Digital Forensics

An introduction into Post-mortem Digital Forensics

CIRCL
Computer Incident
Response Center
Luxembourg

CIRCL *TLP:WHITE*

info@circl.lu

Version 1.0.2 2018 edition

Overview

1. Introduction
2. From data to knowledge
3. Disk Acquisition
4. Disk Cloning / Disk Imaging
5. Disk Analysis
6. File System Analysis
7. Carving
8. Analysing files
9. String Search
10. Windows Registry Analysis
11. Memory Forensics
12. Outlook

CIRCL
FORENSICS Training

1. Introduction

1.1 Admin default behaviour

- Get operational asap:
  - Re-install
  - Re-image
  - Restore from backup
    $\rightarrow$ Destroy of evidences

- Analyse the system on his own:
  - Do some investigations
  - Run AV
  - Apply updates
    $\rightarrow$ Overwrite evidences
    $\rightarrow$ Create big noise

$\rightarrow$ Negative impact on forensics

1.2 Preservation of evidences

- Finding answers:
  - → System compromised
  - → How, when, why
  - → Malware/RAT involved
  - → Persistence mechanisms
  - → Lateral movement inside LAN
  - → Detect the root cause of the incident
  - → Access sensitive data
  - → Data exfiltration
  - → Illegal content
  - → System involved at all

- Legal case:
  - → Collect & safe evidences
  - → Witness testimony for court

1.2 Preservation of evidences

- CRC not sufficient:
  - Example: Checksum

    $4711 \rightarrow 13$
  - Example: Collision

    $12343 \rightarrow 13$

- Cryptographic hash function:
  - Output always same size
  - Deterministic: if $m = m \rightarrow h(m) = h(m)$
  - 1 Bit change in m $\rightarrow$ max. change in $h(m)$
  - One way function: For $h(m)$ impossible to find m
  - Simple collision resistance: For given $h(m1)$ hard to find $h(m2)$
  - Strong collision resistance: For any $h(m1)$ hard to find $h(m2)$

1.3 Forensics Science

- Classical forensic

  Locard's exchange principle

  https://en.wikipedia.org/wiki/Locard%27s_exchange_principle

- Write down everything you see, hear, smell and do

- Chain of custody

  → https://www.nist.gov/sites/default/files/documents/2017/04/28/Sample-Chain-of-Custody-Form.docx

- Scope of the analysis

1.4 Forensic disciplines

- Reverse Engineering
- Code-Deobfuscation
- Memory Forensics
  - → https://www.circl.lu/pub/tr-22/
  - → https://www.circl.lu/pub/tr-30/
- Network Forensics
- Mobile Forensics
- Cloud Forensics
- Post-mortem Analysis
  - → https://www.circl.lu/pub/tr-22/
  - → https://www.circl.lu/pub/tr-30/

1.5 First Responder: Order of volatility

CPU registers → nanoseconds

CPU cache → nanoseconds

RAM memory → tens of nanoseconds

Network state → milliseconds

Processes running → seconds

Disk, system settings, data → minutes

External disks, backup → years

Optical storage, printouts → tens of ears

→ `https://www.circl.lu/pub/tr-22/`

1.5 First Responder: Be prepared

- Prepare your toolbox
  - Photo camera
  - Flash light, magnifying glasses
  - Labelling device, labels, tags, stickers
  - Toolkit, screwdriver kits
  - Packing boxes, bags, faraday bag
  - Cable kits, write blocker, storage devices
  - Anti-static band, network cables
  - Pens, markers, notepads
    $\rightarrow$ Chain of custody
- USB stick
  - 256 GB USB3
  - File system: exFAT
  - Memory dump: Dumpit
  - FTK Imager Lite
  - Encrypted Disk Detector - Edd

1.5 First Responder: First steps

- Did an incident occure
  - Talk with people
  - Take notes
- Mouse jiggler
- Identify potential evidences
  - Tower, desktop, laptop, tablets
  - Screen, printer, storage media
  - Router, switches, access point
  - Paper, notes, ...
- Powered-on versus powered-off
  - Shutdown: Lost of live data
  - Shutdown: Data on disk modified
  - Pull power: Corrupt file system
  - Live analysis: Modify memory and disk
  - Live analysis: Known good binaries?

1.5 First Responder: Live response

- Memory dump
- Live analysis:
  - → System time
  - → Logged-on users
  - → Open files
  - → Network -connections -status
  - → Process information -memory
  - → Process / port mapping
  - → Clipboard content
  - → Services
  - → Command history
  - → Mapped drives / shares
  - → !!! Do not store information on the subject system !!!
- Image of live system (Possible issues)
- Shutdown and image if possible

1.6 Post-mortem Analysis

- Hardware layer & acquisition
  - Best copy (in the safe)
  - Working copy (on a NAS)
  - Disk volumes and partitions
  - Simple tools: dd, dmesg, mount
- File system layer
  - FAT, NTFS
  - File system timeline
  - Restore deleted files
- Data layer
  - Carving: foremost, scalpel, testdisk/photorec
  - String search

1.7 Post-mortem Analysis

- OS layer
  - Registry
  - Event logs
  - Volume shadow copies
  - Prefetch files
- Application layer
  - AV logs
  - Browser history: IE, firefox, chrome
  - Email
  - Office files & PDFs
- Identify malware
  - TEMP folders
  - Startup folders
  - Windows tasks

1.8 Forensic Distributions

- Commercial

  EnCase Forensic
  F-Response
  Forensic Toolkit
  Helix Enterprise
  X-Ways Forensics
  Magnet Axiom

- Open source tools

  Kali Linux
  SANS SIFT
  Digital Evidence and Forensics Toolkit - DEFT
  PlainSight
  Computer Aided INvestigative Environment - CAINE

CIRCL
FORENSICS Training

2. From data to knowledge

2.1 Data in a binary system

- Binary digit $\rightarrow$ BIT

- Data represented as binary patterns

  Ordered sequence

  `x Bits --> 01010000011010010110111001100111 --> y Bits`

  Bit x + 2 = 1

  Bit x + 3 = 0

- Structurise the data: Apply addressing

```
-->   01010000   01101001   01101110   01100111   -->
      --------   --------   --------   --------
-->   Byte 117   Byte 118   Byte 119   Byte 120   -->
```

- Apply interpretative rules on addresses

2.1 Data in a binary system

- Nibble

  `0101 0000 0110 1001 0110 1110 0110 0111`
- Byte

  `01010000  01101001  01101110  01100111`
- Word

  `0101000001101001  0110111001100111`
- Double Word
- Big / Little Endian
- Integer / Signed Integer
- Floating Point
- Binary Coded Decimal
- ASCII, Unicode
- GIF / JPEG / PNG / EXE / ...
- ...

2.2 Example: Integer Bytes

```
0101 0000   0110 1001   0110 1110   0110 0111
---------

0101 0000
|||| ||||__ 0 * 2^0 =   0
|||| |||___ 0 * 2^1 =   0
|||| ||____ 0 * 2^2 =   0
|||| |_____ 0 * 2^3 =   0
||||_______ 1 * 2^4 =  16
|||________ 0 * 2^5 =   0
||_________ 1 * 2^6 =  64
|__________ 0 * 2^7 =   0
                      ---
                       80
```

2.3 Example: Signed Integer Bytes

```
1011 1111
---------
 011 1111
 100 0000
 100 0001
 ||| ||||__ 1 * 2^0 =   1
 ||| |||___ 0 * 2^1 =   0
 ||| ||____ 0 * 2^2 =   0
 ||| |_____ 0 * 2^3 =   0
 |||_______ 0 * 2^4 =   0
 ||________ 0 * 2^5 =   0
 |_________ 1 * 2^6 =  64
                      ---
                      -65
```

```
Two's complement:
      1. Remove the sign
      2. Invert
      3. Add 1
```

2.3 Exercise: Signed Integer Bytes

```
1101 1100
---------
```

Two's complement:

1. Remove the sign
2. Invert
3. Add 1

```
 ||| ||||__ ? * 2^0 =
 ||| |||___ ? * 2^1 =
 ||| ||____ ? * 2^2 =
 ||| |_____ ? * 2^3 =
 |||_______ ? * 2^4 =
 ||________ ? * 2^5 =
 |_________ ? * 2^6 =
                      ---
                      -
```

2.3 Exercise: Signed Integer Bytes

```
1101 1100
---------
 101 1100
 010 0011
 010 0100
 ||| ||||__ 0 * 2^0 =   0
 ||| |||___ 0 * 2^1 =   0
 ||| ||____ 1 * 2^2 =   4
 ||| |_____ 0 * 2^3 =   0
 |||_______ 0 * 2^4 =   0
 ||________ 1 * 2^5 =  32
 |_________ 0 * 2^6 =   0
                      ---
                      -36
```

Two's complement:

1. Remove the sign
2. Invert
3. Add 1

2.4 From Bin to Hex

Example:

```
0001 1000     0101 0101     0000 1111     1010 0110
---------     ---------     ---------     ---------
   0x18          0x55          0x0F          0xA6
```

Exercise:

```
1001 0110     1010 0101     0000 1111     1100 0011
---------     ---------     ---------     ---------
   0x            0x            0x            0x
```

2.4 From Bin to Hex

Exercise:

```
1001 0110     1010 0101     0000 1111     1100 0011
---------     ---------     ---------     ---------
   0x            0x            0x            0x
```

Results:

```
1001 0110     1010 0101     0000 1111     1100 0011
---------     ---------     ---------     ---------
   0x96          0xA5          0x0F          0xC3
```

2.5 Big Endian and Little Endian

```
Big Endian representation:

    2^: 15 14 13 12 11 10  9  8    7  6  5  4  3  2  1  0
         _  _  _  _  _  _  _  _    _  _  _  _  _  _  _  _
         0  0  0  1  1  0  0  0    0  1  0  1  0  1  0  1
         _  _  _  _  _  _  _  _    _  _  _  _  _  _  _  _
Address: 10.000                    10.001


Little Endian representation:

    2^: 7  6  5  4  3  2  1  0   15 14 13 12 11 10  9  8
         _  _  _  _  _  _  _  _    _  _  _  _  _  _  _  _
         0  1  0  1  0  1  0  1    0  0  0  1  1  0  0  0
         _  _  _  _  _  _  _  _    _  _  _  _  _  _  _  _
Address: 10.000                    10.001
```

2.5 Big Endian and Little Endian

Exercise: Convert 2 byte value to Little Endian representation:

```
10010110 10100101
-------- --------
  0x96     0xA5



-------- --------
  0x       0x
```

Exercise: Read 4 byte value in Little Endian representation:

```
0x 1B 2A 01 00
   -- -- -- --

0x
   -- -- -- --
 =
   ======
```

2.5 Big Endian and Little Endian

Exercise: Convert 2 byte value to Little Endian representation:

```
10010110 10100101
-------- --------
  0x96     0xA5

10100101 10010110
-------- --------
  0xA5     0x96
```

Exercise: Read 4 byte value in Little Endian representation:

```
0x 1B 2A 01 00
   -- -- -- --

0x 00 01 2A 1B
   -- -- -- --  11 + 1*16 + 10*16^2 + 2*16^3 + 1*16^4
 = 76.315
   ======
```

2.6 Example: Others

Packed BCD

```
0110 1110 0110 0111
---- ---- ---- ----
  6   na    6    7
```

ASCII

```
0101 000 0110 1001 0110 1110 0110 0111

01010000  01101001  01101110  01100111
--------  --------  --------  --------
     80       105       110       103
      P         i         n         g
```

2.7 Data, files, context

- Sequence of Bits + Addressing + Interpretation → Information
- Information → Stored in files
- Where did you find the string "ping"?
  - Binary inside TEMP folder
  - Autorun folder
  - Registry
  - Browser history
  - Command line history

  → Data → Information → Knowledge

- Files contains data
- Files → Meta data describe files
- Files → File systems organize files and meta data

CIRCL FORENSICS Training

3. Disk Acquisition

3.1 Storage devices / media

- IBM 305 RAMAC
  - Random Access Method of Accounting and Control
  - 1956
- IBM 350 Disk Storage
  - 152 x 172 x 63 cm
  - 50.000 blocks of 100 Characters → 5MB

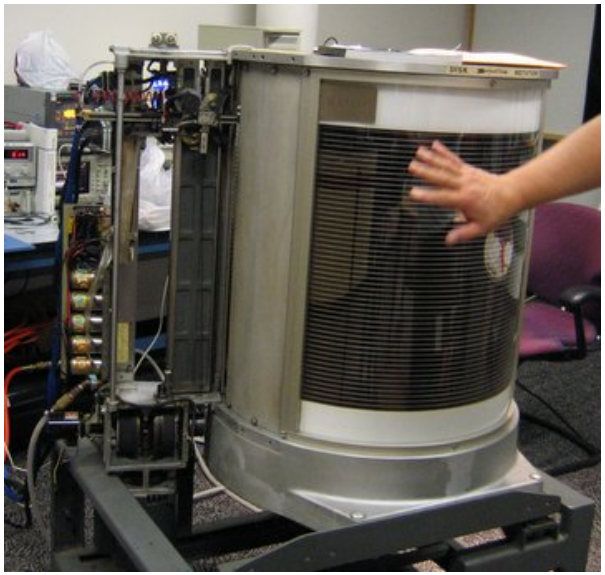


Image (c) wikipedia.org - Image used solely for illustration purposes

3.1 Storage devices / media

- Magnetic storage
  - Tapes
  - Floppy disks
    - 8" - 1971 - 80KB
    - 5.25" - 1976 - 360 KB
    - 3.5" - 1984 - 1.2 MB / - 1986 - 1.44 MB
  - Hard disks
    - IDE / EIDE, Firewire, PATA, SCSI
    - SATA, SAS Serial attached SCSI, USB, Thunderbolt
- Optical storage
  - Compact disks - CD
  - Digital versatile disk - DVD
  - Blu-ray disk
- Non-volatile memory
  - USB flash drive
  - Solid state drive
  - Flash memory cards

3.2 Physical- / Logical layers

```
                Physical disks
 ___________   ___________   ___________
| Disc 1 |    | Disc 2 |    | Disc 3 |
 ___________   ___________   ___________
      |             |             |
      V             V             V
      Logical hard disk volume
 ____________________________________________
| ____________________________________________ |
| |   P1   |      P2      |      P3      | |
| ____________________________________________ |
 ____________________________________________
      |             |             |
      V             V             V
  ___________   ___________   ___________
  |   P1   |   |    P2    |   |    P3    |
  ___________   ___________   ___________
 Volume C:     Volume D:     Volume E:
```

3.3 ATA Disks

- ATA-3: Hard disk password
- ATA-4: HPA - Host Protected Area
  - Vendor area - benefit system vendors
  - Recovery data. persistent data
  - Controlled by firmware not OS
  - `READ_NATIVE_MAX_ADDRESS`
- ATA-6: DCO - Device Configuration Overlay
  - Benefit system vendors
  - Control reported capacity and disk features
  - Use disk from different manufacturers
  - Use disk with different number of sectors
    $\rightarrow$ Makes disks looking uniq
  - `DEVICE_CONFIGURATION_IDENTIFY`
- ATA-7: Serial ATA

3.4 Demo: Hidden Sectors

- New disk

```
dmesg
    sd 1:0:0:0: [sdb] 3904981168 512-byte logical blocks: (2.00 TB/1.82 TiB)

hdparm -N /dev/sdb
    max sectors   = 3907029168/3907029168, ACCESSIBLE MAX ADDRESS disabled
```

- Create hidden message

```
echo -n 'MySecret 123456' | dd of=/dev/sdb seek=3500000000

dd if=/bin/dd of=/dev/sdb seek=3500000001
     148+1 records in
     148+1 records out
     76000 bytes (76 kB, 74 KiB) copied, 0,022659 s, 3,4 MB/s
```

- Create HPA

```
hdparm --yes-i-know-what-i-am-doing -N p3000000000 /dev/sdb
    setting max visible sectors to 3000000000 (permanent)
    max sectors   = 3000000000/3907029168, ACCESSIBLE MAX ADDRESS enabled

Power cycle your device after every ACCESSIBLE MAX ADDRESS
```

3.4 Demo: Hidden Sectors

- Create partition and format

```
dmesg
    sd 1:0:0:0: [sdb] 3000000000 512-byte logical blocks: (1.54 TB/1.40 TiB)

fdisk /dev/sdb
    primary
    2048
    2999999999

mkfs.ntfs -L CIRCL.DFIR -f /dev/sdb1
    Creating NTFS volume structures.
    mkntfs completed successfully. Have a nice day.
```

- Investigate disk layout

```
fdisk -l /dev/sdb
    Device     Boot Start        End    Sectors  Size Id Type
    /dev/sdb1        2048 2999999999 2999997952  1,4T  7 HPFS/NTFS/exFAT
```

- Investigate last accessible sector

```
dd if=/dev/sdb skip=2999999999 status=none| xxd
    00000000: eb52 904e 5446 5320 2020 2000 0208 0000  .R.NTFS    .....
      .......
    000001f0: 0000 0000 0000 0000 0000 0000 0000 55aa  ..............U.
```

3.4 Demo: Hidden Sectors

- Try to access hidden message

```
dd if=/dev/sdb skip=3500000000 count=1 | xxd
    dd: /dev/sdb: cannot skip: Invalid argument
    0+0 records in
```

- Resize HPA

```
hdparm -N /dev/sdb
    max sectors   = 3000000000/3907029168, ACCESSIBLE MAX ADDRESS enabled

hdparm --yes-i-know-what-i-am-doing -N p3900000000 /dev/sdb
    max sectors   = 3900000000/3907029168, ACCESSIBLE MAX ADDRESS enabled

Power cycle your device after every ACCESSIBLE MAX ADDRESS
```

- Investigate disk layout and last sector

```
fdisk -l /dev/sdb
    Device     Boot Start        End    Sectors  Size Id Type
    /dev/sdb1        2048 2999999999 2999997952  1,4T  7 HPFS/NTFS/exFAT

dd if=/dev/sdb skip=2999999999 status=none | xxd | less
dd if=/dev/sdb skip=3899999999 status=none | xxd | less
```

3.4 Demo: Hidden Sectors

- Recover hidden message

```
dd if=/dev/sdb skip=3500000000 count=1 status=none
    00000000: 4d79 5365 6372 6574 2031 3233 3435 3600  MySecret 123456.
```

- Recover hidden dd command

```
dd if=/dev/sdb skip=$(( 3500000001*512 )) count=76000 bs=1 of=dd.exe

md5sum dd.exe
    36a70f825b8b71a3d9ba3ac9c5800683

md5sum /bin/dd
    36a70f825b8b71a3d9ba3ac9c5800683
```

- Feeback: kaplan(at)cert.at

```
https://www.schneier.com/blog/archives/2014/02/swap_nsa_exploi.html
https://en.wikipedia.org/wiki/Host_protected_area
```

- How it works

```
IDENTIFY DEVICE
SET MAX ADDRESS
READ NATIVE MAX ADDRESS
--> HPA aware software (like the BIOS)
```

3.5 Other Hidden Sectors

- Service area, negative sectors
  - Firmware
  - Bad sectors
  - ATA passwords

    ```
    hdparm --security-unlock "myPassWD" /dev/sdb
    ```
  - SMART data

- Self-Monitoring, Analysis and Reporting Technology - SMART

  ```
  apt install smartmontools
  smartctl -x /dev/sdb | less
  ```

  ```
  .....
  SMART Attributes Data Structure revision number: 16
  Vendor Specific SMART Attributes with Thresholds:
  ID# ATTRIBUTE_NAME          FLAGS    VALUE WORST THRESH FAIL RAW_VALUE
    1 Raw_Read_Error_Rate     POSR-K   200   200   051    -    0
    3 Spin_Up_Time            POS--K   234   233   021    -    3258
    4 Start_Stop_Count        -O--CK   100   100   000    -    679
    5 Reallocated_Sector_Ct   PO--CK   200   200   140    -    0
    7 Seek_Error_Rate         -OSR-K   200   200   000    -    0
    9 Power_On_Hours          -O--CK   095   095   000    -    3802
    .....
  ```

3.6 Collecting information from devices

hdparm -I /dev/sdb

```
ATA device, with non-removable media
        Model Number:       WDC WD20NPVT-00Z2TT0
        Serial Number:      WD-WX11A9269540
        Firmware Revision:  01.01A01
        Transport:          Serial, SATA 1.0a, SATA Rev 2.6, SATA Rev 3.0
Standards:
        Supported: 8 7 6 5
        Likely used: 8
        .....
Security:
        Master password revision code = 65534    supported
        not     enabled
        not     locked
        not     frozen
        not     expired: security count
        374min for SECURITY ERASE UNIT.
```

hdparm -I /dev/sda

```
...
Commands/features:
        Enabled Supported:
        ...
           *    Data Set Management TRIM supported (limit 8 blocks)
           *    Deterministic read ZEROs after TRIM
```

3.7 How is the device connected

- Most relevant data with: `dmesg`

```
dmesg -T
    .....
    [Mi Aug  1 13:06:11 2018] usb-storage 1-1:1.0: USB Mass Storage device detected
    [Mi Aug  1 13:06:11 2018] scsi host1: usb-storage 1-1:1.0
    [Mi Aug  1 13:06:13 2018] scsi 1:0:0:0: Direct-Access USB Flash DISK
    [Mi Aug  1 13:06:13 2018] sd 1:0:0:0: Attached scsi generic sg1 type 0
    [Mi Aug  1 13:06:13 2018] sd 1:0:0:0: [sdb] 15826944 512-byte logical blocks
```

- Enumerate host hardware

```
lshw | less
    .....

lshw -businfo -class storage
    Bus info          Device          Class          Description
    ==================================================================
    pci@0000:04:00.0                  storage        Samsung Electronics Co Ltd
    usb@2:3           scsi0           storage
    usb@1:1           scsi1           storage

lshw -businfo -class disk
    Bus info          Device          Class          Description
    ==================================================================
    scsi@0:0.0.0      /dev/sda        disk           SD/MMC CRW
                      /dev/sda        disk
    scsi@1:0.0.0      /dev/sdb        disk           2TB 2000FYYZ-01UL1B2
```

3.7 How is the device connected

- Enumerate PCI bus

```
lspci -d ::0106                    # List SATA controller

lspci -d ::0108                    # List NVME controller
    04:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd Device a808

lspci -d ::0C03                    # List USB, FW, ... controller
    00:14.0 USB controller: Intel Corporation Sunrise Point-LP USB 3.0 xHCI Controller
    3b:00.0 USB controller: Intel Corporation JHL6540 Thunderbolt 3 USB Controller (C
    3e:00.0 USB controller: Fresco Logic FL1100 USB 3.0 Host Controller (rev 10)
    40:00.0 USB controller: Fresco Logic FL1100 USB 3.0 Host Controller (rev 10)
```

- Enumerate block devices

```
lsscsi -v

lsblk /dev/sdb
    NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
    sdb      8:16   0  1,8T  0 disk
     sdb1   8:17   0  1,8T  0 part /media/mich/031F0F30642CBB8B

lsblk -pd -o TRAN,NAME,SERIAL,VENDOR,MODEL,REV,WWN,SIZE,HCTL,SUBSYSTEMS /dev/sdb
    TRAN NAME     SERIAL           VENDOR   MODEL
    usb  /dev/sdb WD-WMC1P0H10ZEX  WT055 WD 2000FYYZ-01UL1B2
             REV WWN                  SIZE HCTL       SUBSYSTEMS
             01.0 0x50014ee05979e023  1,8T 1:0:0:0    block:scsi:usb:pci
```

3.8 USB enumeration

- List attached USB device
  - USB bus
  - Device address
  - Vendor ID
  - Product ID
  - Product details

  ...

`lsusb`

```
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 002: ID 0bda:0328 Realtek Semiconductor Corp.
Bus 002 Device 003: ID 1b1c:1a0e Corsair
Bus 002 Device 004: ID 0951:162b Kingston Technology
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 06cb:009a Synaptics, Inc.
Bus 001 Device 003: ID 04f2:b61e Chicony Electronics Co., Ltd
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

3.8 USB enumeration

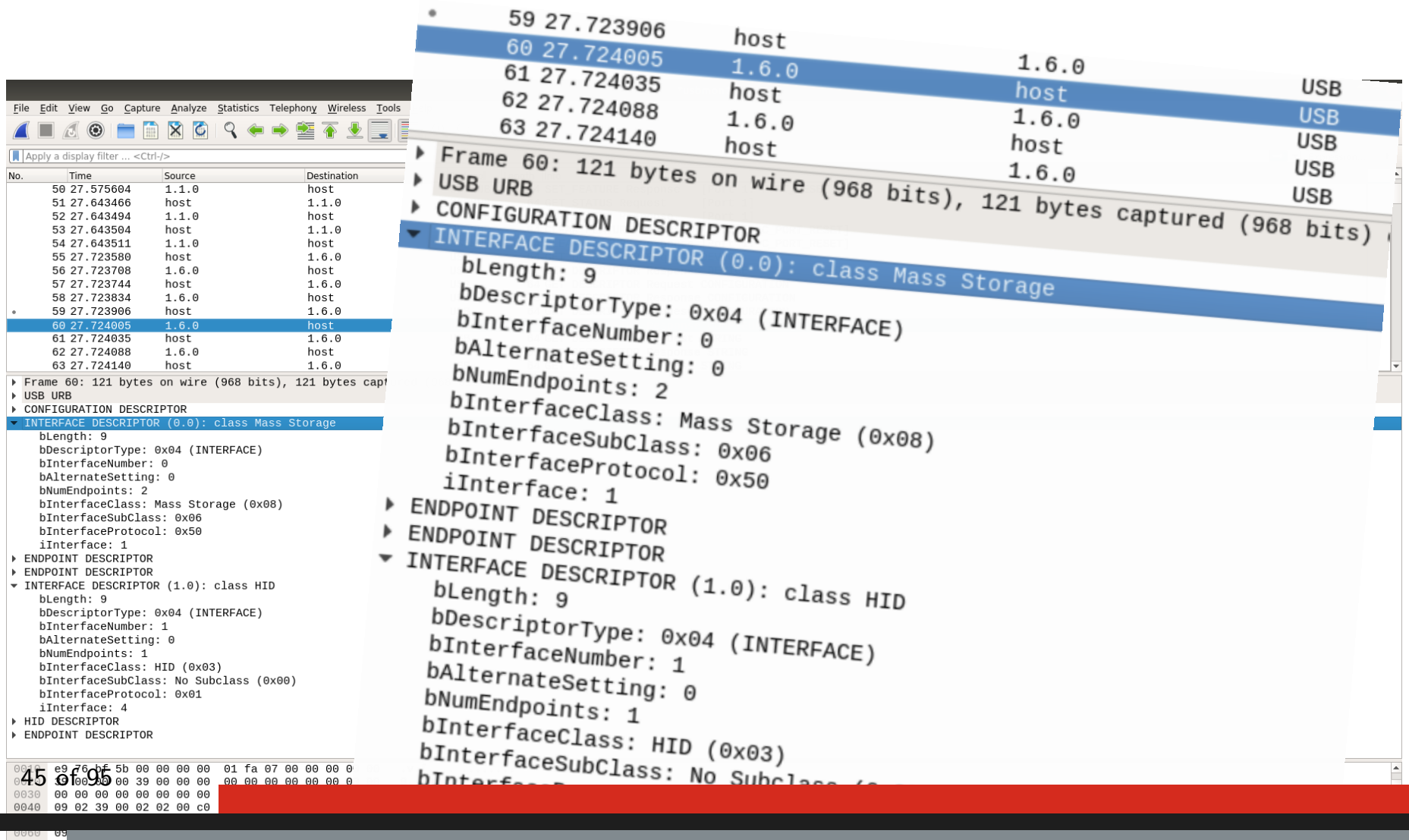
lsusb -t

```
/:  Bus 04.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/2p, 10000M
/:  Bus 03.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/2p, 480M
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/6p, 5000M
    |__ Port 1: Dev 4, If 0, Class=Mass Storage, Driver=usb-storage, 5000M
    |__ Port 2: Dev 3, If 0, Class=Mass Storage, Driver=uas, 5000M
    |__ Port 3: Dev 2, If 0, Class=Mass Storage, Driver=usb-storage, 5000M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/12p, 480M
    |__ Port 8: Dev 3, If 1, Class=Video, Driver=uvcvideo, 480M
    |__ Port 8: Dev 3, If 0, Class=Video, Driver=uvcvideo, 480M
    |__ Port 9: Dev 4, If 0, Class=Vendor Specific Class, Driver=, 12M
```

lsusb -v -d 0951:162b‘

```
...
    Interface Descriptor:
      bLength                 9
      bDescriptorType         4
      bInterfaceNumber        0
      bAlternateSetting       0
      bNumEndpoints           2
      bInterfaceClass         8 Mass Storage
      bInterfaceSubClass      6 SCSI
      bInterfaceProtocol     80 Bulk-Only
...
```

3.9 USB Interface monitoring



CIRCL
FORENSICS Training

4. Disk Cloning / Disk Imaging

4.1 Disk cloning - imaging

- Clone disk-2-disk
  - Different sizes
  - Wipe target disk!
- Clone disk-2-image
  - Clear boundaries
  - One big file
  - Break file into chunks
- Image file format
  - RAW
  - AFF (Advanced Forensic Format)
  - EWF (Expert Witness Format)
  - Please no 3rd party formats
- Write-Blockers
  - Hardware

4.2 Connecting devices

- udev

```
udevadm info /dev/sda                    # userspace /dev
udevadm monitor
```

- /dev/

```
/dev/sd*                # SCSI, SATA
/dev/hd*                # IDE. EIDE
/dev/md*                # RAID
/dev/nvme*n*            # NVME devices

/dev/sda1               # Partition 1 on disk 1
/dev/sda2               # Partition 2 on disk 1
...
```

- Block Devices
  - Attaching
  - Mounting

4.2 Read partition table

- **dmesg**

```
[106834.127269] sd 6:0:0:0: Attached scsi generic sg1 type 0
[106834.127503] sd 6:0:0:0: [sdb] 15826944 512-byte logical blocks: (8.10 GB/7.54 GiB)
[106834.130380] sd 6:0:0:0: [sdb] Write Protect is off
```

- **fdisk -l circl-dfir.dd**

```
Disk circl-dfir.dd: 1536 MB, 1536000000 bytes
4 heads, 7 sectors/track, 107142 cylinders, total 3000000 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x8f7e6594

        Device Boot      Start         End      Blocks   Id  System
circl-dfir.dd1            2048     3000000     1498976+   7  HPFS/NTFS/exFAT
```

- Exercise: Analyze output. Why 1498976? → Conclusions?

```
End:      echo $(( 3000000 * 512 ))                  --> 1536 MB, 1536000000 bytes
          echo $(( 3000000 * 512 / 1024 / 1024 ))    --> 1464

1498976:  echo $(( 1498976 * 2 ))                    --> 2997952
```

4.2 Mounting

- **mount**

```
mkdir /mnt/ntfs                                  # Create mount point
mount /dev/sdb1 /mnt/ntfs                        # Mounting

mount -o ro,remount /dev/sdb1 /mnt/ntfs          # Re-mounting

umount /mnt/ntfs                                 # Un-mounting
umount /dev/sdb1                                 # Also un-mounting


# Mounting readonly, no journaling, no executable
mount -o ro,noload,noexec /dev/sdb1 /mnt/ntfs
mount -o ro,noload,noexec,remount /dev/sdb1 /mnt/ntfs


# Mounting with offset. mounting from image files
mount -o ro,noload,noexec,offset=$((512*2048)) circl-dfir.dd /mnt/ntfs


# Mounting NTFS file systems
mount -o ro,noload,noexec,offset=$((512*2048)),
      show_sys_files,streams_interface=windows circl-dfir.dd /mnt/ntfs
```

4.3 dd - disk imaging rudimentary

Copy files from: `/mnt/ntfs/dd/`

```
$ dd if=img_1.txt of=out_1.txt bs=512

     <input file>      <output file><block size>
                                     (default)
3+0 records in
3+0 records out
1536 bytes (1.5 kB) copied, 0.000126 s, 12.2 MB/s

$ ll
-rw-rw-r-- 1 hamm hamm 1536 May 16 11:20 img_1.txt
-rw-rw-r-- 1 hamm hamm 1536 May 16 11:16 out_1.txt
```

```
$ dd if=img_2.txt of=out_2.txt bs=512
3+1 records in
3+1 records out
1591 bytes (1.6 kB) copied, 0.00016048 s, 9.9 MB/s

$ ll
-rw-rw-r-- 1 hamm hamm 1591 May 16 11:20 img_2.txt
-rw-rw-r-- 1 hamm hamm 1591 May 16 11:26 out_2.txt
```

4.3 dd - disk imaging rudimentary

Demo: `skip` and `count` options

```
dd if=img_3.txt bs=512 skip=0 count=1 status=none | less
dd if=img_3.txt bs=512 skip=1 count=1 status=none | less
dd if=img_3.txt bs=512 skip=2 count=1 status=none | less
```

Exercise: Find the secret password behind sector 3

Exercise: Play with `bs`, `skip` and `count` options

```
dd if=img_3.txt bs=1 skip=$((512*3)) count=16 status=none
dd if=img_3.txt bs=16 skip=$((32*3)) count=1 status=none
```

Exercise: `dd | xxd | less`

```
dd if=img_3.txt bs=512 skip=3 count=1 status=none | xxd | less
        0+1 records in
        0+1 records out
        55 bytes (55 B) copied, 5.04e-05 s, 1.1 MB/s

0000000: 4f76 6572 6865 6164 2031 3233 3435 3637  Overhead 1234567
0000010: 3839 3020 204d 6573 7361 6765 2d31 2020  890  Message-1
0000020: 3039 3837 3635 3433 3231 2020 2020 2020  0987654321
0000030: 2020 2020 2020 20
```

4.3 dd - disk imaging rudimentary

Demo: Continue an interrupted imaging process

```
dd if=img_2.txt of=broken.raw bs=512 skip=0 count=2 status=none

ll  img_2.txt  ..... 1591 Aug 13 14:40 img_2.txt*
ll broken.raw  ..... 1024 Aug 13 15:05 broken.raw

dd if=img_2.txt of=broken.raw bs=512 skip=2 seek=2 status=none

md5sum  img_2.txt f319b1cc9d424a923a8c83c3e67185f1
md5sum broken.raw f319b1cc9d424a923a8c83c3e67185f1
```

Error handling: Bad blocks

```
$ dd if=img_3.txt of=out_3.txt bs=512 conv=noerror,sync
```

Demo: Progress

Option: `status=progress`

Signaling: `'&' and 'kill -10'`

4.4 Disk acquisition

- Forensic features
  - Progress monitoring
  - Error handling & logging
  - Meta data
  - Splitting output files & support of forensic formats
  - Cryptographic hashing & verification checking
- Example: hashing

  `md5sum circl-dfir.dd` → `bd80672b9d1bef2f35b6e902f389e83`
  `sha1sum circl-dfir.dd` → `e5ffc7233a.....7e53b9f783`
- Tools
  - dd
  - ddrescue, gddrescue, dd_rescue
  - dc3dd - Department of Defense Cyber Crime Center
  - dcfldd - Defense Computer Forensic Labs
  - rdd-copy, netcat, socat, ssh
  - Guymager

4.5 Exercise: dc3dd

```
dc3dd if=/mnt/ntfs/carving/deleted.dd                    # Input file
      log=usb.log  -/                                    # Logging
      hash=md5 hash=sha1  -/                             # Hashing
      ofsz=$((8*1024*1024)) ofs=usb.raw.000              # Chunk files of 8MB

ls -l

cat usb.log

cat usb.raw.00* | md5sum                                 # Verify hashes
cat usb.raw.00* | sha1sum

dc3dd wipe=/dev/sdx                                      # Wipe a drive
```

4.6 SuashFS as forensic container

- Embedded systems
- Read only file system
- Supports very large files
- Adding files possible
- Deleting, modifying files not possible
- Compressed

  → Real case: 3*1TB disks stored in 293GB container
- Bruce Nikkel: `http://digitalforensics.ch/sfsimage/`

```
mksquashfs circl-dfir.dd case_123.sfs
mksquashfs analysis.txt case_123.sfs
unsquashfs -ll case_123.sfs
.....
mksquashfs analysis.txt case_123.sfs
.....
sudo mount case_123.sfs /mnt/
```

4.7 Exercise: Modify data on RO mounted device

```
mount
mount -o ro,remount /media/michael/7515-6AA5/
mount


Demo: Modify Document

strings -td /dev/sdb1
    .....
    299106 Hello World!
    .....

echo $((299106/512))
    584

dd if=/dev/sdb1 bs=512 skip=584 count=1 of=584.raw
ll
hexer 584.raw

dd of=/dev/sdb1 bs=512 seek=584 count=1 if=584.raw
mount


Demo: Review Document
```

4.7 Exercise: RO Countermeasures

- Try on board methods:
  - `hdparm -r1 /dev/sdb`
  - `blockdev --setro /dev/sdb`
  - `udev` rules
    → Attack on block device still possible
- Try Forensics Linux Distributions:
  - Live Kali 2018_4 in forensic mode
  - SANS SIFT Workstation 3.0
  - DEFT X 8.2 DFIR Toolkit
    - Some distributions do not auto mount
      → Attack on block device still possible
- Kernel Patch: Linux write blocker (not tested)
  → `https://github.com/msuhanov/Linux-write-blocker`
- Hardware Write Blocker
  → Effectively block attack

CIRCL
FORENSICS Training

5. Disk Analysis

5.1 CHS - Cylinder Head Sector

Track, Head, Cylinder, Sector, Block, Cluster

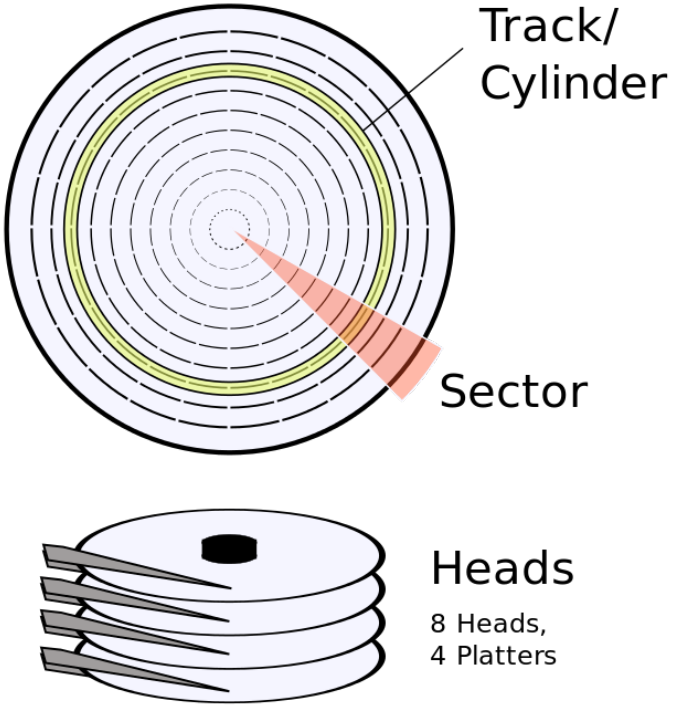


Image (c) wikipedia.org - Image used solely for illustration purposes

5.2 LBA - Logical Block Addressing

```
Logical volume addresses
-------------------------------------------------------------------------------
|  0 |  1 |  2 |  3 |  4 |  5 |  6 |  7 |  8 |  9 | 10 | 11 | 12 | 13 | 14 |
-------------------------------------------------------------------------------
          |                   |                   |                   |
          |                   |                   |                   |
          -------------------------------------------------------------
   ^      |         0         |         1         |         3         |   ^
   |      -------------------------------------------------------------   |
   |      Logical file system addresses - Clusters                        |
  MBR                                                              Volume slack
          *-------------------*
               2.048 Bytes
          *-------------------------------------------------------------*
                                   6.144 Bytes
```

5.3 Low-Level: Sector Structur

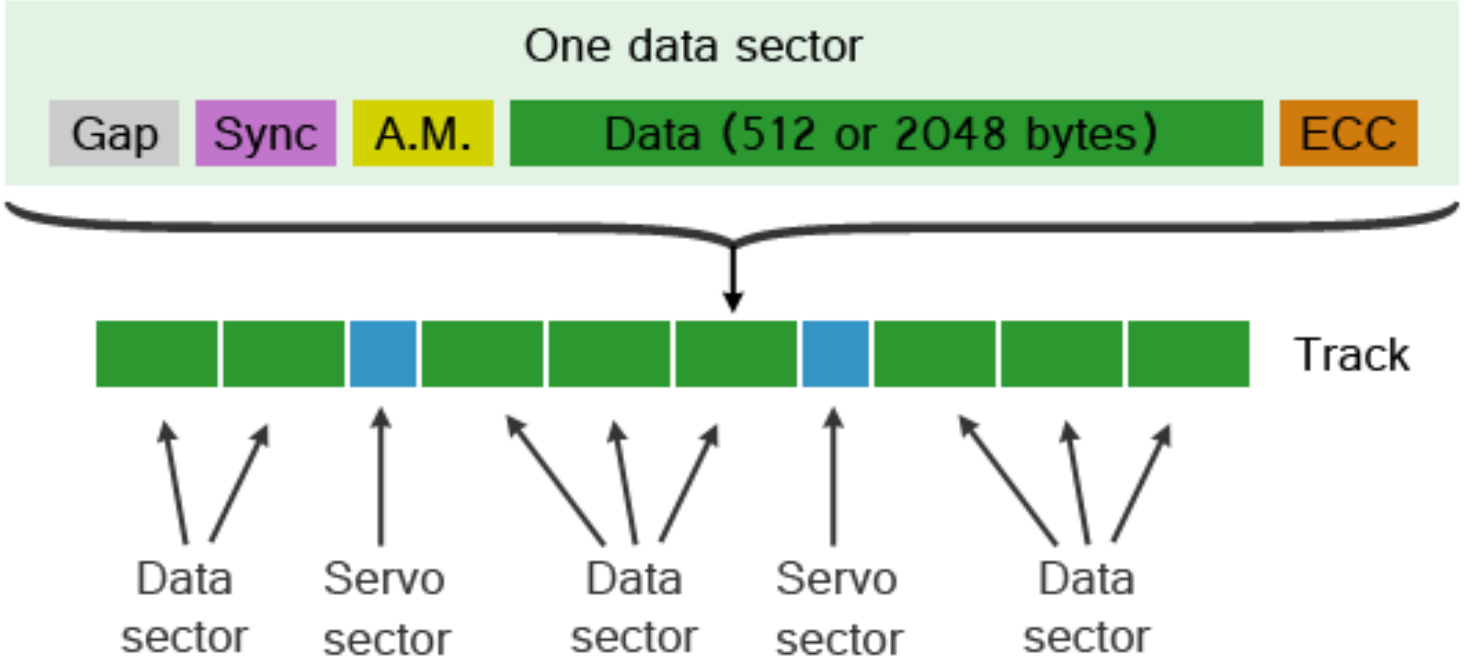


Image (c) forensicfocus.com - Image used solely for illustration purposes

5.3 Low-Level: Encoding digital data

1. FM - Frequency Modulation

```
        ______      ______      ______      ______    Clock
|      |      |      |      |      |      |      |
 ______      ______      ______      ______
.  0  .  0  .  0  .  0  .  0  .  0  .  0  .  0  .
.     .     .     .     .     .     .     .     .
        __    ______      __    __    __    ______    Data + Clock
|      |  |  |      |      |  |  |  |  |  |  |
 ______    __      ______    __    __    __
   0     1     0     0     1     1     1     0
```

2. MFM - Modified Frequency Modulation (Double Density)

```
        ______      ______      ______      ______    Clock
|      |      |      |      |      |      |      |
 ______      ______      ______      ______
.  0  .  0  .  0  .  0  .  0  .  0  .  0  .  0  .
.     .     .     .     .     .     .     .     .
           __ ______          __ __       __ ______    Data - Clock
|        |         |        |      |      |
 ______ __          ______ __       __ __
   0     1     0     0     1     1     1     0
```

3. RLL - Run Length Limited
4. PRML, EPRML - Extended Partial Response Maximun Likehood

5.4 MBR - Master Boot Record

```
# dd if=/dev/sdc bs=512 count=1 skip=0 |xxd

0000000: fab8 0010 8ed0 bc00 b0b8 0000 8ed8 8ec0  ................
0000016: fbbe 007c bf00 06b9 0002 f3a4 ea21 0600  ...|.........!..
0000032: 00be be07 3804 750b 83c6 1081 fefe 0775  ....8.u........u
0000048: f3eb 16b4 02b0 01bb 007c b280 8a74 018b  .........|...t..
0000064: 4c02 cd13 ea00 7c00 00eb fe00 0000 0000  L.....|.........
0000080: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000096: 0000 0000 0000 0000 0000 0000 0000 0000  ................
...
...
0000432: 0000 0000 0000 0000 9af0 0200 0000 0020  ...............
0000448: 2100 0b1b 0299 0008 0000 0080 2500 00a8  !...........%...
0000464: 01a8 071a b327 0058 2900 00c0 5d00 001a  ..... '.X)...]...
0000480: b427 076c dad2 0018 8700 00c0 6800 0000  .'.l........h...
0000496: 0000 0000 0000 0000 0000 0000 0000 55aa  ..............U.
```

```
000 – 439        0x000 – 0x1B7    Boot code
440 – 443        0x1B8 – 0x1BB    Disc signature
444 – 445        0x1BC – 0x1BD    Reserved
446 – 509        0x1BE – 0x1FD    Partitiontable
510 – 511        0x1FE – 0x1FF    0x55 0xAA
```

5.5 MBR - DOS Partition Table

```
# dd if=/dev/sdc bs=512 count=1 skip=0 |xxd

0000000: fab8 0010 8ed0 bc00 b0b8 0000 8ed8 8ec0   ................
0000016: fbbe 007c bf00 06b9 0002 f3a4 ea21 0600   ...|.........!..
0000032: 00be be07 3804 750b 83c6 1081 fefe 0775   ....8.u........u
0000048: f3eb 16b4 02b0 01bb 007c b280 8a74 018b   .........|...t..
0000064: 4c02 cd13 ea00 7c00 00eb fe00 0000 0000   L.....|.........
0000080: 0000 0000 0000 0000 0000 0000 0000 0000   ................
0000096: 0000 0000 0000 0000 0000 0000 0000 0000   ................
...
...
0000432: 0000 0000 0000 0000 9af0 0200 0000 0020   ................
0000448: 2100 0b1b 0299 0008 0000 0080 2500 00a8   !...........%...
0000464: 01a8 071a b327 0058 2900 00c0 5d00 001a   .....'.X)...]...
0000480: b427 076c dad2 0018 8700 00c0 6800 0000   .'.l........h...
0000496: 0000 0000 0000 0000 0000 0000 0000 55aa   ..............U.
```

```
Partitiontable:
  Offset: 0     Size: 1 Value: 0x80    --> Bootable
  Offset: 1     Size: 3 Value:         --> Starting CHS address
  Offset: 4     Size: 1 Value: 0x0b    --> FAT32
                                0x07    --> NTFS
  Offset: 5     Size: 3 Value:         --> Ending CHS address
  Offset: 8     Size: 4 Value:         --> Starting LBA address
  Offset:12     Size: 4 Value:         --> LBA size in sectors
```

5.5 MBR - DOS Partition Table

```
0000432: 0000 0000 0000 0000 9af0 0200 0000 0020   ................
0000448: 2100 0b1b 0299 0008 0000 0080 2500 00a8   !...........%...
0000464: 01a8 071a b327 0058 2900 00c0 5d00 001a   .....'.X)...]...
0000480: b427 076c dad2 0018 8700 00c0 6800 0000   .'.l........h...
0000496: 0000 0000 0000 0000 0000 0000 0000 55aa   ..............U.

Partitiontable:
  Offset: 0     Size: 1 Value: 0x80     --> Bootable
  Offset: 1     Size: 3 Value:          --> Starting CHS address
  Offset: 4     Size: 1 Value: 0x0b     --> FAT32
                                0x07     --> NTFS
  Offset: 5     Size: 3 Value:          --> Ending CHS address
  Offset: 8     Size: 4 Value:          --> Starting LBA address
  Offset:12     Size: 4 Value:          --> LBA size in sectors

Addressable space:
  CHS: echo $((2**8 * 2**6 * 2**10 * 512 / 1024**2))  ==  8192 MByte
  LBA: echo $((2**32 * 512 / 1024**3))                ==  2048 GByte
```

- Exercise: Calculate the size if the partitions
  1. Take LBA size
  2. Apply Little Endian
  3. Apply sector size

5.5 MBR - DOS Partition Table

```
0000432: 0000 0000 0000 0000 9af0 0200 0000 0020   ................
0000448: 2100 0b1b 0299 0008 0000 0080 2500 00a8   !...........%...
0000464: 01a8 071a b327 0058 2900 00c0 5d00 001a   .....'.X)...]...
0000480: b427 076c dad2 0018 8700 00c0 6800 0000   .'.l........h...
0000496: 0000 0000 0000 0000 0000 0000 0000 55aa   ..............U.
```

- Exercise: Calculate the size if the partitions

	LBA size	Little Endian		Sector size		
Part1:	0x00802500	0x00258000	2457600	* 512	1258291200	1.2 GB
Part2:	0x00c05d00	0x005dc000	6144000	* 512	3145728000	3.0 GB
Part3:	0x00c06800	0x0068c000	6864896	* 512	3514826752	3.4 GB

- Demo: Change partition type with hexeditor

  `fdisk -l /dev/sdb; hexedit /dev/sdb; F2, CTRL+x`

- Exercise: Find password in unused space before first partition

5.6 Extended Partition - EBR

```
...
0000432: 0000 0000 0000 0000 0000 0000 0000 0020  ................
0000448: 2100 0b1b 0299 0008 0000 0080 2500 00a8  !...........%...
0000464: 01a8 071a b327 0058 2900 00c0 5d00 0000  .....'.X)...]...
0000480: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000496: 0000 0000 0000 0000 0000 0000 0000 55aa  ..............U.
```

```
Partition table:
446 - 461        0x1BE - 0x1CD    1th entry - This logical partition
462 - 477        0x1CE - 0x1DD    2nd entry - Empty OR Next EBR - Extended Boot Record
478 - 493        0x1DE - 0x1ED    Unused
494 - 509        0x1EE - 0x1FD    Unused
```

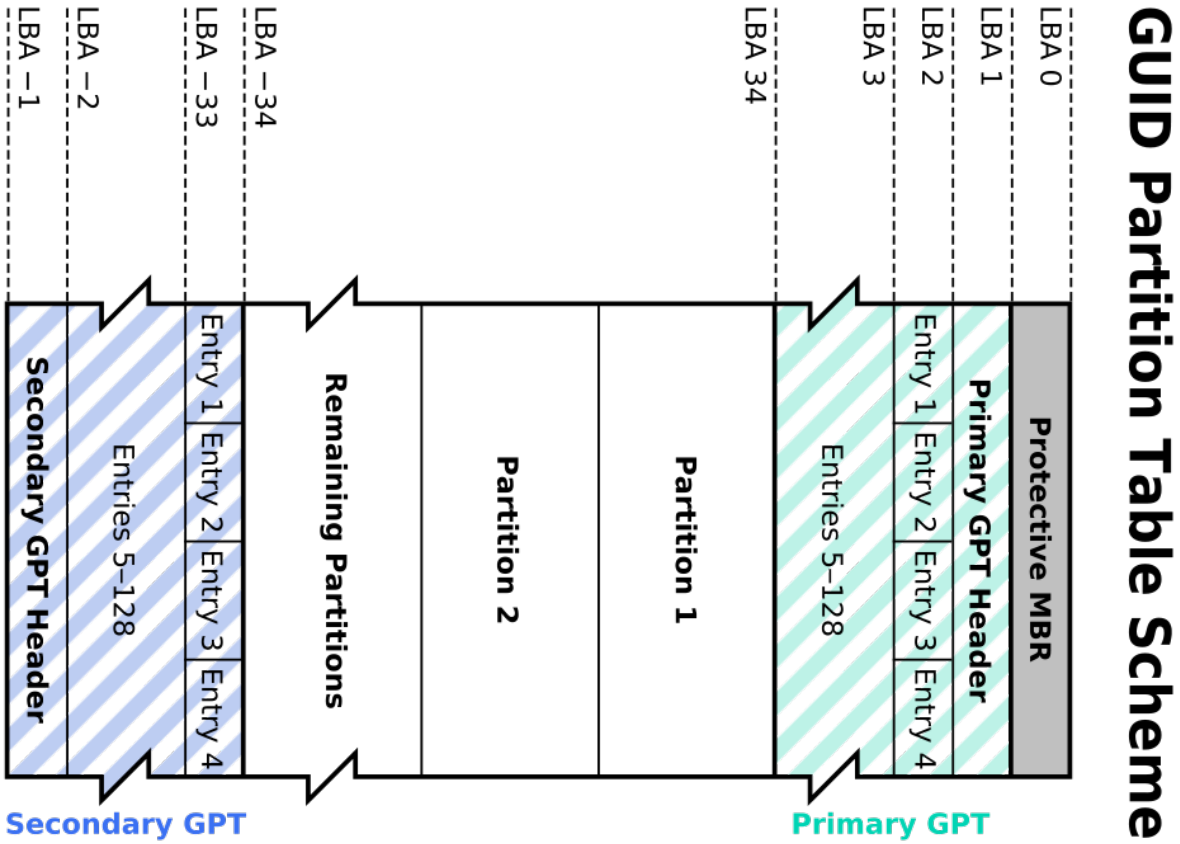
```
| Extended Partition                                                    |
| EBR | Logical        | Extended partition                           |
|     | Partition      |
| --> |                | EBR | Logical          | Extended partition |
| ----+----------------> |     |                  |
|     |                | --> |                  | EBR | Logical      |
|     |                | ----+------------------> |     |              |
|     |                |     |                  | --> |              |
|     |                |     |                  | 000 |              |
```

5.7 GPT - GUID Partition Table

- BIOS → UEFI - Unified Extensible Firmware Interface
- GUID - Globally Unique Identifier for each partition
  → GUID Partition Table
- Protective MBR at LBA0
  - One single entry covering the entire disk
  - Partition type 0xEE
    if 0xEE unknown → Not empty → Not formatted
- GPT header at LBA1
- GPT entries at LBA2 → LBA34
- GPT entries: 128 Bytes
- GPT backup at end of disk

5.7 GPT - GUID Partition Table

Figure: Image (c) wikipedia.org - Image used solely for illustration purposes



5.8 Exercise: Investigate disk with strange PT

- Fix the first partition table entry! `mmls mbr_ex.raw`

```
      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  --------  0000000000   0000002049   0000002050   Unallocated
002:  000:000   0000002050   0000067585   0000065536   Win95 FAT32 (0x0c)
003:  000:001   0000067586   0000133119   0000065534   Win95 FAT32 (0x0c)
004:  000:002   0000133120   0000262142   0000129023   Win95 FAT32 (0x0c)
005:  --------  0000262143   0000262143   0000000001   Unallocated
```

- Search for partition 1 signature

```
sigfind -o 510 -l AA55 mbr_ex.raw
```

5.8 Exercise: Investigate disk with strange PT

- The fixed partition table:

	Slot	Start	End	Length	Description
000:	Meta	0000000000	0000000000	0000000001	Primary Table (#0)
001:	———	0000000000	0000002047	0000002048	Unallocated
002:	000:000	0000002048	0000067583	0000065536	Win95 FAT32 (0×0c)
003:	———	0000067584	0000067585	0000000002	Unallocated
004:	000:001	0000067586	0000133119	0000065534	Win95 FAT32 (0×0c)
005:	000:002	0000133120	0000262142	0000129023	Win95 FAT32 (0×0c)
006:	———	0000262143	0000262143	0000000001	Unallocated

- Investigate partition 3 boundaries

```
dd if=mbr_ex.raw count=2049 | xxd | less
dd if=mbr_ex.raw skip=67583 count=4 | xxd | less
dd if=mbr_ex.raw skip=262142 | xxd | less
```

5.9 VBR - Volume Boot Record - Boot Sector

```
# dd if=/dev/sdc1 bs=512 count=1 skip=0 |xxd

0000000: eb58 906d 6b64 6f73 6673 0000 0208 2000  .X.mkdosfs.... .     # 0xeb 0x58 0x90
0000010: 0200 0000 00f8 0000 3e00 f800 0000 0000  ........>.......     # JMP  2+88 NOP
0000030: 0100 0600 0000 0000 0000 0000 0000 0000  ................
0000040: 0000 29a2 20e9 9c46 4154 2020 2020 2020  ..). ..FAT
0000050: 2020 4641 5433 3220 2020 0e1f be77 7cac    FAT32   ...w|.
0000060: 22c0 740b 56b4 0ebb 0700 cd10 5eeb f032  ".t.V.......^..2
...
...
00001f0: 0000 0000 0000 0000 0000 0000 0000 55aa  ..............U.


0 - 2            Size:  3    Jump to bootstrap code
3 - 10           Size:  8    OEM-ID: mkdosfs
11 - 12          Size:  2    Bytes per sector: 0x0002 -> 0x0200 (little endian)-> 512
13 (0xD)         Size:  1    Sectors per cluster: 0x08 -> 4096 bytes per cluster
50 (0x32) - 51   Size:  2    Boot sector backup: 0x0600 -> 0x0006 -> at sector 6
67 (0x43) - 70   Size:  4    Volume serial number: 0xa220e99c -> 0x9ce920a2
71 (0x47)        Size: 11    Volume label: FAT
82 (0x52)        Size:  8    Partition type: FAT32
90 (0x5A)- 509 (0x1FD)       Bootstrap code
510 (0x1FE)      Size:  2    Signature: 0x55AA
```

- Demo: Sleuthkit tools: `mmstat`, `mmls`, `fsstat`

CIRCL
FORENSICS Training

9. String Search

9.1 What is 'String Search'?

- Search the disk image for known words
  - Terms used in a secret document
  - IBAN ot other banking details
  - Email addresses or URLs
  - File names or shell commands

- Search thrue all the blocks
  - Allocated blocks
  - File slack
  - Non allocated blocks
  - Outside the partition borders

- Goal
  - Proof that the data was there once
  - May even recover deleted files
  - Identify intresting data that are close

9.2 Steps to do a String Search

1. Identify block/cluster size
   `mmls, fsstat`
2. Search for the string and the offset
   `blkls | srch_strings | grep`
3. Calculate block/cluster of the string
   `xxxxxxxxxx / 4096 = yyyy`
4. Review block/cluster content
   `blkcat`
5. Identify inode of the block/cluster
   `ifind`
6. Identify associated file
   `ffind`
7. Recover file
   `icat`
   Or mount and copy file

9.3 Exercise: What about Paulas cat?

1. Identify cluster size

```
mmls circl-dfir.dd

     1       Slot      Start        End          Length       Description
 000:  Meta        0000000000   0000000000   0000000001   Primary Table (#0)
 001:  --------    0000000000   0000002047   0000002048   Unallocated
 002:  000:000     0000002048   0004917247   0004915200   NTFS / exFAT (0x07)
```

```
fsstat -o 2048 circl-dfir.dd

     File System Type: NTFS
     Volume Serial Number: 7B6E5F9427919882
     OEM Name: NTFS
     Volume Name: CIRCL-DFIR
     Version: Windows XP

     .....

     Sector Size: 512
     Cluster Size: 4096
     Total Cluster Range: 0 - 614398
     Total Sector Range: 0 - 4915198
```

9.3 Exercise: What about Paulas cat?

2. Search for the string `'Paula'`

```
blkls -e -o 2048 circl-dfir.dd | strings -a -td | grep -i paula

     157342 Paula's cat is fat.........
     157370 Paula's cat is fat.........
     .....
     157510 Paula's cat is fat.........
     157538 Paula's cat is fat.........
```

3. Calculate cluster of the string

```
echo $((157342/4096))
     38

echo $((157538/4096))
     38
```

4. Review cluster content

```
blkcat -o 2048 circl-dfir2dd 38 | strings
     .....
     Paula's cat is fat.........
     Paula's cat is fat.........
     Paula's cat is fat.........
     .....
```

9.3 Exercise: What about Paulas cat?

5. Identify inode of the cluster

```
ifind -o 2048 -d 38 circl-dfir.dd
  0-128-1
```

6. Identify associated file

```
ffind -o 2048 circl-dfir.dd 0-128-1
  //$MFT
```

7. Recover file

```
icat -o 2048 circl-dfir.dd 0-128-1 > MFT
```

Exercise: Manual approach - Learn from errors

```
dd if=circl-dfir.dd bs=4096 skip=38 count=1 | xxd | less
dd if=circl-dfir.dd bs=4096 skip=$((2048 + 38)) count=1 | xxd | less
dd if=circl-dfir.dd bs=4096 skip=$((2048/8 + 38)) count=1 | xxd | less
```

CIRCL
FORENSICS Training

12. Memory Forensics

12.1 About Memory Forensics

- Information expected
  - Network connections
  - Processes (hidden)
  - Services (listening)
  - Malware
  - Registry content
  - DLL analysis
  - Passwords in clear text
- History
  - 2005: String search
  - $\rightarrow$ EProcess structures
- Finding EProcess structures
  - Find the doubly linked list (ntoskrnl.exe)
  - Brute Force searching

12.2 Get your memory dump

- Page file, swap area: `pagefile.sys`
- Memory dump

  `http://www.msuiche.net`

  `DumpIt.exe`

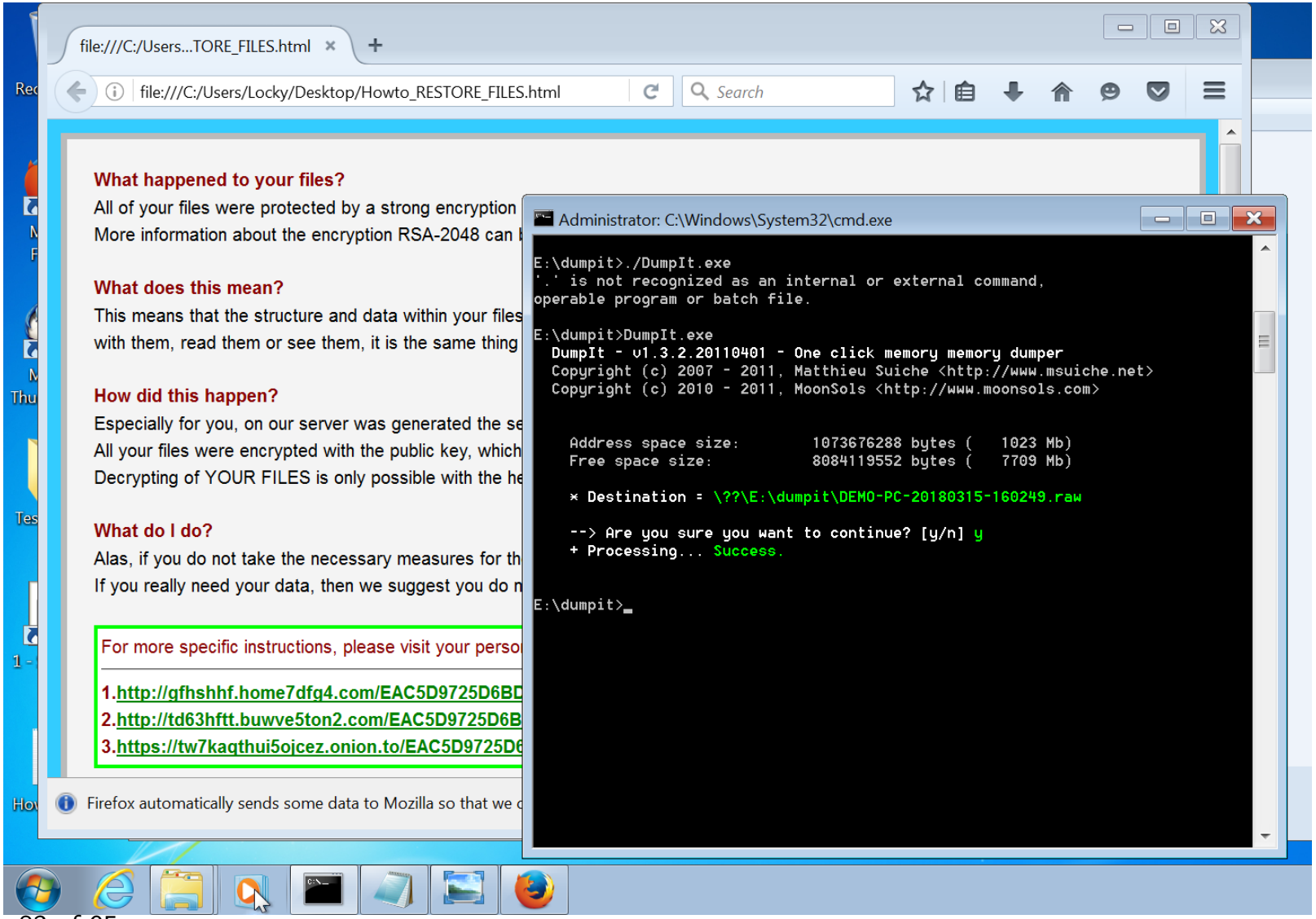
```
E:\dumpit>DumpIt.exe
  DumpIt - v1.3.2.20110401 - One click memory memory dumper
  Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.msuiche.net>
  Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>


    Address space size:        1073676288 bytes (   1023 Mb)
    Free space size:           2401239040 bytes (   2290 Mb)

    * Destination = \??\E:\dumpit\WIN7WS-20190411-151517.raw

    --> Are you sure you want to continue? [y/n] y
    + Processing... Success.


E:\dumpit>
```

- Hibernation file: `hiberfil.sys`

  `powercfg /h[ibernate] [on|off]`

  `psshutdown -h`

12.2 DumpIt


file:///C:/Users...TORE_FILES.html
file:///C:/Users/Locky/Desktop/Howto_RESTORE_FILES.html
Search
What happened to your files?
All of your files were protected by a strong encryption
More information about the encryption RSA-2048 can
What does this mean?
This means that the structure and data within your files
with them, read them or see them, it is the same thing
How did this happen?
Especially for you, on our server was generated the se
All your files were encrypted with the public key, which
Decrypting of YOUR FILES is only possible with the he
What do I do?
Alas, if you do not take the necessary measures for th
If you really need your data, then we suggest you do n
For more specific instructions, please visit your perso
1.http://gfhshhf.home7dfg4.com/EAC5D9725D6BD
2.http://td63hftt.buwve5ton2.com/EAC5D9725D6B
3.https://tw7kaqthui5ojcez.onion.to/EAC5D9725D6
Firefox automatically sends some data to Mozilla so that we
Administrator: C:\Windows\System32\cmd.exe
E:\dumpit>./DumpIt.exe
'.' is not recognized as an internal or external command,
operable program or batch file.
E:\dumpit>DumpIt.exe
DumpIt - v1.3.2.20110401 - One click memory memory dumper
Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.msuiche.net>
Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>
Address space size: 1073676288 bytes (1023 Mb)
Free space size: 8084119552 bytes (7709 Mb)
* Destination = \??\E:\dumpit\DEMO-PC-20180315-160249.raw
--> Are you sure you want to continue? [y/n] y
+ Processing... Success.
E:\dumpit>

12.3 Mandiant Redline - Malware Risk Index

Process Name	MRI Score	PID	Path	Arguments	Start Time
owxxb-a.exe	93	3432	C:\Users\John\AppData\Roaming	C:\Users\John\AppData\Roaming\owxxb-a.exe	04/15/2019 15:07:13
svchost.exe	93	3728	C:\Windows\System32	C:\Windows\System32\svchost.exe -k swprv	04/15/2019 15:07:23
csrss.exe	59	360	C:\Windows\system32	%SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024...	04/15/2019 15:02:54
csrss.exe	57	324	C:\Windows\system32	%SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024...	04/15/2019 15:02:54
Explorer.EXE	56	920	C:\Windows	C:\Windows\Explorer.EXE	04/15/2019 15:03:42
svchost.exe	55	2884	C:\Windows\System32	C:\Windows\System32\svchost.exe -k secsvcs	04/15/2019 15:05:41
powershell.exe	52	2748	C:\Windows\System32\WindowsPowerSh...	powershell	04/15/2019 15:05:26
spoolsv.exe	52	1296	C:\Windows\System32	C:\Windows\System32\spoolsv.exe	04/15/2019 15:03:02
lsass.exe	52	464	C:\Windows\system32	C:\Windows\system32\lsass.exe	04/15/2019 15:02:55
svchost.exe	52	852	C:\Windows\system32	C:\Windows\system32\svchost.exe -k netsvcs	04/15/2019 15:02:58
WzPreloader.exe	52	1852	C:\Program Files\WinZip	"C:\Program Files\WinZip\WzPreloader.exe"	04/15/2019 15:03:44
svchost.exe	47	1444	C:\Windows\system32	C:\Windows\system32\svchost.exe -k LocalServiceAndNoImpersonation	04/15/2019 15:03:03
services.exe	47	456	C:\Windows\system32	C:\Windows\system32\services.exe	04/15/2019 15:02:55

12.3 Mandiant Redline - Malware Risk Index

Process Name	PID	Path	State	Created	Local IP Address	Local...	Remote IP Add...	Re...	Protocol
owxxb-a.exe	3432	C:\Users\John\AppData\Roaming	ESTABLISHED		10.0.2.15	49161	216.239.32.21	443	TCP
owxxb-a.exe	3432	C:\Users\John\AppData\Roaming	CLOSED		10.0.2.15	49164	139.99.68.76	80	TCP
owxxb-a.exe	3432	C:\Users\John\AppData\Roaming	ESTABLISHED		10.0.2.15	49160	216.239.32.21	80	TCP
owxxb-a.exe	3432	C:\Users\John\AppData\Roaming	ESTABLISHED		10.0.2.15	49162	2.17.201.8	80	TCP

12.3 Mandiant Redline - Malware Risk Index

Malware Risk Index Report

svchost.exe (3728)

Process Details

Username:	
Path:	C:\Windows\System32
Parent:	services.exe (456)
Parent Process Path:	C:\Windows\system32
Arguments:	C:\Windows\System32\svchost.exe -k swprv
Start Time:	2019-04-15 15:07:23Z
Kernel Time Elapsed:	00:00:00
User Time Elapsed:	00:00:00
SID:	S-1-5-18
SID Type:	
Malware Risk Index:	93

Malware Risk Index Hits

This process was spawned with unexpected arguments: "C:\Windows\System32\svchost.exe -k swprv"

Named Memory Sections

Negative Factors	45%
Positive Factors	55%
Ignored Factors	0%

12.3 Mandiant Redline - Hierarchical

System	0	4		04/15/2019 15:02:52		0
smss.exe	47	248	\SystemRoot\System32\smss.exe	04/15/2019 15:02:52	System	4
csrss.exe	57	324	%SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection...	04/15/2019 15:02:54		308
wininit.exe	47	368	wininit.exe	04/15/2019 15:02:54		308
services.exe	47	456	C:\Windows\system32\services.exe	04/15/2019 15:02:55	wininit.exe	368
taskhost.exe	47	352	"taskhost.exe"	04/15/2019 15:03:42	services.exe	456
csrss.exe	59	360	%SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection...	04/15/2019 15:02:54	taskhost.exe	352
conhost.exe	47	2552	\??\C:\Windows\system32\conhost.exe	04/15/2019 15:04:43	csrss.exe	360
winlogon.exe	47	396	winlogon.exe	04/15/2019 15:02:54	taskhost.exe	352
svchost.exe	47	564	C:\Windows\system32\svchost.exe -k DcomLaunch	04/15/2019 15:02:57	services.exe	456
wmiprvse.exe	47	3268		04/15/2019 15:06:52	svchost.exe	564
VBoxService.exe	47	624	C:\Windows\System32\VBoxService.exe	04/15/2019 15:02:57	services.exe	456

powershell.exe	52	2748	powershell	04/15/2019 15:05:26		2544
owxxb-a.exe	93	3432	C:\Users\John\AppData\Roaming\owxxb-a.exe	04/15/2019 15:07:13		3368
NOTEPAD.EXE	52	3820	"C:\Windows\system32\NOTEPAD.EXE" C:\Users\John\Desktop\Howto_RESTORE_FILES.txt	04/15/2019 15:08:05	owxxb-a.exe	3432
iexplore.exe	52	3832	"C:\Program Files\Internet Explorer\iexplore.exe" -nohome	04/15/2019 15:08:06	owxxb-a.exe	3432
iexplore.exe	47	3908	"C:\Program Files\Internet Explorer\iexplore.exe" SCODEF:3832 CREDAT:14337	04/15/2019 15:08:07	iexplore.exe	3832

12.3 Mandiant Redline - Timeline

04/15/2019 15:05:26	Process/StartTime	**Name:** powershell.exe	**PID:** 2748	**Path:** C:\Windows\System32\WindowsPowerShell\v1.0			**Args:** powershell	
04/15/2019 15:05:41	Process/StartTime	**Name:** svchost.exe	**PID:** 2884	**Path:** C:\Windows\System32			**Args:** C:\Windows\System32\svchost.exe -k secsvcs	
04/15/2019 15:05:41	Process/StartTime	**Name:** sppsvc.exe	**PID:** 2844	**Path:** C:\Windows\system32			**Args:** C:\Windows\system32\sppsvc.exe	
04/15/2019 15:06:50	Port/CreationTime	**Remote:** *:*:0	**Local:** 0.0.0.0:0	**Protocol:** UDP	**State:** LISTENING	**PID:** 2748		**Process:** powershell.exe
04/15/2019 15:06:50	Port/CreationTime	**Remote:** *:*:0	**Local:** 00:00:00:00:00:00:00:00:0	**Protocol:** UDP	**State:** LISTENING	**PID:** 2748		**Process:** powershell.exe
04/15/2019 15:06:50	Port/CreationTime	**Remote:** *:*:0	**Local:** 0.0.0.0:0	**Protocol:** UDP	**State:** LISTENING	**PID:** 2748		**Process:** powershell.exe
04/15/2019 15:06:50	Port/CreationTime	**Remote:** *:*:0	**Local:** 00:00:00:00:00:00:00:00:0	**Protocol:** UDP	**State:** LISTENING	**PID:** 2748		**Process:** powershell.exe
04/15/2019 15:06:52	Process/StartTime	**Name:** wmiprvse.exe	**PID:** 3268	**Path:** C:\Windows\system32\wbem			**Args:**	
04/15/2019 15:07:13	Process/StartTime	**Name:** owxxb-a.exe	**PID:** 3432	**Path:** C:\Users\John\AppData\Roaming			**Args:** C:\Users\John\AppData\Roaming\owxxb-a.exe	
04/15/2019 15:07:22	Process/StartTime	**Name:** vssvc.exe	**PID:** 3676	**Path:** C:\Windows\system32			**Args:** C:\Windows\system32\vssvc.exe	
04/15/2019 15:07:23	Process/StartTime	**Name:** svchost.exe	**PID:** 3728	**Path:** C:\Windows\System32			**Args:** C:\Windows\System32\svchost.exe -k swprv	

04/15/2019 15:07:13	**Name:** owxxb-a.exe	**PID:** 3432	**Path:** C:\Users\John\AppData\Roaming	**Args:** C:\Users\John\AppData\Roaming\owxxb-a.exe
04/15/2019 15:07:22	**Name:** vssvc.exe	**PID:** 3676	**Path:** C:\Windows\system32	**Args:** C:\Windows\system32\vssvc.exe
04/15/2019 15:07:23	**Name:** svchost.exe	**PID:** 3728	**Path:** C:\Windows\System32	**Args:** C:\Windows\System32\svchost.exe -k swprv
04/15/2019 15:08:05	**Name:** NOTEPAD.EXE	**PID:** 3820	**Path:** C:\Windows\system32	**Args:** "C:\Windows\system32\NOTEPAD.EXE" C:\Users\John\Desktop\H...
04/15/2019 15:08:06	**Name:** iexplore.exe	**PID:** 3832	**Path:** C:\Program Files\Internet Explorer	**Args:** "C:\Program Files\Internet Explorer\iexplore.exe" -nohome
04/15/2019 15:08:07	**Name:** iexplore.exe	**PID:** 3908	**Path:** C:\Program Files\Internet Explorer	**Args:** "C:\Program Files\Internet Explorer\iexplore.exe" SCODEF:3832 C...
04/15/2019 15:08:07	**Name:** DllHost.exe	**PID:** 3928	**Path:** C:\Windows\system32	**Args:** C:\Windows\system32\DllHost.exe /Processid:{AB8902B4-09CA-4...

12.4 Volatility: Overview

```
volatility -h

...
imagecopy       Copies a physical address space out as a raw DD image
imageinfo       Identify information for the image
...
pslist          Print all running processes by following the EPROCESS lists
psscan          Scan Physical memory for _EPROCESS pool allocations
pstree          Print process list as a tree
psxview         Find hidden processes with various process listings
...
sockets         Print list of open sockets
sockscan        Scan Physical memory for _ADDRESS_OBJECT objects (tcp sockets)
...
```

```
volatility -f [filename] [plugin] [options]

volatility -f DEMO-PC-20180315.raw imageinfo
```

12.4 Volatility: Overview

```
volatility -f Win-Enc-20190415.raw imageinfo
```

```
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...

          Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
                     AS Layer1 : IA32PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace
                      PAE type : No PAE
                           DTB : 0x185000L
                          KDBG : 0x82968c28L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0x82969c00L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2019-04-15 15:08:11 UTC+0000
     Image local date and time : 2019-04-15 17:08:11 +0200
```

```
volatility --profile=Win7SP1x86 -f [filename] [plugin]
[options]
```

12.5 Volatility: Process Analysis

`pslist`

- Running processes
- Process IP - PID
- Parent PIP - PPID
- Start time

`pstree`

- Like `pslist`
- Visual child-parent relation

`psscan`

- Brute Force
- Find inactive and/or hidden processes

`psxview`

- Run and compare some tests
- Correlate `psscan` and `pslist`

12.5 Volatility: Process Analysis

```
volatility --profile=Win7SP1x86 -f Win-Enc-20190415.raw pslist
```

Offset(V)	Name	PID	PPID	Thds	Hnds	Ses	Wow64	Start
0x84233af0	System	4	0	70	505	——	0	2019-04-15 15:02:52 UTC+0000
0x848d8288	smss.exe	248	4	2	29	——	0	2019-04-15 15:02:52 UTC+0000
0x8487a700	csrss.exe	324	308	9	384	0	0	2019-04-15 15:02:54 UTC+0000
0x84fbb530	csrss.exe	360	352	7	274	1	0	2019-04-15 15:02:54 UTC+0000
0x84fc3530	wininit.exe	368	308	3	77	0	0	2019-04-15 15:02:54 UTC+0000
0x84fd0530	winlogon.exe	396	352	4	112	1	0	2019-04-15 15:02:54 UTC+0000
0x85048a18	services.exe	456	368	8	203	0	0	2019-04-15 15:02:55 UTC+0000
0x8505ac00	lsass.exe	464	368	7	580	0	0	2019-04-15 15:02:55 UTC+0000
0x8505caa0	lsm.exe	472	368	10	145	0	0	2019-04-15 15:02:55 UTC+0000
...								
...								
...								
0x85050b60	WmiPrvSE.exe	3268	564	9	175	0	0	2019-04-15 15:06:52 UTC+0000
0x8438bd40	owxxb-a.exe	3432	3368	15	471	1	0	2019-04-15 15:07:13 UTC+0000
0x84394030	VSSVC.exe	3676	456	6	123	0	0	2019-04-15 15:07:22 UTC+0000
0x84394488	svchost.exe	3728	456	6	70	0	0	2019-04-15 15:07:23 UTC+0000
0x84a243c8	notepad.exe	3820	3432	1	64	1	0	2019-04-15 15:08:05 UTC+0000
0x846d8030	iexplore.exe	3832	3432	19	427	1	0	2019-04-15 15:08:06 UTC+0000
0x846d2d40	iexplore.exe	3908	3832	11	293	1	0	2019-04-15 15:08:07 UTC+0000
0x846e5a58	dllhost.exe	3928	564	6	94	1	0	2019-04-15 15:08:07 UTC+0000
0x84684d40	dllhost.exe	4012	564	10	212	1	0	2019-04-15 15:08:08 UTC+0000

12.5 Volatility: Process Analysis

```
volatility --profile=Win7SP1x86 -f Win-Enc-20190415.raw pslist
```

Offset(P)	Name	PID	pslist	psscan	thrdproc	pspcid	csrss	session	deskthrd
.....									
.....									
0x3f60f030	taskhost.exe	352	True	True	True	True	True	True	True
0x3fa84d40	dllhost.exe	4012	True	True	True	True	True	True	True
0x3ec23148	spoolsv.exe	1296	True	True	True	True	True	True	True
0x3f63f470	explorer.exe	920	True	True	True	True	True	True	True
0x3ff0bd40	owxxb—a.exe	3432	True	True	True	True	True	True	True
0x3f3d0530	winlogon.exe	396	True	True	True	True	True	True	True
0x3f3c3530	wininit.exe	368	True	True	True	True	True	True	True
0x3ec9f030	svchost.exe	688	True	True	True	True	True	True	True
0x3ef3d758	VBoxTray.exe	1832	True	True	True	True	True	True	True
0x3fae5a58	dllhost.exe	3928	True	True	True	True	True	True	True
0x3ec50b60	WmiPrvSE.exe	3268	True	True	True	True	True	True	True
0x3ec88b90	svchost.exe	564	True	True	True	True	True	True	True
0x3ecd3768	svchost.exe	820	True	True	True	True	True	True	True
0x3ef4f030	SearchIndexer.	2008	True	True	True	True	True	True	True
0x3ec08d40	svchost.exe	1444	True	True	True	True	True	True	True
0x3ed10d40	svchost.exe	1008	True	True	True	True	True	True	True
0x3f6243c8	notepad.exe	3820	True	True	True	True	True	True	True
0x3ecd95f8	svchost.exe	852	True	True	True	True	True	True	True
0x3fad2d40	iexplore.exe	3908	True	True	True	True	True	True	True
.....									
.....									

12.6 Volatility: Network Analysis

- Windows XP and 2003 Server
  - `connections`
  - `connscan`
  - `sockets`
- Windwos 7
  - `netscan`

```
volatility --profile=Win7SP1x86 -f Win-Enc-20190415.raw netscan
```

```
Proto    Local Address        Foreign Address      State            Pid      Owner
.....
UDPv4    0.0.0.0:0            *:*                                   2748     powershell.exe
UDPv6    :::0                 *:*                                   2748     powershell.exe
TCPv4    0.0.0.0:49155        0.0.0.0:0            LISTENING         456     services.exe
TCPv4    0.0.0.0:49156        0.0.0.0:0            LISTENING         464     lsass.exe
TCPv6    :::49156             :::0                 LISTENING         464     lsass.exe
TCPv4    10.0.2.15:49167      2.17.201.11:80       ESTABLISHED      1128     svchost.exe
TCPv4    10.0.2.15:49166      93.184.220.29:80     ESTABLISHED      1128     svchost.exe
TCPv4    10.0.2.15:49165      50.62.124.1:80       ESTABLISHED      3432     owxxb-a.exe
TCPv4    10.0.2.15:49160      216.239.32.21:80     ESTABLISHED      3432     owxxb-a.exe
TCPv4    10.0.2.15:49162      2.17.201.8:80        ESTABLISHED      3432     owxxb-a.exe
TCPv4    10.0.2.15:49168      13.107.21.200:80     ESTABLISHED      3832     iexplore.exe
TCPv4    10.0.2.15:49159      94.23.7.52:80        CLOSE_WAIT       2748     powershell.exe
.....
```

12.7 Volatility: Exercise

```
volatility --profile=Win7SP1x86 -f Win-Enc-20190415.raw malfind
```

```
Process: owxxb-a.exe Pid: 3432 Address: 0x400000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 134, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00400000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00400010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00400020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00400030  00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00   ................

0x00400000 4d                DEC EBP
0x00400001 5a                POP EDX
0x00400002 90                NOP
```

```
volatility --profile=Win7SP1x86 -f Win-Enc-20190415.raw getsids
```

```
powershell.exe (2748): S-1-5-21-3408732720-2018246097-660081352-1000 (John)
owxxb-a.exe (3432): S-1-5-21-3408732720-2018246097-660081352-1000 (John)
notepad.exe (3820): S-1-5-21-3408732720-2018246097-660081352-1000 (John)
iexplore.exe (3832): S-1-5-21-3408732720-2018246097-660081352-1000 (John)
iexplore.exe (3908): S-1-5-21-3408732720-2018246097-660081352-1000 (John)
dllhost.exe (3928): S-1-5-21-3408732720-2018246097-660081352-1000 (John)
```

Create memdump of malicious process and search for suspicious URLs!