, location of volume structures, volume bitmap and the size of logical structures. Alternate Master Directory Block (Alternate MDB) is a duplicate of the MDB located in the opposite end of the volume and second to last logical block. Disk utilities primarily use the alternate Master Directory Block and update it only when the extents overflow file or catalog file are large in size.

The Logical block 3 is the first block of the volume bitmap. It keeps track of the allocation blocks in use as well as the blocks that are free.

The Hierarchical File System allows the user to:

- Hierarchically organize files
- Use longer file names that include embedded spaces
- Use as many levels of directories and subdirectories as needed
- Use a CICS transaction to observe and maintain HFS files



Hierarchical File System Plus (HFS+)

HFS+ is a successor of HFS and used as a primary file system in Macintosh

It supports large files and uses Unicode for naming the items (files and folders)

It is also called Mac OS Extended (HFS Extended) and is one of the formats used in the Apple iPod

The HFS Plus allows user to:

- Efficiently use hard disk space
- Use only international-friendly file names
- Easily boot on non-Mac OS operating systems

HFS Plus (HFS+) is a successor of HFS and is a primary file system in Macintosh. It supports large files and uses Unicode for naming the items (files and folders).

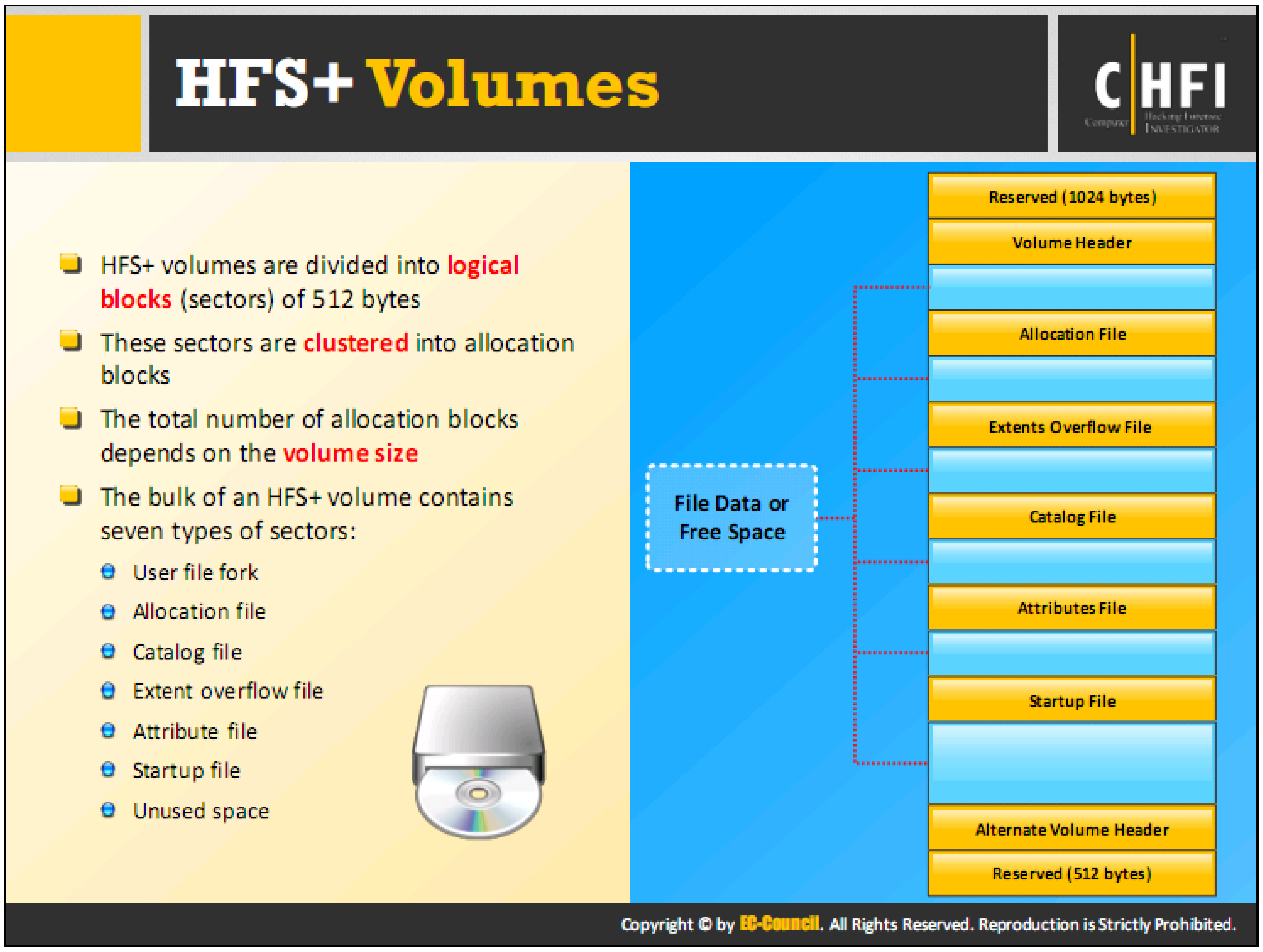
A few of the features added to HFS Plus are:

- HFS Plus uses B-tree to store the data
- It supports files with 64 bits in length
- It permits file names of 255 characters in length
- It uses a 32-bit allocation table for the mapping table, unlike 16 bits in HFS

HFS Plus allows the user to:

- Efficiently use hard disk space.
- Use only international-friendly file names.
- Easily boot on non-Mac OS operating systems.

Also called as Mac OS Extended (HFS Extended), it is the file system of some Apple iPods as well.



HFS+ Volumes

- HFS+ volumes are divided into **logical blocks** (sectors) of 512 bytes
- These sectors are **clustered** into allocation blocks
- The total number of allocation blocks depends on the **volume size**
- The bulk of an HFS+ volume contains seven types of sectors:
  - User file fork
  - Allocation file
  - Catalog file
  - Extent overflow file
  - Attribute file
  - Startup file
  - Unused space

Reserved (1024 bytes)
Volume Header
Allocation File
Extents Overflow File
Catalog File
Attributes File
Startup File
Alternate Volume Header
Reserved (512 bytes)
File Data or Free Space

HFS+ volumes have further divisions called logical blocks (sectors) of 512 bytes size. These sectors cluster into allocation blocks. The total number of allocation blocks depends on the volume size.

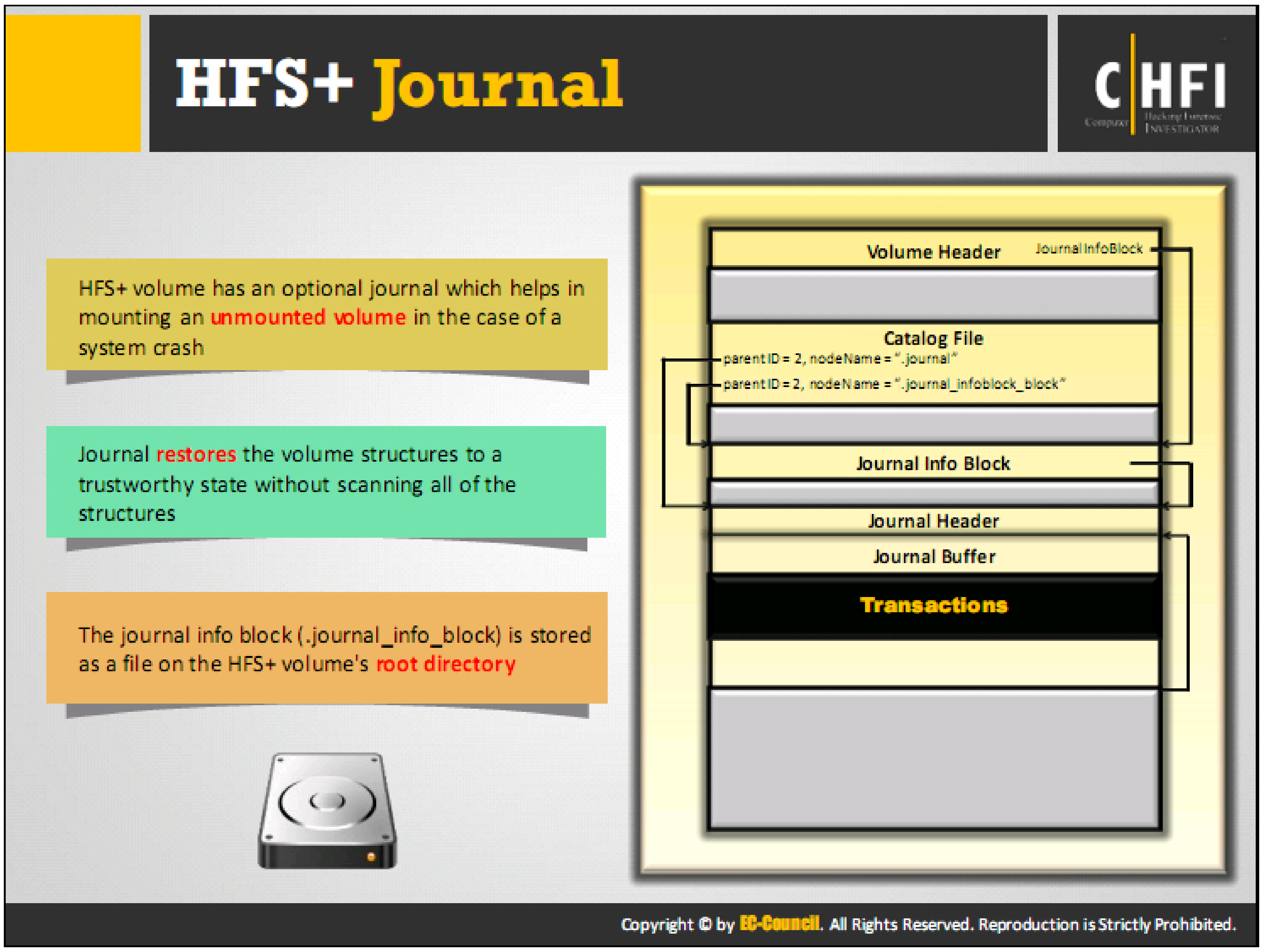
The bulk of an HFS+ volume consists of seven types of sectors:

- User file fork
- Allocation file
- Catalog file
- Extents overflow file
- Attribute file
- Startup file
- Unused space

An HFS+ volume is made of 9 main structures:

- **Boot Blocks:** Present in sectors 0 and 1 and are similar to the boot blocks found in HFS.
- **Volume Header:** Found in sector 2 and stores data about the volume such as timestamps, allocation blocks sizes, and locations volume structures like Extent Overflow File or Catalog File.

- **Allocation File:** This file maintains records of the blocks that are free as well as blocks in use and denoted by one bit. It is a regular file and does not have a reserved space at the beginning of the volume. Its size is variable and does not need contiguous storage within a volume.
- **Catalog File:** This is a B-tree which contains records for all files and directories stored in the volume. The HFS+ offers bigger records to provide more fields and also allows larger fields. The record fields in HFS+ can also vary based on the data stored by them.
- **Extents Overflow File:** This is a B-tree which records the allocation blocks allocated to each file as extents. The Catalog Files can record up to eight extents for each fork of a file and store the extra extents in the Extents Overflow File. They record even bad blocks as extents. The default file sizes are 1 KB and 4 KB in Mac OS and Mac OS X, respectively.
- **Attributes File:** This is a B-tree only present in HFS+ and can store three kinds of 4 KB records including Fork Data Attribute records, Inline Data Attribute records, and Extension Attribute records.
- **Startup File:** Developed for non-Mac OS systems, which do not have HFS or HFS+ support, the file is equivalent to the HFS volume Boot Blocks.
- **Alternate Volume Header:** It is present in the last but one sector of an HFS+ volume and is similar to HFS Alternate Master Directory Block.
- **Last sector:** Reserved for Apple’s use at the time of the computer manufacturing.

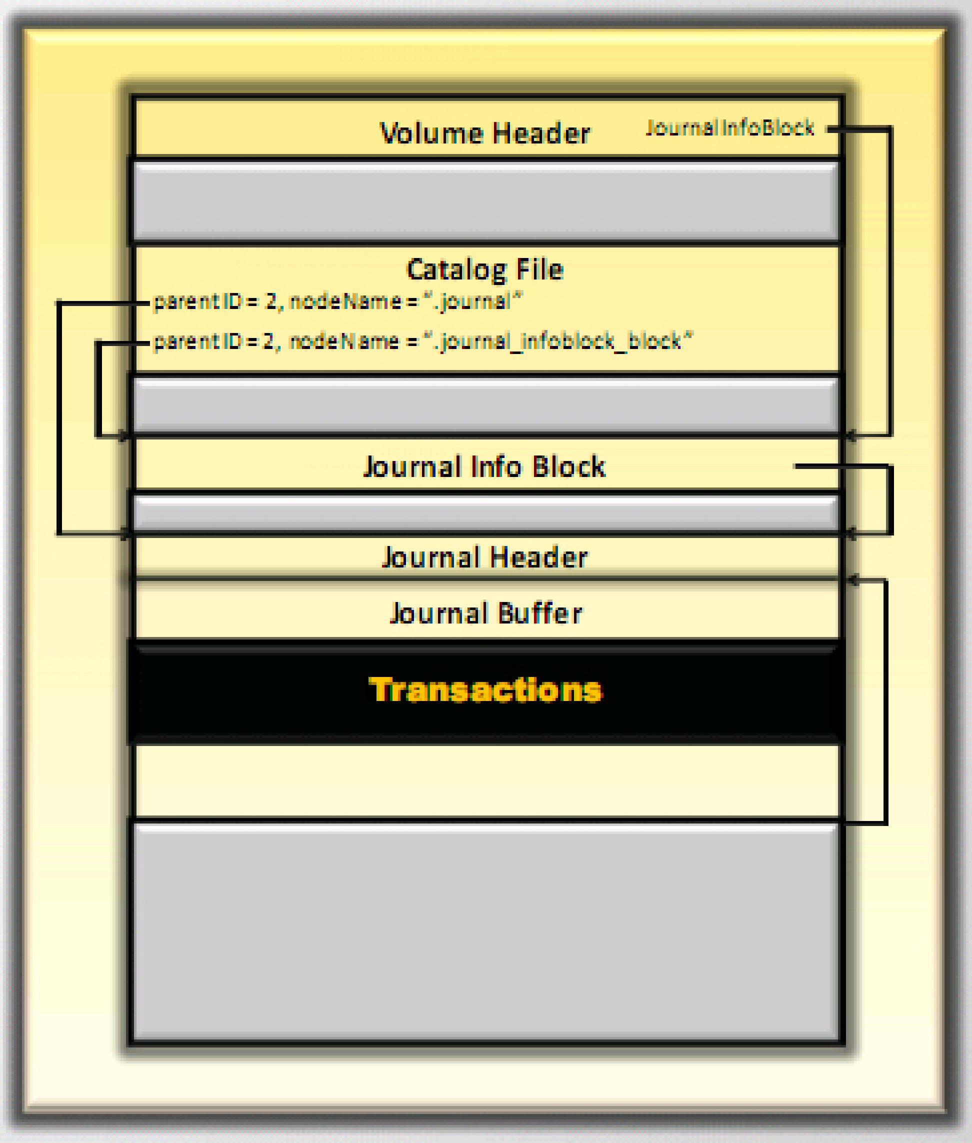


HFS+ Journal

HFS+ volume has an optional journal which helps in mounting an unmounted volume in the case of a system crash

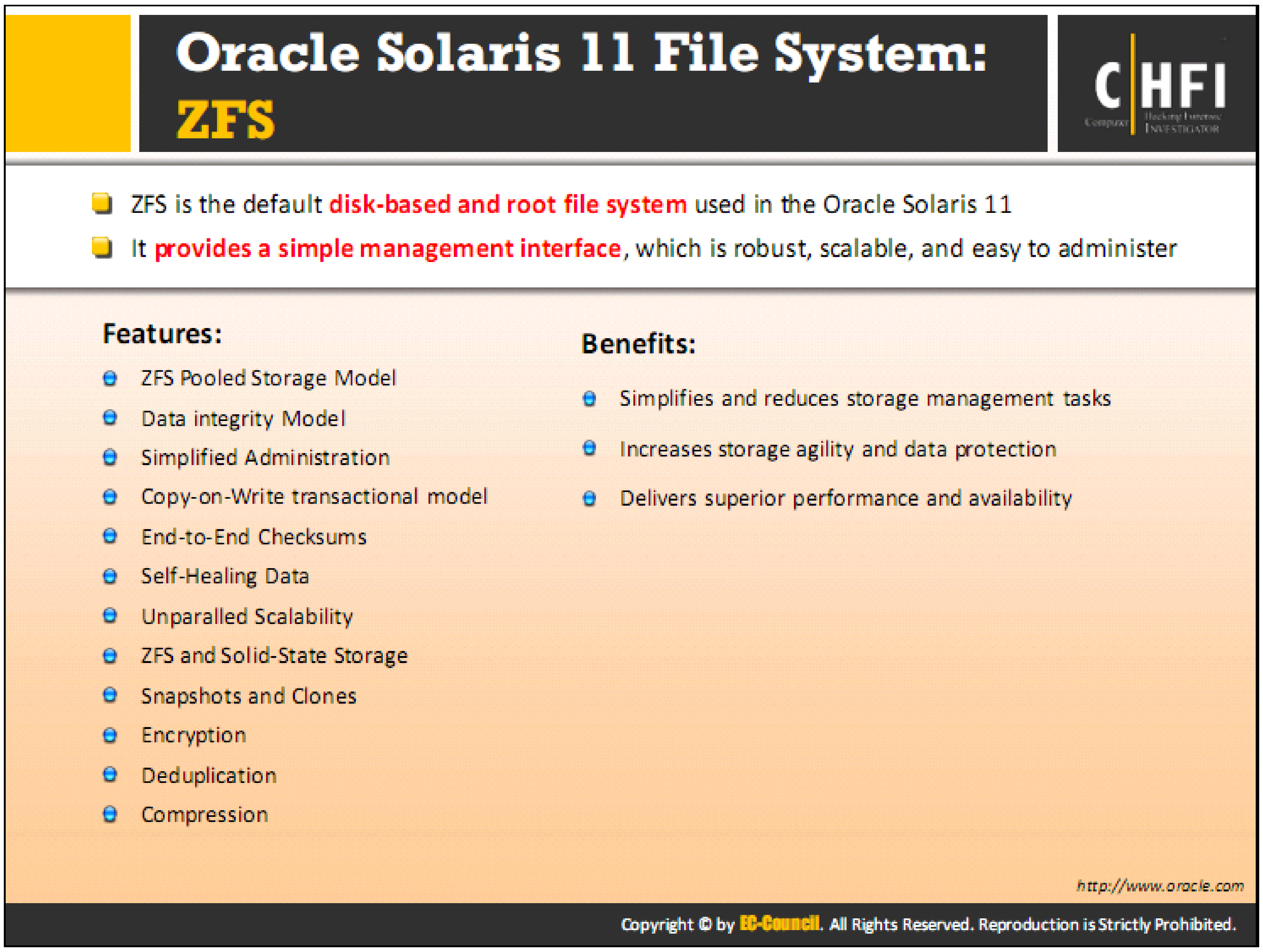
Journal restores the volume structures to a trustworthy state without scanning all of the structures

The journal info block (.journal_info_block) is stored as a file on the HFS+ volume's root directory



HFS+ volumes have an optional journal, which helps in mounting an unmounted volume in the case of a system crash. The journal restores the volume structures to a reliable state without scanning all of the structures. The journal info block (.journal_info_block) is stored as a file on the HFS+ volume's root directory.

Apple added journaling features to HFS Plus with the release of Mac OS X 10.2.2 update in 2002 to improved data reliability. All HFS Plus volumes on all Macs are set with the journaling feature as default from OS X v10.3 version. The HFS Plus volume with a journal is denoted as HFSJ.



Oracle Solaris 11 File System: ZFS

- ZFS is the default **disk-based and root file system** used in the Oracle Solaris 11
- It **provides a simple management interface**, which is robust, scalable, and easy to administer

Features:

- ZFS Pooled Storage Model
- Data integrity Model
- Simplified Administration
- Copy-on-Write transactional model
- End-to-End Checksums
- Self-Healing Data
- Unparalled Scalability
- ZFS and Solid-State Storage
- Snapshots and Clones
- Encryption
- Deduplication
- Compression

Benefits:

- Simplifies and reduces storage management tasks
- Increases storage agility and data protection
- Delivers superior performance and availability

http://www.oracle.com

ZFS is a file system as well as a logical volume manager developed by Sun Microsystems. The ZFS offers features like high storage capability, data protection, data compression, volume management, copy-on-write clones, data integrity checking, and automatic repair.

Features

- **ZFS Pooled Storage Model**

  This file system uses the storage pool model, which defines the storage characteristics and acts as a random data store from the point of creation of file systems. The storage allows the ZFS file systems to share disk space with all other file systems. The users do not have to know the file system size, because the file systems are present within the disk space allotted to the storage pool. On addition of a new storage, all file systems in the pool will be able to use this additional disk space.

- **Data integrity Model**

  It protects disk data from data damage caused by current spikes, data degradation, phantom writes, bugs in disk firmware, DMA parity errors between the array, and server memory or from the driver, driver errors, etc.

- **Simplified Administration**

  ZFS provides a simple administration model. It simplifies creation and handling of file systems by using the property inheritance, hierarchical file system layout, NFS share semantics, and automatic management of mount points. These features allow it to manage the file systems without multiple commands or editing configuration files.

- **Copy-on-Write transactional model**

  The file system implements a copy-on-write transactional object model. All the block pointers in the file system have a 256-bit checksum or 256-bit hash of the target block. It does not overwrite the blocks containing active data but allocates a new block and writes the modified data to it. Then it reads, reallocates, and writes any metadata blocks referencing to it in the same manner. In order to reduce the overhead of this process, the file system gathers several updates into transaction groups and uses a ZIL (intent log) write cache when it needs synchronous write semantics.

- **End-to-End Checksums and Self-Healing Data**

  ZFS verifies all data and metadata using a user-selectable checksum algorithm. The traditional file systems used per-block checksum algorithm leading to data errors. ZFS performs checksums at the file system layer making them transparent to the applications and to minimize the shortcomings.

  ZFS provides self-healing data and supports storage pools with different levels of data redundancy. When ZFS detects a bad data block, it gets similar data from a redundant copy, repairs the bad data and replaces it.

- **Un-paralleled Scalability**

  ZFS is 128-bit file system, which allows 256 quadrillion zettabytes of storage. It allocates all the metadata dynamically and does not require pre-allocation of inodes or limit the scalability of the file system during creation. The directories can contain up to 256 trillion entries with no limits on the number of files in a file system.

- **Snapshots and Clones**

  ZFS includes the copy-on-write capability, which allows it to take space-efficient snapshots that acquire less storage and require very less capture. The file system tracks only the changes made to the data.

  It can also capture writeable snapshots or clones that result in two independent file systems, which share a set of blocks. When these clone file systems observe any changes, the file system creates new data blocks to reflect those changes.

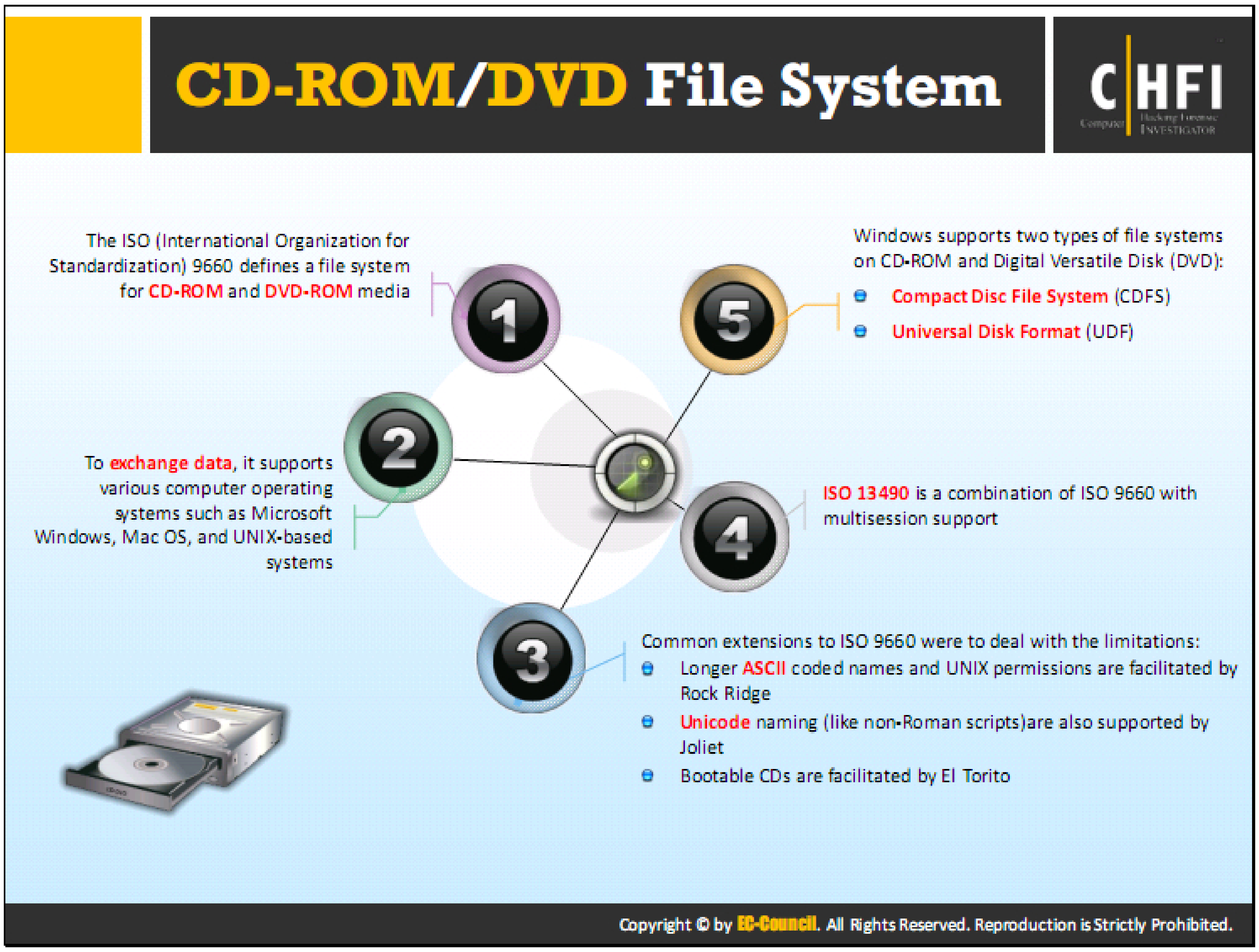
- **Deduplication, ZFS and Solid-State Storage and Compression**

  ZFS enables data deduplication, which requires large RAM capacity of up to 1 and 5 GB for every TB of storage for better use of deduplication. The lack required physical memory or ZFS cache will lead to virtual memory thrashing at the time of deduplication and also lower performance or result in complete memory starvation. Use speed up deduplication solid-state drives (SSDs) to cache deduplication tables and speed up deduplication.

- **Encryption**

  ZFS has the encryption feature embedded into the I/O pipeline and the encryption policy set at the dataset level. The file system can compress, encrypt, checksum, and de-duplicate a block during writes. It also allows the users to change the encryption keys at any time without taking the file system offline.

Source: *http://www.oracle.com*



CD-ROM/DVD File System

1. The ISO (International Organization for Standardization) 9660 defines a file system for CD-ROM and DVD-ROM media
2. To exchange data, it supports various computer operating systems such as Microsoft Windows, Mac OS, and UNIX-based systems
3. Common extensions to ISO 9660 were to deal with the limitations:
   - Longer ASCII coded names and UNIX permissions are facilitated by Rock Ridge
   - Unicode naming (like non-Roman scripts)are also supported by Joliet
   - Bootable CDs are facilitated by El Torito
4. ISO 13490 is a combination of ISO 9660 with multisession support
5. Windows supports two types of file systems on CD-ROM and Digital Versatile Disk (DVD):
   - Compact Disc File System (CDFS)
   - Universal Disk Format (UDF)

The computer systems require file systems, such as NTFS or UNIX, to exchange and access the data contained in files easily and quickly. They divide data stored on CD-ROMs into sectors, containing both user data and error correction codes. Users need not worry about which data is stored in which sector, but should have an understanding of the CD-ROM file structure.

ISO 9660

ISO (International Organization for Standardization) 9660 is a standard that defines uses for file systems of CD-ROM and DVD media. Its aim is to support different operating systems like MS Windows, Mac OS, and systems that follow the UNIX specification in order to exchange data.

There are some well-known extensions to ISO 9660 to deal with the limitations. For example, the Rock Ridge extension allows usage of almost all ASCII characters for file or folder names, and enables longer file names (up to 255 characters). Users can select ASCII code to store the owner's permission, for deeper directory and also for symbolic links. The El Torito Bootable CD specification allows machines to boot with the help of a CD-ROM.

ISO 13490

This standard has several improvements over 9660, the later having few drawbacks, namely POSIX attributes and multi-byte characters. It addresses the file name completely. It is an efficient format that allows incremental recording and also permit the ISO 9660 format and the ISO/IEC 13490 format to exist on the same media. It specifies using multicasting properly.

ISO 9660 Specifications

A reserved area of 32,768 bytes at the beginning of the disk is present for use in booting CD-ROMs on a computer (system area). However, it is important to note that, the ISO 9660 standard did not specify its use.

Immediately a series of volume descriptors details the contents and kind of information contained on the disk (something like the partition table of MS-DOS).

A volume descriptor describes the characteristics of the file system information present on a given CD-ROM, or volume. It has two parts: the type of volume descriptor, and the characteristics of the descriptor.

Construction of the volume descriptor is in such a manner that if a program reading the disk does not understand a particular descriptor, it can just skip over until it recognizes one , thus allowing the use of many different types of information on one CD-ROM. Also, if an error were to render a descriptor unreadable, a subsequent redundant copy of a descriptor could then allow for fault recovery. While checking CD-ROMs with a dump utility, each descriptor is present in a single logical sector on itself, and also a backup of the descriptor, a few logical sectors further.

An ISO 9660–compliant disc contains at least a primary descriptor describing the ISO 9660 file system and a terminating descriptor for indicating the end of the descriptor sequence. Joliet and UDF are examples of such file systems that are adding more descriptors to this sequence.

The primary volume descriptor acts much like the super block of the UNIX file system, providing details on the ISO 9660–compliant portion of the disc. The primary volume descriptor contains the root directory record that describes the location of the contiguous root directory. (As in UNIX, directories appear as files for the operating system's special use.) Directory entries are successively stored within this region. Evaluation of the ISO 9660 file names begins at this location. The root directory is stored as an extent, or as sequential series of sectors, that contains each of the directory entries appearing in the root. In addition, since ISO 9660 works by segmenting the CD-ROM into logical blocks, the size of these blocks is present in the primary volume descriptor as well.

The first field in a volume descriptor is the volume descriptor type (type), which can have the following values:

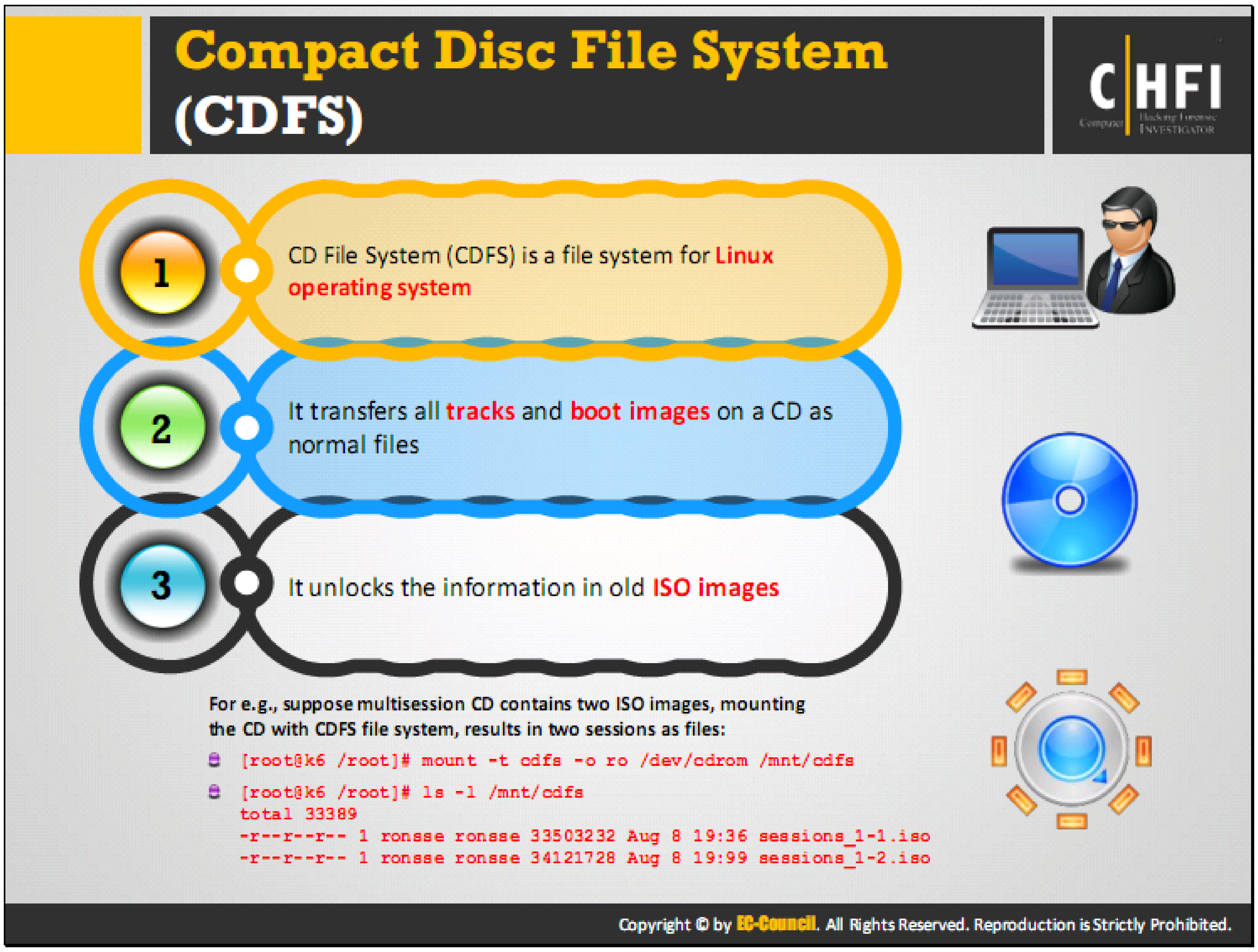
- **Number 0**: refers that the volume descriptor is a boot record
- **Number 1**: refers that the volume descriptor is a primary volume descriptor
- **Number 2**: refers that the volume descriptor is a supplementary volume descriptor
- **Number 3**: refers that the volume descriptor is a volume partition descriptor
- **Number 255**: refers that the volume descriptor is a volume descriptor set terminator

The second field is the standard identifier and is set to CD001 for a CD-ROM compliant to the ISO 9660 standard.

Another interesting field is the volume space size, which contains the amount of data available on the CD-ROM.

File attributes are simple in ISO 9660. The most important file attribute is determining whether the file is a directory or an ordinary file. File attributes for the file described by the directory entry are stored in the directory entry and optionally, in the extended attribute record.

Source: *http://en.wikipedia.org*



Compact Disc File System (CDFS)

1. CD File System (CDFS) is a file system for **Linux operating system**
2. It transfers all **tracks** and **boot images** on a CD as normal files
3. It unlocks the information in old **ISO images**

For e.g., suppose multisession CD contains two ISO images, mounting the CD with CDFS file system, results in two sessions as files:

```
[root@k6 /root]# mount -t cdfs -o ro /dev/cdrom /mnt/cdfs
```

```
[root@k6 /root]# ls -l /mnt/cdfs
total 33389
-r--r--r-- 1 ronsse ronsse 33503232 Aug 8 19:36 sessions_1-1.iso
-r--r--r-- 1 ronsse ronsse 34121728 Aug 8 19:99 sessions_1-2.iso
```

The CD File System (CDFS) is a file system for the Linux operating system. It transfers all tracks and boot images on a CD, as normal files. These files can then be mounted (for example, for ISO and boot images), copied, and played (audio and video CD tracks). The primary goal for developing this file system was to unlock information in old ISO images.

For instance, if there is a multisession CD with two ISO images that contains the file "a," only the file "a" in the second session is seen only if the ISO 9660 file system is used:

```
[root@k6 /root]# mount -t iso9660 -o ro /dev/cdrom /mnt/cdrom

[root@k6 /root]# ls -l /mnt/cdrom

total 2

-r-xr-xr-x 1 root root 2 Aug 8 19:16 a

-r-xr-xr-x 1 root root 2 Aug 8 19:19 b
```

If we mount the CD with the CDFS file system, we get the two sessions as files:

```
[root@k6 /root]# mount -t cdfs -o ro /dev/cdrom /mnt/cdfs

[root@k6 /root]# ls -l /mnt/cdfs

total 33389

-r--r--r-- 1 ronsseronsse 33503232 Aug 8 19:36 sessions_1-1.iso

-r--r--r-- 1 ronsseronsse 34121728 Aug 8 1999 sessions_1-2.iso
```

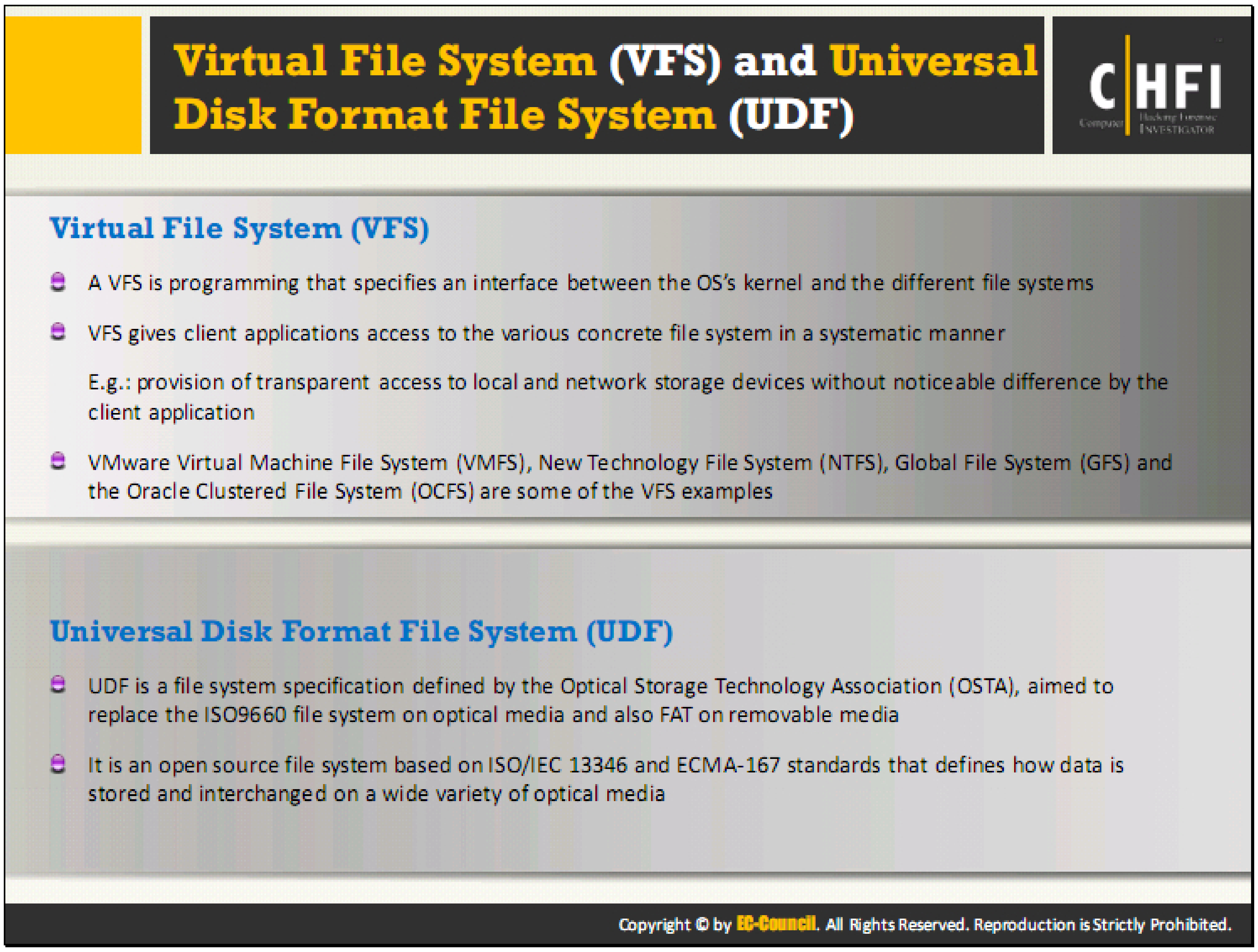
These files can then be mounted loop back:

```
[root@k6 /root]# mount -t iso9660 -o loop /cdfs/sessions_1-1.iso /mnt/loop1
[root@k6 /root]# mount -t iso9660 -o loop /cdfs/sessions_1-2.iso /mnt/loop2
```

File "a" can be accessed in both sessions:

```
[root@k6 /root]# ls -l /mnt/loop1
total 9889
-r-xr-xr-x 1 root root 10104236 Aug 8 17:34 a
[root@k6 /root]# ls -l /mnt/loop2
total 2
-r-xr-xr-x 1 root root 2 Aug 8 19:16 a
-r-xr-xr-x 1 root root 2 Aug 8 19:19 b
```

Source: *http://users.elis.ugent.be*



Virtual File System (VFS) and Universal Disk Format File System (UDF)

Virtual File System (VFS)

- A VFS is programming that specifies an interface between the OS's kernel and the different file systems
- VFS gives client applications access to the various concrete file system in a systematic manner

  E.g.: provision of transparent access to local and network storage devices without noticeable difference by the client application
- VMware Virtual Machine File System (VMFS), New Technology File System (NTFS), Global File System (GFS) and the Oracle Clustered File System (OCFS) are some of the VFS examples

Universal Disk Format File System (UDF)

- UDF is a file system specification defined by the Optical Storage Technology Association (OSTA), aimed to replace the ISO9660 file system on optical media and also FAT on removable media
- It is an open source file system based on ISO/IEC 13346 and ECMA-167 standards that defines how data is stored and interchanged on a wide variety of optical media

Virtual File System (VFS)

In Linux, the real file systems remain detached from the operating system and system services, and are connected through an interface layer known as the Virtual File system, or VFS. A VFS is a programming that acts as an interface between the OS's kernel and the different file systems. It provides client applications with constant access to various file system through a common interface. For example, provision of transparent access to local and network storage devices without noticeable difference by the client application. VMware Virtual Machine File System (VMFS), New Technology File System (NTFS), Global File System (GFS) and the Oracle Clustered File System (OCFS) are some of the VFS examples.

The Linux VFS provides fast and efficient access to the files, while ensuring proper storage of files. The VFS also caches information in memory from each file system when mounted and used.

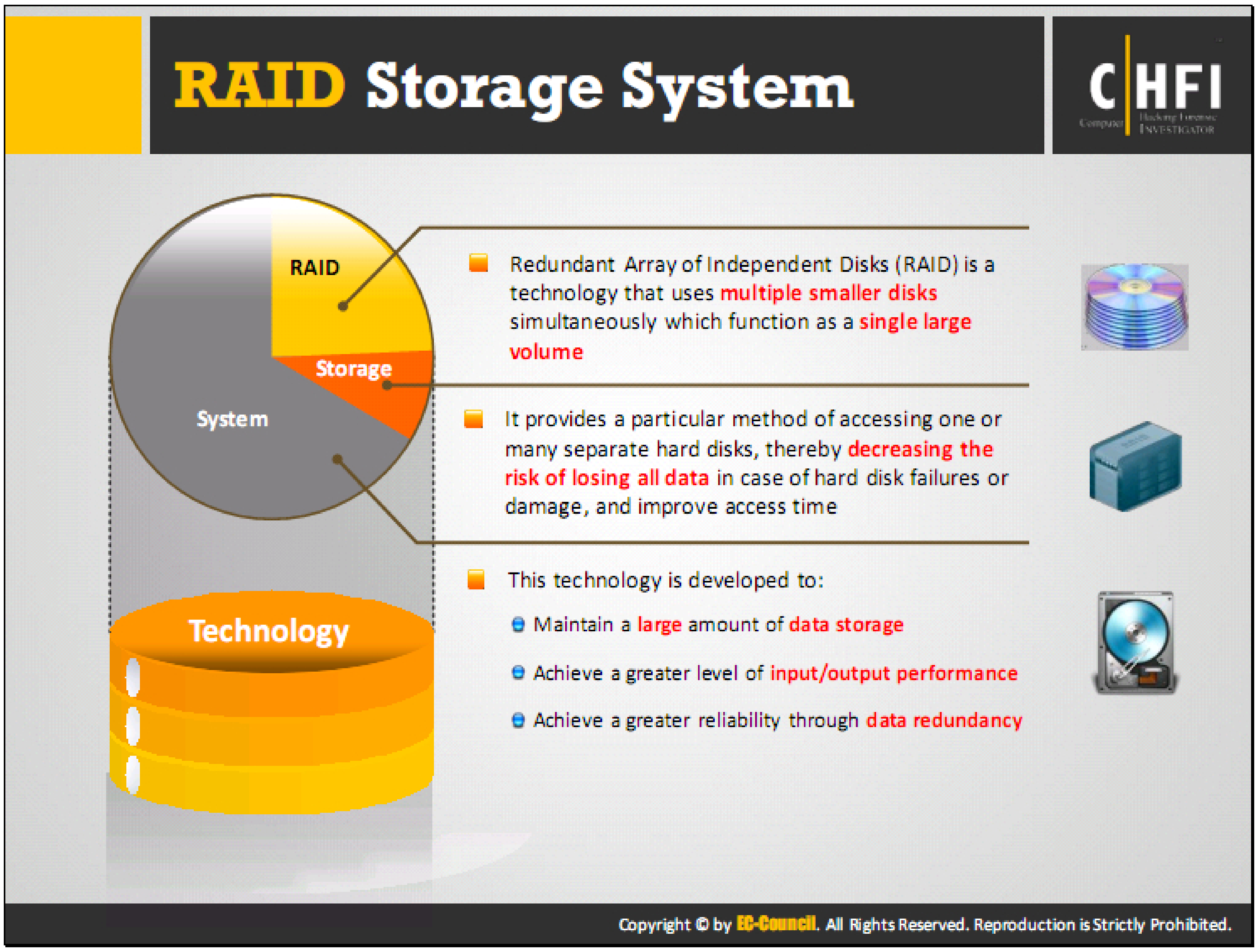
Universal Disk Format File System (UDF)

UDF is a file system specification defined by the Optical Storage Technology Association (OSTA), meant to replace the ISO9660 file system on optical media and also FAT on removable media. It is an open source file system based on ISO/IEC 13346 and ECMA-167 standards that defines how a variety of optical media store and interchange the data.

DVD-ROMs need UDF, as DVDs, CD-R and CD-RW use it to store MPEG audio/video streams in a process called packet writing, which offers more efficiency in time consumption and disk space required.

The UDF standard defines three file system variations, also called builds. These includes:

- Plain (Random Read/Write Access)
- Virtual Allocation Table also known as VAT (Incremental Writing)
- Spared (Limited Random Write Access)



RAID Storage System

RAID
Storage
System
Technology

- Redundant Array of Independent Disks (RAID) is a technology that uses multiple smaller disks simultaneously which function as a single large volume
- It provides a particular method of accessing one or many separate hard disks, thereby decreasing the risk of losing all data in case of hard disk failures or damage, and improve access time
- This technology is developed to:
  - Maintain a large amount of data storage
  - Achieve a greater level of input/output performance
  - Achieve a greater reliability through data redundancy

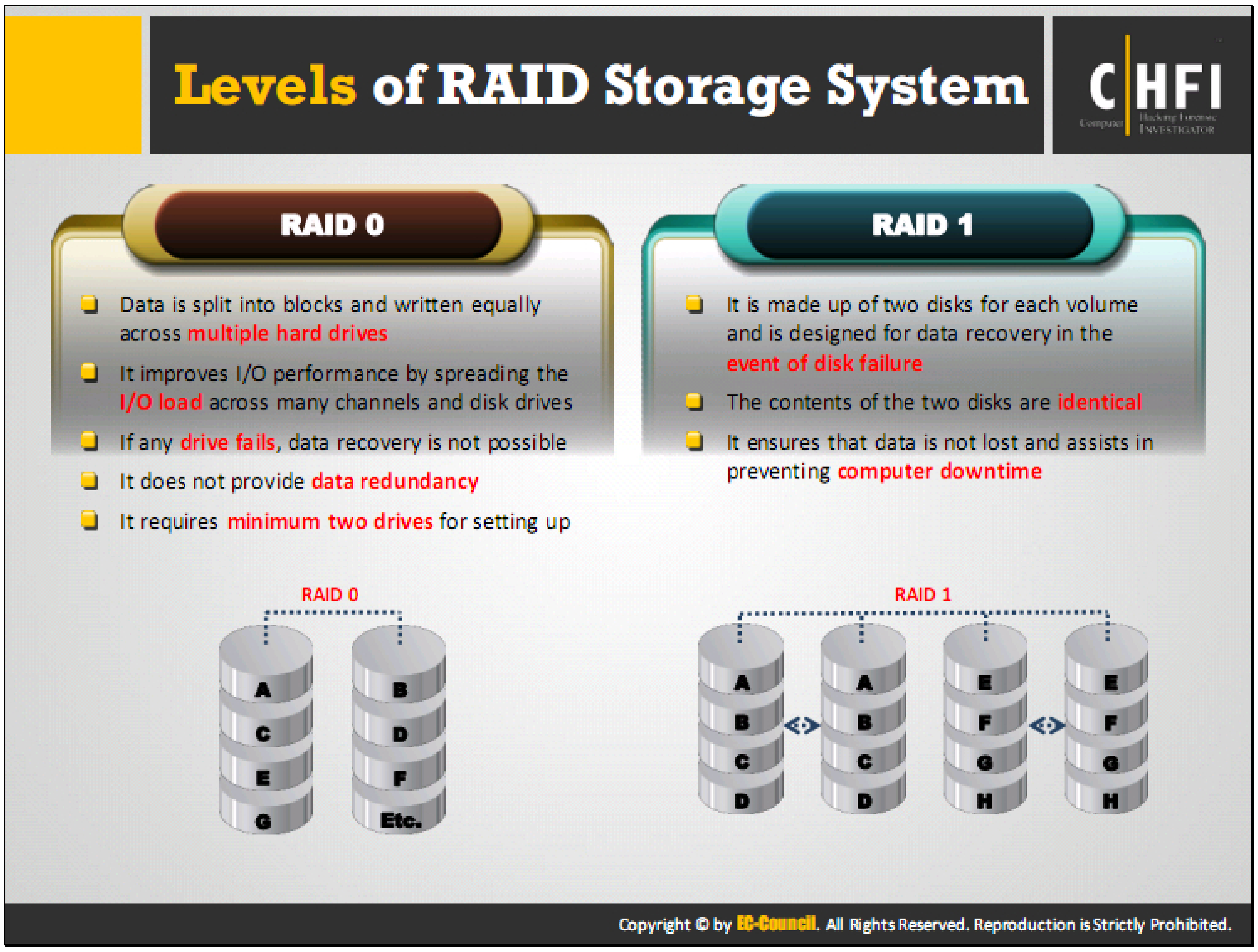
Redundant Array of Independent Disks (RAID) is a technology that uses multiple smaller disks simultaneously, which function as a single large volume. It provides a particular method of accessing one or many separate hard disks, thereby decreasing the risk of losing all data if at all a hard disk fails or is prone to damages, and it also helps in improving access time.

The RAID technology helps users to:

- Maintain a large amount of data storage.
- Achieve a greater level of input/output performance.
- Achieve greater reliability through data redundancy.

The basic idea of the RAID storage system is to group multiple small and cheap hard disks into an array of hard disks that provides performance greater than that of single large hard disk. A RAID storage system is a collection of hard disks that work as and also appear to be a single large capacity disk drive to the user. The main advantage of the RAID system is that if any disk in the RAID array fails or is susceptible to damage; the system still continues to function without any loss of data. This is possible because the data of each separate hard disk is stored on another disk in an array.

The RAID system allows multiple simultaneous accesses to different files on different hard disks, which reduces the time required to find the data on a hard disk. Data transfer considerably increases on a RAID system over a single hard disk.



Levels of RAID Storage System

RAID 0

- Data is split into blocks and written equally across multiple hard drives
- It improves I/O performance by spreading the I/O load across many channels and disk drives
- If any drive fails, data recovery is not possible
- It does not provide data redundancy
- It requires minimum two drives for setting up

RAID 1

- It is made up of two disks for each volume and is designed for data recovery in the event of disk failure
- The contents of the two disks are identical
- It ensures that data is not lost and assists in preventing computer downtime

RAID Level 0: Disk Striping

It is the simplest RAID level, which does not involve any redundancy and fragments the file into user-defined stripe size of the array. Then it sends these stripes to every disk in the array. As RAID 0 does not have redundancy, it allows this RAID level to offer the best overall performance characteristics of the single RAID levels.

Following are the features of RAID level 0: Disk Striping

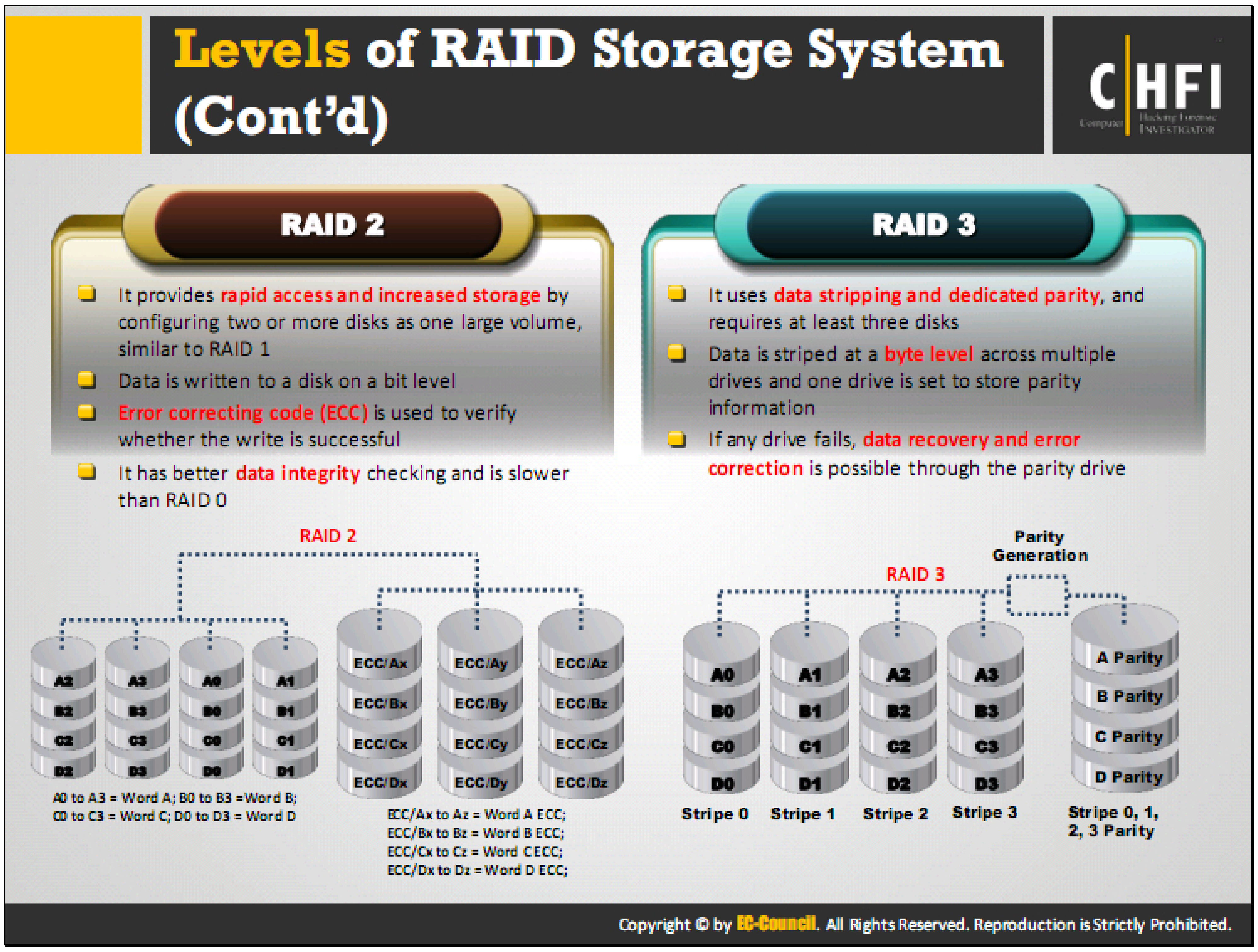
- Split data into blocks and writes equally across multiple hard drives
- If any drive fails, data recovery is not possible
- It does not provide data redundancy
- It requires a minimum of two drives to set up
- The main benefit of this level is its high level of input/output performance, as it spreads the inputs/outputs across multiple channels and hard drives
- Data transfer rate is fast and most efficient

RAID Level 1: Disk Mirroring

RAID 1 generally executes mirroring as it duplicates or copies the drive data on to two different drives using a hardware RAID controller or a software. If one of the drives fail, the other will function as a single drive until a user replaces the failed drive with a new one.

Following are the features of disk mirroring:

- Writes multiple copies of data to multiple drives at the same time
- It provides data redundancy by completely duplicating the drive data to multiple drives
- If one drive fails, data recovery is possible
- It requires a minimum of two drives to set up
- The performance of this level is faster on reading data and slower on writing data compared to the single hard drive because the data is distributed between multiple hard drives
- This level provides the best protection of data
- The main disadvantage of this level is that data access speed is slow
- It provides high data consistency and continued data availability in case of failure of any hard disk drive
- It gives 100% data redundancy and does not require rebuild time



Levels of RAID Storage System (Cont'd)

RAID 2

- It provides rapid access and increased storage by configuring two or more disks as one large volume, similar to RAID 1
- Data is written to a disk on a bit level
- Error correcting code (ECC) is used to verify whether the write is successful
- It has better data integrity checking and is slower than RAID 0

RAID 3

- It uses data stripping and dedicated parity, and requires at least three disks
- Data is striped at a byte level across multiple drives and one drive is set to store parity information
- If any drive fails, data recovery and error correction is possible through the parity drive

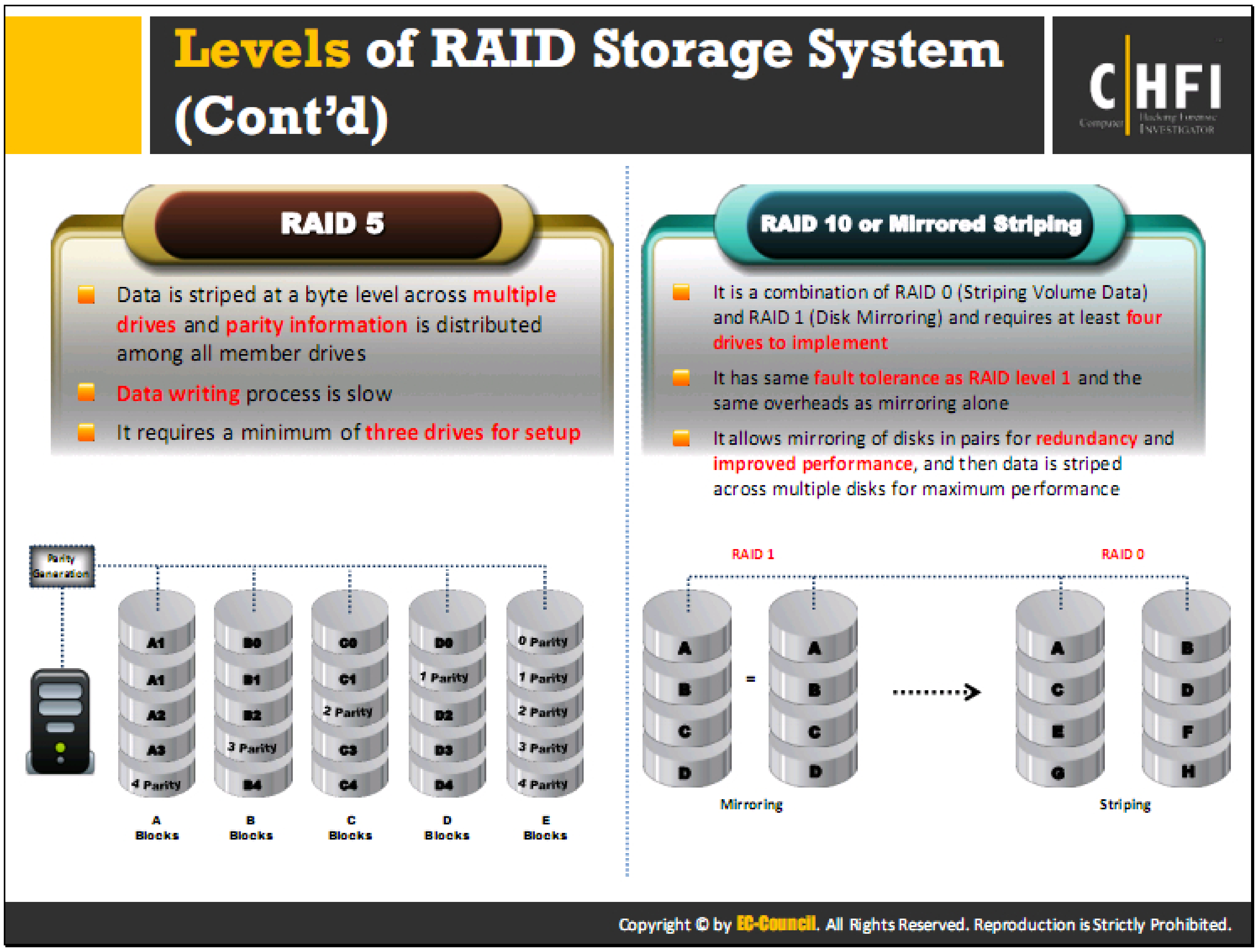
RAID 2

RAID 2 is the only level among all the RAID levels that does not implement even one of the standard techniques of parity, mirroring or striping. It uses a technique similar to striping with parity. It includes splitting of data at the bit level and distributing it to numerous data disks and redundancy disks.

Hamming codes, which is a form of error correcting code (ECC), help to calculate these redundant bits. When a user writes something to an array, the system calculates the codes and writes them together with the data to dedicated ECC disks.

RAID 3

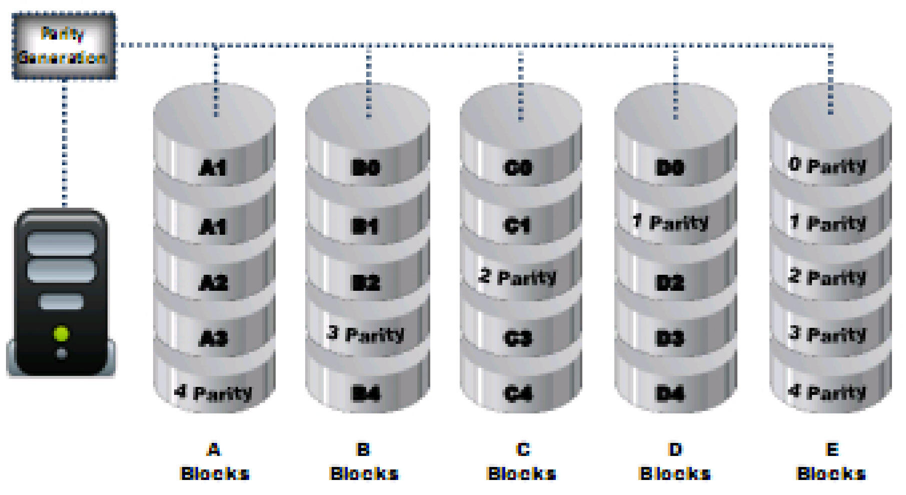
RAID 3 uses byte-level stripping with a dedicated parity disk, which stores checksums. It also supports a special processor for parity codes calculation. This RAID cannot cater multiple data requests simultaneously. If a failure occurs, it enables data recovery by an applicable calculation of the parity bytes, and the remaining bytes which relate with them.



Levels of RAID Storage System (Cont'd)

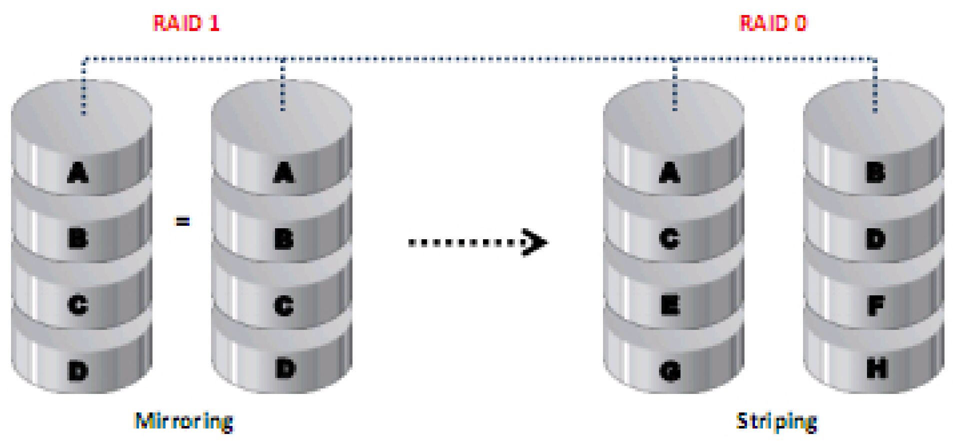
RAID 5

- Data is striped at a byte level across multiple drives and parity information is distributed among all member drives
- Data writing process is slow
- It requires a minimum of three drives for setup



RAID 10 or Mirrored Striping

- It is a combination of RAID 0 (Striping Volume Data) and RAID 1 (Disk Mirroring) and requires at least four drives to implement
- It has same fault tolerance as RAID level 1 and the same overheads as mirroring alone
- It allows mirroring of disks in pairs for redundancy and improved performance, and then data is striped across multiple disks for maximum performance

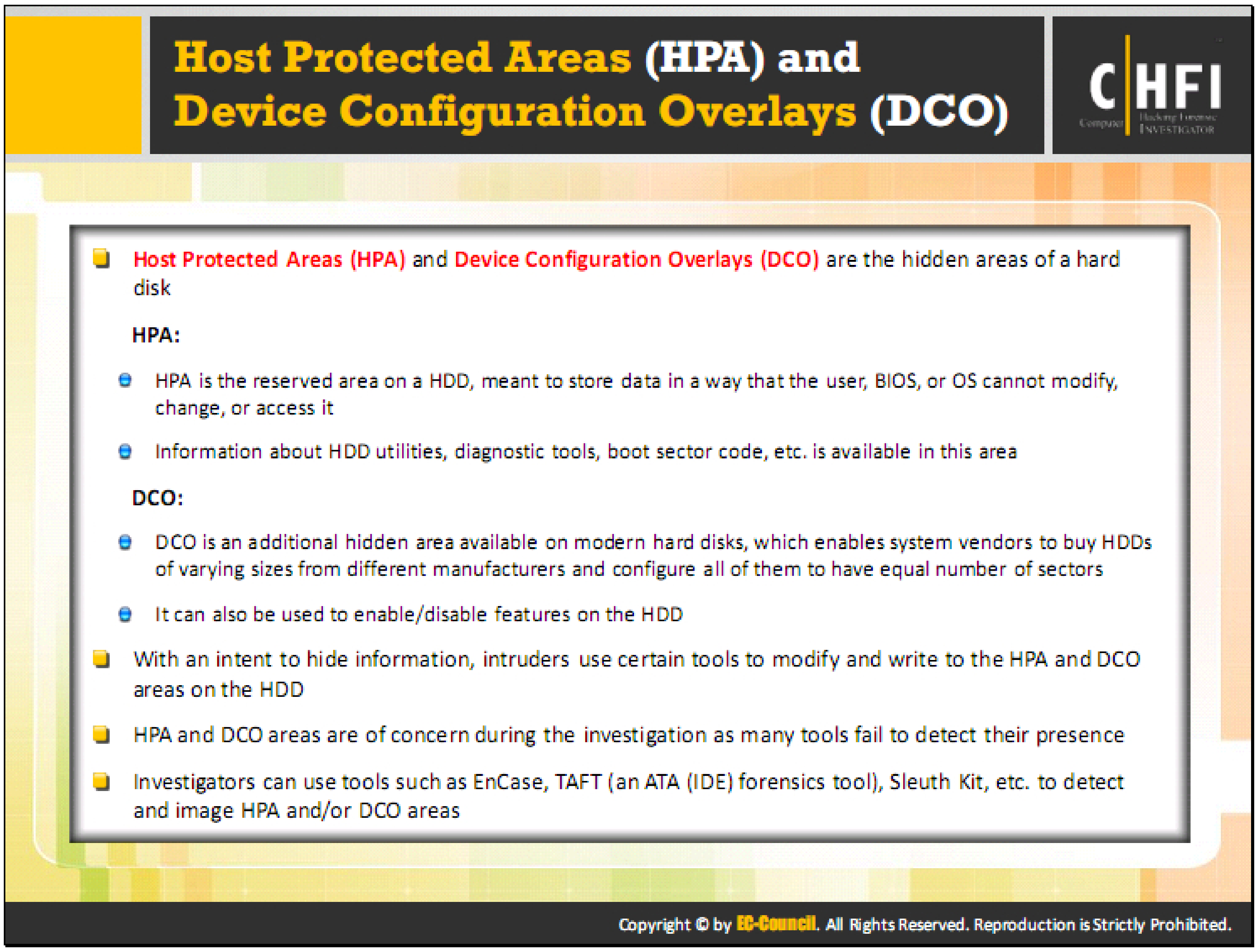


RAID 5

Uses byte level data striping across multiple drives, and distributes the parity information among all member drives. Data writing process is slow. It requires a minimum of three drives to set up. The RAID stripes and distributes the error detection and correction code or Data and parity code across three or more drives.

RAID 10

RAID 10, also known as RAID 1+0, is a combination of RAID 0 (Striping Volume Data) and RAID 1 (Disk Mirroring) to protect data. It requires at least four drives to implement. It has same fault tolerance as RAID level 1 and the same overheads as mirroring alone. It allows mirroring of disks in pairs for redundancy and improved performance, and then stripes data across multiple disks for maximum performance. The user retrieves data from the RAID if one disk in each mirrored pair is working; however, if two disks in the same mirrored pair fail, the data is not available.



Host Protected Areas (HPA) and Device Configuration Overlays (DCO)

- Host Protected Areas (HPA) and Device Configuration Overlays (DCO) are the hidden areas of a hard disk

  HPA:
  - HPA is the reserved area on a HDD, meant to store data in a way that the user, BIOS, or OS cannot modify, change, or access it
  - Information about HDD utilities, diagnostic tools, boot sector code, etc. is available in this area

  DCO:
  - DCO is an additional hidden area available on modern hard disks, which enables system vendors to buy HDDs of varying sizes from different manufacturers and configure all of them to have equal number of sectors
  - It can also be used to enable/disable features on the HDD
- With an intent to hide information, intruders use certain tools to modify and write to the HPA and DCO areas on the HDD
- HPA and DCO areas are of concern during the investigation as many tools fail to detect their presence
- Investigators can use tools such as EnCase, TAFT (an ATA (IDE) forensics tool), Sleuth Kit, etc. to detect and image HPA and/or DCO areas

Hard disk drives and other storage media can have hidden areas such as Host Protected Areas (HPA) and Device Configuration Overlays (DCO).

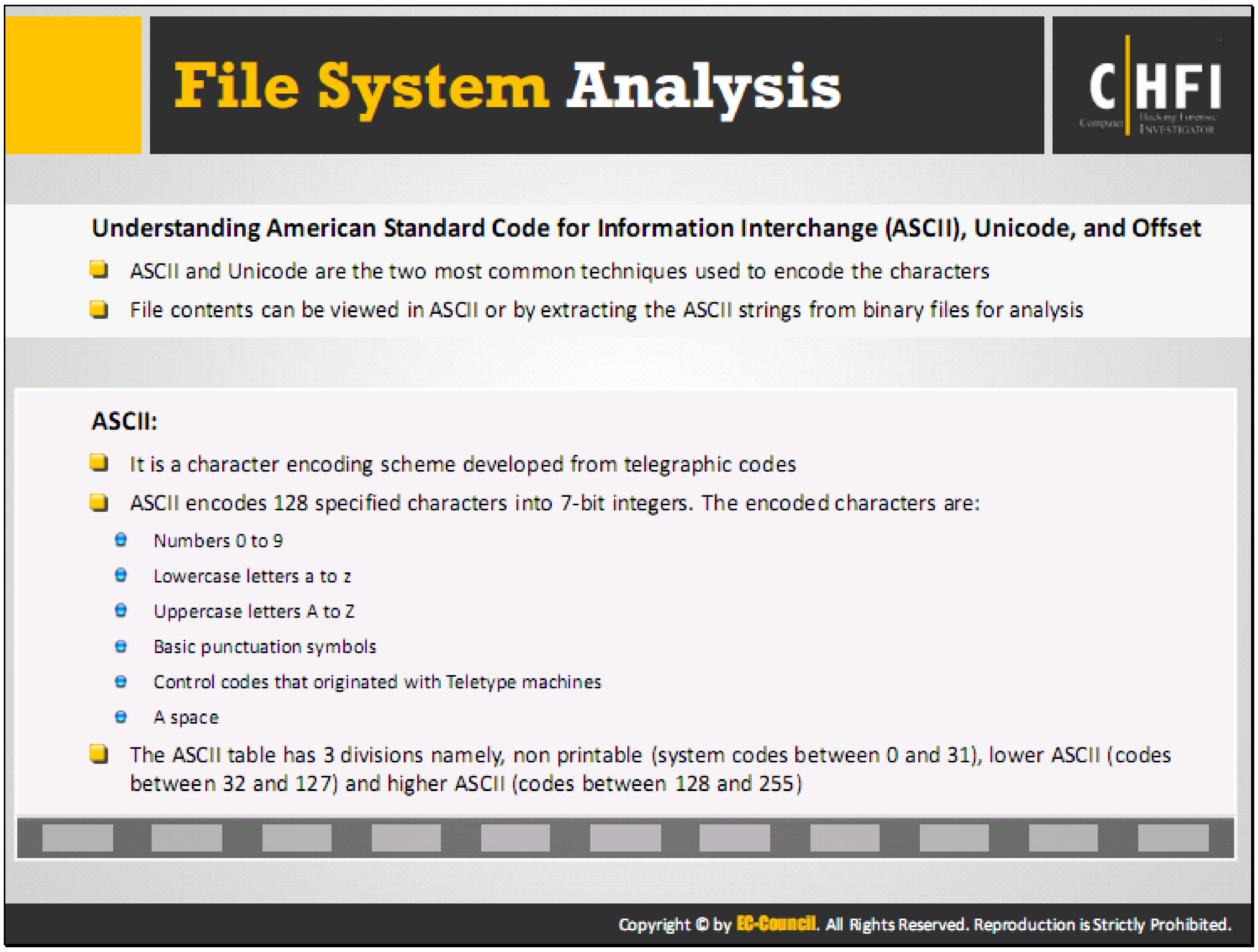
- Host Protected Areas (HPA): First introduced in the ATA-4 standard, HPA is a reserved area on a Hard Disk Drive (HDD) or a Solid State Drive (SSD) that is not visible to the operating system. This space stores data, which the users, BIOS or OS of a system cannot modify, change, or access it. It can store information about HDD utilities, diagnostic tools, boot sector code, etc.

  The three ATA commands that can help users to create and use a hidden protected area are:

  - IDENTIFY DEVICE
  - SET MAX ADDRESS
  - READ NATIVE MAX ADDRESS

- Device Configuration Overlays (DCO): First introduced in the ATA-6 standard, DCO is an additional hidden area available on modern hard disks, which enables system vendors to buy HDDs of varying sizes from different manufactures and configure all of them to have equal number of sectors. DCO refers to the area hidden from the system, BIOS, and users. It can help the users to enable/disable features on the HDD. To determine the actual size and features of a disk, it is required to use the command the DEVICE_CONFIGURATION_IDENTIFY.

These areas can prove to be tricky to the users as perpetrators can hide data in them without knowledge of the investigators. The intruders use certain tools to modify and write to the HPA and DCO areas on the HDD. Investigators can use tools such as EnCase, TAFT (an ATA (IDE) forensics tool), Sleuth Kit, etc. to detect and image HPA and/or DCO areas.



File System Analysis

Understanding American Standard Code for Information Interchange (ASCII), Unicode, and Offset

- ASCII and Unicode are the two most common techniques used to encode the characters
- File contents can be viewed in ASCII or by extracting the ASCII strings from binary files for analysis

ASCII:

- It is a character encoding scheme developed from telegraphic codes
- ASCII encodes 128 specified characters into 7-bit integers. The encoded characters are:
  - Numbers 0 to 9
  - Lowercase letters a to z
  - Uppercase letters A to Z
  - Basic punctuation symbols
  - Control codes that originated with Teletype machines
  - A space
- The ASCII table has 3 divisions namely, non printable (system codes between 0 and 31), lower ASCII (codes between 32 and 127) and higher ASCII (codes between 128 and 255)

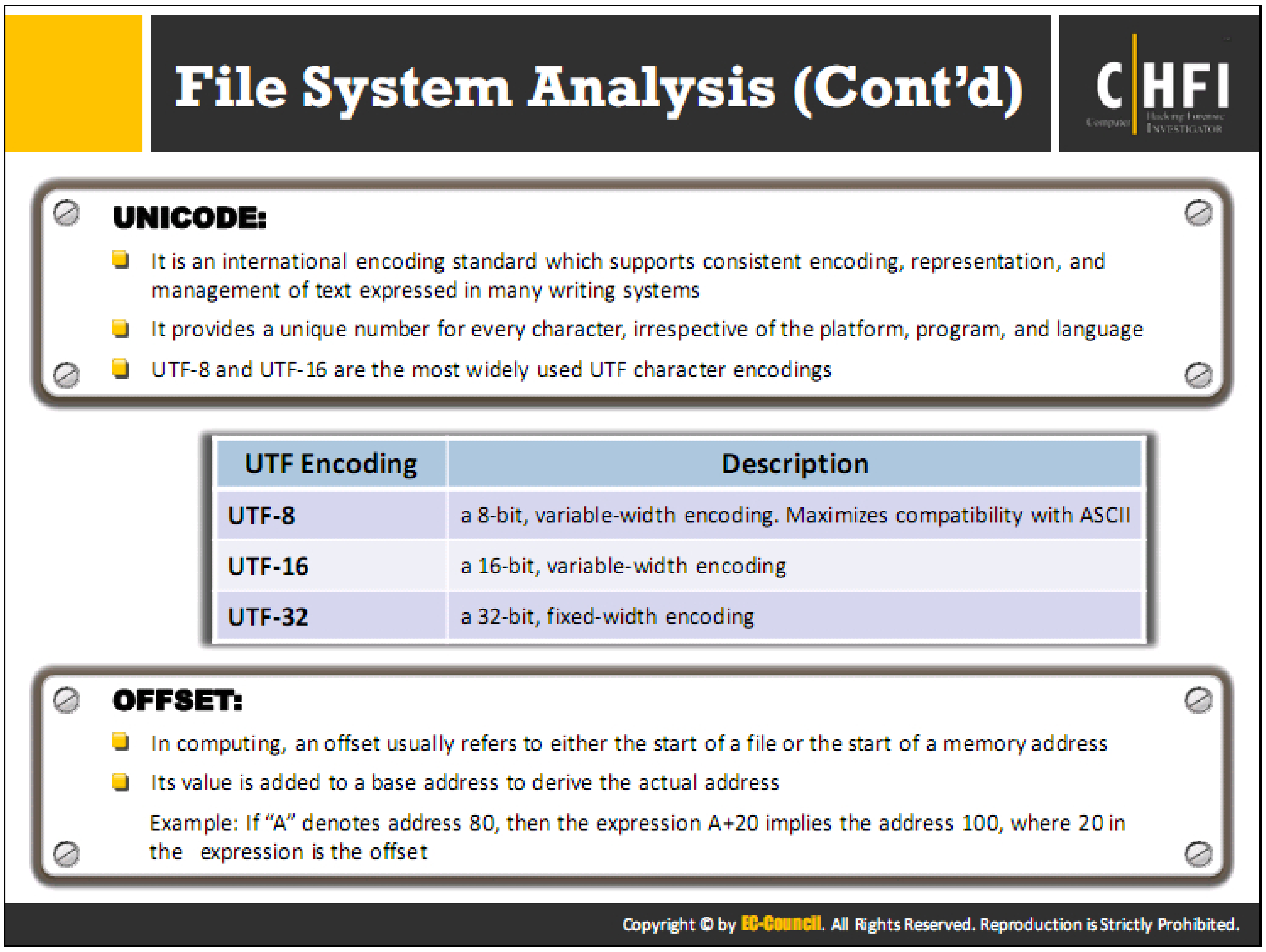
Understanding ASCII, Unicode, and Offset

American Standard Code for Information Interchange (ASCII)

Developed from telegraph codes, ASCII is a character encoding standard used in digital devices such as computers. The standard has 128 specified characters coded into 7-bit integers. Source code of a program, batch files, macros, scripts, HTML and XML documents are also ASCII files. The characters encoded are:

- Numbers 0 to 9
- Lowercase letters a to z
- Uppercase letters A to Z
- Basic punctuation symbols
- Control codes that originated with teletype machines
- A space

ASCII is a machine readable language, used in major digital operations such as sending and receiving emails. The ASCII table has 3 divisions namely, non-printable (system codes between 0 and 31), lower ASCII (codes between 32 and 127), and higher ASCII (codes between 128 and 255). The graphics files and documents use non-ASCII characters made in word processers, spreadsheet or database programs and sent as email file attachments.



File System Analysis (Cont'd)

UNICODE:

- It is an international encoding standard which supports consistent encoding, representation, and management of text expressed in many writing systems
- It provides a unique number for every character, irrespective of the platform, program, and language
- UTF-8 and UTF-16 are the most widely used UTF character encodings

UTF Encoding	Description
UTF-8	a 8-bit, variable-width encoding. Maximizes compatibility with ASCII
UTF-16	a 16-bit, variable-width encoding
UTF-32	a 32-bit, fixed-width encoding

OFFSET:

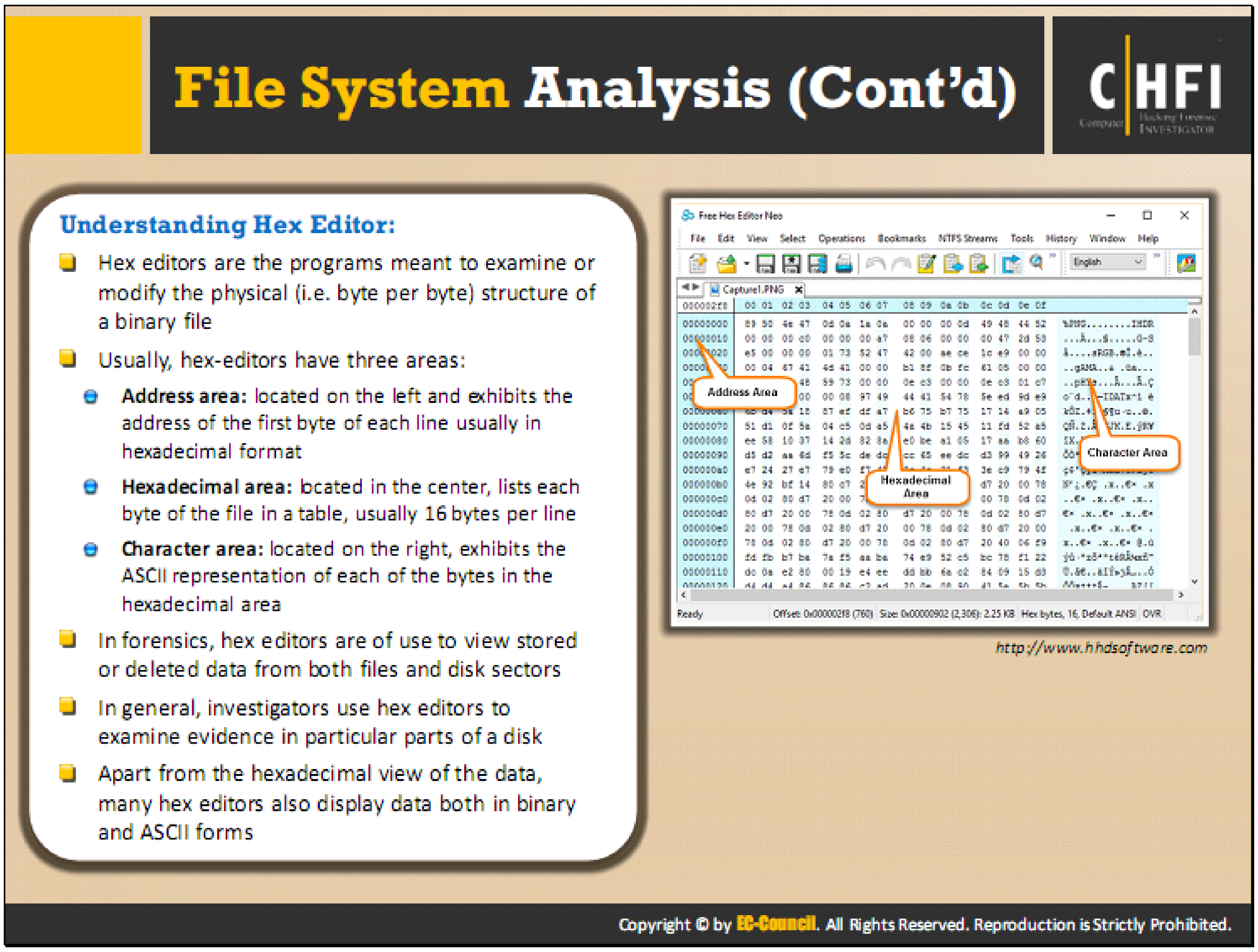
- In computing, an offset usually refers to either the start of a file or the start of a memory address
- Its value is added to a base address to derive the actual address

  Example: If "A" denotes address 80, then the expression A+20 implies the address 100, where 20 in the expression is the offset

UNICODE

Unicode is a computing standard, developed along with the Universal Coded Character Set (UCS) standard for encoding, representation, and management of texts, which most of the world's writing systems use. It provides a unique number for every character, irrespective of the platform, program, and language.

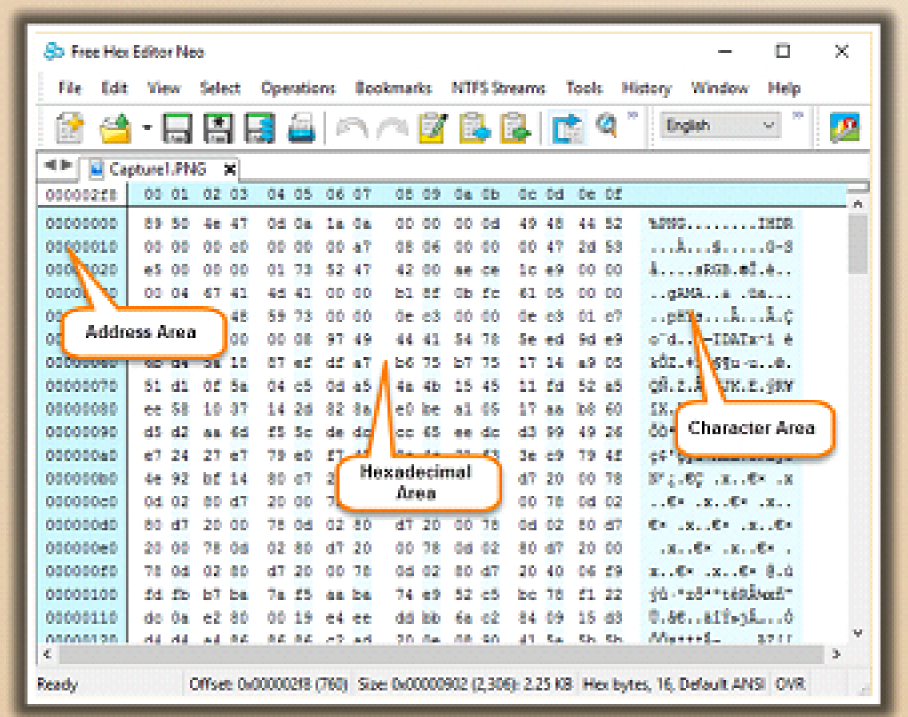
Unicode contains more than 128,000 characters from about 135 modern and historic scripts. Technologies such as modern operating systems, XML, Java, and the Microsoft .NET Framework have adopted the Unicode standards.



File System Analysis (Cont'd)

Understanding Hex Editor:

- Hex editors are the programs meant to examine or modify the physical (i.e. byte per byte) structure of a binary file
- Usually, hex-editors have three areas:
  - **Address area:** located on the left and exhibits the address of the first byte of each line usually in hexadecimal format
  - **Hexadecimal area:** located in the center, lists each byte of the file in a table, usually 16 bytes per line
  - **Character area:** located on the right, exhibits the ASCII representation of each of the bytes in the hexadecimal area
- In forensics, hex editors are of use to view stored or deleted data from both files and disk sectors
- In general, investigators use hex editors to examine evidence in particular parts of a disk
- Apart from the hexadecimal view of the data, many hex editors also display data both in binary and ASCII forms


Free Hex Editor Neo
File Edit View Select Operations Bookmarks NTFS Streams Tools History Window Help
English
Capture1.PNG
Address Area
Hexadecimal Area
Character Area

```
000002f8  00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
00000000  89 50 4e 47  0d 0a 1a 0a  00 00 00 0d  49 48 44 52   ‰PNG........IHDR
00000010  00 00 00 c0  00 00 00 a7  08 06 00 00  00 47 2d 53   ...À...§.....G-S
00000020  e5 00 00 00  01 73 52 47  42 00 ae ce  1c e9 00 00   å....sRGB.®Î.é..
00000030  00 04 67 41  4d 41 00 00  b1 8f 0b fc  61 05 00 00   ..gAMA..±..üa...
[illegible]  48  59 73 00 00  0e c3 00 00  0e c3 01 c7   ..pHYs...Ã...Ã.Ç
[illegible]  00  00 08 97 49  44 41 54 78  5e ed 9d e9   [illegible]IDATx^í.é
[illegible]  18  87 ef df a7  b6 75 b7 75  17 14 a9 05   [illegible]
00000070  51 d1 0f 5a  04 c5 0d a5  4a 4b 15 45  11 fd 52 a5   [illegible]JK.E.ýR¥
00000080  ee 58 10 37  14 2d 82 8a  e0 be a1 65  17 aa b8 60   îX[illegible]
00000090  d5 d2 aa 6d  f5 5c de [illegible]  cc 65 ee dc  d3 99 49 26   Õ[illegible]
000000a0  e7 24 27 e7  79 e0 [illegible]  3e c9 79 4f   ç$'[illegible]
000000b0  4e 92 bf 14  80 c7 [illegible]  d7 20 00 78   N'¿.€Ç .x..€× .x
000000c0  0d 02 80 d7  20 00 [illegible]  00 78 0d 02   ..€× .x..€× .x..
000000d0  80 d7 20 00  78 0d 02 80  d7 20 00 78  0d 02 80 d7   €× .x..€× .x..€×
000000e0  20 00 78 0d  02 80 d7 20  00 78 0d 02  80 d7 20 00    .x..€× .x..€× .
000000f0  78 0d 02 80  d7 20 00 78  0d 02 80 d7  20 40 06 f9   x..€× .x..€× @.ù
00000100  fd fb b7 ba  7a f5 aa ba  74 e9 52 c5  bc 78 f1 22   ýû·ºzõªºtéRÅ¼xñ"
00000110  dc 0a e2 80  00 19 e4 ee  dd bb 6a c2  84 09 15 d3   Ü.â€..äîÝ»jÂ„..Ó
```

```
Ready    Offset: 0x000002f8 (760)    Size: 0x00000902 (2,306): 2.25 KB    Hex bytes, 16, Default ANSI    OVR
```

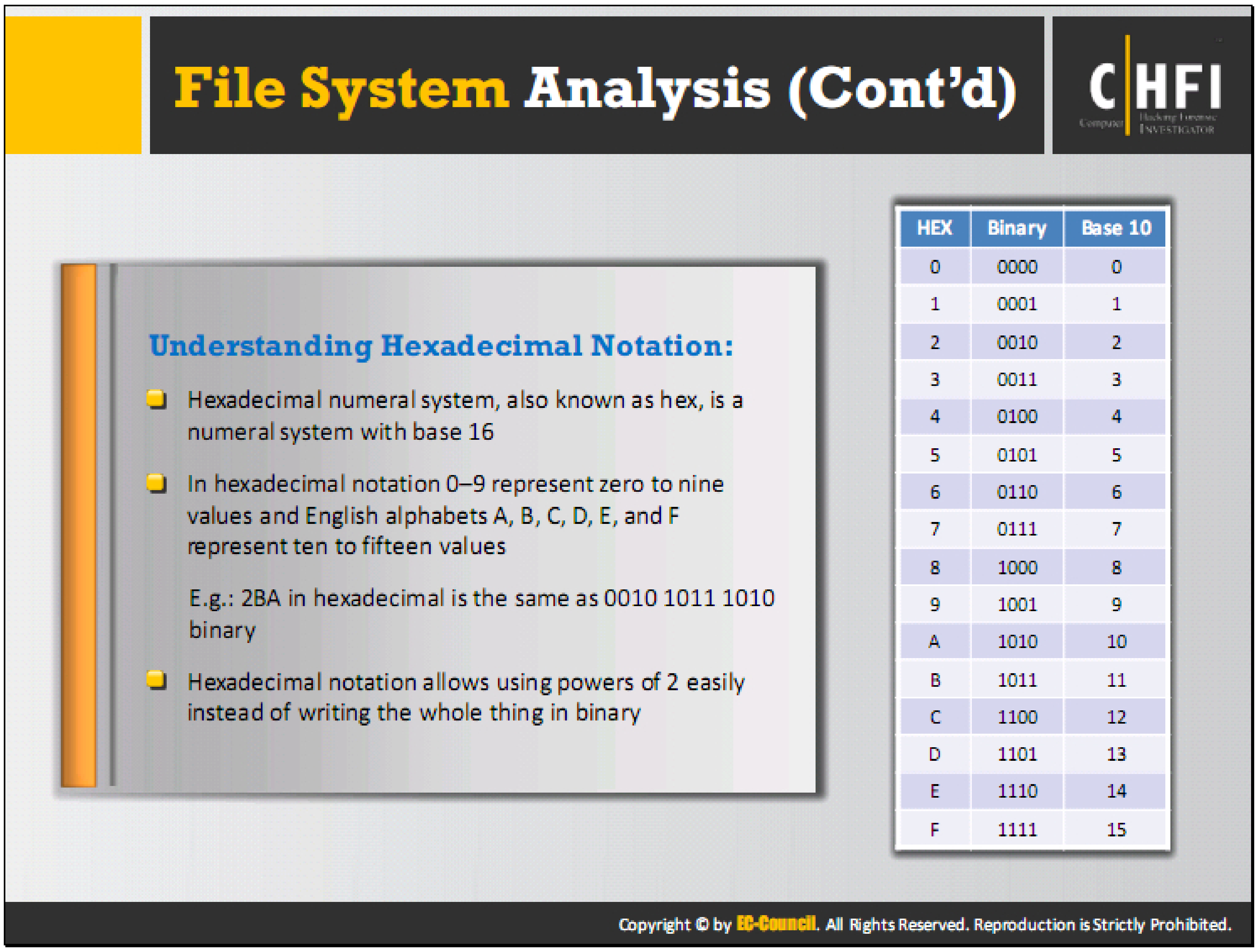
http://www.hhdsoftware.com

Understanding Hex Editor

A hex editor is a program that allows users to modify the fundamental binary data of a file. Using a hex editor, the user can see or edit the contents of a file. A hex editor has three display areas including an address area, a hexadecimal area, and a character area. Each area shows different values.

In digital forensic investigations, the hex editors allow the investigators to view any data stored in disk and also search for the remnants of deleted files. A hex editor allows investigators to view the physical contents stored on a disk, including the files, directories, or partitions.

Other functions include cracking of copy-protected software, studying of how computer viruses work, identify and retrieve hidden information.



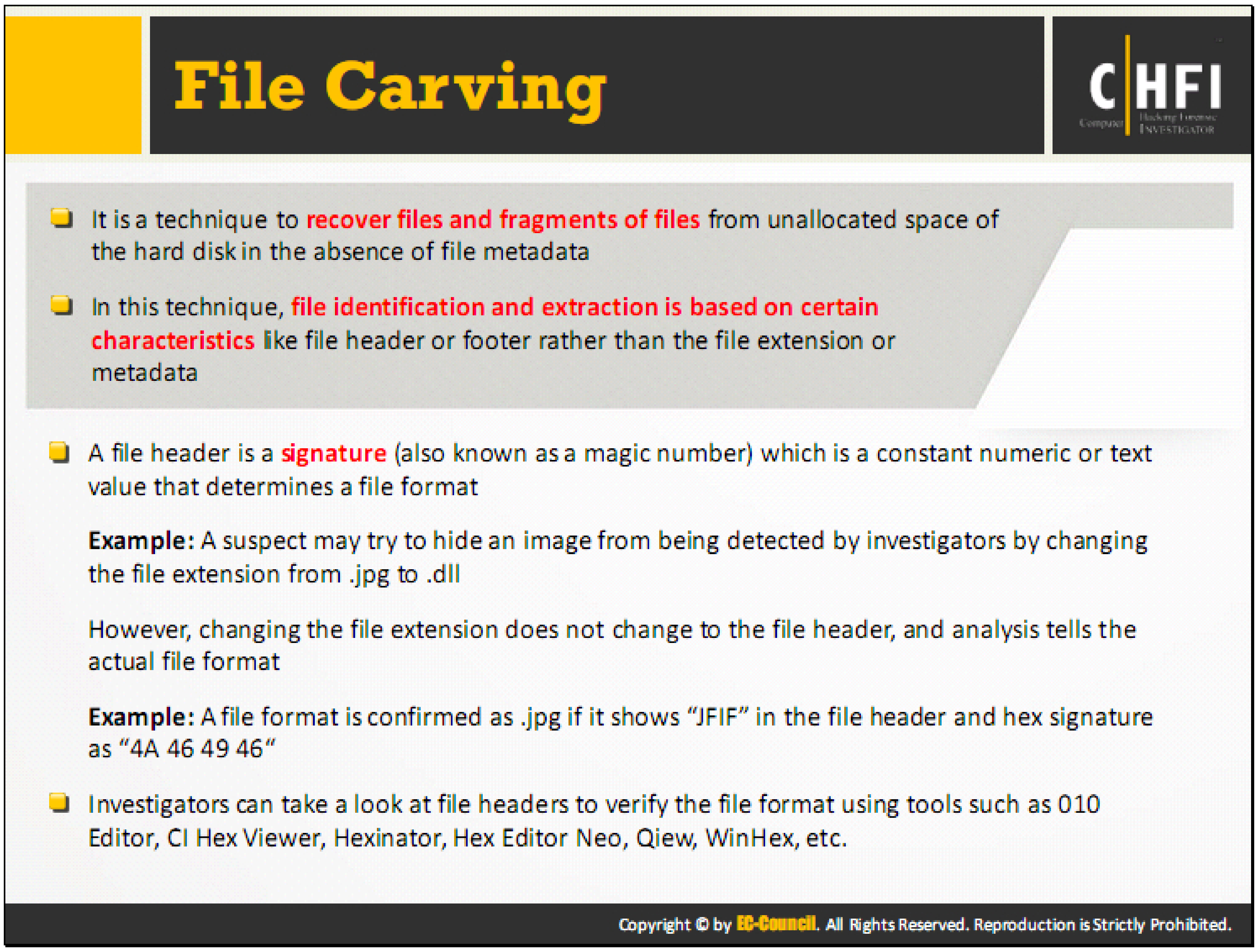
File System Analysis (Cont'd)

Understanding Hexadecimal Notation:

- Hexadecimal numeral system, also known as hex, is a numeral system with base 16
- In hexadecimal notation 0–9 represent zero to nine values and English alphabets A, B, C, D, E, and F represent ten to fifteen values

  E.g.: 2BA in hexadecimal is the same as 0010 1011 1010 binary
- Hexadecimal notation allows using powers of 2 easily instead of writing the whole thing in binary

HEX	Binary	Base 10
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15



File Carving

- It is a technique to recover files and fragments of files from unallocated space of the hard disk in the absence of file metadata
- In this technique, file identification and extraction is based on certain characteristics like file header or footer rather than the file extension or metadata
- A file header is a signature (also known as a magic number) which is a constant numeric or text value that determines a file format

  Example: A suspect may try to hide an image from being detected by investigators by changing the file extension from .jpg to .dll

  However, changing the file extension does not change to the file header, and analysis tells the actual file format

  Example: A file format is confirmed as .jpg if it shows "JFIF" in the file header and hex signature as "4A 46 49 46"
- Investigators can take a look at file headers to verify the file format using tools such as 010 Editor, CI Hex Viewer, Hexinator, Hex Editor Neo, Qiew, WinHex, etc.

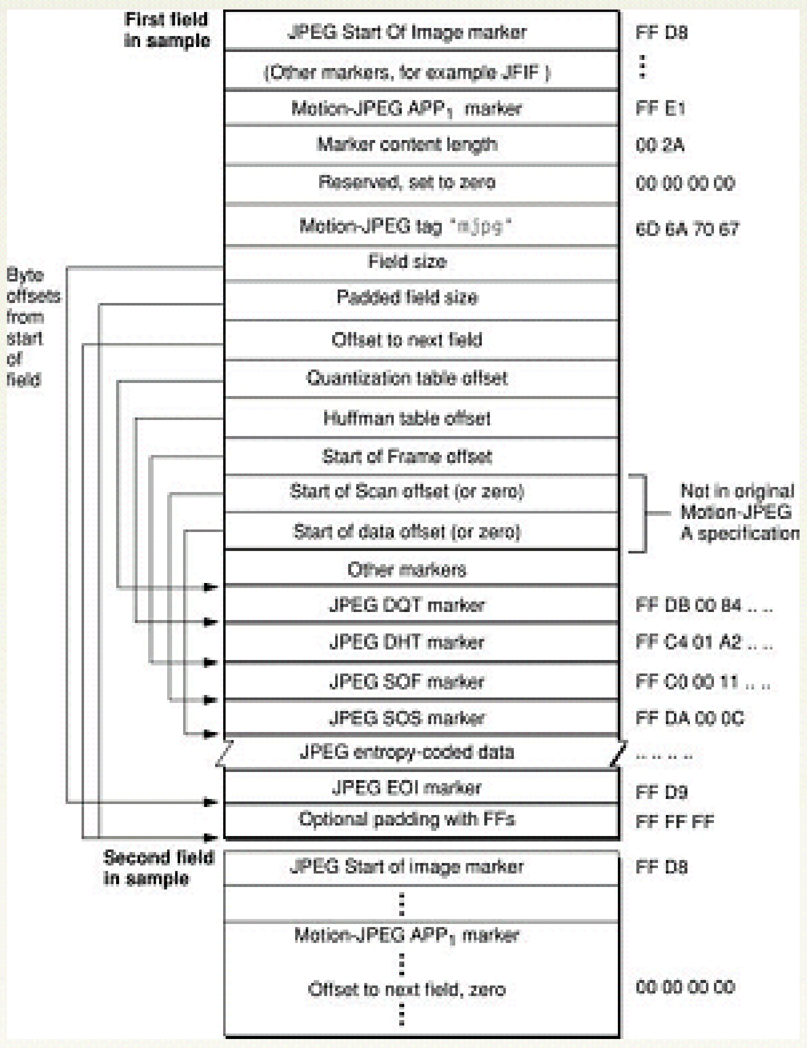
File Carving is the process of recovering files from their fragments and pieces from unallocated space of the hard disk in the absence of file system metadata. In computer forensics, it helps investigators to extract data from a storage media without any support of the file system used in creation of the file.

Unallocated space refers to the hard disk space that does not contain any file information, but store file data without the details of its location. Investigators can identify the files using certain characteristics like file header (the first few bytes) and footer (the last few bytes).

For example, a suspect may try to hide an image from detection by investigators by changing the file extension from .jpg to .dll. However, changing the file extension does not make changes to the file header, which on analyzing will reveal the actual file format.

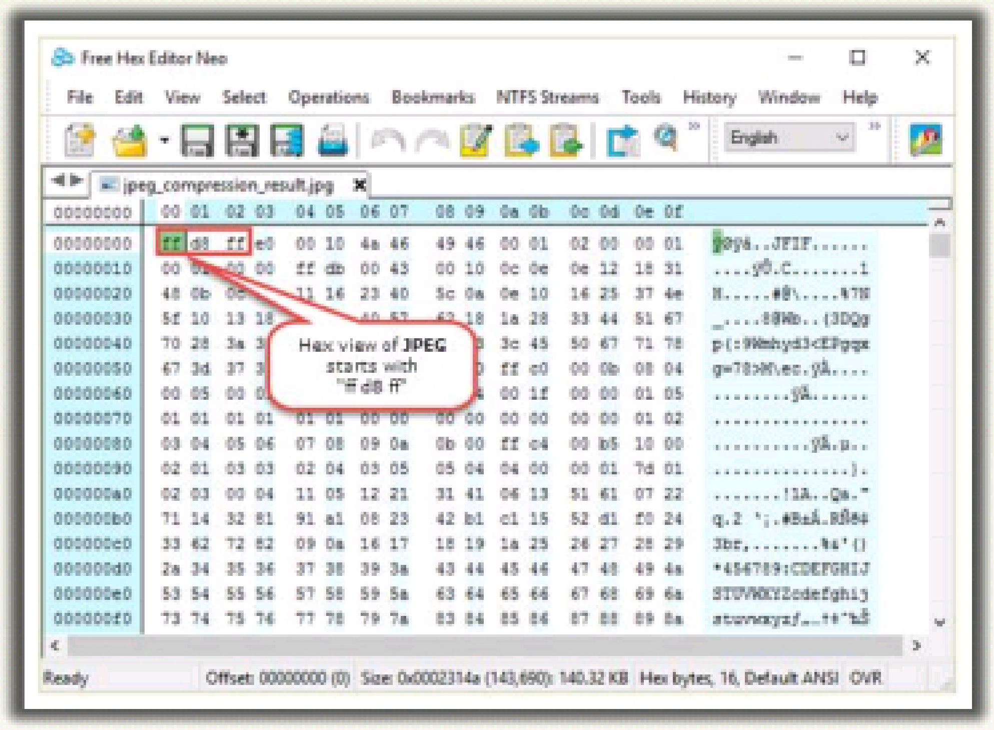
File carving methods may vary based on different elements such as the fragments of data present, deletion technique used, type of storage media, etc. This process depends on the information about the format of the existing files of interest and guesses of the file information layout on other devices. Investigators can take a look at file headers to verify the file format using tools such as 010 Editor, CI Hex Viewer, Hexinator, Hex Editor Neo, Qiew, WinHex, etc.

Image File Formats: JPEG



https://developer.apple.com

- The JPEG is a commonly used method to compress photographic images
- It uses a compression algorithm to minimize the size of the file without affecting the quality of the image



```
00000000  00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
00000000  ff d8 ff e0  00 10 4a 46  49 46 00 01  02 00 00 01  ÿØÿà..JFIF......
00000010  00 [illegible] [illegible] 00  ff db 00 43  00 10 0c 0e  0e 12 18 31  ....ÿÛ.C.......1
00000020  48 0b [illegible] [illegible]  11 16 23 40  5c 0a 0e 10  16 25 37 4e  H.....#@\....%7N
00000030  5f 10 13 18  [illegible]  62 18 1a 28  33 44 51 67  _....@gb..(3DQg
00000040  70 28 3a [illegible]  [illegible]  [illegible] 3c 45  50 67 71 78  p(:9Wmhyd3<EPgqx
00000050  67 3d 37 [illegible]  [illegible]  [illegible] ff c0  00 0b 08 04  g=78>M\ec.ÿÀ....
00000060  00 05 00 [illegible]  [illegible]  [illegible] 00 1f  00 00 01 05  ........ÿÄ......
00000070  01 01 01 01  01 01 00 00  00 00 00 00  00 00 01 02  ................
00000080  03 04 05 06  07 08 09 0a  0b 00 ff c4  00 b5 10 00  ..........ÿÄ.µ..
00000090  02 01 03 03  02 04 03 05  05 04 04 00  00 01 7d 01  ..............}.
000000a0  02 03 00 04  11 05 12 21  31 41 06 13  51 61 07 22  .......!1A..Qa."
000000b0  71 14 32 81  91 a1 08 23  42 b1 c1 15  52 d1 f0 24  q.2 ';.#BtÁ.RÑð$
000000c0  33 62 72 82  09 0a 16 17  18 19 1a 25  26 27 28 29  3br,.......%&'()
000000d0  2a 34 35 36  37 38 39 3a  43 44 45 46  47 48 49 4a  *456789:CDEFGHIJ
000000e0  53 54 55 56  57 58 59 5a  63 64 65 66  67 68 69 6a  STUVWXYZcdefghij
000000f0  73 74 75 76  77 78 79 7a  83 84 85 86  87 88 89 8a  stuvwxyz/...†‡ˆŠ
```

Ready | Offset: 00000000 (0) | Size: 0x0002314a (143,690): 140.32 KB | Hex bytes, 16, Default ANSI | OVR

Hex View of JPEG File Format

JPEG is an abbreviation for Joint Photographic Experts Group, the committee that created the JPEG standard and JPED is the term used for representing any graphic image file produced by using a JPEG standard.

It is a method of lossy compression for digital images and allows users to adjust the degree of compression, which has selectable impact on the storage size and image quality. JPEG files allow compression ratio of 90%, which is one-tenth of the size of the data.

File Structure

A JPEG bit stream contains a sequence of data chunks. Every chunk starts with the marker value, each marker having a 16-bit integer value, and it is stored in big endian byte format. The most significant bit of marker is set to `0xff`. The lower byte of the marker determines the type of marker.

The first bits of a file represent the file type and JPEG files start with binary value 0xffd8 (SOI—start of image) and end with binary value 0xffd9 (EOI—end of image). Therefore, ffd8 (the 0x is implied) at the beginning represents a JPEG file when viewed with a hex editor. A JPEG bit-stream contains a sequence of data chunks, or segments, and every chunk starts with a marker value. The basic format of a segment is the 16-bit integer value that determines the file size value. The most significant byte of the marker (the left-most bit) is 0xff. The lower byte of the marker determines the type of marker.

The basic format of a segment is as follows: 0xff marker number (1 byte) data; size (2 bytes); and data (n bytes)

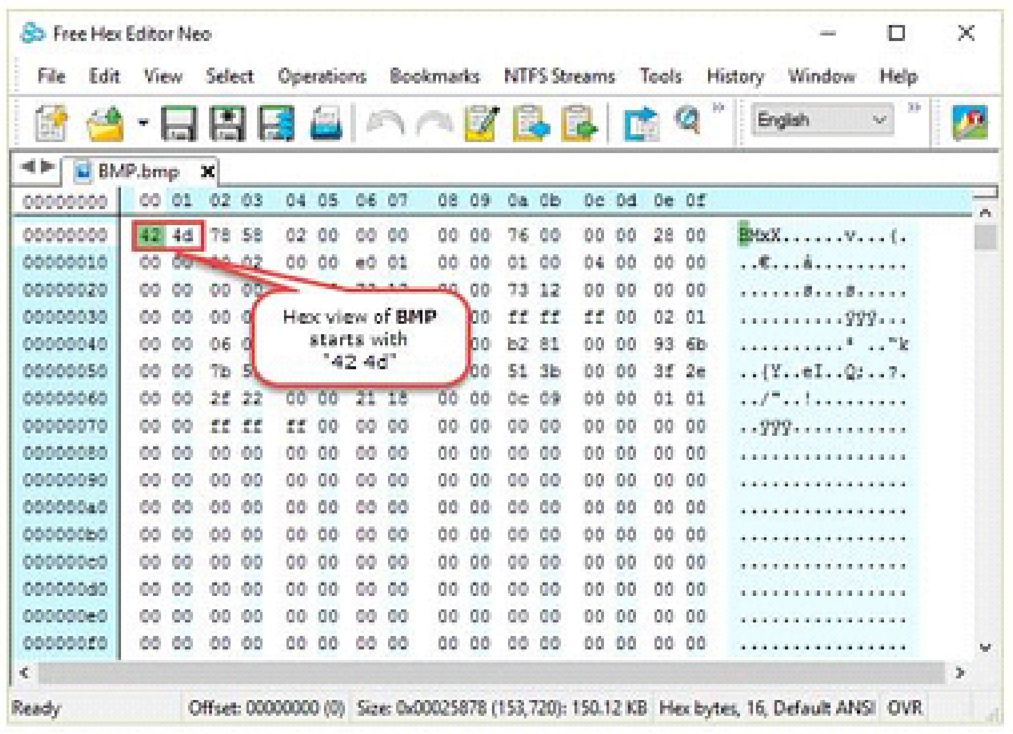
For example, for the marker FF E1 00 0E, the marker (0xFFE1) has 0x000E (which equals 14) bytes of data. But the data size 14 includes the data size descriptor (2 bytes); thus, only 12 bytes of data follow after 0x000E. Figure 5-3 shows a JPEG file structure. Figure 5-4 shows possible kinds of segment markers in JPEGs.

Image File Formats: BMP

- The Bitmap (BMP) is a standard file format for a Windows Device Independent Bitmap (DIB) file
- The size and color of these images can vary from 1 bit per pixel (black and white) to 24-bit color (16.7 million colors)

Basic BMP File Format

Name			Size
Header			14 bytes
	Signature		2 bytes
	File Size		4 bytes
	reserved		4 bytes
	DataOffset		4 bytes
Color Table			4 * NumColors bytes
		Red	1 byte
		Green	1 byte
		Blue	1 byte
		reserved	1 byte
	Repeated NumColors times		
Raster Data			Info ImageSize bytes


Free Hex Editor Neo
File Edit View Select Operations Bookmarks NTFS Streams Tools History Window Help
English
BMP.bmp
Hex view of BMP starts with "42 4d"

```
00000000  00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
00000000  42 4d 78 58  02 00 00 00  00 00 76 00  00 00 28 00   BMxX......v...(.
00000010  00 00 [illegible] 02  00 00 e0 01  00 00 01 00  04 00 00 00   ..€...à.........
00000020  00 00 00 00  [illegible]  00 00 73 12  00 00 00 00   .......8...8.....
00000030  00 00 00 [illegible]  [illegible]  00 ff ff  ff 00 02 01   ...........ÿÿÿ...
00000040  00 00 06 [illegible]  [illegible]  00 b2 81  00 00 93 6b   ..........² ..“k
00000050  00 00 7b 5[illegible]  [illegible]  00 51 3b  00 00 3f 2e   ..{Y..eI..Q;..?.
00000060  00 00 2f 22  00 00 21 18  00 00 0c 09  00 00 01 01   ../"..!.........
00000070  00 00 ff ff  ff 00 00 00  00 00 00 00  00 00 00 00   ..ÿÿÿ...........
00000080  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
00000090  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
000000a0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
000000b0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
000000c0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
000000d0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
000000e0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
000000f0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
Ready | Offset: 00000000 (0) | Size: 0x00025878 (153,720): 150.12 KB | Hex bytes, 16, Default ANSI | OVR
```

Hex View of BMP File Format

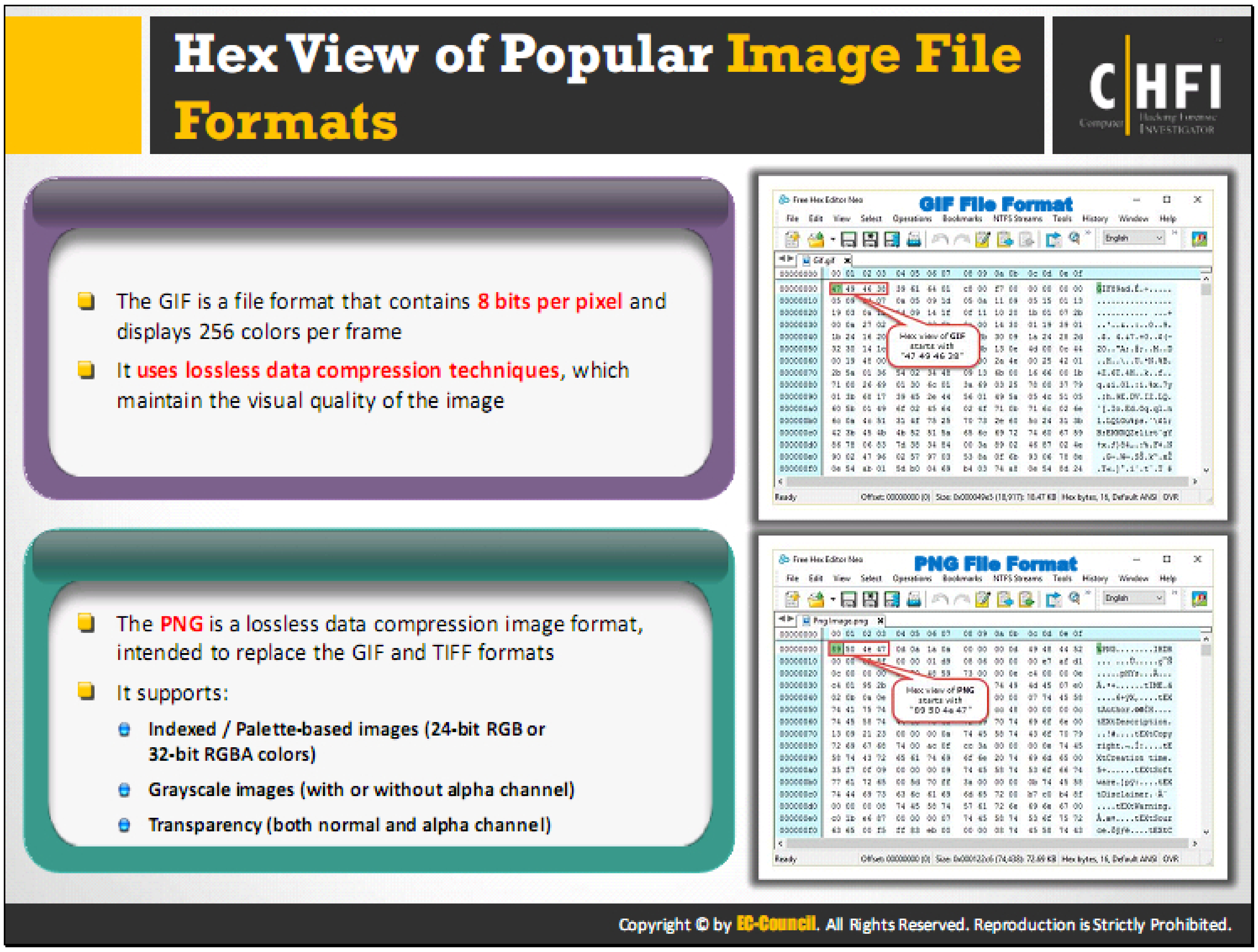
BMP file format, also called as bitmap image file or device independent bitmap (DIB) file format or a bitmap, is a standard graphics image file format used to store images on Windows operating systems. Microsoft developed this format so that Windows can display the image on any type of screens. Bitmap images can include animations. The size and color of these images can vary from 1 bit per pixel (black and white) to 24-bit color (16.7 million colors).

File Structure

BMP File Structure: Every bitmap file contains the following data structure:

- **File header:** The first part of the header that includes the data about the type, size, and layout of a file.
- **Information header:** A header component that contains the dimensions, compression type, and color format for the bitmap.
- **The RGBQUAD array:** A color table that comprises the array of elements equal to the colors present in the bitmap; this color table does not support bitmaps with 24 color bits, as each pixel is represented by24-bit RGB values in the actual bitmap.
- **Image data:** The array of bytes that contains bitmap image data; image data comprises color and shading information for each pixel.

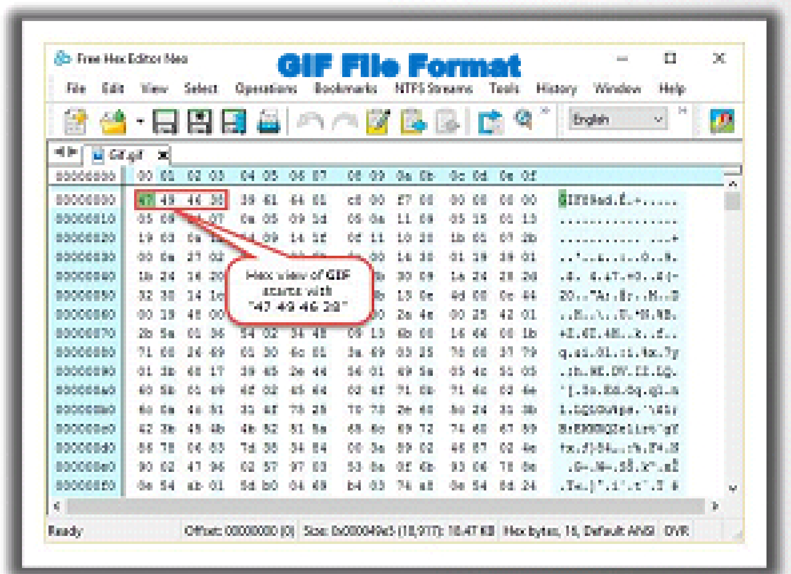
A bitmap file always has 42 4D as the first characters in a hexadecimal representation. These characters translate to BM in the ASCII code.

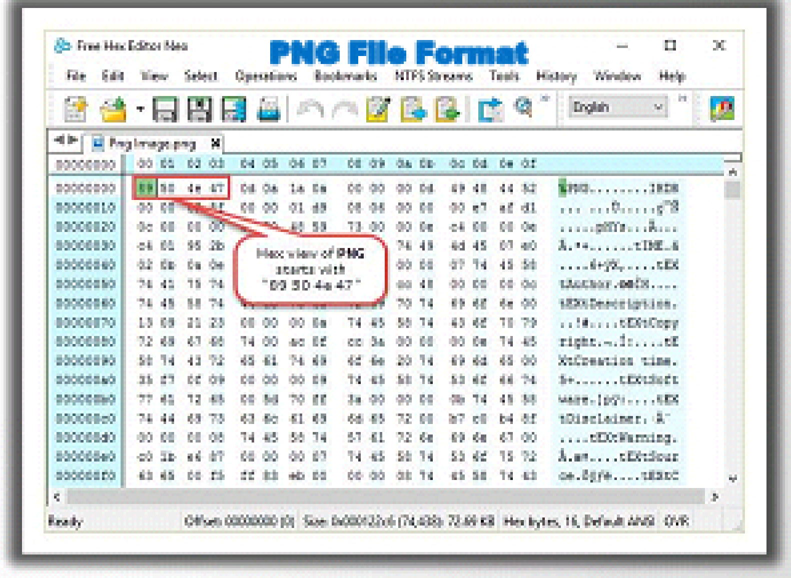


Hex View of Popular Image File Formats

- The GIF is a file format that contains **8 bits per pixel** and displays 256 colors per frame
- It **uses lossless data compression techniques**, which maintain the visual quality of the image

- The **PNG** is a lossless data compression image format, intended to replace the GIF and TIFF formats
- It supports:
  - Indexed / Palette-based images (24-bit RGB or 32-bit RGBA colors)
  - Grayscale images (with or without alpha channel)
  - Transparency (both normal and alpha channel)


Free Hex Editor Neo
GIF File Format
Gif.gif
Hex view of GIF starts with "47 49 46 38"
Ready
Offset: 00000000 (0) Size: 0x000049e5 (18,917); 18.47 KB Hex bytes, 16, Default ANSI OVR


Free Hex Editor Neo
PNG File Format
PngImage.png
Hex view of PNG starts with "89 50 4e 47"
Ready
Offset: 00000000 (0) Size: 0x000122c6 (74,438); 72.69 KB Hex bytes, 16, Default ANSI OVR

GIF File Format

GIF is a file format that contains 8 bits per pixel and displays 256 colors per frame. CompuServe generated the GIF format in 1987. GIF uses lossless data compression techniques, which maintain the visual quality of the image.

There are currently two versions of GIF:

- **GIF 87a:** Developed in 1987; it is the first version of GIF that uses the LZW file compression technique and supports features such as interlacing, 256-color palettes, and multiple image storage.
- **GIF 89a:** Created in 1989; this version supports features like background transparency, delay times, and image replacement parameters. These features are useful for storing multiple images as animations.

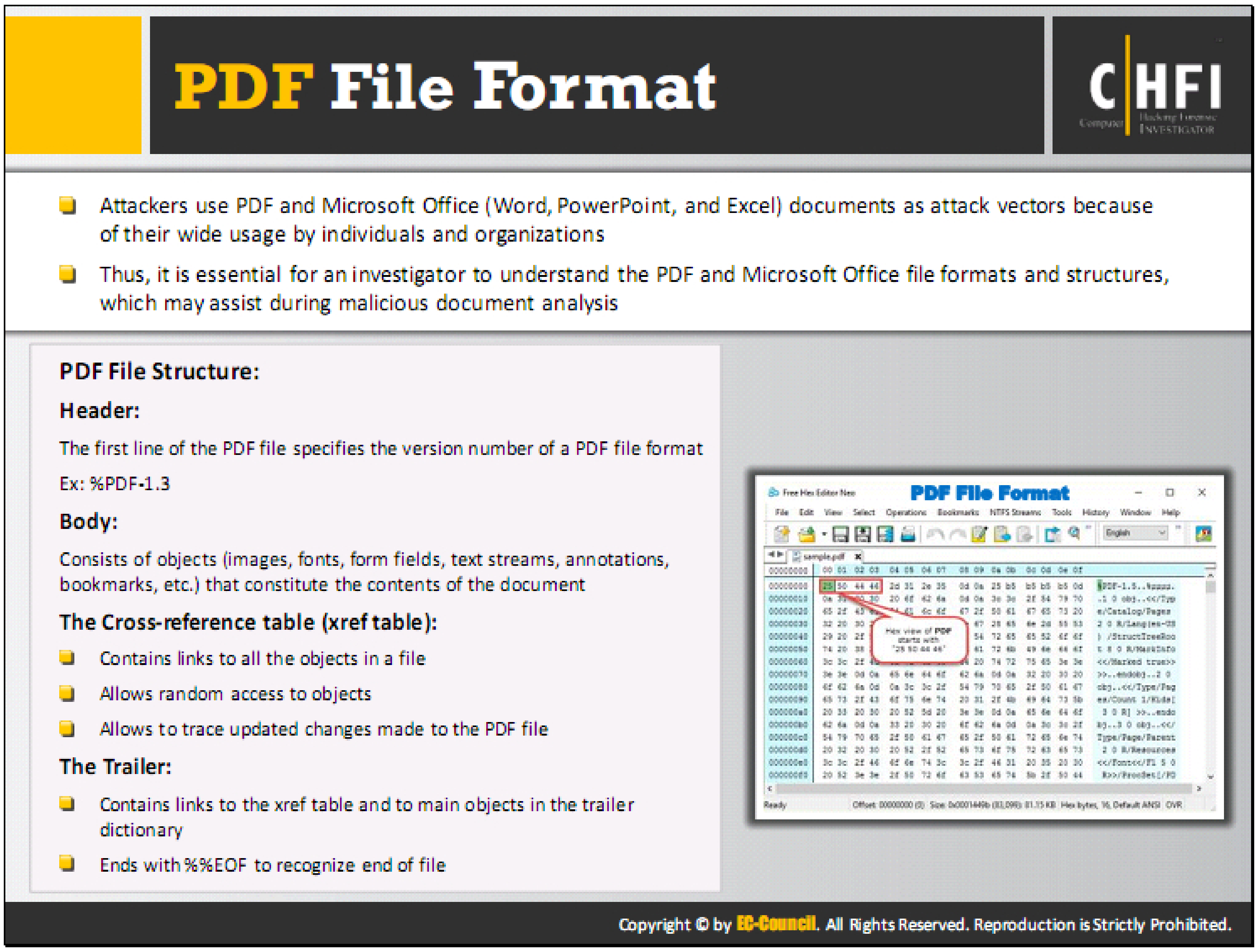
GIF file structure includes a header, image data, optional metadata, and footer. The hex value of a GIF image file starts with the values 47 49 46, which represent the GIF file name.

PNG File Format

PNG, short for Portable Network Graphics, is a lossless image format intended to replace the GIF and TIFF formats. PNG improves the GIF file format and replaces it with the image file format. It is copyright and license free. PNG file format supports 24-bit true color, transparency in both the normal and alpha channels as well as indexed/palette-based images of 24-bit RGB or 32-bit RGBA colors and grayscale images.

PNG file signature consists of the reminder of the file having single PNG image. These images are comprised of a series of chunks, starting with an IHDR chunk and ending with an IEND chunk.

PNG file hex values begin with 89 50 4e, which is the hex value for GIF.



PDF File Format

- Attackers use PDF and Microsoft Office (Word, PowerPoint, and Excel) documents as attack vectors because of their wide usage by individuals and organizations
- Thus, it is essential for an investigator to understand the PDF and Microsoft Office file formats and structures, which may assist during malicious document analysis

PDF File Structure:

Header:

The first line of the PDF file specifies the version number of a PDF file format

Ex: %PDF-1.3

Body:

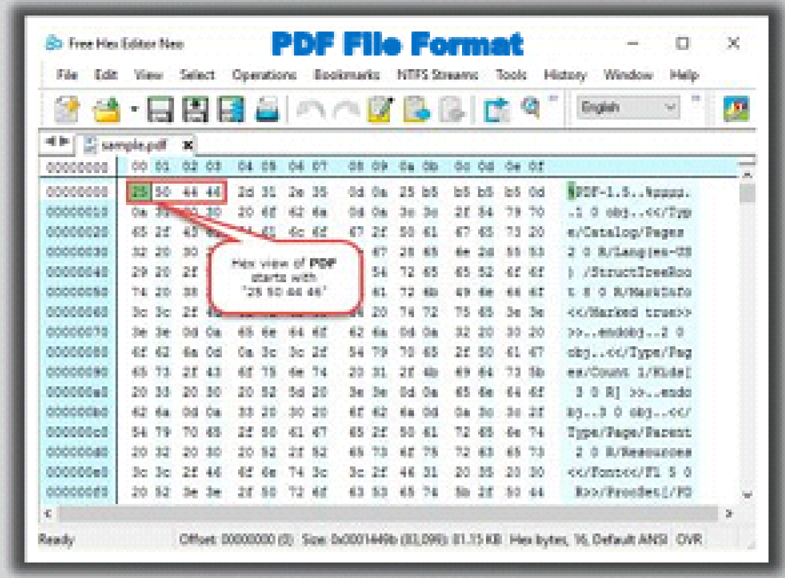
Consists of objects (images, fonts, form fields, text streams, annotations, bookmarks, etc.) that constitute the contents of the document

The Cross-reference table (xref table):

- Contains links to all the objects in a file
- Allows random access to objects
- Allows to trace updated changes made to the PDF file

The Trailer:

- Contains links to the xref table and to main objects in the trailer dictionary
- Ends with %%EOF to recognize end of file


Free Hex Editor Neo
PDF File Format
File Edit View Select Operations Bookmarks NTFS Streams Tools History Window Help
English
sample.pdf
Hex view of PDF starts with '25 50 44 46'

```
00000000  00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
00000000  25 50 44 46  2d 31 2e 35  0d 0a 25 b5  b5 b5 b5 0d   %PDF-1.5..%µµµµ.
00000010  0a 31 20 30  20 6f 62 6a  0d 0a 3c 3c  2f 54 79 70   .1 0 obj..<</Typ
00000020  65 2f 43 61  74 61 6c 6f  67 2f 50 61  67 65 73 20   e/Catalog/Pages
00000030  32 20 30 20  52 2f 4c 61  6e 67 28 65  6e 2d 55 53   2 0 R/Lang(en-US
00000040  29 20 2f 53  74 72 75 63  74 54 72 65  65 52 6f 6f   ) /StructTreeRoo
00000050  74 20 38 20  30 20 52 2f  4d 61 72 6b  49 6e 66 6f   t 8 0 R/MarkInfo
00000060  3c 3c 2f 4d  61 72 6b 65  64 20 74 72  75 65 3e 3e   <</Marked true>>
00000070  3e 3e 0d 0a  65 6e 64 6f  62 6a 0d 0a  32 20 30 20   >>..endobj..2 0
00000080  6f 62 6a 0d  0a 3c 3c 2f  54 79 70 65  2f 50 61 67   obj..<</Type/Pag
00000090  65 73 2f 43  6f 75 6e 74  20 31 2f 4b  69 64 73 5b   es/Count 1/Kids[
000000a0  20 33 20 30  20 52 5d 20  3e 3e 0d 0a  65 6e 64 6f    3 0 R] >>..endo
000000b0  62 6a 0d 0a  33 20 30 20  6f 62 6a 0d  0a 3c 3c 2f   bj..3 0 obj..<</
000000c0  54 79 70 65  2f 50 61 67  65 2f 50 61  72 65 6e 74   Type/Page/Parent
000000d0  20 32 20 30  20 52 2f 52  65 73 6f 75  72 63 65 73    2 0 R/Resources
000000e0  3c 3c 2f 46  6f 6e 74 3c  3c 2f 46 31  20 35 20 30   <</Font<</F1 5 0
000000f0  20 52 3e 3e  2f 50 72 6f  63 53 65 74  5b 2f 50 44    R>>/ProcSet[/PD

Ready    Offset: 00000000 (0)  Size: 0x0001449b (83,099): 81.15 KB  Hex bytes, 16, Default ANSI  OVR
```

Adobe Inc. developed the Portable Document Format (PDF) file format in 1992. It helps users to easily view, save, and print a document independent of any platform, operating system, hardware or any software, which they are using. It consists of text with media components like images, links etc. An Acrobat reader can open and display the PDF files.

Advantages

- PDF files are device independent and support different systems like MAC, Linux, etc.
- These files support different compression algorithms
- They also support several multimedia elements
- It allows password protection

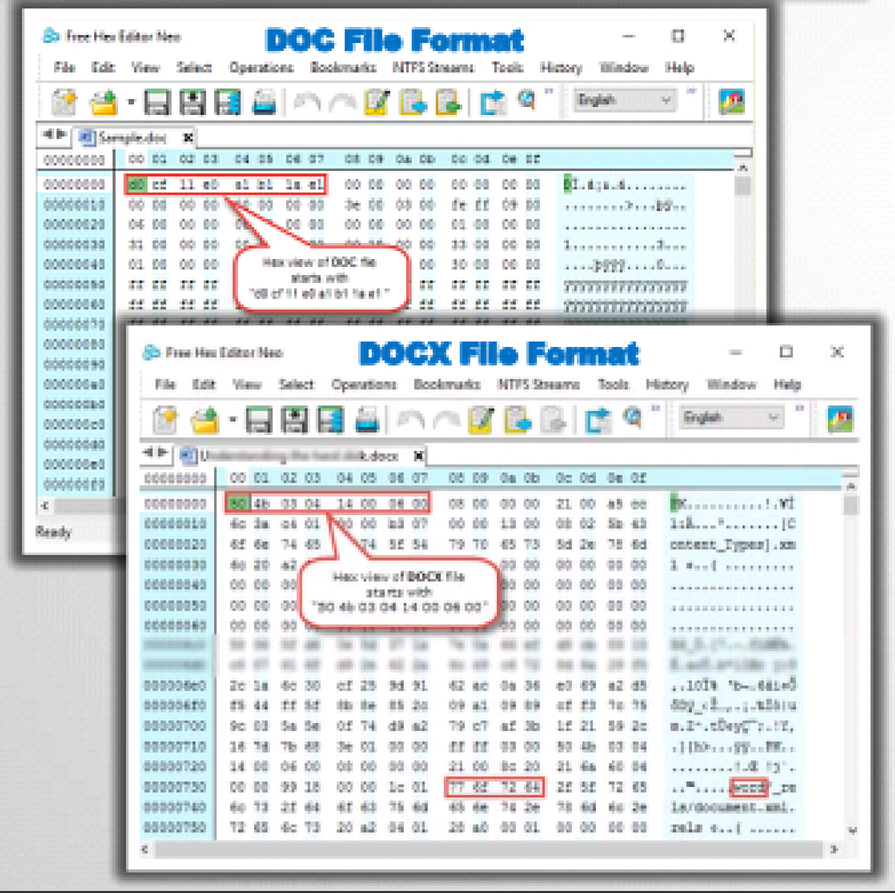
Hex values for a PDF begin with 25 50 44 46, which is the signature of every PDF file representing the %PDF values in hexadecimal form. The file version of PDF follows the signature while the file ends with %EOF value, representing the end of file.

Word File Formats

MS Office Documents – File Format:

Binary File Format

- Usually, attackers target MS Office documents of the binary format because they are still in use and the file structures are highly complex


DOC File Format
Hex view of DOC file starts with "d0 cf 11 e0 a1 b1 1a e1"
DOCX File Format
Hex view of DOCX file starts with "50 4b 03 04 14 00 06 00"

Microsoft Word File Structure (.doc/.docx):

WordDocument Stream/Main Stream

- Contains binary data of the Word document and Word file header (known as the File Information Block (FIB)) located at the offset 0
- FIB contains information about the document, file length, and specifies pointers to elements in the document file

Summary Information Streams

- The summary information is stored in two storage streams: Summary Information and DocumentSummaryInformation

Table Stream (0Table or 1Table)

- Contains data referenced from the FIB and other parts of the file
- stores various plex of character positions (PLCs) and tables defining document's structure
- Has predefined structure only for encrypted files

Data Stream (Optional)

- No predefined structure
- Contains data referenced from the FIB in the mainstream or other parts of the file

Object Streams

- Holds binary data for embedded OLE objects within the .doc file

PowerPoint File Formats

Microsoft PowerPoint Presentation File Structure (.ppt/.pptx):

Current User Stream

- Maintains CurrentUserAtom record, that identifies the last user's name to open/modify a target PPT and location of the most recent user edit

PowerPoint Document Stream

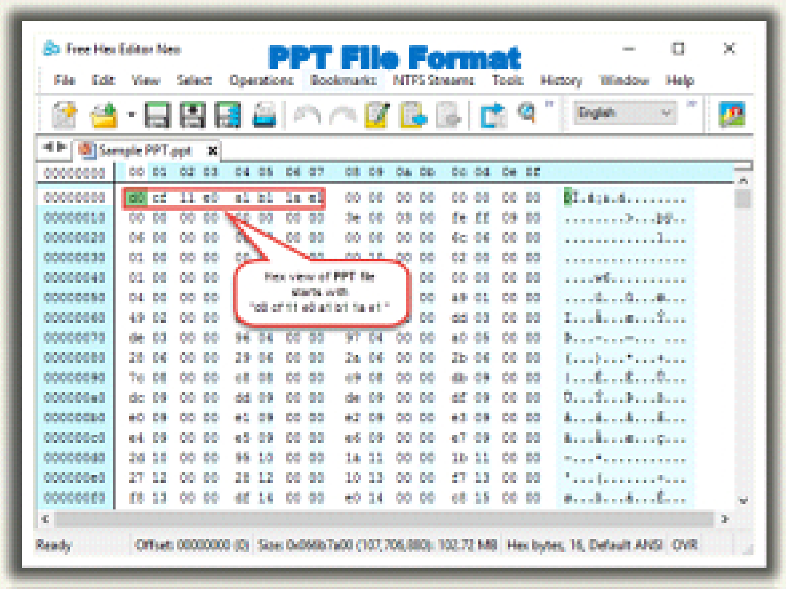
- Contains information about the presentation layout and its contents

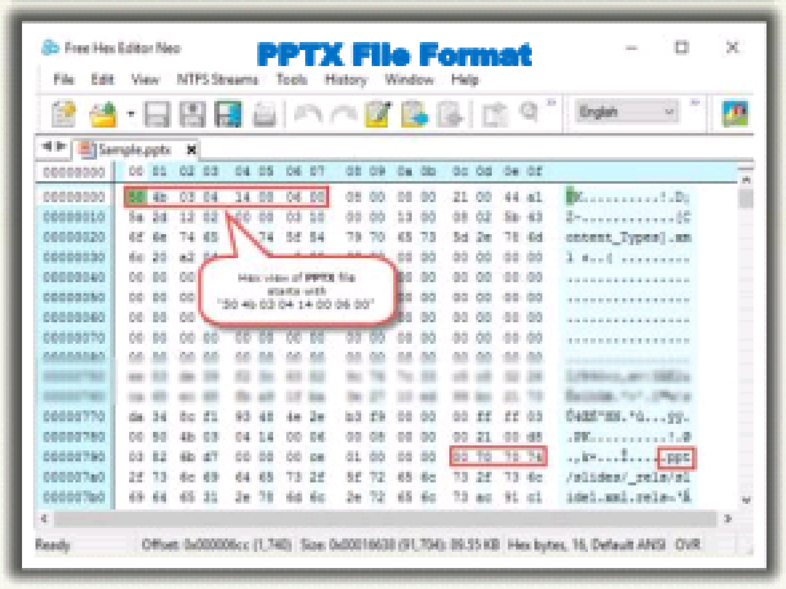
Pictures Stream (Optional)

- Contains information about embedded image files within the presentation

Summary Information Streams (Optional)

- The summary information is stored in two storage streams: SummaryInformation and DocumentSummaryInformation





Excel File Formats

Microsoft Excel File Structure (.xls/.xlsx):

An OLE compound file saved in Binary Interchange File Format (BIFF)

Streams

- Workbook stream is the primary stream in .xls file, which contains many substreams

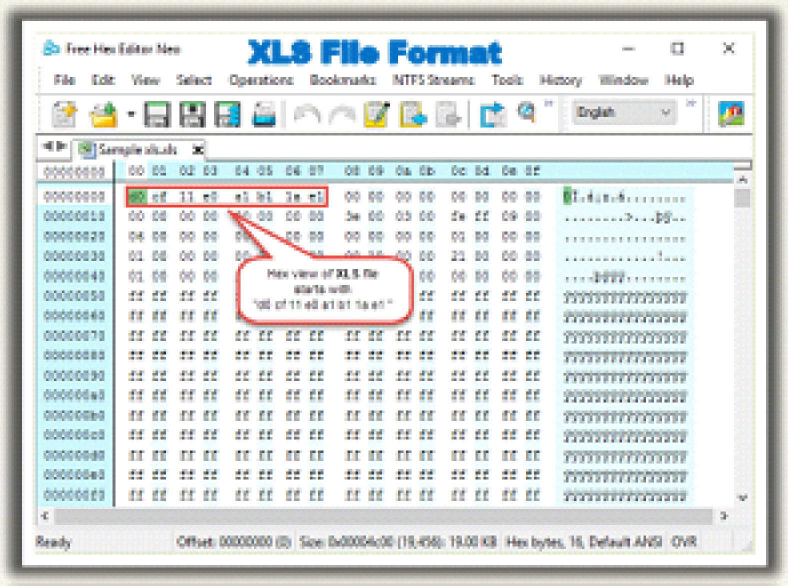
Substreams

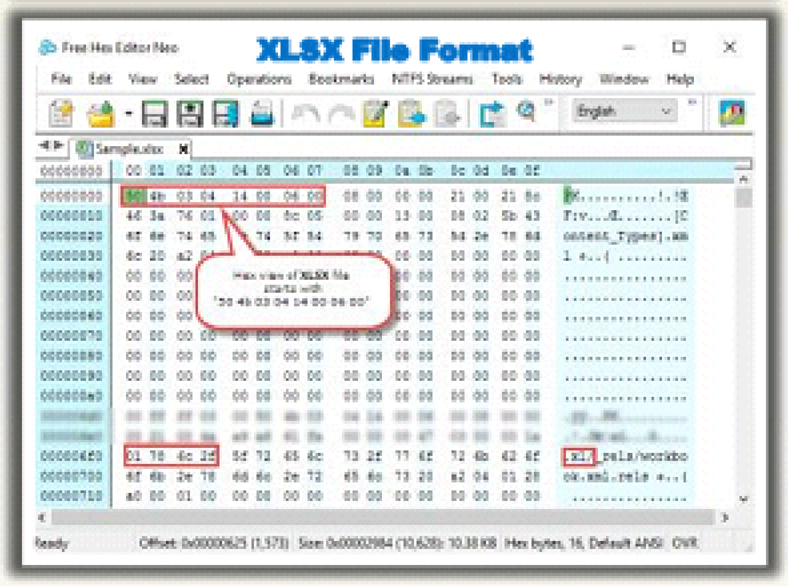
- Global substream - specifies global properties and data in a workbook
- Worksheet substream - specifies a sheet in a workbook

Records

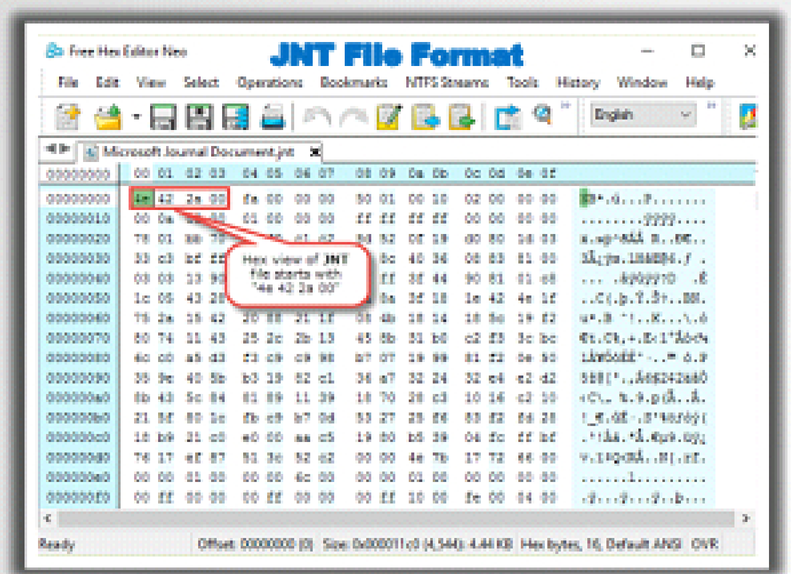
- Holds information about each workbook's features
- Components include record size, record type, and record data

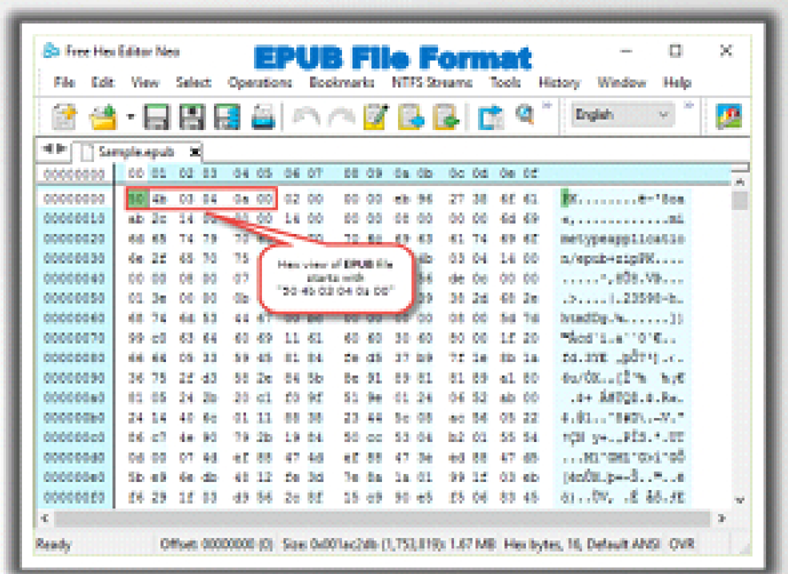
Note: Office Open XML Format (MS Office 2007 and above) is less vulnerable compared to the binary format and is therefore not widely used by attackers as a vector of attack

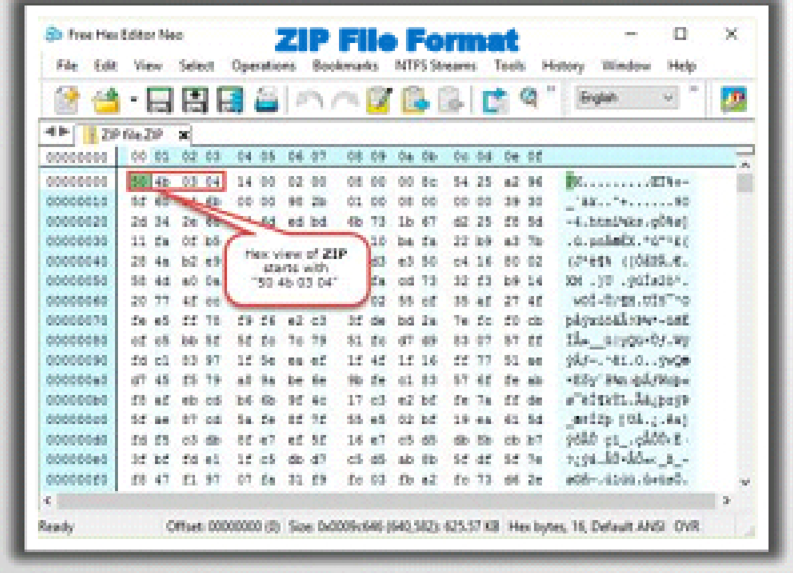


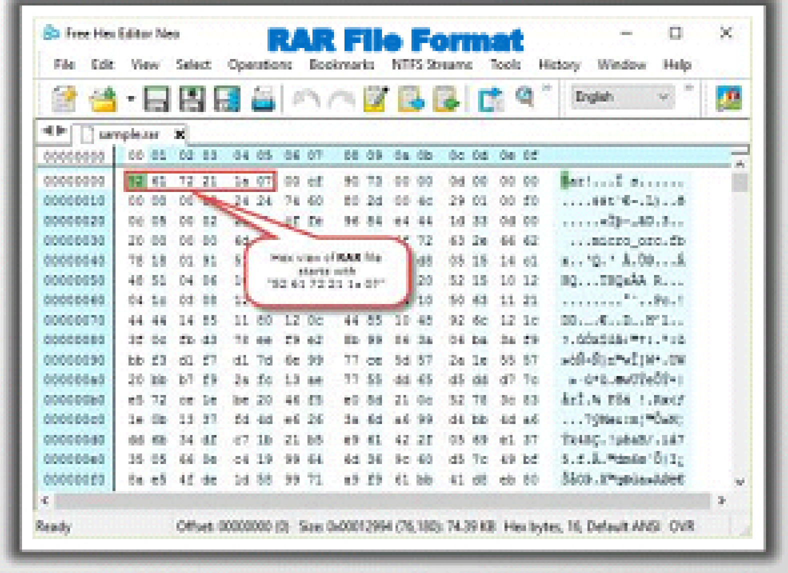


Hex View of Other Popular File Formats

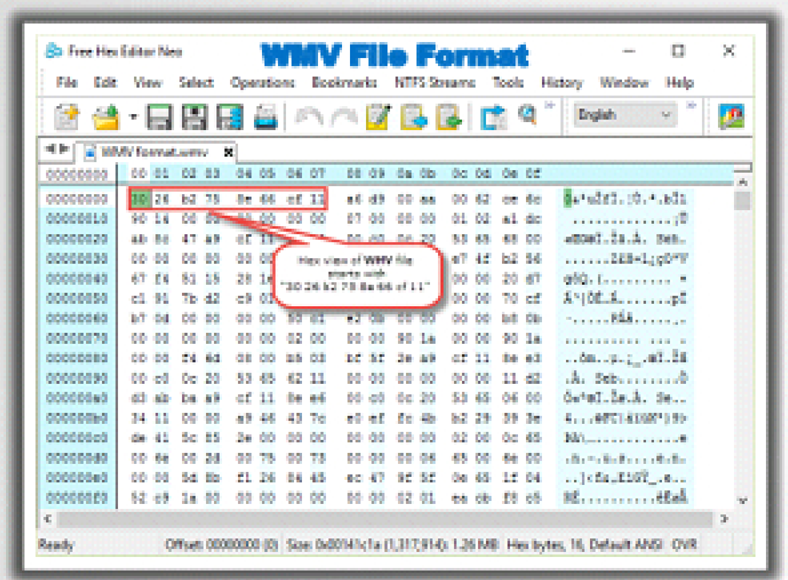


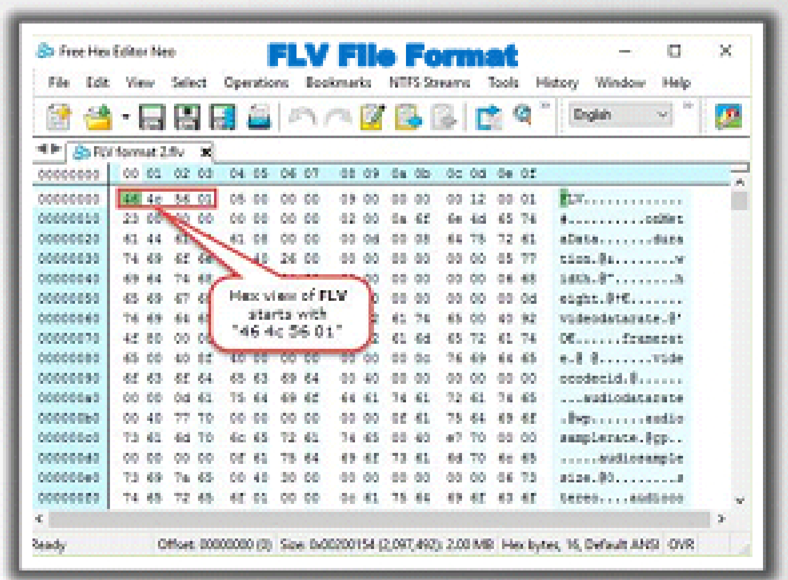


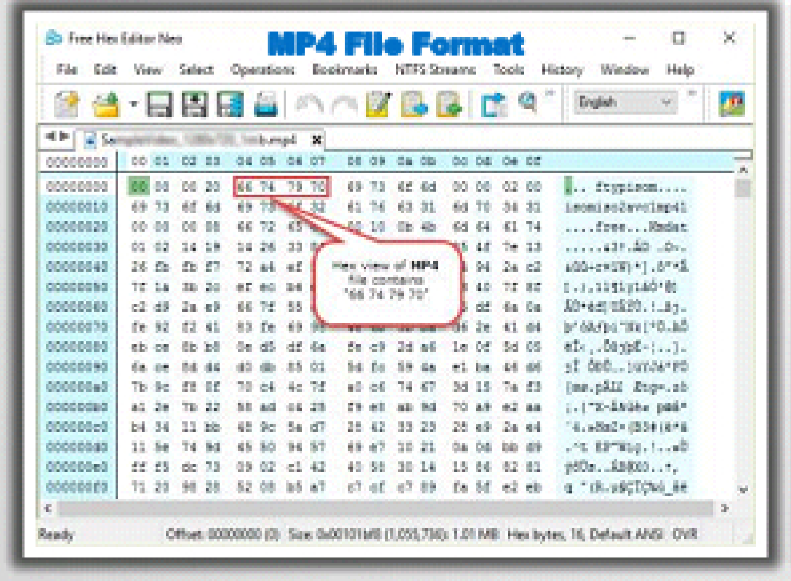


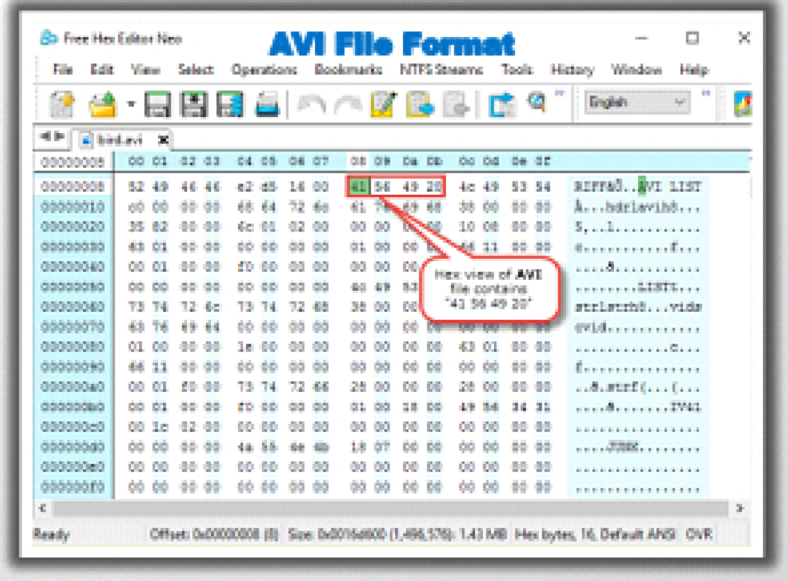


Hex View of Popular Video File Formats

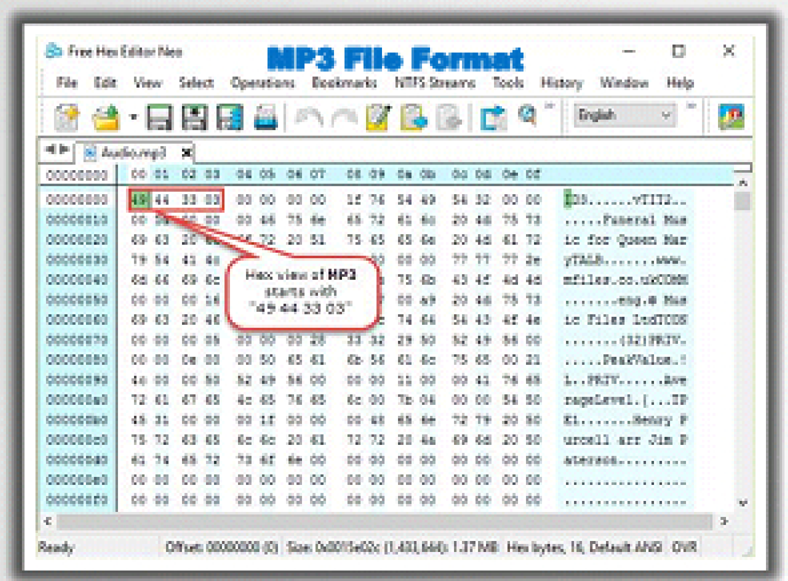


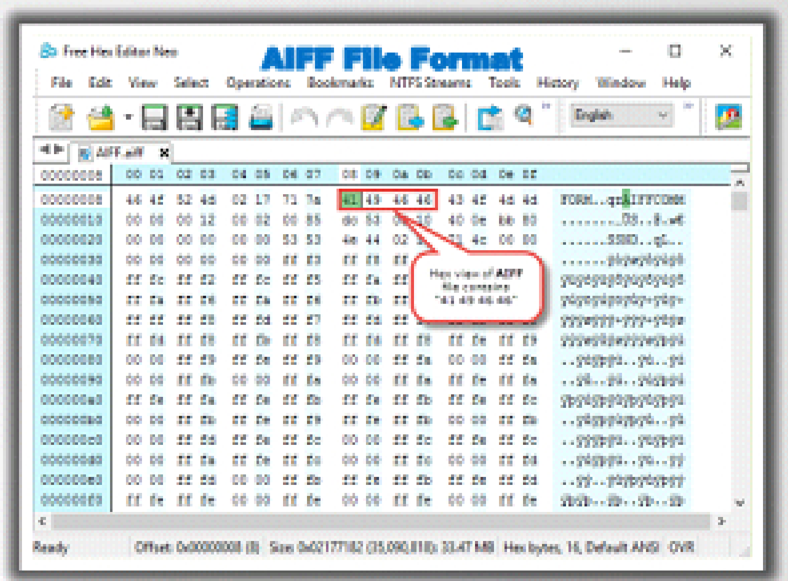


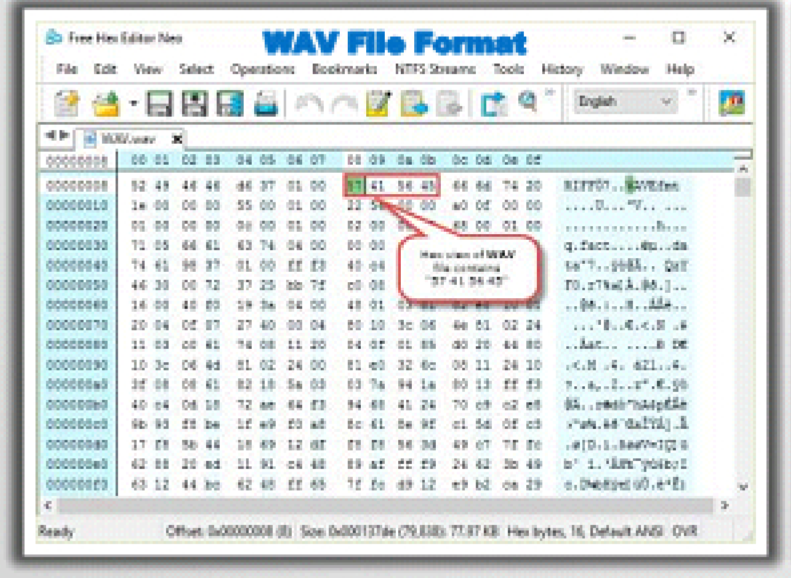


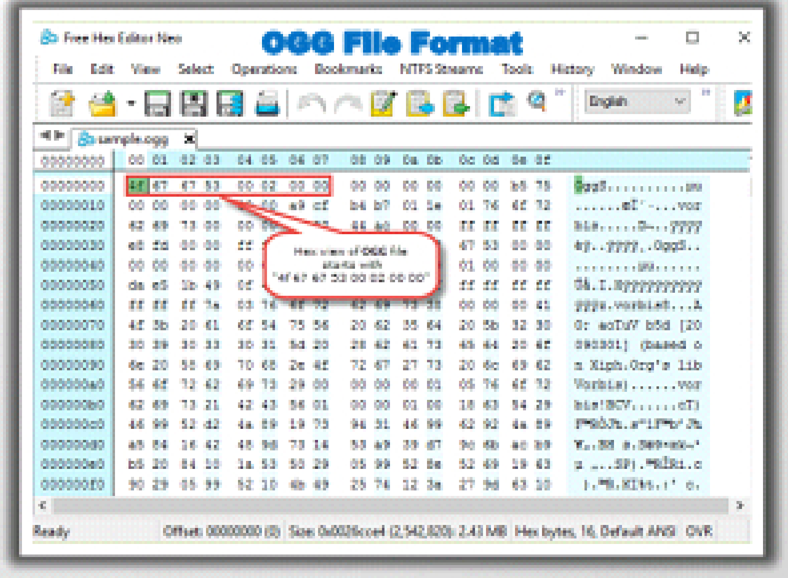


Hex View of Popular Audio File Formats



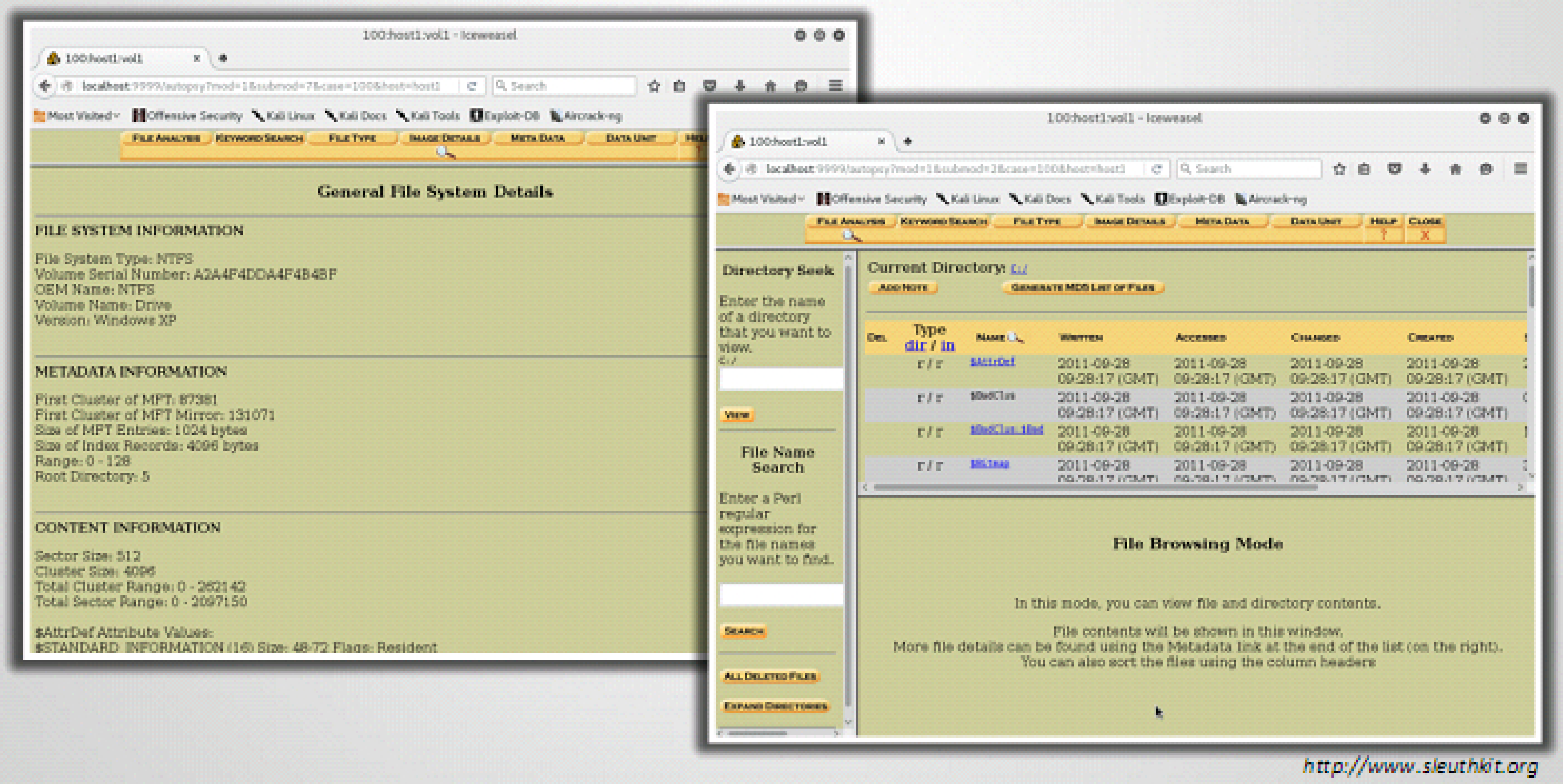






File System Analysis Using Autopsy

- Autopsy is a digital forensics platform and graphical interface to The Sleuth Kit and other digital forensics tools that can be used to investigate activities which happened on a computer



General File System Details

FILE SYSTEM INFORMATION

File System Type: NTFS
Volume Serial Number: A2A4F4DDA4F4B4BF
OEM Name: NTFS
Volume Name: Drive
Version: Windows XP

METADATA INFORMATION

First Cluster of MFT: 87381
First Cluster of MFT Mirror: 131071
Size of MFT Entries: 1024 bytes
Size of Index Records: 4096 bytes
Range: 0 - 128
Root Directory: 5

CONTENT INFORMATION

Sector Size: 512
Cluster Size: 4096
Total Cluster Range: 0 - 262142
Total Sector Range: 0 - 2097150

$AttrDef Attribute Values:
$STANDARD_INFORMATION (16) Size: 48-72 Flags: Resident

Directory Seek

Enter the name of a directory that you want to view.

File Name Search

Enter a Perl regular expression for the file names you want to find.

Current Directory: C:/

Del	Type dir / in	Name	Written	Accessed	Changed	Created
	r / r	$AttrDef	2011-09-28 09:28:17 (GMT)	2011-09-28 09:28:17 (GMT)	2011-09-28 09:28:17 (GMT)	2011-09-28 09:28:17 (GMT)
	r / r	$BadClus	2011-09-28 09:28:17 (GMT)	2011-09-28 09:28:17 (GMT)	2011-09-28 09:28:17 (GMT)	2011-09-28 09:28:17 (GMT)
	r / r	$BadClus:$Bad	2011-09-28 09:28:17 (GMT)	2011-09-28 09:28:17 (GMT)	2011-09-28 09:28:17 (GMT)	2011-09-28 09:28:17 (GMT)
	r / r	$Bitmap	2011-09-28	2011-09-28	2011-09-28	2011-09-28

File Browsing Mode

In this mode, you can view file and directory contents.

File contents will be shown in this window.
More file details can be found using the Metadata link at the end of the list (on the right).
You can also sort the files using the column headers

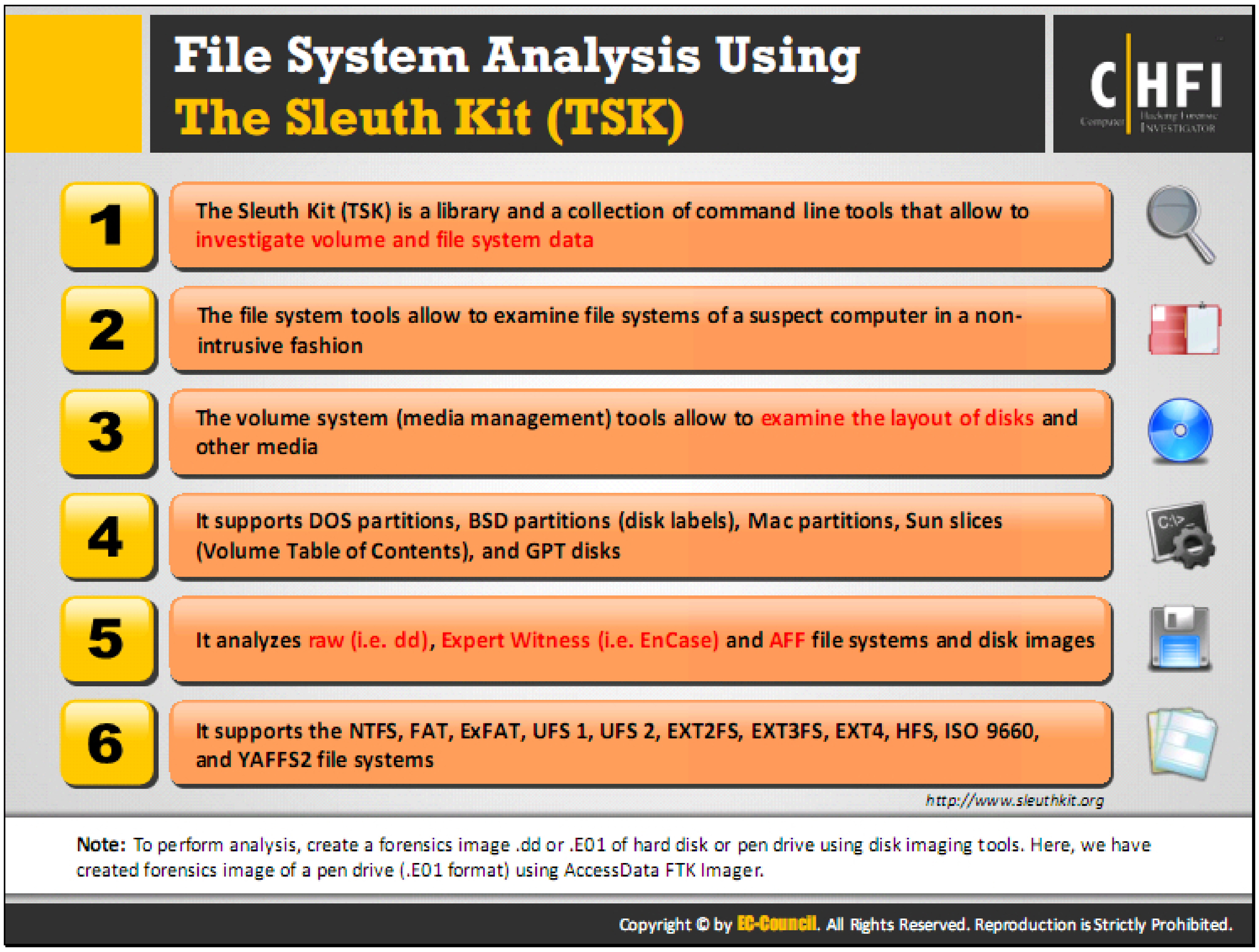
http://www.sleuthkit.org

Autopsy is a digital forensics platform and graphical interface to The Sleuth Kit® and other digital forensics tools. Law enforcement, military, and corporate examiners use it to investigate what happened on a computer. You can even use it to recover photos from your camera's memory card.

Autopsy is an end-to-end platform with modules that come with it out of the box and others that are available from third-parties. Some of the modules provide:

- Timeline Analysis - Advanced graphical event viewing interface (video tutorial included).
- Hash Filtering - Flag known bad files and ignore known good.
- Keyword Search - Indexed keyword search to find files that mention relevant terms.
- Web Artifacts - Extract history, bookmarks, cookies from Firefox, Chrome, and IE.
- Data Carving - Recover deleted files from unallocated space using PhotoRec.
- Multimedia - Extract EXIF from pictures and watch videos.
- Indicators of Compromise - Scan a computer using STIX.

Source: http://www.sleuthkit.org



File System Analysis Using The Sleuth Kit (TSK)

1. The Sleuth Kit (TSK) is a library and a collection of command line tools that allow to investigate volume and file system data
2. The file system tools allow to examine file systems of a suspect computer in a non-intrusive fashion
3. The volume system (media management) tools allow to examine the layout of disks and other media
4. It supports DOS partitions, BSD partitions (disk labels), Mac partitions, Sun slices (Volume Table of Contents), and GPT disks
5. It analyzes raw (i.e. dd), Expert Witness (i.e. EnCase) and AFF file systems and disk images
6. It supports the NTFS, FAT, ExFAT, UFS 1, UFS 2, EXT2FS, EXT3FS, EXT4, HFS, ISO 9660, and YAFFS2 file systems

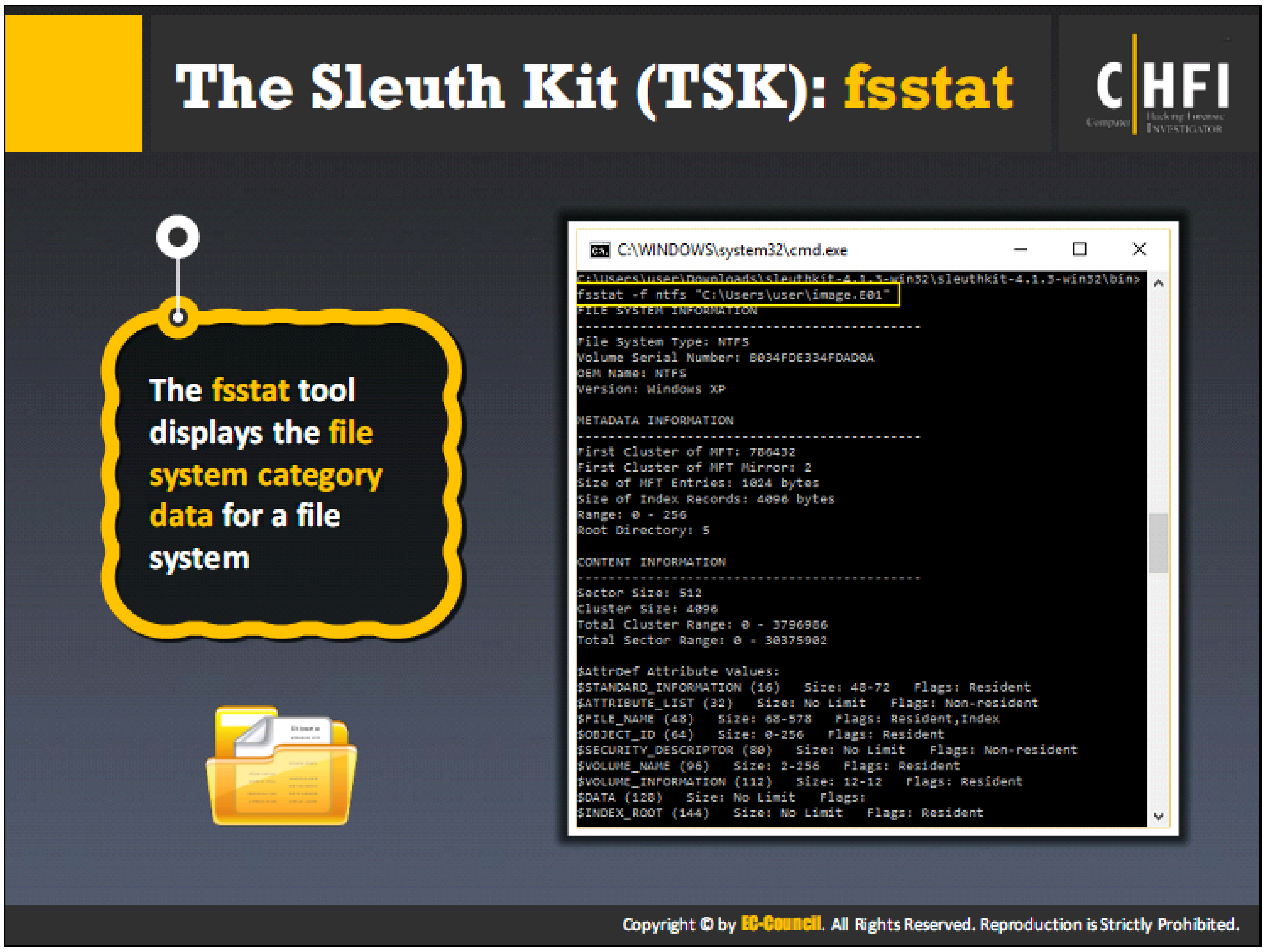
http://www.sleuthkit.org

Note: To perform analysis, create a forensics image .dd or .E01 of hard disk or pen drive using disk imaging tools. Here, we have created forensics image of a pen drive (.E01 format) using AccessData FTK Imager.

The Sleuth Kit® (TSK) is a library and collection of command line tools that allow you to investigate disk images. The core functionality of TSK allows you to analyze volume and file system data. The plug-in framework allows you to incorporate additional modules to analyze file contents and build automated systems. The library can be incorporated into larger digital forensics tools and the command line tools can be directly used to find evidence.

- Volume and File System Analysis
- Plug-in Framework
- Download
- Documents
- History
- Licenses

Source: http://sleuthkit.org



The Sleuth Kit (TSK): fsstat

The **fsstat** tool displays the **file system category data** for a file system

C:\WINDOWS\system32\cmd.exe

```
C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
fsstat -f ntfs "C:\Users\user\image.E01"
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: NTFS
Volume Serial Number: B034FDE334FDAD0A
OEM Name: NTFS
Version: Windows XP

METADATA INFORMATION
--------------------------------------------
First Cluster of MFT: 786432
First Cluster of MFT Mirror: 2
Size of MFT Entries: 1024 bytes
Size of Index Records: 4096 bytes
Range: 0 - 256
Root Directory: 5

CONTENT INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 4096
Total Cluster Range: 0 - 3796986
Total Sector Range: 0 - 30375902

$AttrDef Attribute Values:
$STANDARD_INFORMATION (16)   Size: 48-72   Flags: Resident
$ATTRIBUTE_LIST (32)   Size: No Limit   Flags: Non-resident
$FILE_NAME (48)   Size: 68-578   Flags: Resident,Index
$OBJECT_ID (64)   Size: 0-256   Flags: Resident
$SECURITY_DESCRIPTOR (80)   Size: No Limit   Flags: Non-resident
$VOLUME_NAME (96)   Size: 2-256   Flags: Resident
$VOLUME_INFORMATION (112)   Size: 12-12   Flags: Resident
$DATA (128)   Size: No Limit   Flags:
$INDEX_ROOT (144)   Size: No Limit   Flags: Resident
```

fsstat - Display general details of a file system

Syntax

```
fsstat [-f fstype ] [-i imgtype] [-o imgoffset] [-b dev_sector_size] [-tvV]
image [images]
```

Description

fsstat displays the details associated with a file system. The output of this command is file system specific. At a minimum, the range of meta-data values (inode numbers) and content units (blocks or clusters) are given. Also given are details from the Super Block, such as mount times and features. For file systems using groups (FFS and EXT2FS), the tool lists the layout of each group.

For a FAT file system, the FAT table is in a condensed format. Note that the data is in sectors and not in clusters.

Arguments

- -t type

  Print the file system type only.

- -f fstype

  Specify the file system type. Use '-f list' to list the supported file system types. If not given, autodetection methods are used.

- -i imgtype

  Identify the type of image file, such as raw. Use '-i list' to list the supported types. If not given, autodetection methods are used.

- -o imgoffset

  The sector offset where the file system starts in the image.

- -b dev_sector_size

  The size, in bytes, of the underlying device sectors. If not given, the value in the image format is used (if it exists) or 512-bytes is assumed.

- -v

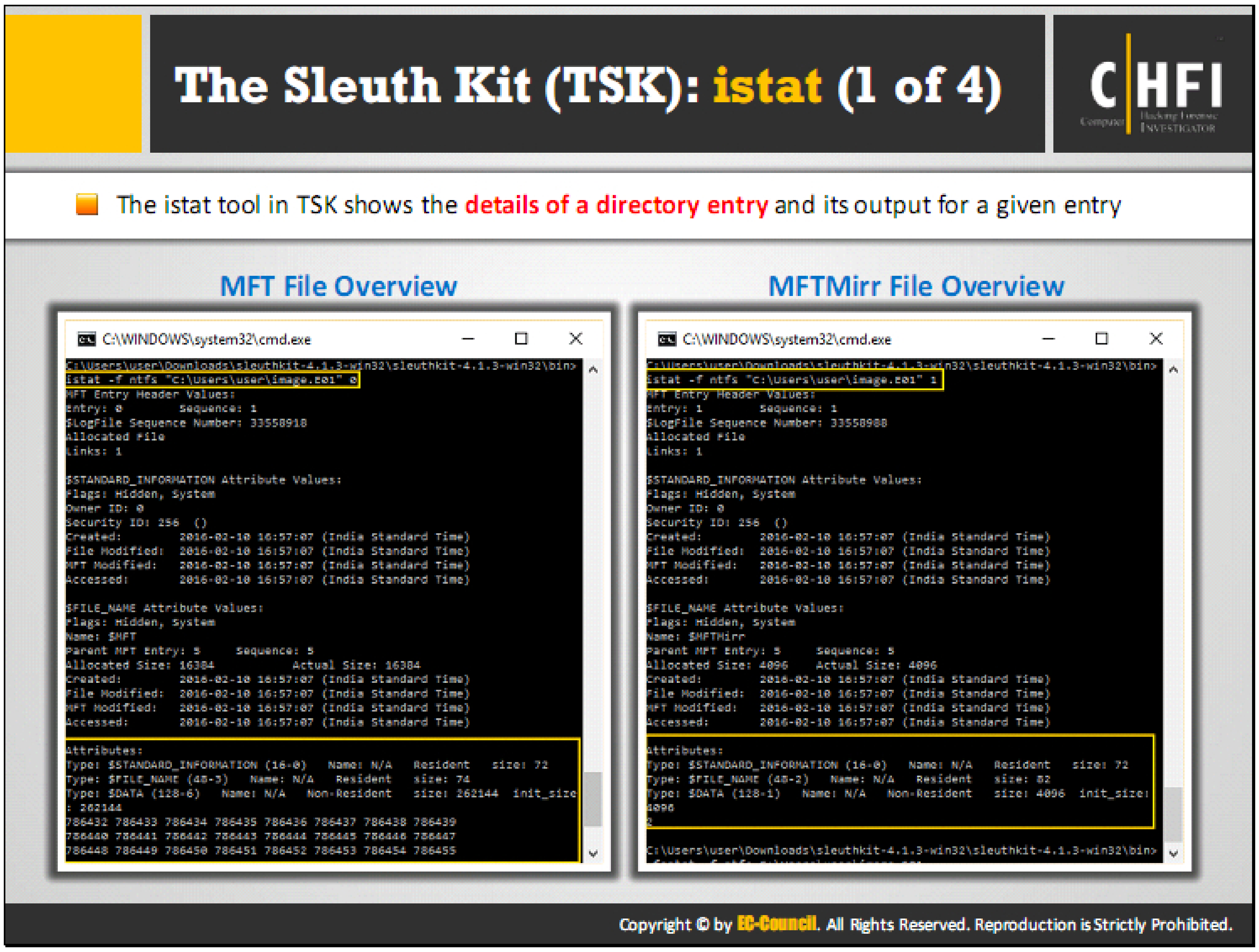
  Verbose output of debugging statements to stderr

- -V

  Display version

In the above image, the investigator uses fsstat command line tool from the Sleuth Kit to view details of an NTFS image named image.E01.

Source: *http://www.sleuthkit.org*



The Sleuth Kit (TSK): istat (1 of 4)

CHFI Computer Hacking Forensic INVESTIGATOR

- The istat tool in TSK shows the **details of a directory entry** and its output for a given entry

MFT File Overview

C:\WINDOWS\system32\cmd.exe

```
C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
istat -f ntfs "c:\Users\user\image.E01" 0
MFT Entry Header Values:
Entry: 0        Sequence: 1
$LogFile Sequence Number: 33558918
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Hidden, System
Owner ID: 0
Security ID: 256  ()
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

$FILE_NAME Attribute Values:
Flags: Hidden, System
Name: $MFT
Parent MFT Entry: 5     Sequence: 5
Allocated Size: 16384           Actual Size: 16384
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-3)   Name: N/A   Resident   size: 74
Type: $DATA (128-6)   Name: N/A   Non-Resident   size: 262144  init_size
: 262144
786432 786433 786434 786435 786436 786437 786438 786439
786440 786441 786442 786443 786444 786445 786446 786447
786448 786449 786450 786451 786452 786453 786454 786455
```

MFTMirr File Overview

C:\WINDOWS\system32\cmd.exe

```
C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
istat -f ntfs "c:\Users\user\image.E01" 1
MFT Entry Header Values:
Entry: 1        Sequence: 1
$LogFile Sequence Number: 33558988
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Hidden, System
Owner ID: 0
Security ID: 256  ()
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

$FILE_NAME Attribute Values:
Flags: Hidden, System
Name: $MFTMirr
Parent MFT Entry: 5     Sequence: 5
Allocated Size: 4096    Actual Size: 4096
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-2)   Name: N/A   Resident   size: 82
Type: $DATA (128-1)   Name: N/A   Non-Resident   size: 4096  init_size:
4096
2

C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
```

istat - Display details of a meta-data structure (i.e. inode)

Syntax

```
istat [-B num ] [-f fstype ] [-i imgtype] [-o imgoffset] [-b dev_sector_size]
[-vV] [-z zone ] [-s seconds ] image [images] inode
```

Description

istat displays the uid, gid, mode, size, link number, modified, accessed, changed times, and all the disk units a structure has allocated.

The options are as follows:

- -B num: Display the addresses of num disk units. Useful when the inode is unallocated with size 0 but still has block pointers.
- -f fstype: Specify the file system type. Use '-f list' to list the supported file system types. If not given, autodetection methods are used.
- -s seconds: The time skew of the original system in seconds. For example, if the original system was 100 seconds slow, this value would be -100.
- -i imgtype: Identify the type of image file, such as raw. Use '-i list' to list the supported types. If not given, autodetection methods are used.
- -o imgoffset: The sector offset where the file system starts in the image.

- -b dev_sector_size: The size, in bytes, of the underlying device sectors. If not given, the value in the image format is used (if it exists) or 512-bytes is assumed.
- -v: Verbose output of debugging statements to stderr
- -V: Display version
- -z zone: An ASCII string of the original system's time zone. For example, EST5EDT or GMT. These strings are defined by the operating system and may vary. NOTE: This has changed since TCTUTILs.
- image [images]: The disk or partition image to read, whose format is given with '-i'. Multiple image file names can be given if the image is split into multiple segments. If only one image file is given and its name is the first in a sequence (e.g., as indicated by ending in '.001'), subsequent image segments will be included automatically.
- Inode: Meta-data number to display

Segment Number	File Name	Purpose
0	$MFT	Describes all files on the volume, including file names, timestamps, stream names, and lists of cluster numbers where data streams reside, indexes, security identifiers, and file attributes like "read only", "compressed", "encrypted", etc.
1	$MFTMirr	Duplicate of the first vital entries of $MFT, usually 4 entries (4 Kilobytes).
2	$LogFile	Contains transaction log of file system metadata changes
3	$Volume	Contains information about the volume, namely the volume object identifier, volume label, file system version, and volume flags
4	$AttrDef	A table of MFT attributes that associates numeric identifiers with names.
5	.	Root directory. Directory data is stored in $INDEX_ROOT and $INDEX_ALLOCATION attributes both named $I30.
6	$Bitmap	An array of bit entries: each bit indicates whether its corresponding cluster is used (allocated) or free (available for allocation).

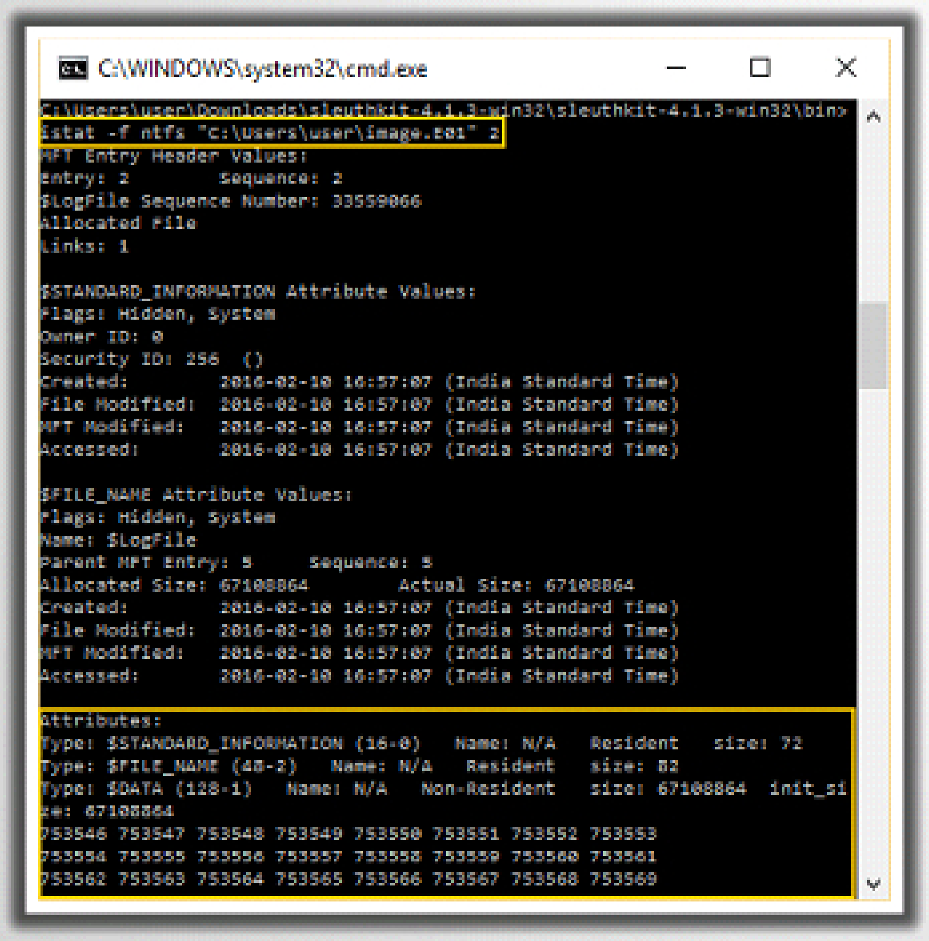
7	$Boot	Volume boot record. This file is always located at the first clusters on the volume. It contains bootstrap code (see NTLDR/BOOTMGR) and a BIOS parameter block including a volume serial number and cluster numbers of $MFT
8	$BadClus	A file that contains all the clusters marked as having bad sectors. This file simplifies cluster management by the chkdsk utility, both as a place to put newly discovered bad sectors, and for identifying unreferenced clusters. This file contains two data streams, even on volumes with no bad sectors: an unnamed stream contains bad sectors—it is zero length for perfect volumes; the second stream is named $Bad and contains all clusters on the volume not in the first stream.
9	$Secure	Access control lists database that reduces overhead having many identical ACLs stored with each file, by uniquely storing these ACLs in this database only
10	$UpCase	A table of unicode uppercase characters for ensuring case-insensitivity in Win32 and DOS namespaces.

TABLE 3.2: NTFS Metadata files

Source: *http://www.sleuthkit.org*

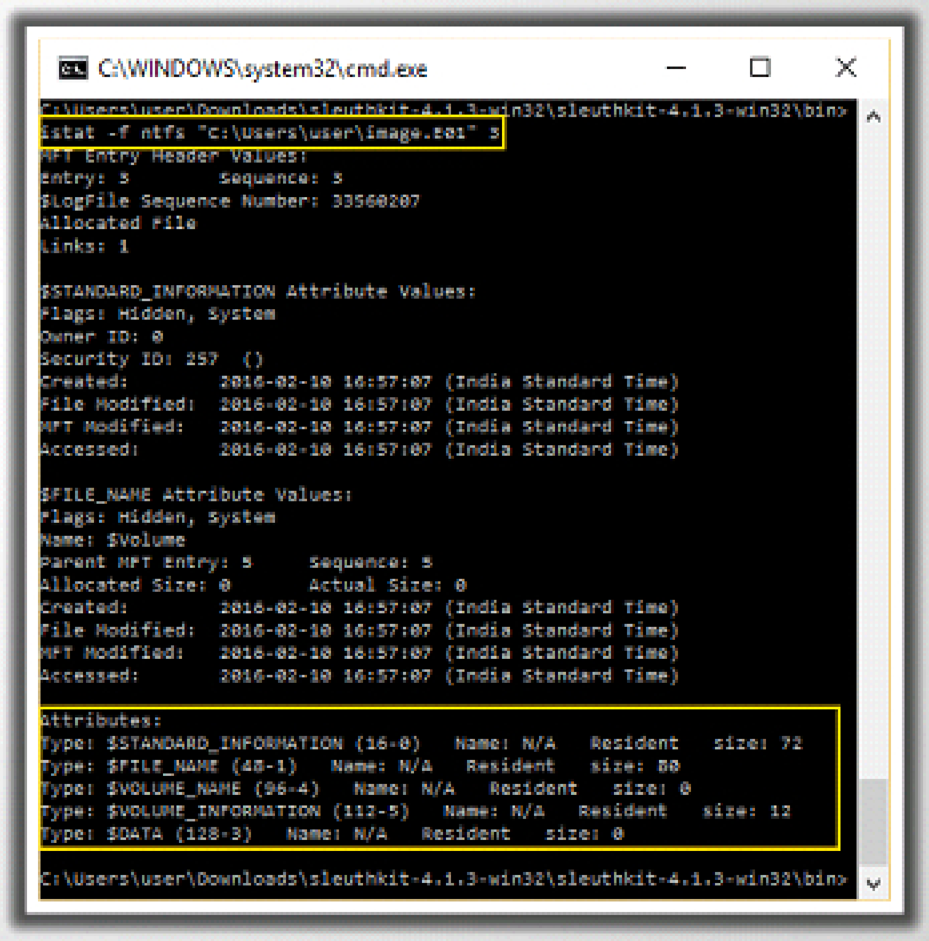
The Sleuth Kit (TSK): istat (2 of 4)

LogFile Overview



```
C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
istat -f ntfs "c:\Users\user\image.E01" 2
MFT Entry Header Values:
Entry: 2        Sequence: 2
$LogFile Sequence Number: 33559066
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Hidden, System
Owner ID: 0
Security ID: 256  ()
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

$FILE_NAME Attribute Values:
Flags: Hidden, System
Name: $LogFile
Parent MFT Entry: 5     Sequence: 5
Allocated Size: 67108864        Actual Size: 67108864
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-2)   Name: N/A   Resident   size: 82
Type: $DATA (128-1)   Name: N/A   Non-Resident   size: 67108864  init_si
ze: 67108864
753546 753547 753548 753549 753550 753551 753552 753553
753554 753555 753556 753557 753558 753559 753560 753561
753562 753563 753564 753565 753566 753567 753568 753569
```

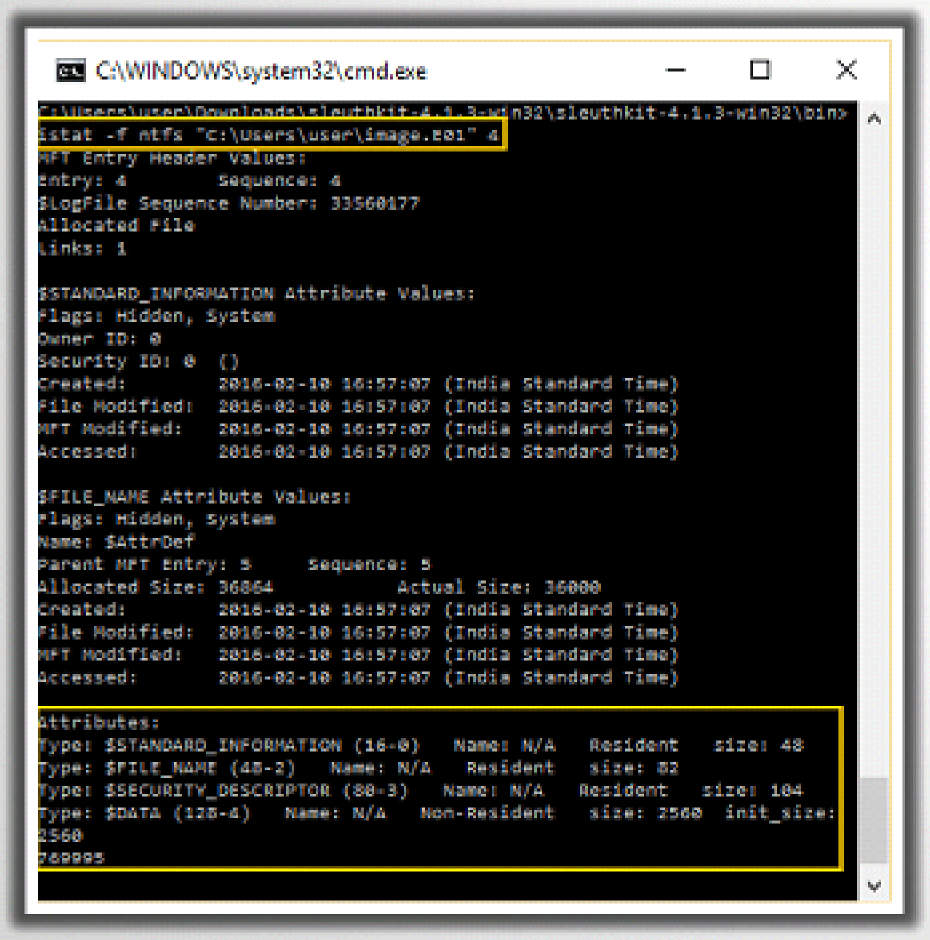
Volume File Overview



```
C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
istat -f ntfs "c:\Users\user\image.E01" 3
MFT Entry Header Values:
Entry: 3        Sequence: 3
$LogFile Sequence Number: 33560207
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Hidden, System
Owner ID: 0
Security ID: 257  ()
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

$FILE_NAME Attribute Values:
Flags: Hidden, System
Name: $Volume
Parent MFT Entry: 5     Sequence: 5
Allocated Size: 0       Actual Size: 0
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-1)   Name: N/A   Resident   size: 80
Type: $VOLUME_NAME (96-4)   Name: N/A   Resident   size: 0
Type: $VOLUME_INFORMATION (112-5)   Name: N/A   Resident   size: 12
Type: $DATA (128-3)   Name: N/A   Resident   size: 0

C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
```

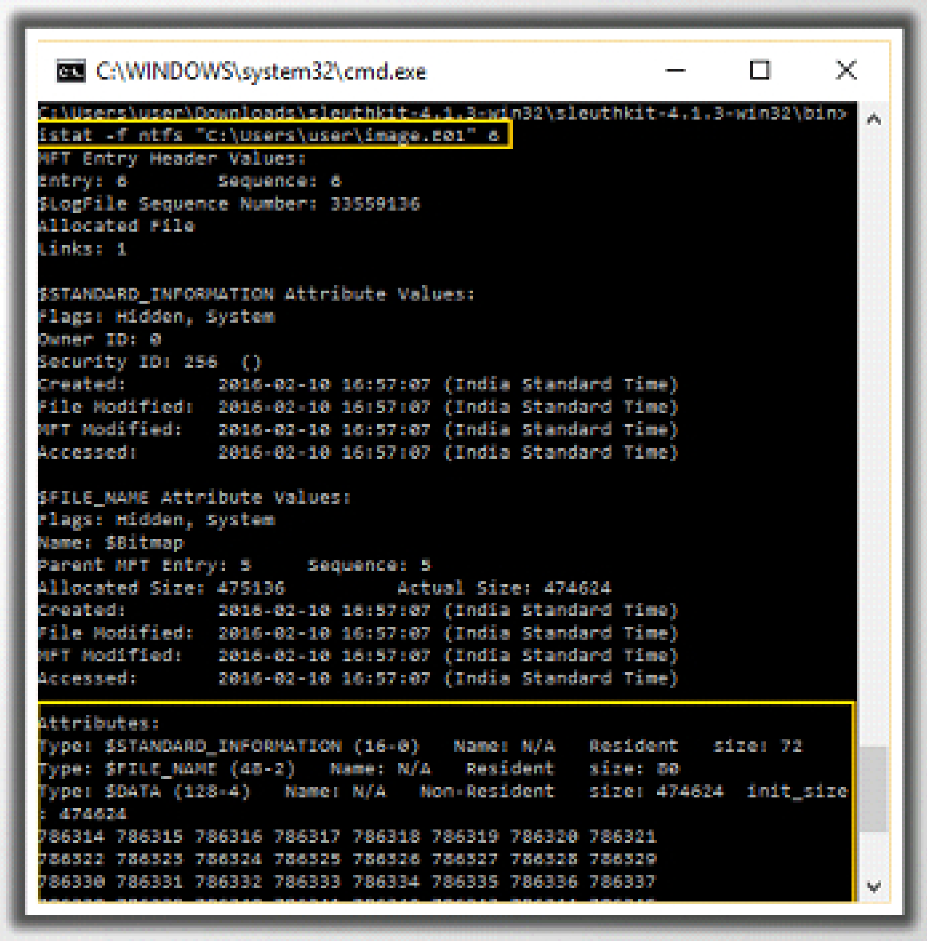
The Sleuth Kit (TSK): istat (3 of 4)

AttrDef File Overview



```
C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
istat -f ntfs "c:\Users\user\image.E01" 4
MFT Entry Header Values:
Entry: 4        Sequence: 4
$LogFile Sequence Number: 33560177
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Hidden, System
Owner ID: 0
Security ID: 0  ()
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

$FILE_NAME Attribute Values:
Flags: Hidden, System
Name: $AttrDef
Parent MFT Entry: 5     Sequence: 5
Allocated Size: 36864           Actual Size: 36000
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 48
Type: $FILE_NAME (48-2)   Name: N/A   Resident   size: 82
Type: $SECURITY_DESCRIPTOR (80-3)   Name: N/A   Resident   size: 104
Type: $DATA (128-4)   Name: N/A   Non-Resident   size: 2560  init_size:
2560
769995
```

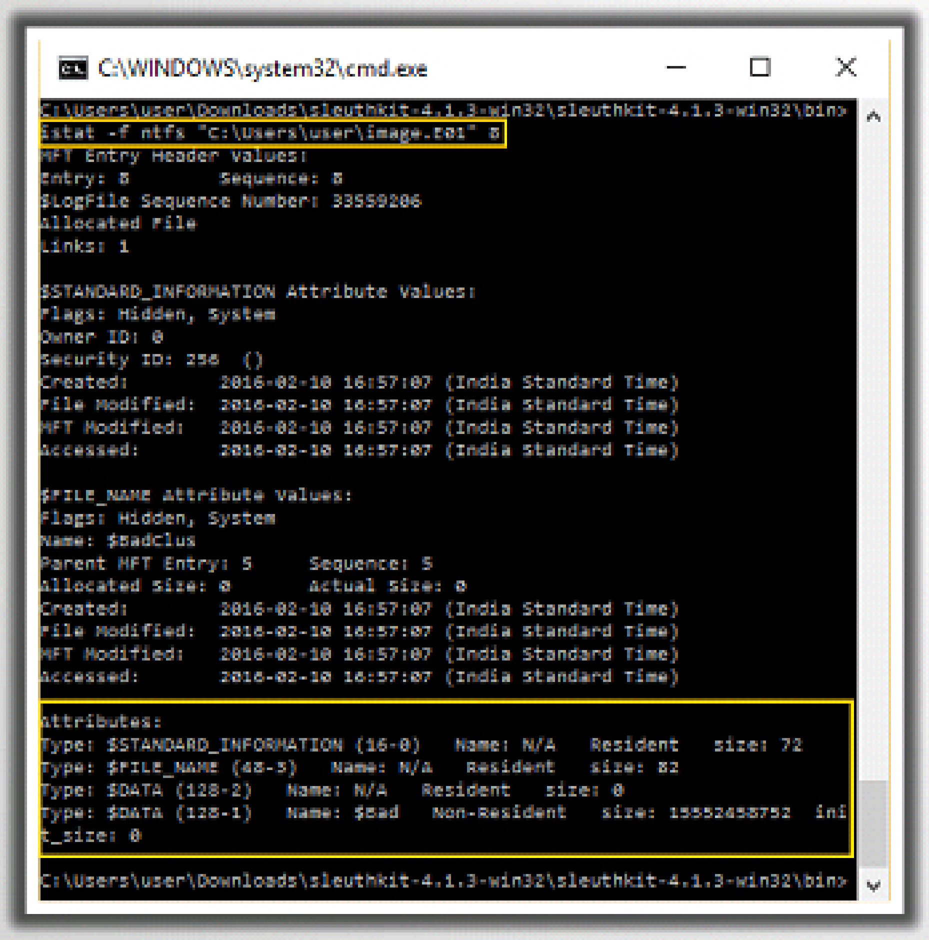
Bitmap File Overview



```
C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
istat -f ntfs "c:\users\user\image.E01" 6
MFT Entry Header Values:
Entry: 6        Sequence: 6
$LogFile Sequence Number: 33559136
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Hidden, System
Owner ID: 0
Security ID: 256  ()
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

$FILE_NAME Attribute Values:
Flags: Hidden, System
Name: $Bitmap
Parent MFT Entry: 5     Sequence: 5
Allocated Size: 475136          Actual Size: 474624
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-2)   Name: N/A   Resident   size: 80
Type: $DATA (128-4)   Name: N/A   Non-Resident   size: 474624  init_size
: 474624
786314 786315 786316 786317 786318 786319 786320 786321
786322 786323 786324 786325 786326 786327 786328 786329
786330 786331 786332 786333 786334 786335 786336 786337
```

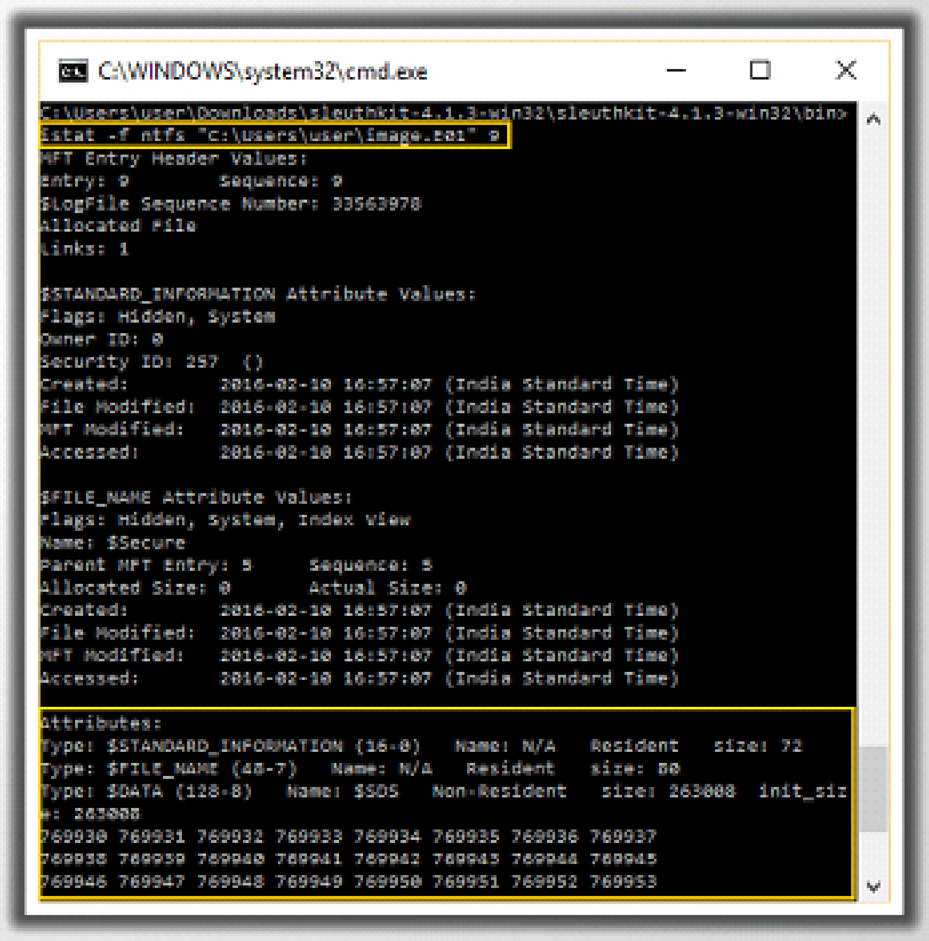
The Sleuth Kit (TSK): istat (4 of 4)

BadClus File Overview



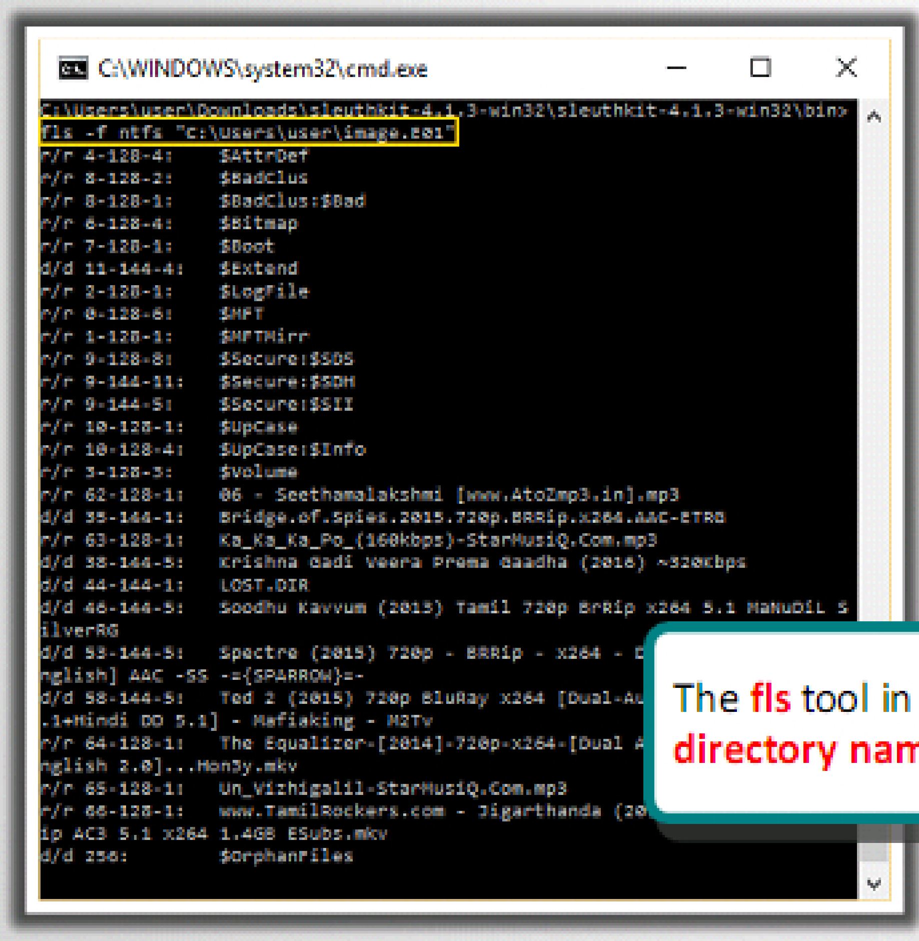
```
C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
istat -f ntfs "C:\Users\user\image.E01" 8
MFT Entry Header Values:
Entry: 8        Sequence: 8
$LogFile Sequence Number: 33559206
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Hidden, System
Owner ID: 0
Security ID: 256  ()
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

$FILE_NAME Attribute Values:
Flags: Hidden, System
Name: $BadClus
Parent MFT Entry: 5     Sequence: 5
Allocated Size: 0       Actual Size: 0
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-3)   Name: N/A   Resident   size: 82
Type: $DATA (128-2)   Name: N/A   Resident   size: 0
Type: $DATA (128-1)   Name: $Bad   Non-Resident   size: 15552458752  ini
t_size: 0

C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
```

Secure File Overview



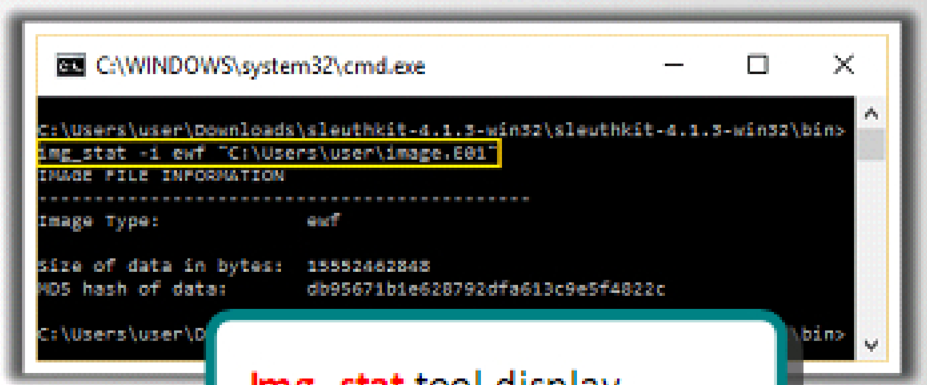
```
C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
istat -f ntfs "C:\Users\user\image.E01" 9
MFT Entry Header Values:
Entry: 9        Sequence: 9
$LogFile Sequence Number: 33563978
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Hidden, System
Owner ID: 0
Security ID: 257  ()
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

$FILE_NAME Attribute Values:
Flags: Hidden, System, Index View
Name: $Secure
Parent MFT Entry: 5     Sequence: 5
Allocated Size: 0       Actual Size: 0
Created:        2016-02-10 16:57:07 (India Standard Time)
File Modified:  2016-02-10 16:57:07 (India Standard Time)
MFT Modified:   2016-02-10 16:57:07 (India Standard Time)
Accessed:       2016-02-10 16:57:07 (India Standard Time)

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-7)   Name: N/A   Resident   size: 80
Type: $DATA (128-8)   Name: $SDS   Non-Resident   size: 263008  init_siz
e: 263008
769930 769931 769932 769933 769934 769935 769936 769937
769938 769939 769940 769941 769942 769943 769944 769945
769946 769947 769948 769949 769950 769951 769952 769953
```

The Sleuth Kit (TSK): fls and img_stat



```
C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
fls -f ntfs "C:\Users\user\image.E01"
r/r 4-128-4:    $AttrDef
r/r 8-128-2:    $BadClus
r/r 8-128-1:    $BadClus:$Bad
r/r 6-128-4:    $Bitmap
r/r 7-128-1:    $Boot
d/d 11-144-4:   $Extend
r/r 2-128-1:    $LogFile
r/r 0-128-6:    $MFT
r/r 1-128-1:    $MFTMirr
r/r 9-128-8:    $Secure:$SDS
r/r 9-144-11:   $Secure:$SDH
r/r 9-144-5:    $Secure:$SII
r/r 10-128-1:   $UpCase
r/r 10-128-4:   $UpCase:$Info
r/r 3-128-3:    $Volume
r/r 62-128-1:   06 - Seethamalakshmi [www.AtoZmp3.in].mp3
d/d 35-144-1:   Bridge.of.Spies.2015.720p.BRRip.x264.AAC-ETRG
r/r 63-128-1:   Ka_Ka_Ka_Po_(160kbps)-StarMusiQ.Com.mp3
d/d 38-144-5:   Krishna Gadi Veera Prema Gaadha (2016) ~320Kbps
d/d 44-144-1:   LOST.DIR
d/d 46-144-5:   Soodhu Kavvum (2013) Tamil 720p BrRip x264 5.1 MaNuDiL S
ilverRG
d/d 53-144-5:   Spectre (2015) 720p - BRRip - x264 - D
nglish] AAC -SS -={SPARROW}=-
d/d 58-144-5:   Ted 2 (2015) 720p BluRay x264 [Dual-Au
.1+Hindi DD 5.1] - Mafiaking - M2Tv
r/r 64-128-1:   The Equalizer-[2014]-720p-x264-[Dual A
nglish 2.0]...Hon3y.mkv
r/r 65-128-1:   Un_Vizhigalil-StarMusiQ.Com.mp3
r/r 66-128-1:   www.TamilRockers.com - Jigarthanda (20
ip AC3 5.1 x264 1.4GB ESubs.mkv
d/d 256:        $OrphanFiles
```

The **fls** tool in TSK **list file and directory names** in a disk image



```
C:\Users\user\Downloads\sleuthkit-4.1.3-win32\sleuthkit-4.1.3-win32\bin>
img_stat -i ewf "C:\Users\user\image.E01"
IMAGE FILE INFORMATION
--------------------------------------------
Image Type:             ewf

Size of data in bytes:  15552462848
MD5 hash of data:       db95671b1e628792dfa613c9e5f4822c

C:\Users\user\D                                                         \bin>
```

Img_stat tool display details of an **image file**

fls

fls - List file and directory names in a disk image.

Syntax

fls [-adDFlpruvV] [-m mnt] [-z zone] [-f fstype] [-s seconds] [-i imgtype] [-o imgoffset] [-b dev_sector_size] image [images] [inode]

Description

fls lists the files and directory names in the image and can display file names of recently deleted files for the directory using the given inode. If the inode argument is not given, the inode value for the root directory is used. For example, on an NTFS file system it would be 5 and on an Ext3 file system it would be 2.

Arguments

- -a

  Display the "." and ".." directory entries (by default it does not)
- -d

  Display deleted entries only
- -D

  Display directory entries only
- -f fstype

  Displays the type of file system. Use '-f list' to list the supported file system types. If not given, auto detection methods are used.
- -F

  Display file (all non-directory) entries only.
- -l

  Display file details in long format. The following contents are displayed:

  file_type inode file_name mod_time acc_time chg_time cre_time size uid gid
- -m mnt

  Display files in time machine format so that a timeline can be gid created with mactime(1). The string given as mnt will be prepended to the file names as the mounting point (for example /usr).
- -p

  Display the full path for each entry. By default, it denotes the directory depth on recursive runs with a '+' sign.
- -r

  Recursively display directories. This will not follow deleted directories, because it can't.

- -s seconds

  The time skew of the original system in seconds. For example, if the original system was 100 seconds slow, this value would be -100. This is only used if -l or -m are given.

- -i imgtype

  Identify the type of image file, such as raw. Use '-i list' to list the supported types. If not given, auto detection methods are used.

- -o imgoffset

  The sector offset where the file system starts in the image.

- -b dev_sector_size

  The size, in bytes, of the underlying device sectors. If not given, the value in the image format is used (if it exists) or 512-bytes is assumed.

- -u

  Display undeleted entries only

- -v

  Verbose output to stderr.

- -V

  Display version.

- -z zone

  The ASCII string of the time zone of the original system. For example, EST or GMT. These strings must be defined by your operating system and may vary.

img_stat

img_stat - Display details of an image file

Syntax

img_stat [-i imgtype] [-b dev_sector_size] [-tvV] image [images]

Description

img_stat displays the details associated with an image file. The output of this command is image format specific. At a minimum, the size will be given and the byte range of each file will be given for split image formats.

Arguments

- -i imgtype

  Identify the type of image file, such as raw. Use '-i list' to list the supported types. If not given, autodetection methods are used.

- -b dev_sector_size

  The size, in bytes, of the underlying device sectors. If not given, the value in the image format is used (if it exists) or 512-bytes is assumed.

- -t

  Print the image type only.

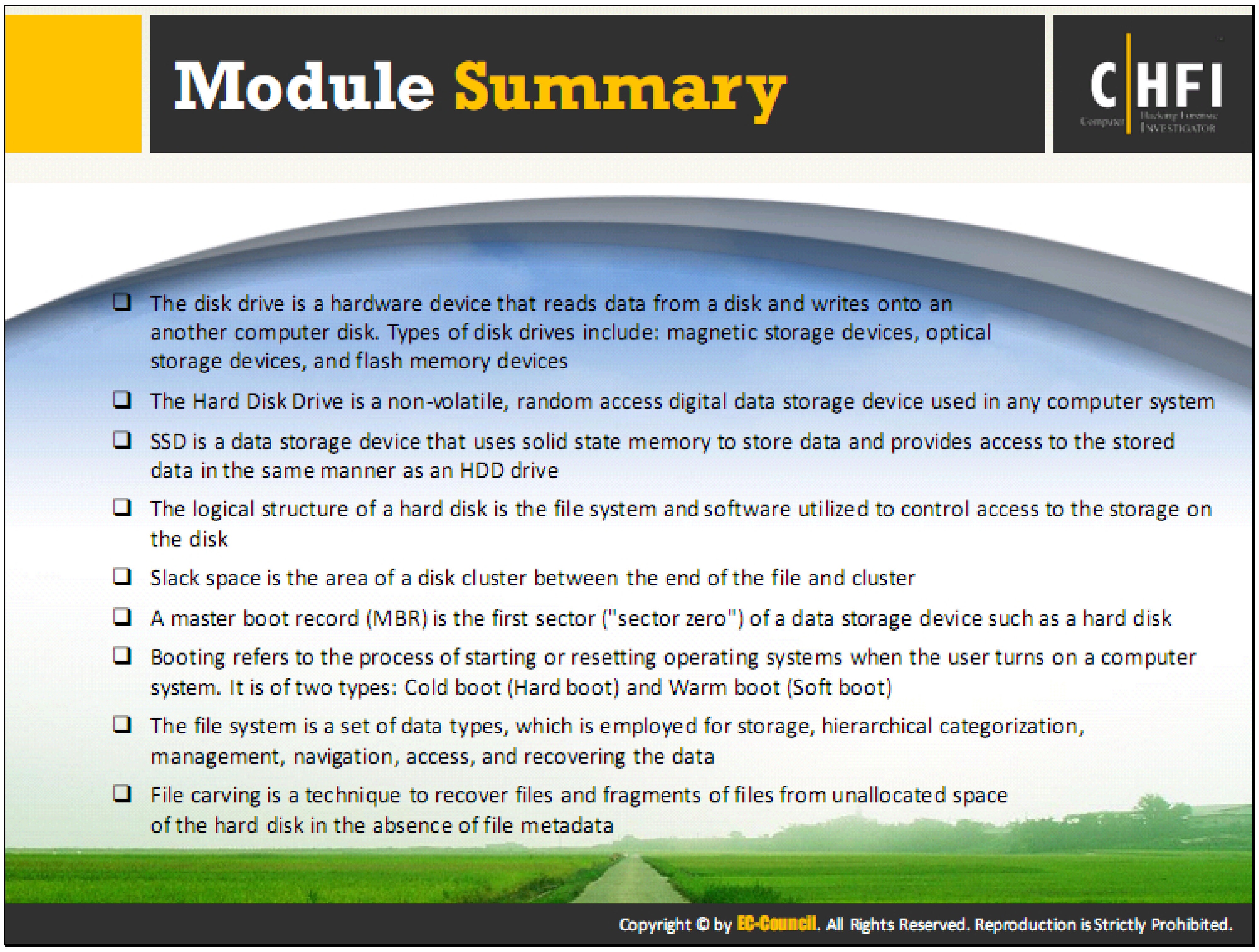
- -v

  Verbose output of debugging statements to stderr

- -V

  Display version

Source: http://www.sleuthkit.org



Module Summary

- The disk drive is a hardware device that reads data from a disk and writes onto an another computer disk. Types of disk drives include: magnetic storage devices, optical storage devices, and flash memory devices
- The Hard Disk Drive is a non-volatile, random access digital data storage device used in any computer system
- SSD is a data storage device that uses solid state memory to store data and provides access to the stored data in the same manner as an HDD drive
- The logical structure of a hard disk is the file system and software utilized to control access to the storage on the disk
- Slack space is the area of a disk cluster between the end of the file and cluster
- A master boot record (MBR) is the first sector ("sector zero") of a data storage device such as a hard disk
- Booting refers to the process of starting or resetting operating systems when the user turns on a computer system. It is of two types: Cold boot (Hard boot) and Warm boot (Soft boot)
- The file system is a set of data types, which is employed for storage, hierarchical categorization, management, navigation, access, and recovering the data
- File carving is a technique to recover files and fragments of files from unallocated space of the hard disk in the absence of file metadata

In this module, you have learnt about the hard disks and file systems, as well as the structure and behavior of the hard disk. This module also exposes the process of working of different types of hard disks on various operating systems such as Windows, MAC OS, and Linux. Investigators can also learn the process of analyzing a hard disk, CD-ROM/DVD, and RAID storage system in this module. The next module will discuss the data acquisition and duplication process in detail.