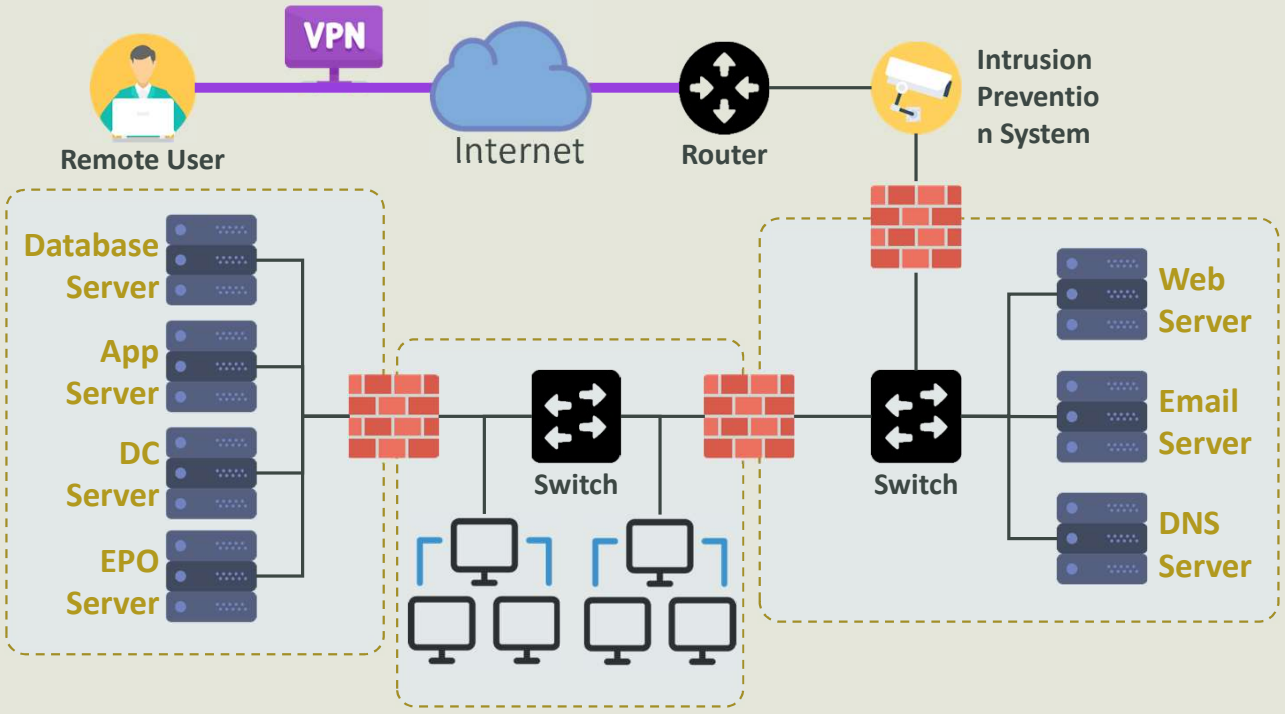
# Securitate cibernetică

Modul 7: Investigații informatice specifice efectuate la nivel de sistem și/sau rețea

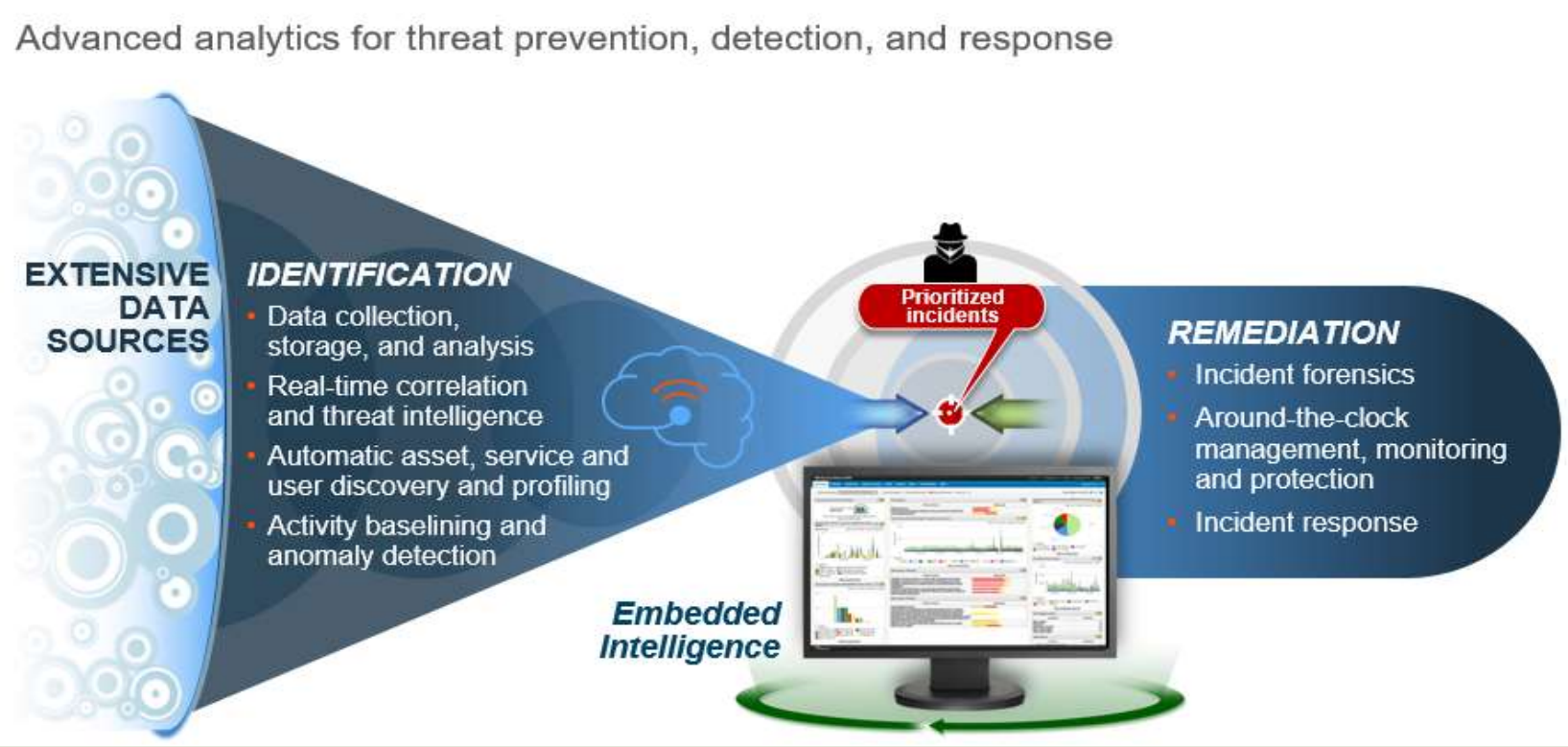
Alin PUNCIOIU
Ionuț GEORGESCU

# Controale de securitate

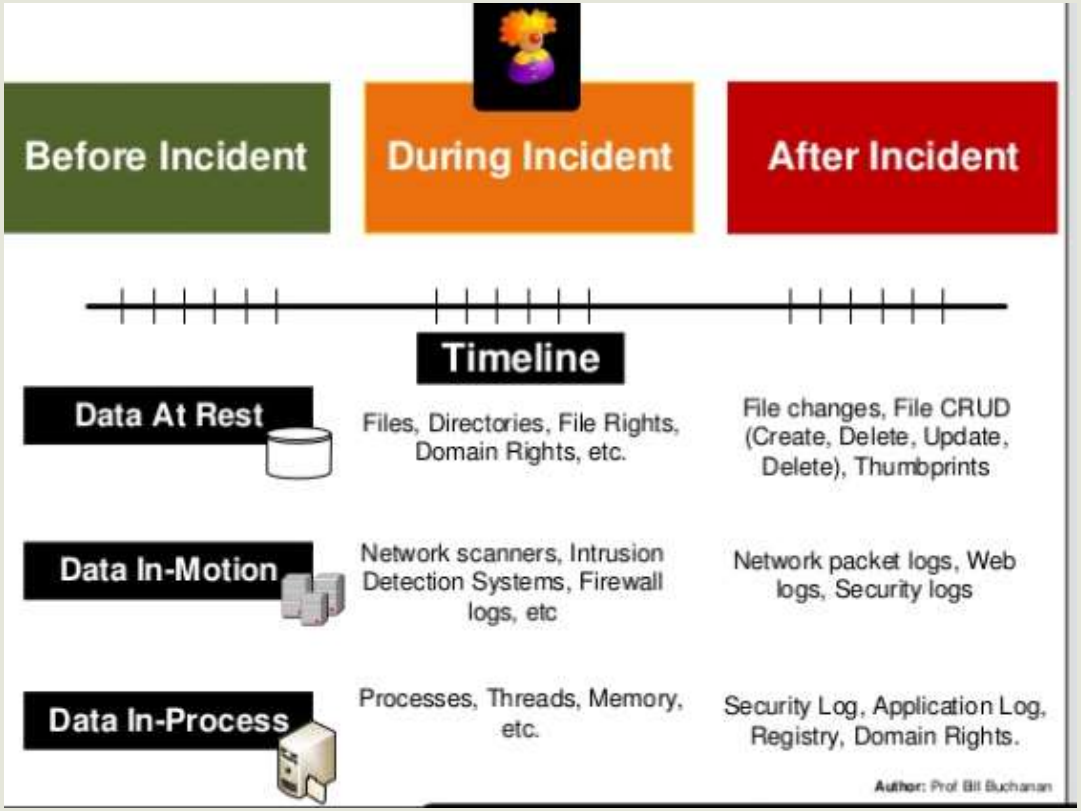
- Ce se întâmplă?
  **Alerte**
- Ce s-a întâmplat?
  **Repoarte, Analiză, căutare de evenimente**
- Ce s-a schimbat?
  **Module schimbate**
- Compliance



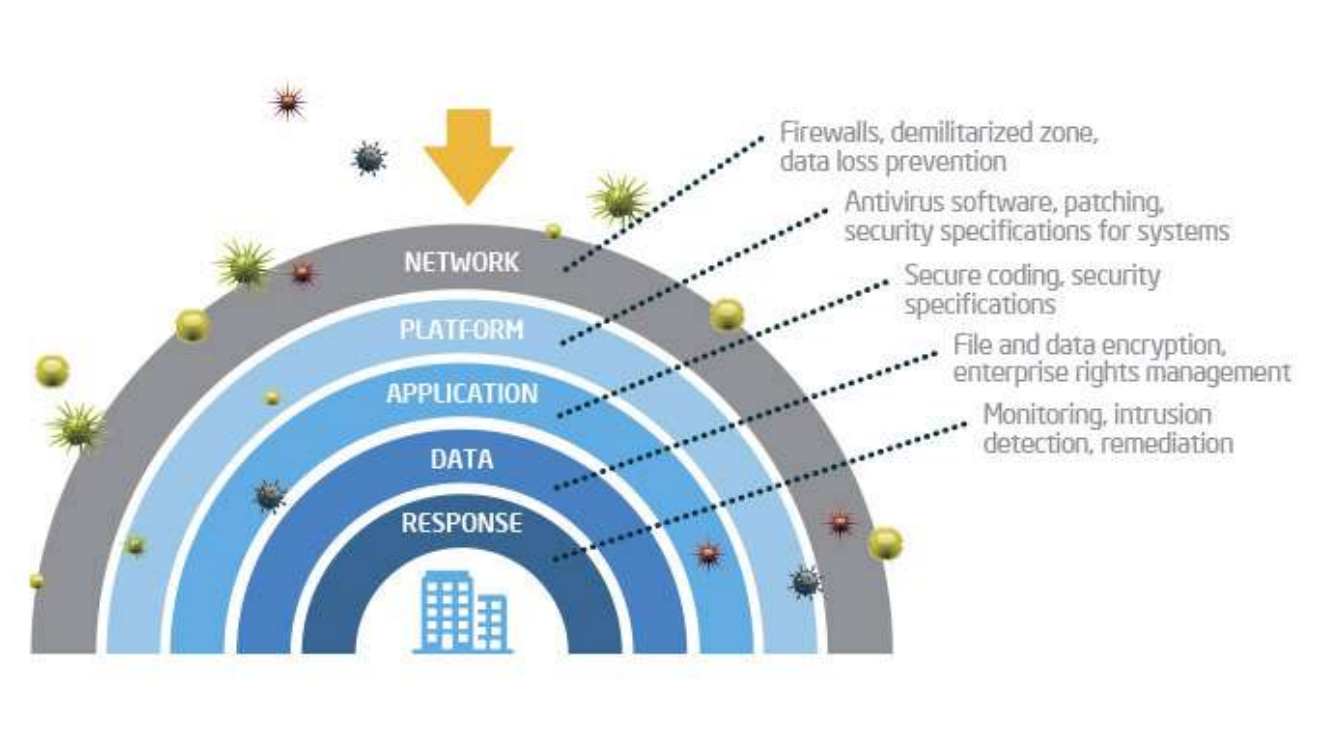
# Controale de securitate



# Controale de securitate

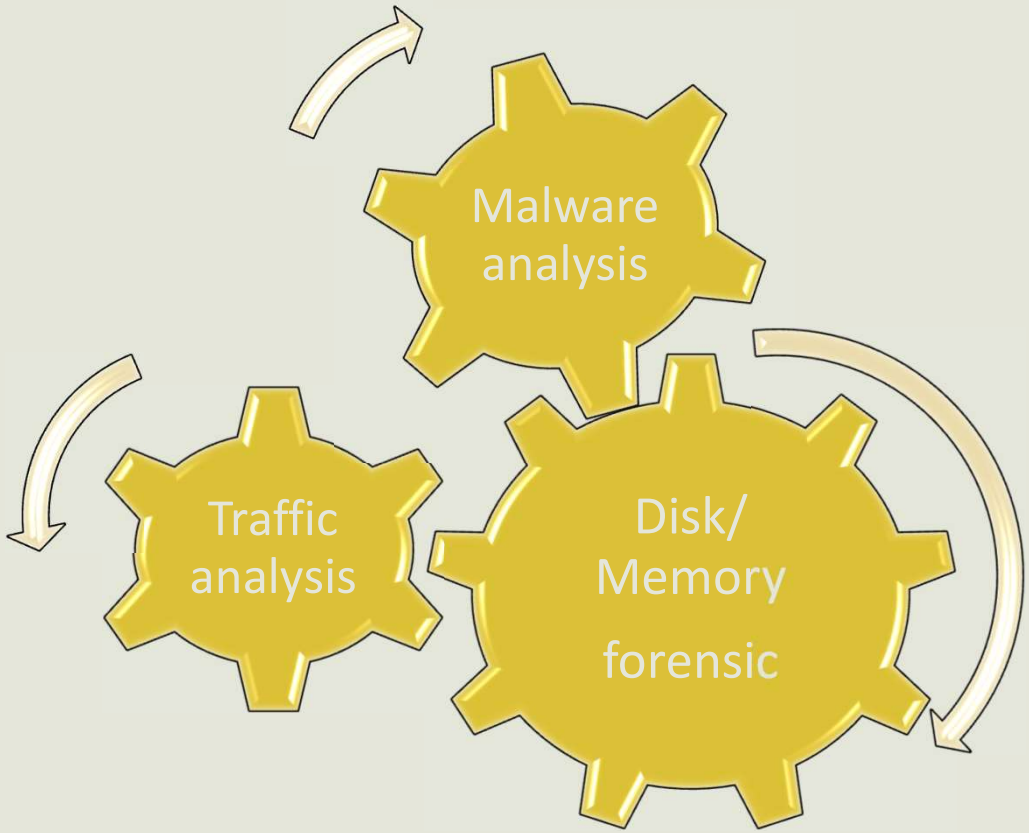



# Defense In-Depth



# Fazele KillChain

# Host/ network investigations



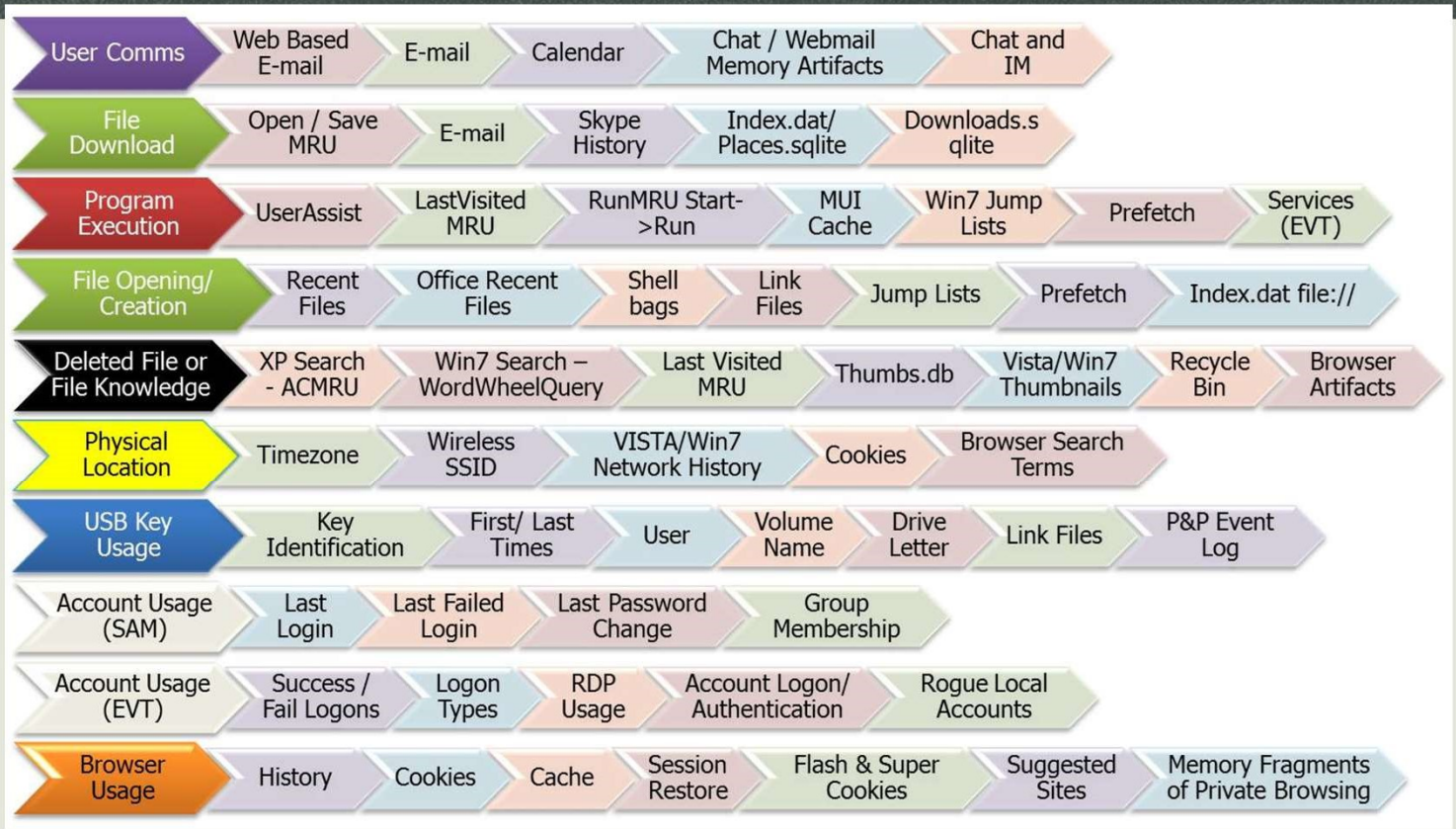
# Quick evidence review

- Real evidence - physical objects that play a relevant role in the crime
  - Physical HDD or USB
  - Computer – box, keyboard, etc.
- Best evidence - can be produced in court
  - Recovered file
  - Bit – for – bit snapshot of transaction
- Direct evidence – eye witness
- Circumstantial evidence – linked with other evidence to draw conclusion
  - Email signature
  - USB serial number
- Hearsay – second-hand information
  - Text file containing personal letter
- Business records – routinely generated documentation
  - Contracts and employee policies
  - Logs
- Digital evidence – electronic evidence
  - Emails / IM
  - Logs

# Computer Forensic Capabilities

- Recover deleted files
- Find out what external devices have been attached and what users accessed them
- Determine what programs ran
- Recover webpages
- Recover emails and users who read them
- Recover chat logs
- Determine file servers used
- Discover document's hidden history
- Recover phone records and SMS text messages from mobile devices
- Find malware and data collected

# artefact collection and analysis



# Investigative methodology

- OSCAR [3]
  - Obtain information
  - Strategize
  - Collect evidence
  - Analyze
  - Report

# Data acquisition types

- **Live** data acquisition
  - Involves collecting **volatile information** that resides in registries, cache and RAM
  - Contamination is harder to control because tools and commands may change file access dates and times, use shared libraries or DLLs, trigger the execution of malicious software or even force a reboot and lose all volatile data.
  - Types of volatile information:
    - System information: current configuration, current date and time, running processes, open files, clipboard data etc.
    - Network information: open connections and ports, routing information and configuration, ARP cache.
- **Static** data acquisition
  - Collecting data that remains unaltered even if the system is powered off

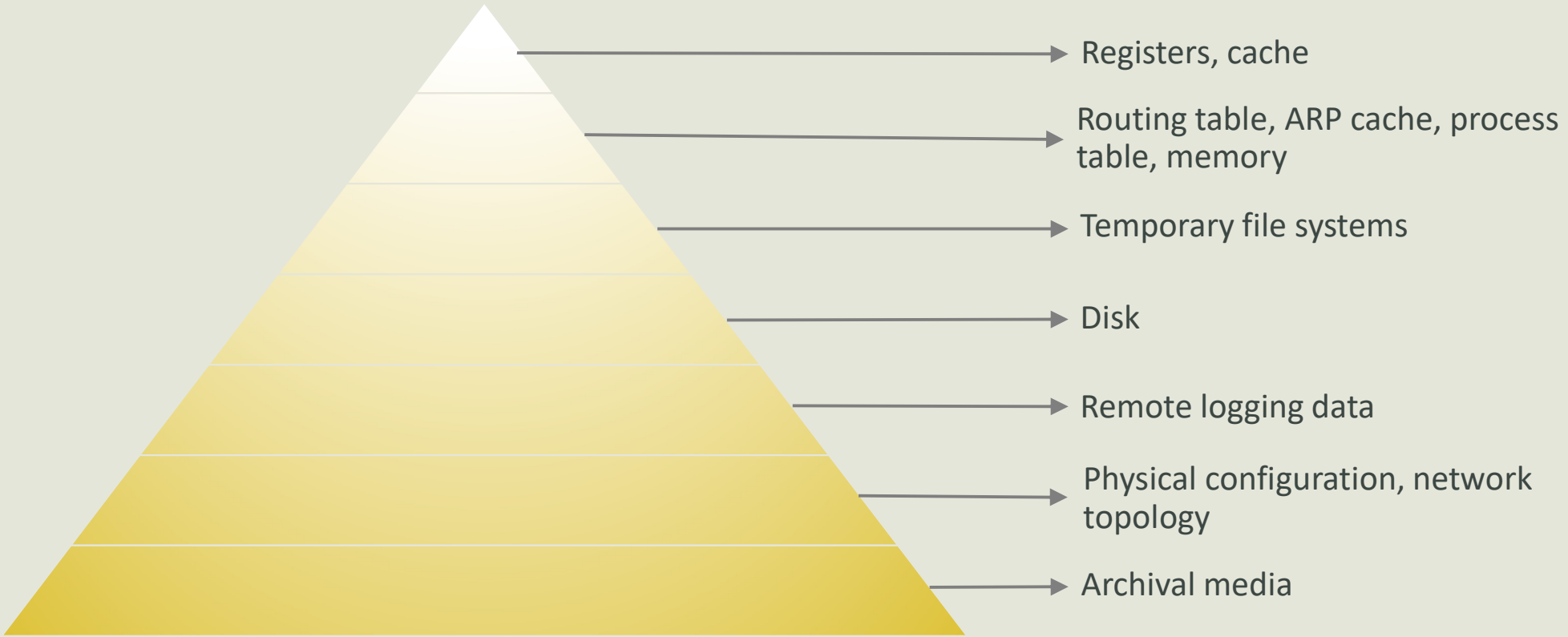
Based on the resulting image, data acquisition can also be:

- **Logical**: only active content, does not include deleted data.
- **Physical**: bit-by-bit image of the device, including deleted data

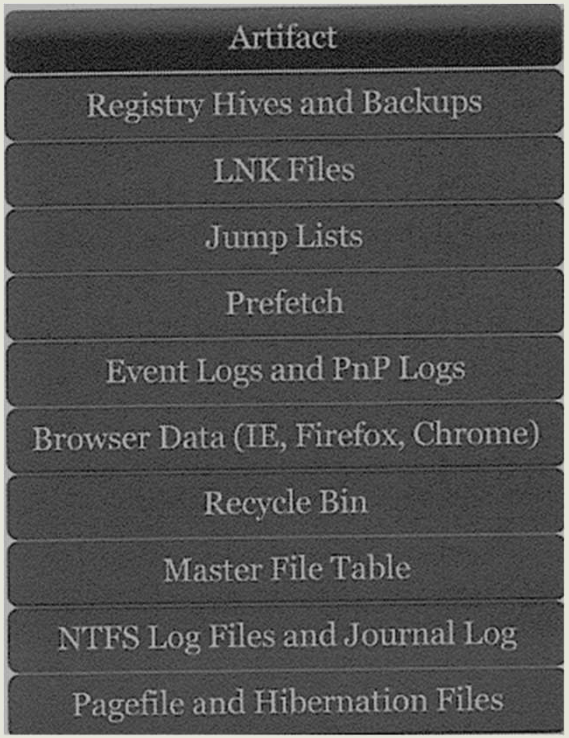
# Evidence and forensic backups

- **Evidence**: anything that can be presented in support of an assertion.
- **Valid evidence** means that a forensic examination should involve two critical areas:
  - *Use sterile and validated media for backups/clones*
    - overwrite with known/random hex value in order to eliminate any previous data.
    - forensically sterile media : use the known character **0x00** to overwrite previous data
    - validate by running a **checksum-64** hash against the media
  - *Use validated forensic backup practices*
- **Forensic backups**:
  - "*Forensic copy*": original media copied directly to target media (same or larger capacity), any remaining space is overwritten with 0x00. Also called disk-to-disk.
  - "*Forensic evidence files*": one or more files containing a bit-for-bit copy of the data found on the source media. Also called disk-to-file.
    - Linux dd (data dump) uses .001 .002 etc. file extensions
    - EnCase Evidence File use .E01, .E02 etc. file extensions

# Order of volatility



- Windows Artifacts



# Windows Artifacts

## Common Windows folder structures

- In the Windows Operating System, after the installation is completed, there is a standard set of folder structures and special files created:

  - **PROGRAM FILES** – contains most of the applications installed on the system
    - **Program Files location**:

| Folder location | Windows version |
|---|---|
| C:\Program Files | 7, 8, 10 |
| C:\Program Files (x86) | 7, 8, 10 |
| C:\Program Data | 7, 8 |

    - **The Program Files (x86)** folder exists only in the 64-bit version of Windows 7, 8 and 10 operating system environment. The Program Files (x86) is the default location for all the 32-bit applications

# Windows Artifacts

## Common Windows folder structures

- **User Account Profiles** – contains all of the configuration settings and files for each individual user account on a Windows system. This includes settings for all of the application software on the system that are specific to that user account.

.

- By default, each user who logs on to a Windows system has a user profile. This profile is created when the user logs on for the first time.
  - **Location**:

| Folder path | Windows version |
|---|---|
| C:\Users\%UserName% | 7, 8, 10 |

**Application Data** – contains application-specific data

- **Location**

| Location | Windows version |
|---|---|
| C:\Users\%UserName%/AppData | 7, 8, 10 |

# Exploring the Organization of the Windows Registry

- Registry terminology:
  - Registry
  - Registry Editor
  - HKEY
  - Key
  - Subkey
  - Branch
  - Value
  - Default value
  - Hives

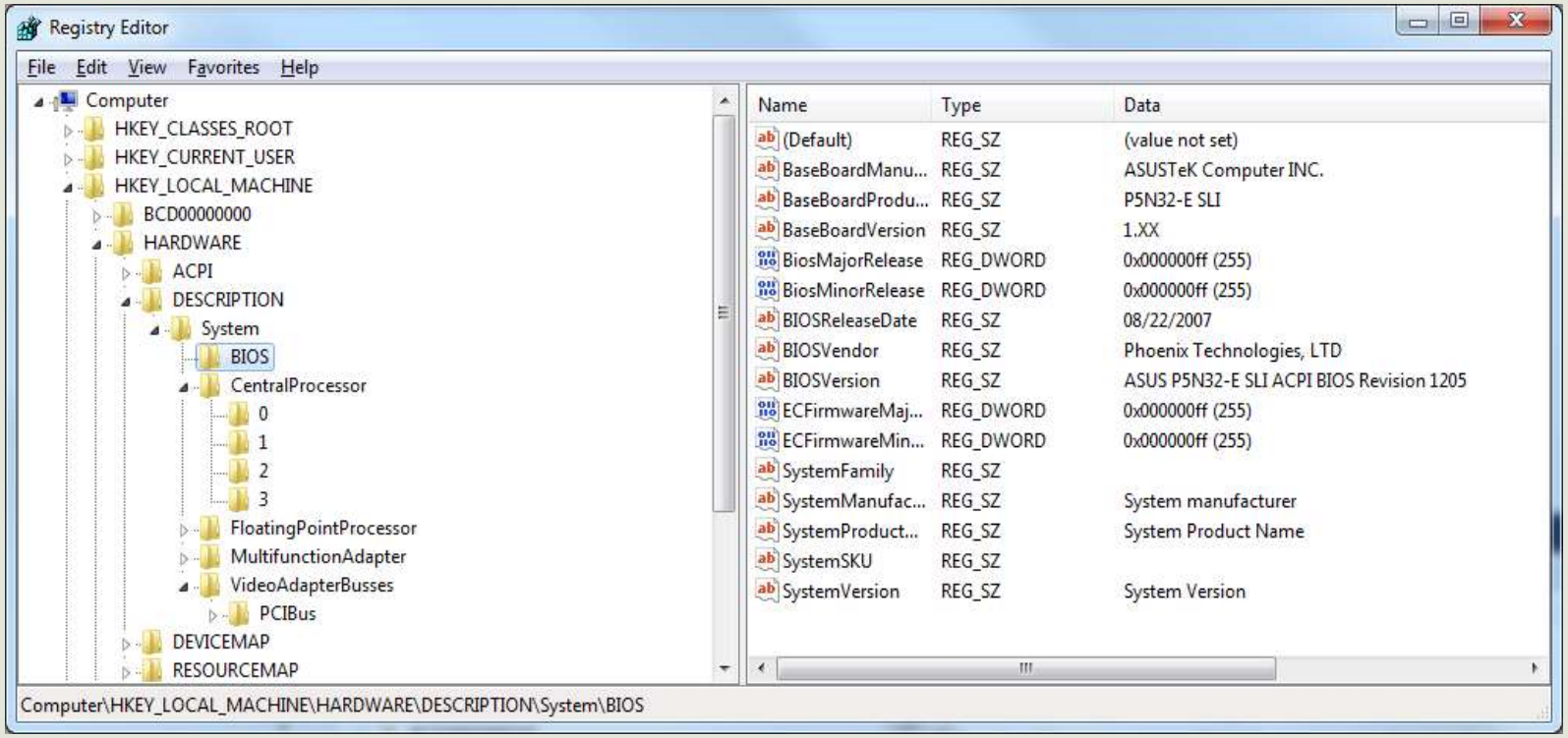
## Hive Properties

- HKEY_USERS – all loaded user data
- HKEY_CURRENT_USER – currently logged on user (NTUSER.DAT)
- HKEY_LOCAL_MACHINE – array of software and hardware settings
- HKEY_CURRENT_CONFIG – hardware and software settings at start-up
- HKEY_CLASSES_ROOT – contains information about application needs to be used to open files

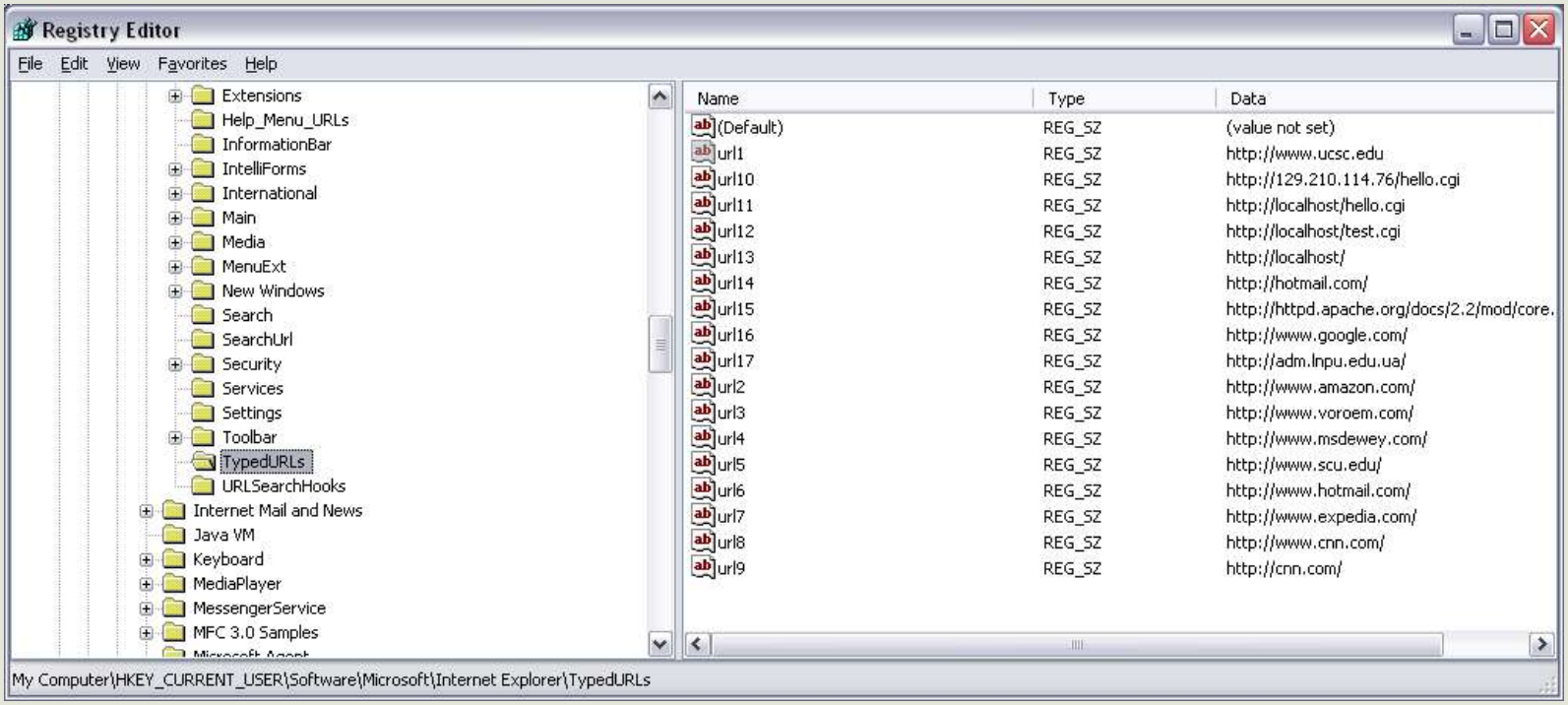
# File Locations and Purpose

| Filename and location | Purpose of file |
|---|---|
| **Windows 9x/Me** | |
| Windows\System.dat | User-protected storage area; contains installed program settings, usernames and passwords associated with installed programs, and system settings |
| Windows\User.dat<br>Windows\profile*UserAccount* | Contains the most recently used (MRU) files list and desktop configuration settings; every user account created on the system has its own user data file |
| **Windows NT, 2000, XP, and Vista** | |
| Documents and Settings*user-account*\ Ntuser.dat (in Vista, Users*UserAccount*\ Ntuser.dat) | User-protected storage area; contains the MRU files list and desktop configuration settings |
| Winnt\system32\config\Default | Contains the computer's system settings |
| Winnt\system32\config\SAM | Contains user account management and security settings |
| Winnt\system32\config\Security | Contains the computer's security settings |
| Winnt\system32\config\Software | Contains installed programs settings and associated usernames and passwords |
| Winnt\system32\config\System | Contains additional computer system settings |

# Forensic Analysis - Hardware

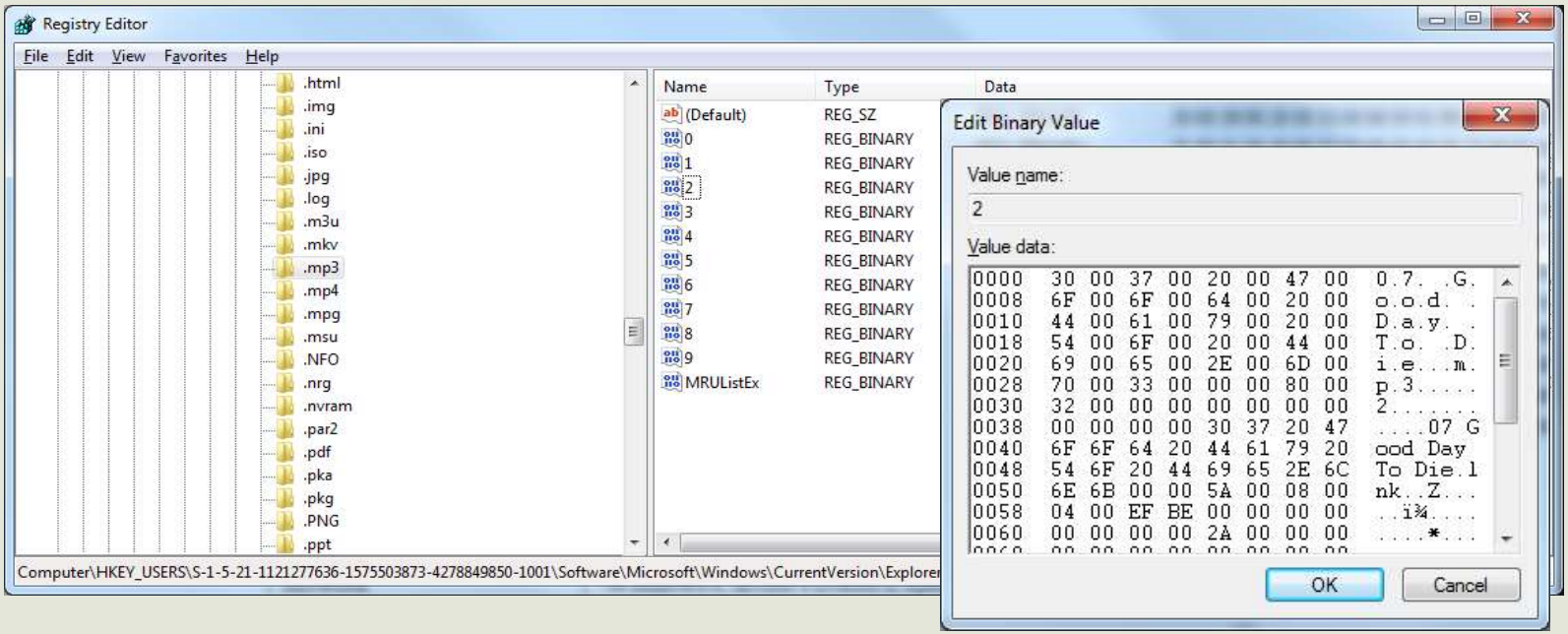


# Forensics Analysis - NTUSER.DAT
# IE explorer Typed URLs

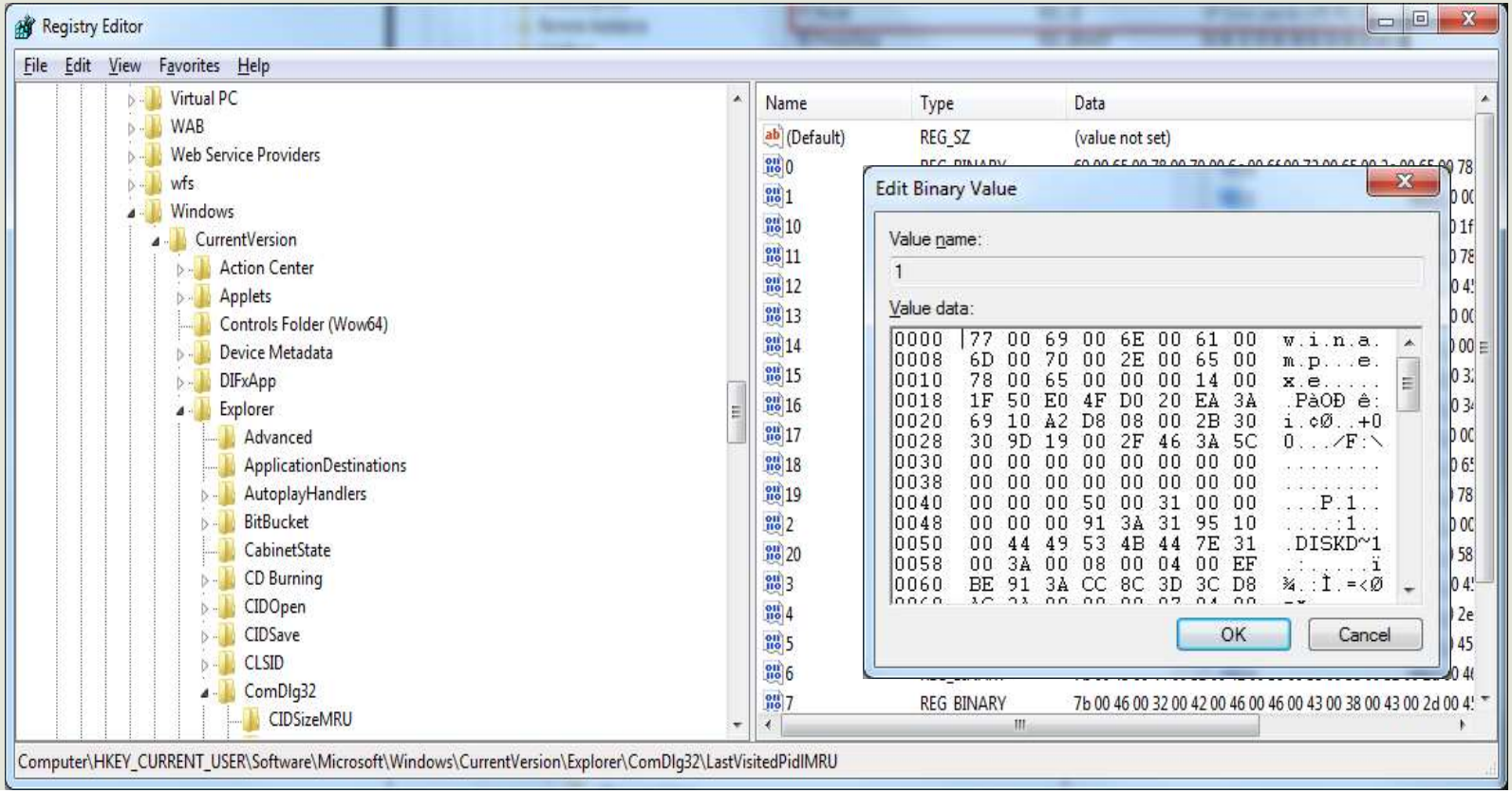


# Forensic Analysis – MRU List

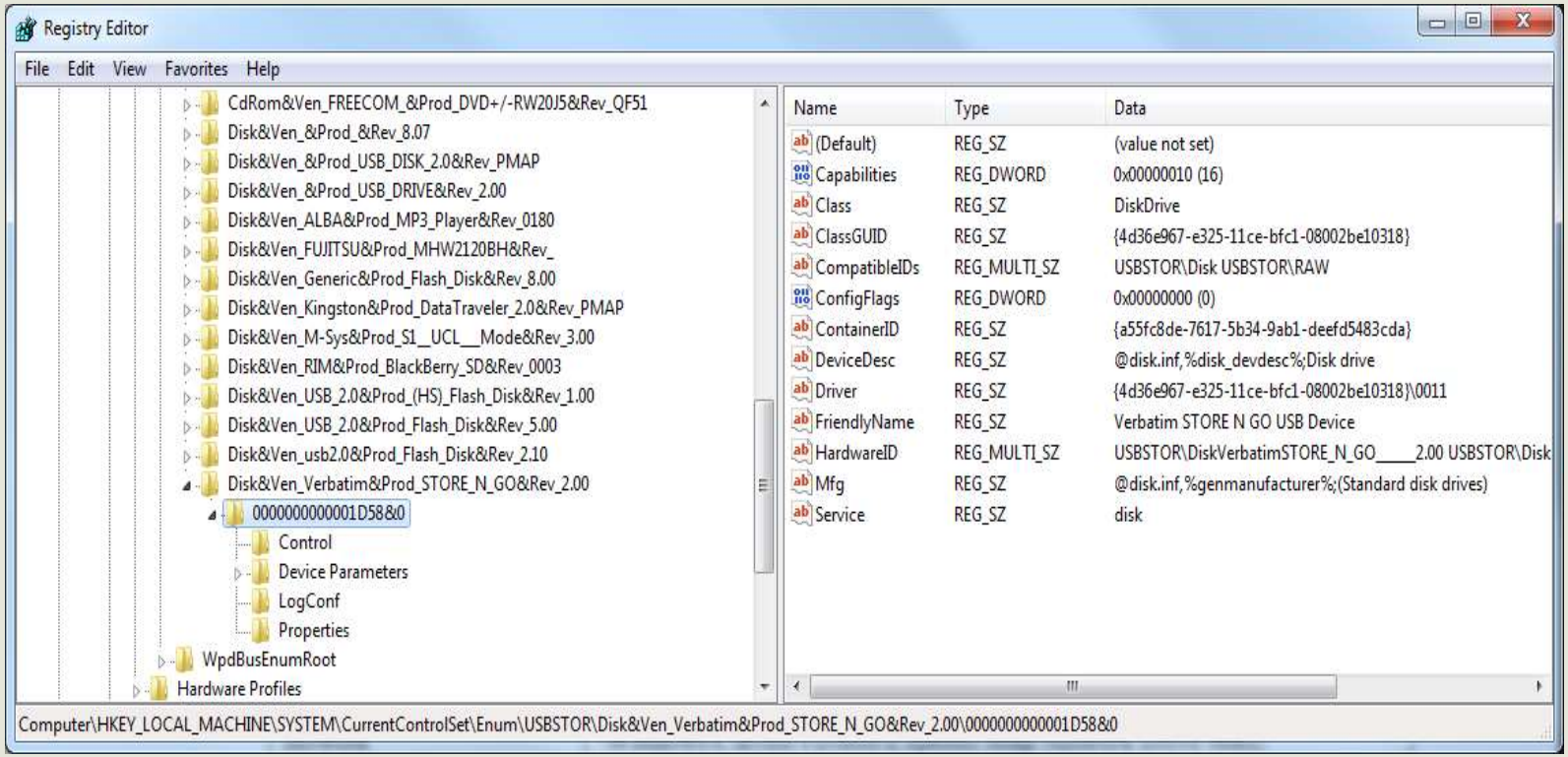
A “Most Recently Used List” contains entries made due to specific actions performed by the user. There are numerous MRU list locations throughout various Registry keys.
These lists are maintained in case the user returns to them in the future. Essentially, their function is similar to how the history and cookies act in a web browser.



# Forensic Analysis – Last Opened Application in Windows



# Forensic Analysis – USB Devices

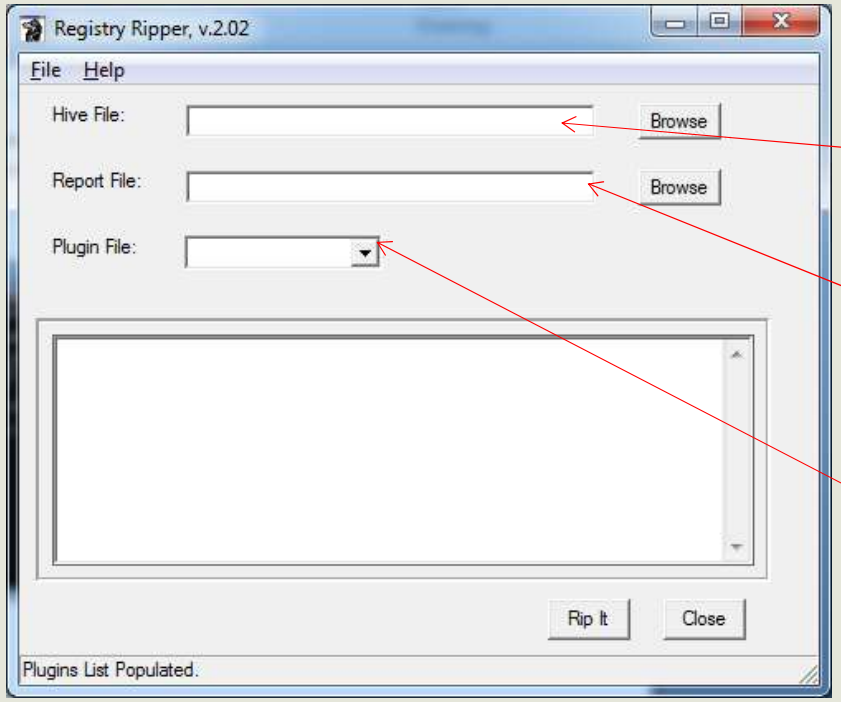


# Forensic Analysis – USB Devices

| Win7 USB Drive Enclosures | |
|---|---|
| **1. Write Down Vendor, Product, Version** | |
| SYSTEM\CurrentControlSet\Enum\USBSTOR | |
| **2. Write Down Serial Numbers** | |
| SYSTEM\CurrentControlSet\Enum\USBSTOR | Serial Number = |
| **3. Determine Vendor-ID (VID) and Product-(PID)** | |
| SYSTEM\CurrentControlSet\Enum\USB -> Perform search for S/N | VID_XXXX = PID_YYYY = |
| **4. Determine Disk Identity** | |
| Examine MBR of hard drive (Sector 0)-> 4 byte value starting at offset 440 | Disk ID = |
| **5. Determine Drive Letter Device Mapped To** | |
| SYSTEM\MountedDevices-> Perform search for Disk ID in the Drive Letters | Drive = |
| **6. Write Down Volume GUIDs** | |
| SYSTEM\MountedDevices-> Perform Search for Disk ID in the GUIDs | GUID = |
| **7. Find User That Used The Specific USB Device** | |
| NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2-> Search for Device GUID | User = |
| **8. Discover First Time Device Connected** | |
| C:\Windows\inf\setupapi.dev.log -> Perform search for Serial Number | Time/Timezone = |
| **9. Determine First Time Device Connected After Last Reboot** | |
| SYSTEM\CurrentControlSet\Enum\USBSTOR\ Vendor_Product_Version -> Perform search for Serial Number (Last Written Time of Serial Number Key) | Time/Timezone = |
| **10. Determine Last Time Device Connected** | |
| SYSTEM\CurrentControlSet\Enum\USB\ VID_XXXX&PID_YYYY -> Perform search for Serial Number (Last Written Time of Serial Number Key) | Time/Timezone = |

# RegRipper Interface

The RegRipper is an open-source application for extracting, correlating, and displaying specific information from Registry hive files from the Windows NT (2000, XP, 2003, Vista and 7) family of operating systems.




Which hive file will be analyzed

Where to put the report

Which Plugins file to use

| HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName | Computer name |
|---|---|
| HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\BIOS | BIOS version |
| HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\CentralProcessor\0 | Processor info |
| HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Windows | Shutdown time |
| HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList | Users profiles |

| Registry key | Information |
|---|---|
| HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run | Startup software |
| HKEY_LOCAL_MACHINE\SOFTWARE\RegisteredApplications | Application on the system |
| HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\NetworkList\Nla\Cache\Intranet | Intranet network connected |
| HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU | Open files and commands ran on the machine |
| HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce<br>HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnceEx<br>HKLM\Software\Microsoft\Windows\CurrentVersion\Run<br>HKCU\Software\Microsoft\Windows\CurrentVersion\Run<br>HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce<br>HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run | Locations checked by Windows at startup for changes |

# WINDOWS VS LINUX

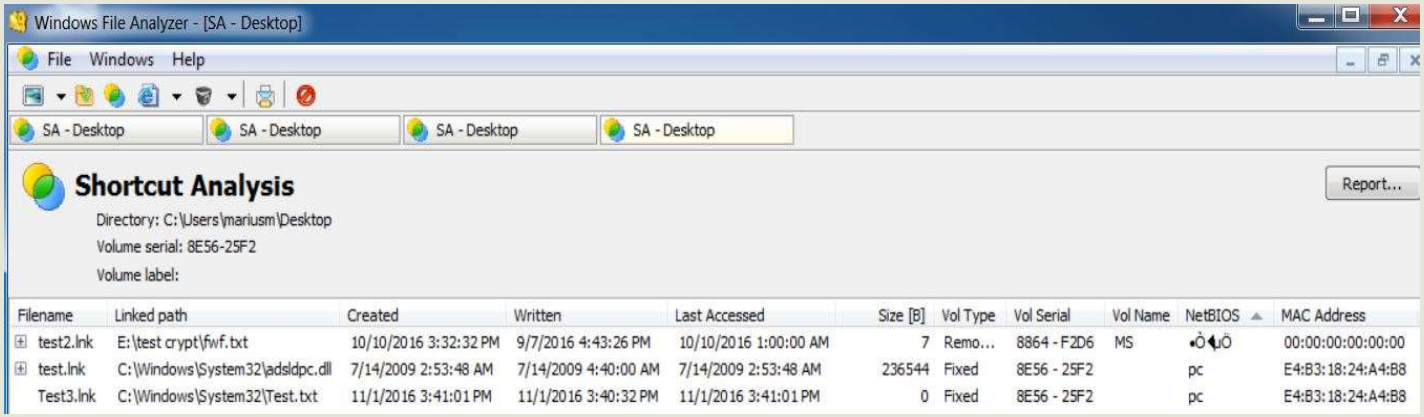
| Evidence | Linux | Windows |
|---|---|---|
| System and application specific logs | /etc | Windows/system32/config |
| Activity Logs | Var/log | Windows event logs (*.evtx) |
| User Profile | /home/$USER | C:\Users\userPrfoile |
| Operating System Information | /etc/os-release | Computer\Hkey_Local_Machine_Software\ Microsoft\WindowsNT\ Current version\ProductName |
| Operating system installation Date | /root/install.log | Computer\Hkey_Local_Machine_Software\ Microsoft\WindowsNT\ Current version\InstallDate |
| Hostname/Computer Name | /etc/hostname | HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\ComputerName |
| IP Address, DNS Server, Lease obtained time | /var/log | HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parametres\Interfaces\DHCPIPAddress |
| Time Zone Information | /etc/timezone | HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\TimeZoneInformation |
| User Login History | /var/log/auth.log | NTUSER.DAT of specific user |
| Connected USB Devices history | /var/log/syslog | C:\Windows\inf\setupapi.dev.log HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USBSTOR |
| Recently accessed files | /home/username/.local/share/recently-used.xbel | C:\Users\(User Name)\AppData\Roaming\Microsoft\Windows\Recent Items |

- Windows Artifacts:
  - Shell Link Files
  - User assist
  - Shellbags
  - System Resource Usage Monitor (SRUM)
  - Jump Lists
  - Prefetch / Superfetch

# Windows Artifacts

## Shell Link Files

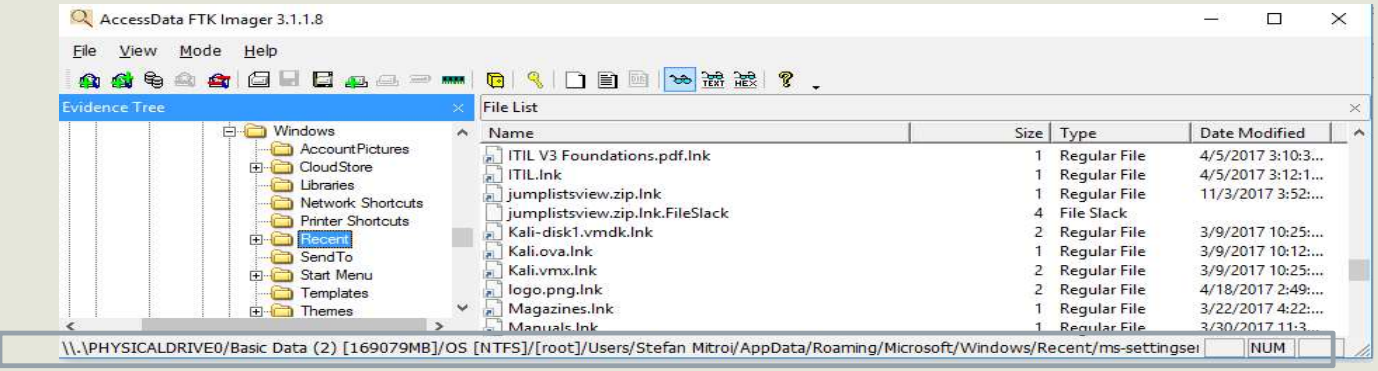
- A **shell link file is commonly referred to as a link file or shortcut**. It is a special file that contains “links” or ”pointers” to other resources, for example, programs, data files and folders.
- During an examination of a Windows system many link files (.lnk) will be found. These files contain very useful information about the target, including:
  - File Attributes
  - MAC Times
  - File Size
  - Volume Type
  - Volume Serial Number
  - Volume Label
  - Original File Path

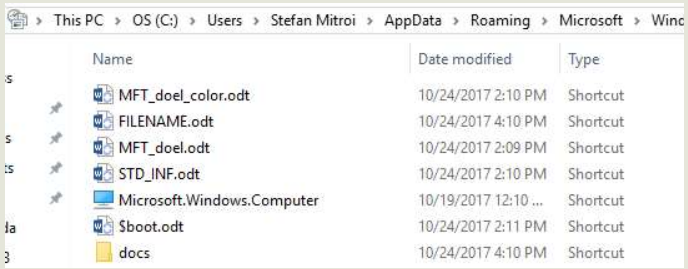



- A useful and free tool for parsing .Lnk files is Windows File Analyzer - http://mitec.cz/wfa.html

# Windows Artifacts

- Shell Link Files
  - The best thing about a Link file is that it will **often demonstrate a user's knowledge of a file and his/her interaction with that file**.
  - A link file's embedded time becomes very powerful when the examiner can check the MAC times of the target within the file system – any date and time entries that are after those embedded within the link file show that a user has interacted with the file
  - There are many forensics implications relating to the content of these files. **The Volume Serial Number can be used to tie a specific thumb drive, USB drive, memory card or other removable media to a specific computer system**.
  - **By default, when a file or document is opened in Windows, a link (.lnk) file is created in the Recent folder**



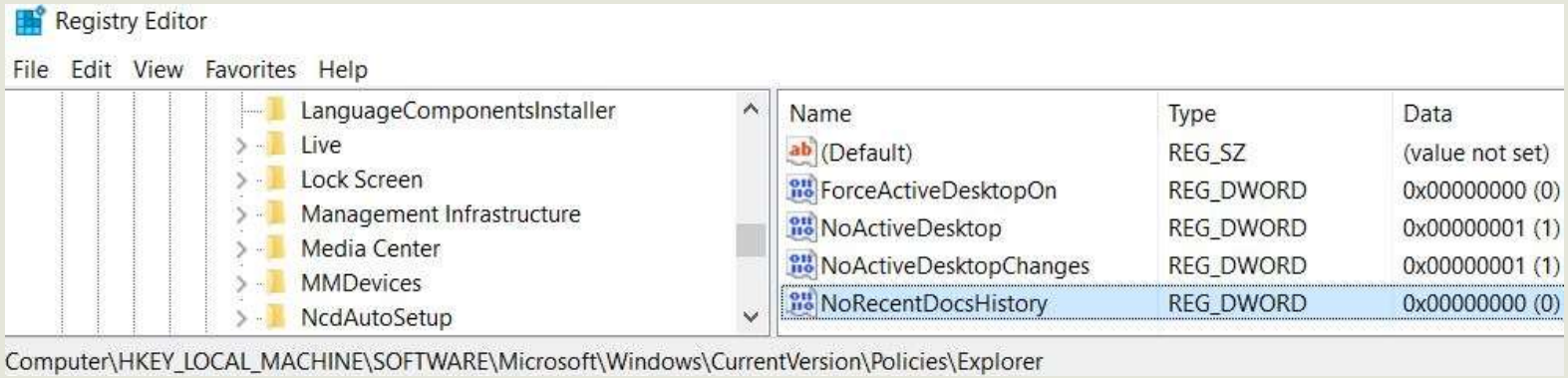



# Windows Artifacts

- Shell Link Files

- **Recent items .lnk file creation can be disabled by the user. It can be done for an individual user in their NTUser.dat at:**

  - Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\NoRecentDocsHistory with a dword value of 0000 0001



- Shellbags
  - **"Shellbags" is a commonly used term to describe a collection of registry keys that allow the Windows operating system to track user window viewing preferences specific to Windows Explorer**. These keys can contain a wealth of information relevant for a forensic investigation and can help paint a clearer picture of **user activity on a machine**. For example, the **following information** can be found in Shellbags:
    - Windows sizes and preferences
    - Icon and folder view settings
    - Metadata such as MAC timestamps
    - Most recently used files and file type (zip, directory, installer)
    - Files, folders, zip files, installers that existed at one point on the system (even if deleted).
    - Network Shares and folders within the shares
    - Metadata associated with any of the above types which may include timestamps and absolute paths

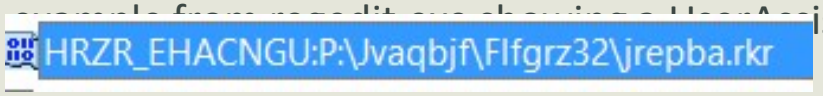
# Windows Artifacts

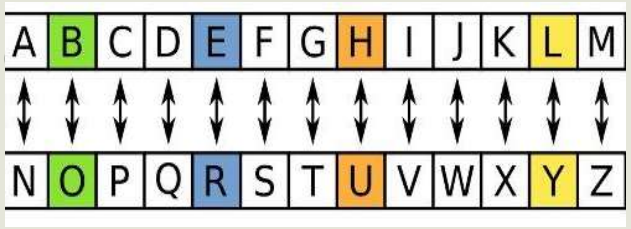
- User assist
- The **UserAssist Registry key keeps track of programs executed from the explorer shell** (Desktop) specifically the "GUI" environment the user interacts with. Unlike prefetch files, **UserAssist entries can be correlated to a specific user**. The UserAssist registry key at:
  - NTUSER.DAT\Software\Microsoft\Windows\Currentversion\Explorer\UserAssist
- It is the key used to **track file execution and the execution** of shortcuts related to executed files. To do this, it uses Globally Unique Identifiers (GUID) within the UserAssist key:
  - CEBFF5CD → Executable File Execution
  - F4E57C4B → Shortcut File Execution

# Windows Artifacts

- User assist

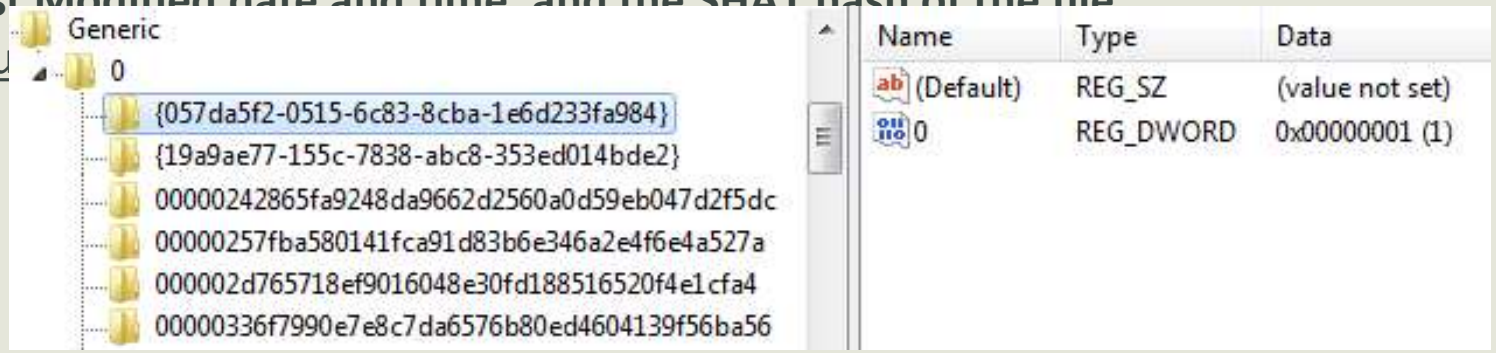
- The data in the **UserAssist key is ROT-13 encoded**. ROT-13 is a **simple substitution** cipher using the English standard alphabet (26 characters), replacing an alphabetical letter with another letter **13 letters** away.
  - **Example A → Rot-13 → N**
- This key is evidence of a user executing a binary through interaction with Explorer.
- Useful Information that is contained within this key includes:
  - Frequency of program execution
  - Key values to distinguish entries
  - Evidence of deleted/moved programs
  - Name including Path (Rot13)
  - Run Counter – How many times the program was run
  - Last Run Time
- Here is an example from regedit.exe showing a UserAssist entry, an



# Windows Artifacts

- Amcache and Shimcache
- **Amcache and Shimcache can provide a timeline of which program was executed and when it was first run and last modified**
- In addition, these **artifacts provide program information regarding the file path, size, and hash depending on the OS version.**
  - C:\Windows\AppCompat\Programs\Amcache.hve
- To sum up, the information it contains includes: **Filename, Standard Information Attribute Last Modified date and time, and the SHA1 hash of the file**
(https://github

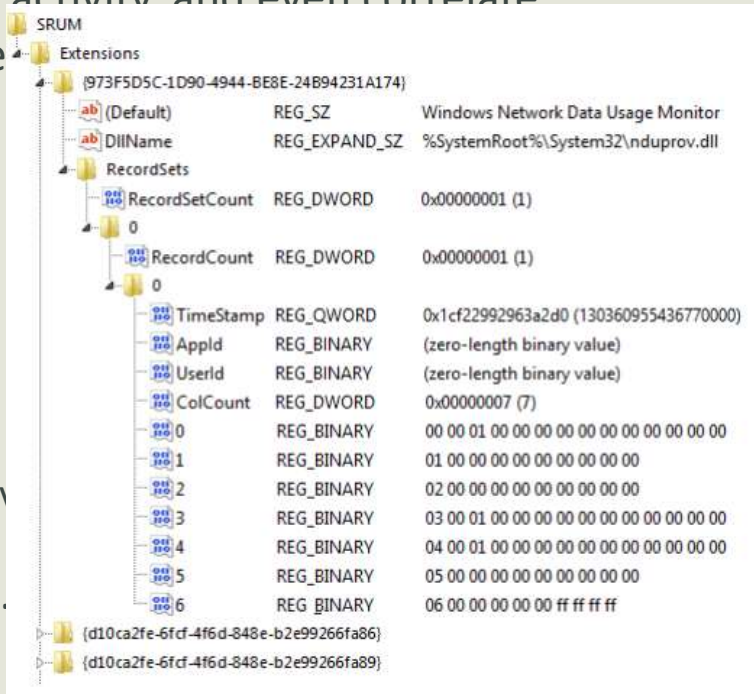


# Windows Artifacts

- Amcache and Shimcache
- **Shimcache**, also known as **AppCompatCache**, is a component of the **Application Compatibility Database**, which was created by **Microsoft** (beginning in **Windows XP**) and used by the **operating system to identify application compatibility issues**.
- The cache **stores various file metadata** depending on the operating system, such as:
  - File Full Path
  - File Size
  - **$Standard_Information** (SI) Last Modified time
  - Shimcache Last Updated time
  - Process Execution Flag
- Similar to a log file, the **Shimcache** also “rolls” data, meaning that the oldest data is replaced by new entries.
  The amount of data retained varies by operating system
- The Registry Key related to this cache is located at
  - HKLM\SYSTEM\CurrentControlSet\Control\SessionManager\AppCompatCache\AppCompatCache
- Shimcache can be investigated using ShimCacheParser.py, by Mandiant:
  - https://github.com/mandiant/ShimCacheParser

# Windows Artifacts

- System Resource Usage Monitor (SRUM)
- **SRUM** was first introduced in Windows 8, and was a new feature designed to **track system resource utilization such as CPU cycles, network activity, power consumption,** etc. Analysts can use the data collected by SRUM to paint a picture of a user's activity, and even correlate that activity with network-related events, data transfer, processe
  - **Network Connectivity**
  - **Network Data usage**
  - **Application Resource usage**
  - **Windows push notifications**
  - **Energy usage**
- Registry is temporary location for holding data
  - HKLM\SOFTWARE \Microsoft\Windows NT\CurrentVersion\SRUM\
- Data is periodically moved to C:\Windows\System32\sru\SRUDB.



# Windows Artifacts

- Jump Lists

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

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

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

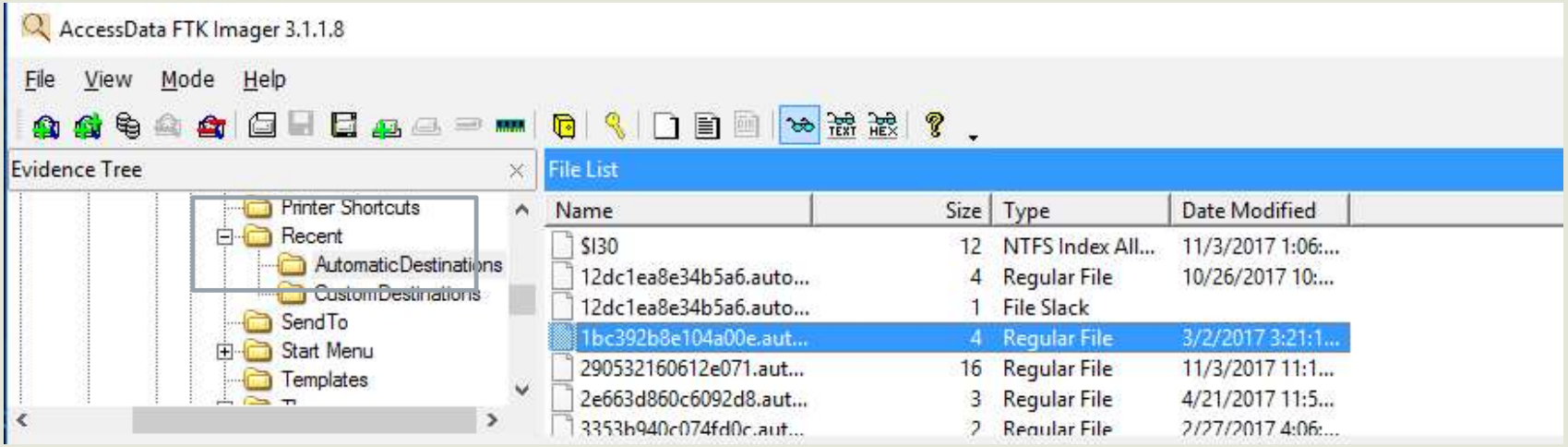
- There are two main types of Jump Lists:
  - **Automatic** – this Jump List is automatically populated by the system
  - **Custom** – this Jump List is maintained by the individual application

# Windows Artifacts

- Jump Lists

- **Jump List** data for all applications is stored in the users profile path: %UserProfile%\AppData\Roaming\Microsoft\Windows\Recent

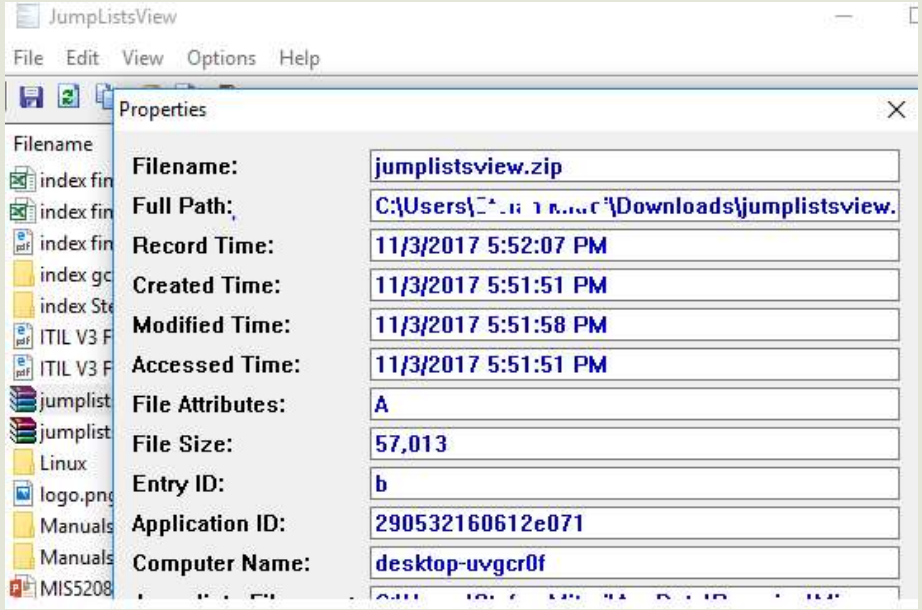
- When this folder is viewed with a forensic tool, additional folders appear:
  - %UserProfile%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations
  - %UserProfile%\AppData\Roaming\Microsoft\Windows\Recent\CustomDestinations



# Windows Artifacts

- Jump Lists

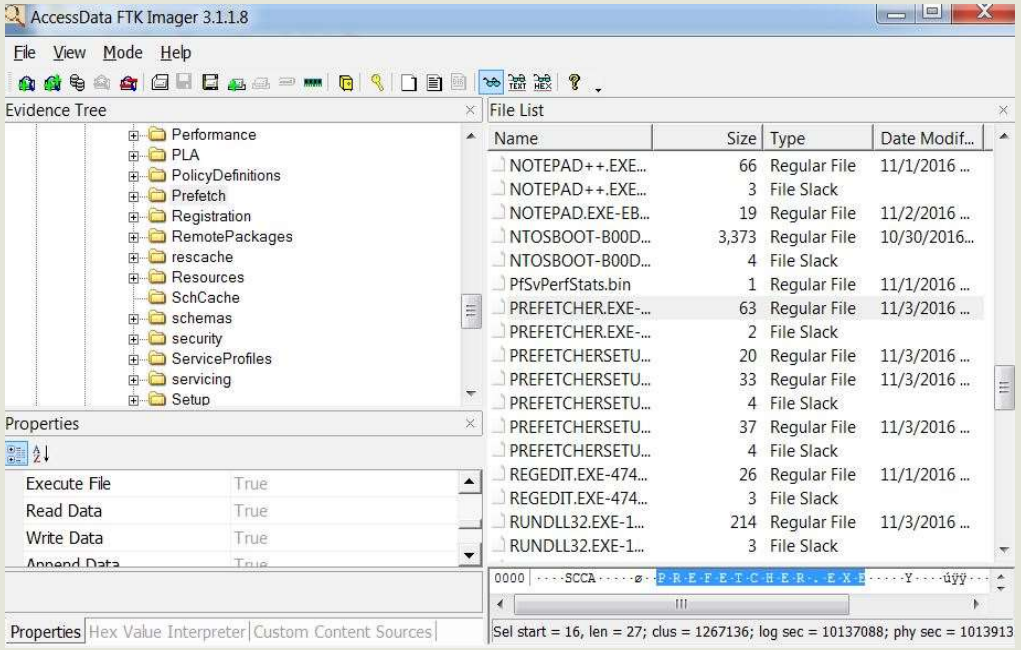
- The recent item data from the Jump Lists populate those two folders. Each program will have its own file name, referred to as a ”Jump List ID”. By examining **each file with a text editor, it can be determined which file links correspond to which program’s Jump List entry.**
- More jump list IDs can be found at: http://www.forensicswiki.org/wiki/List_of_Jump_List_IDs.
- A useful tool is available for examining “lists”, called JumpListView and it’s available at: http://www.nirsoft.net/utils/jump_lists_view.html
- **JumpListView** will display a list of all items in the Jump List, their path, date and time and the entry number for each item.



# Windows Artifacts

- Prefetch / Superfetch

- Prefetching **speeds up computer performance by bringing the data and code pages of programs used during boot process** and in subsequent program launches into memory from the disk before that data and code is actually demanded.
- The prefetch files that are created as a result of the tracing process that occurs are located in the folder %WINDOWS%\Prefetch
- The file's name is the name of the application to which the trace applies followed by a dash and the hexadecimal representation of a hash of the file's path ending with a .PF file extension
- **The prefetch folder will never grow larger than 129 entries**
- Looking at the actual content of the .PF file, the name of the executable file being traced is located at offset 10h an is visible in plaintext. The file will also contain the run count, last run date and a list of files used by the application when it loads.



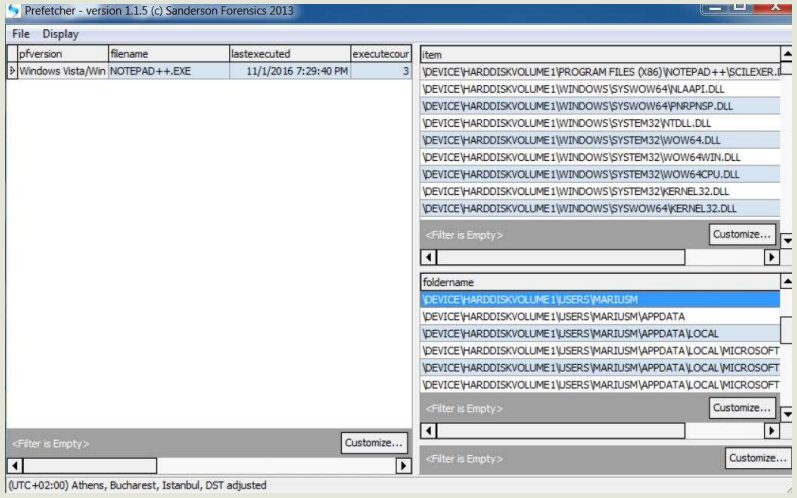
# Windows Artifacts

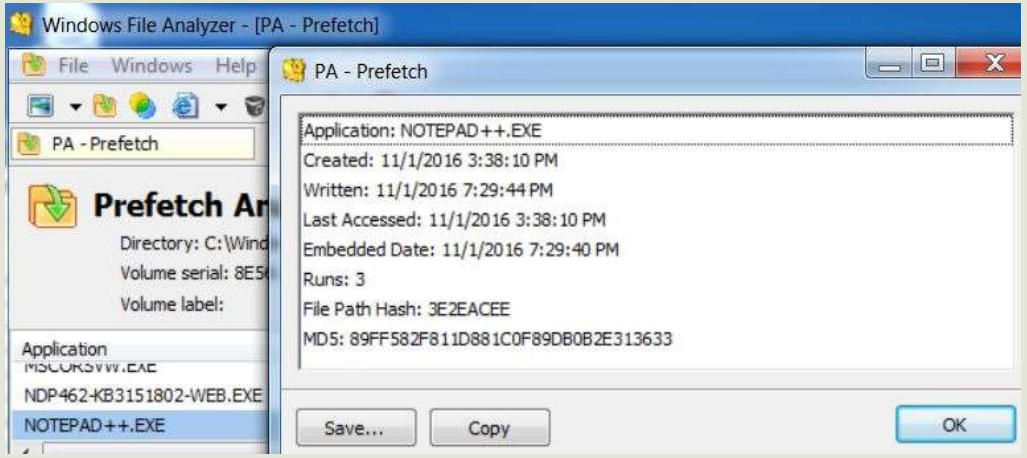
- Prefetch / Superfetch

- **Superfetch** was introduced in Windows Vista. It doesn't replace prefetch files but adds additional functionality by **keeping track of when and how often  a program is run**. It has more granularity and better algorithms to better anticipate  what data will be needed

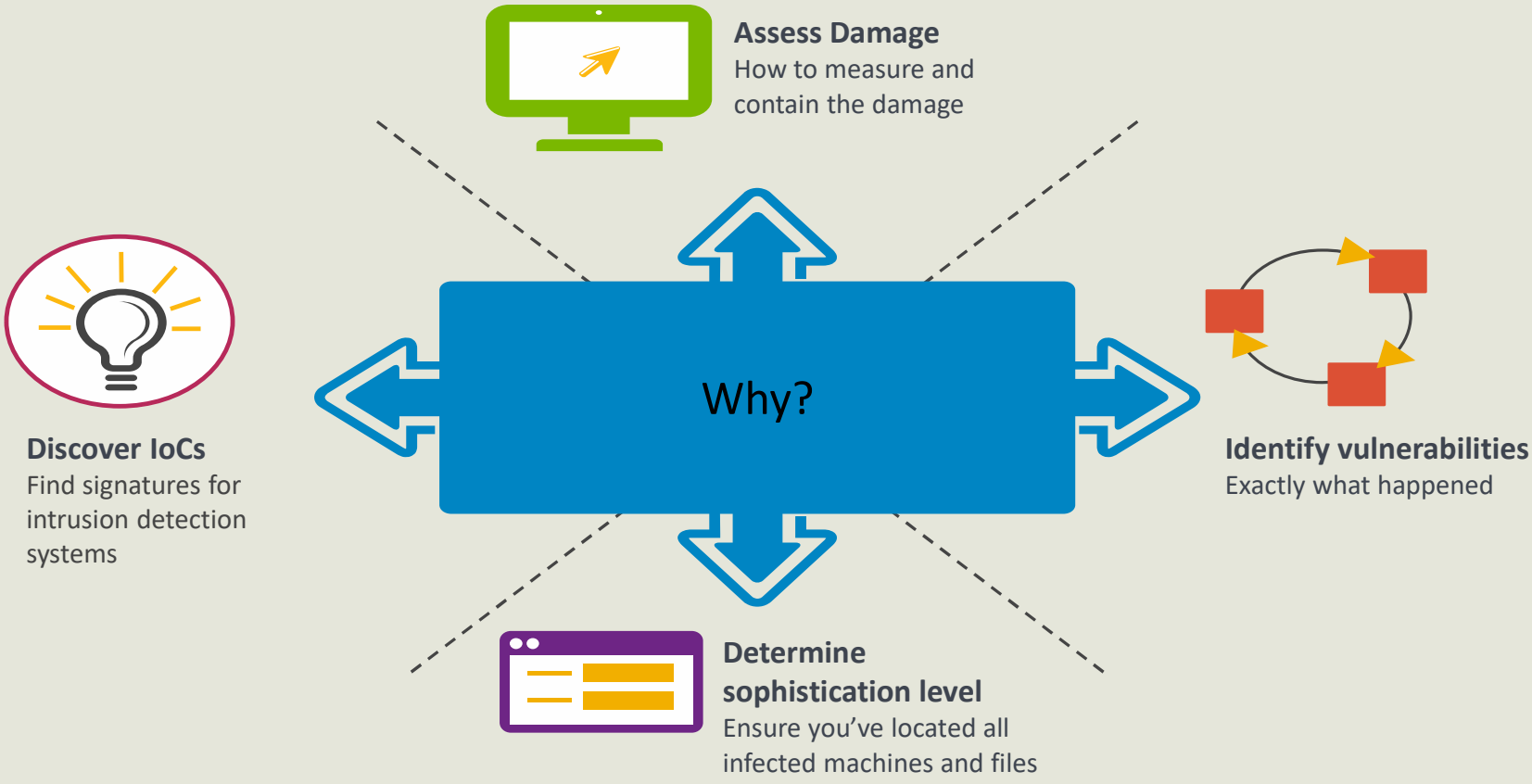
- In a forensic examination, **prefetch files can be used to help determine when an application was last run**. This is useful for creating a timeline of events or if attempting to determine if a virus or other exploit is active on a computer

- Examining the files and directories accessed during the launch of an application can be very beneficial because it can reveal hidden directories, point to user accounts or show that an application was accessed from an external storage drive.

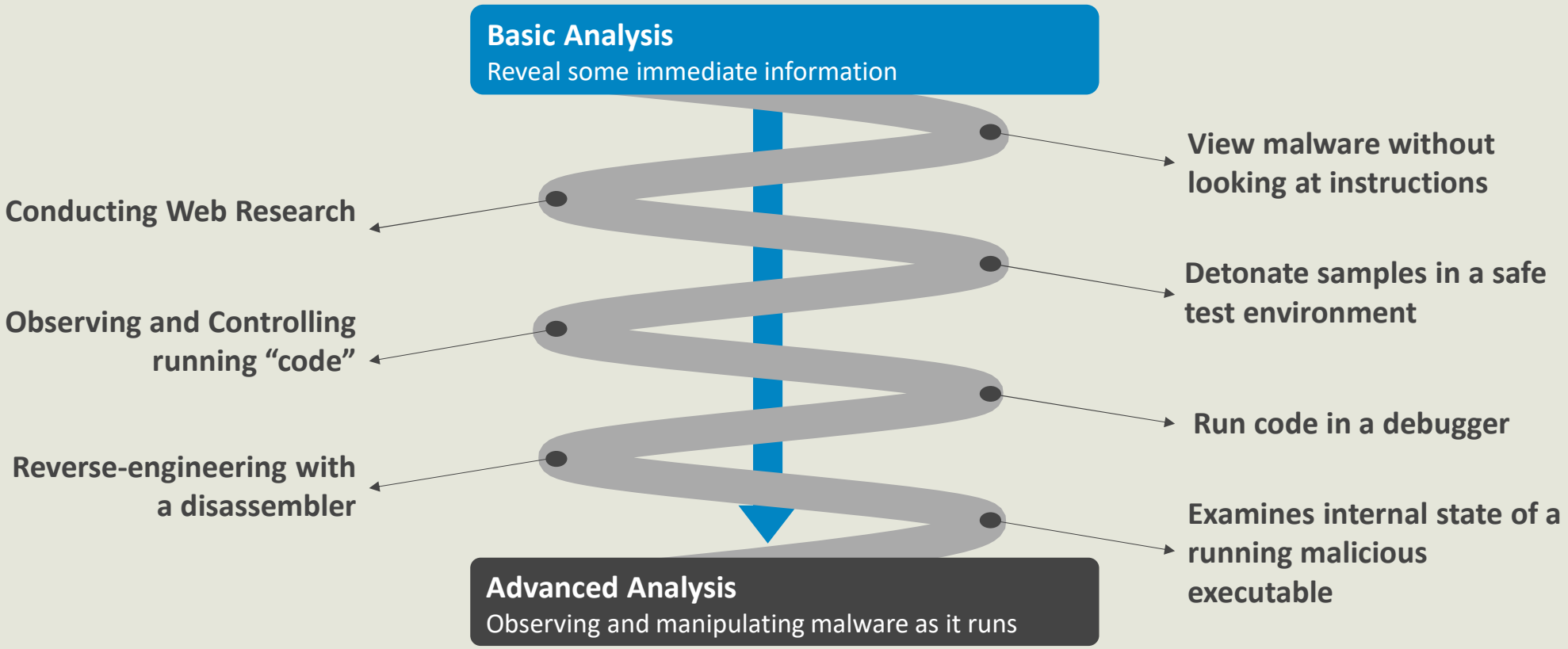


# Malware Analysis



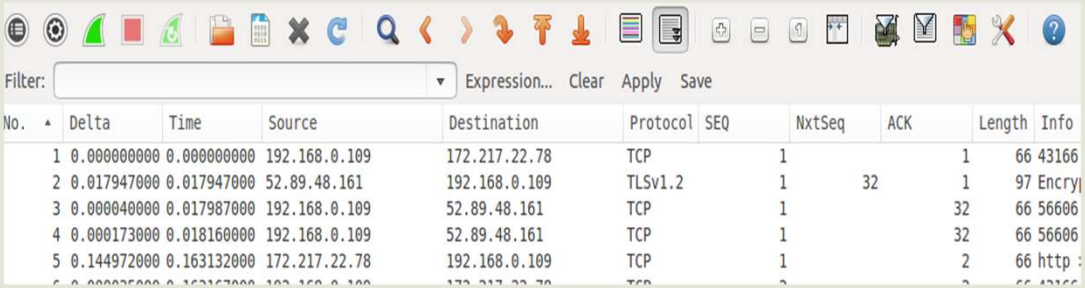
# Malware Analysis



# Wireshark

**Wireshark** is the world's foremost and widely-used network protocol analyzer. It lets you see what's happening on your network at a microscopic level and is the de facto standard across many commercial and non-profit enterprises, government agencies, and educational institutions.

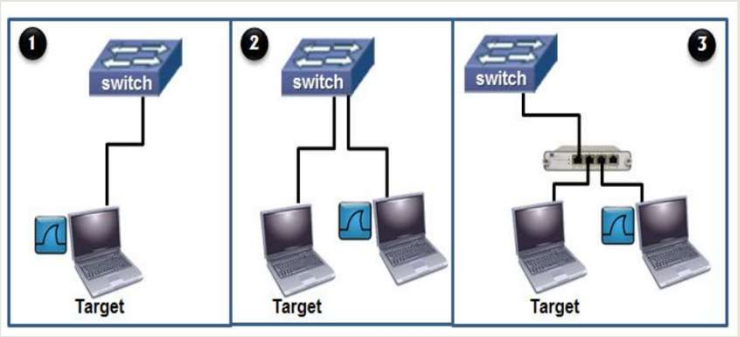
**Wireshark** development thrives thanks to the volunteer contributions of networking experts around the globe and is the continuation of a project started by **Gerald Combs in 1998**.

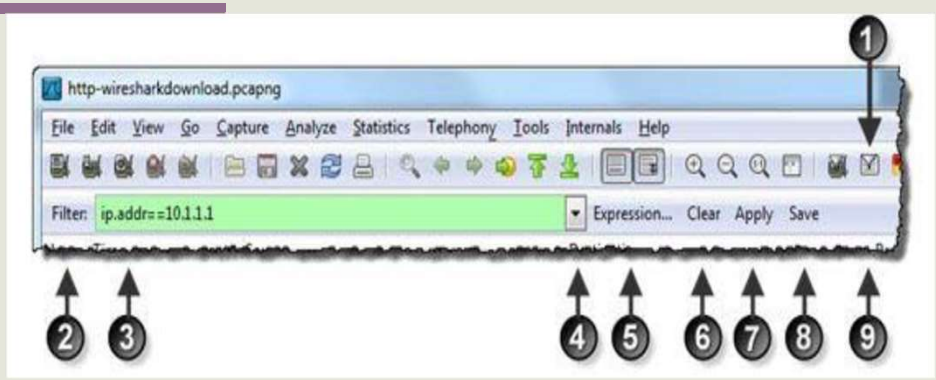


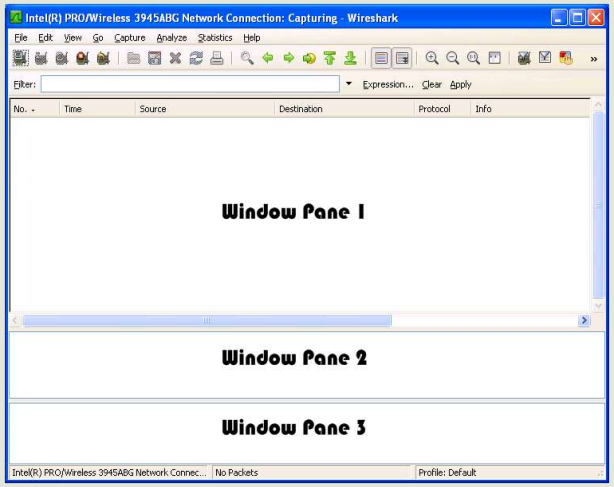

**Wireshark** has a rich feature set which includes the following:

- Deep inspection of hundreds of protocols, with more being added all the time
- Live capture and offline analysis
- Multi-platform: Runs on Windows, Linux, macOS, Solaris, FreeBSD, NetBSD, and many others
- Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
- Read/write many different capture file formats: tcpdump (libpcap), Pcap NG, Catapult DCT2000, Cisco Secure IDS iplog, Microsoft Network Monitor, Network General Sniffer® and many others
- Capture files compressed with gzip can be decompressed on the fly
- Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform)
- Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
- Output can be exported to XML, PostScript®, CSV, or plain text

# Wireshark

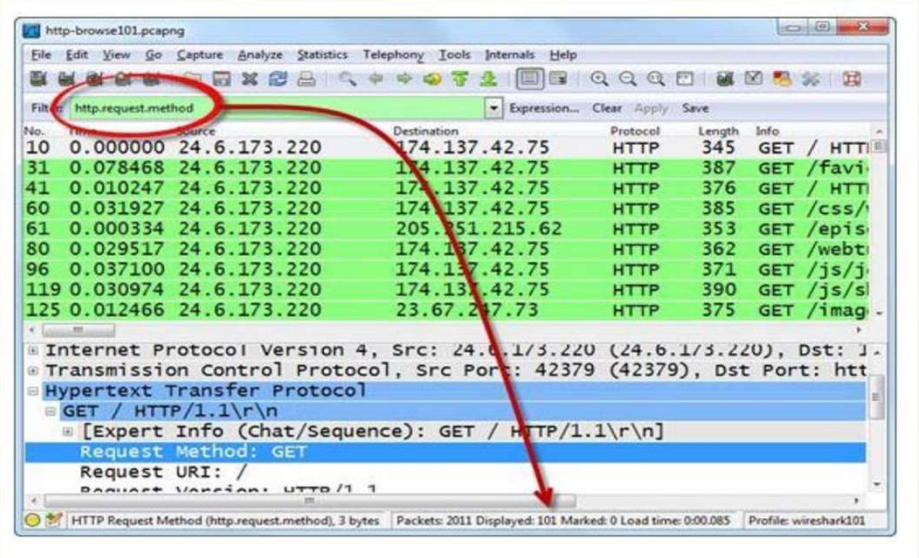


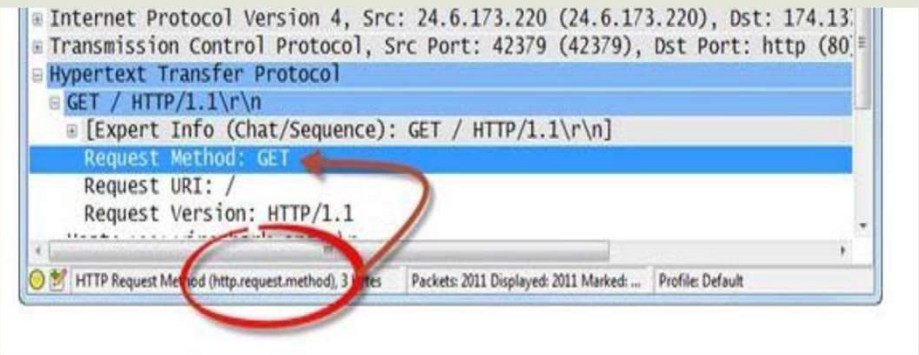



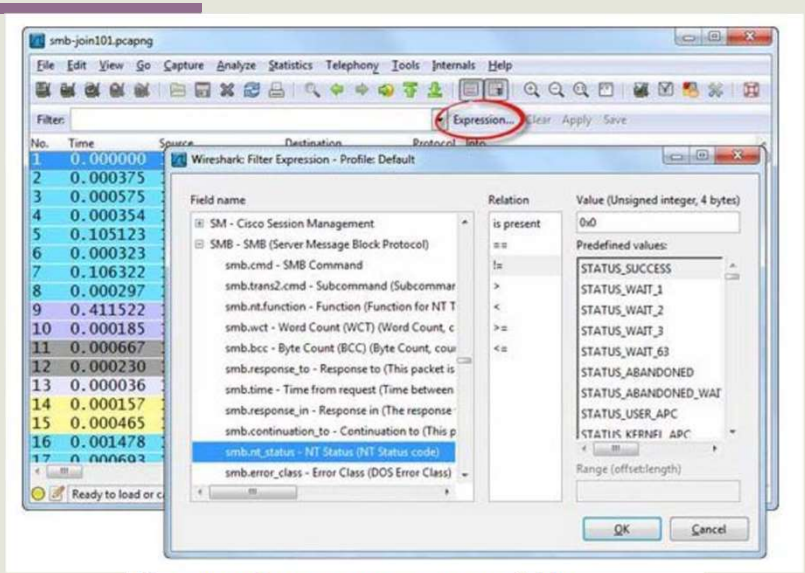




1. View, edit and create display filters (main toolbar)
2. Display Filters button (same as (1) another way to view, edit and create display filters)
3. Display Filter Area (includes auto-complete and error detection)
4. Last used display filter drop down list
5. Expressions to walk you through creating display filters
6. Clears the display filter area so no display filter is applied to the trace file
7. Applies the currently shown display filter during a live capture or to an opened trace file
8. Saves the display filter as a Filter Expression button
9. Filter Expression Button area (blank until new buttons are created)

# Wireshark




Six comparison operators are available:

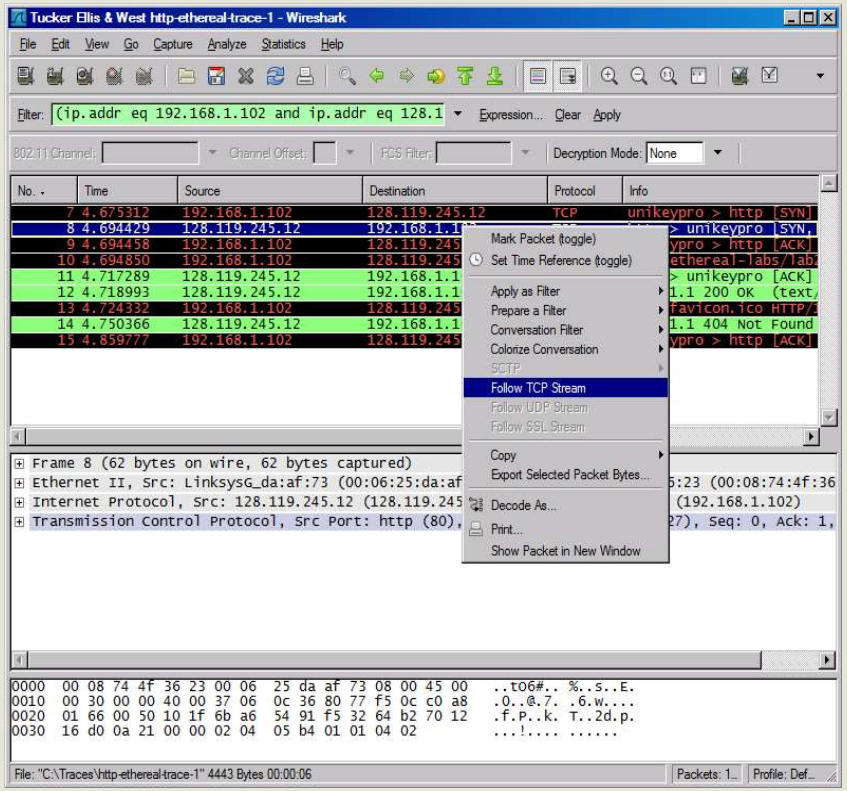
| English format: | C like format: | Meaning: |
|---|---|---|
| eq | == | Equal |
| ne | != | Not equal |
| gt | > | Greater than |
| lt | < | Less than |
| ge | >= | Greater or equal |
| le | <= | Less or equal |

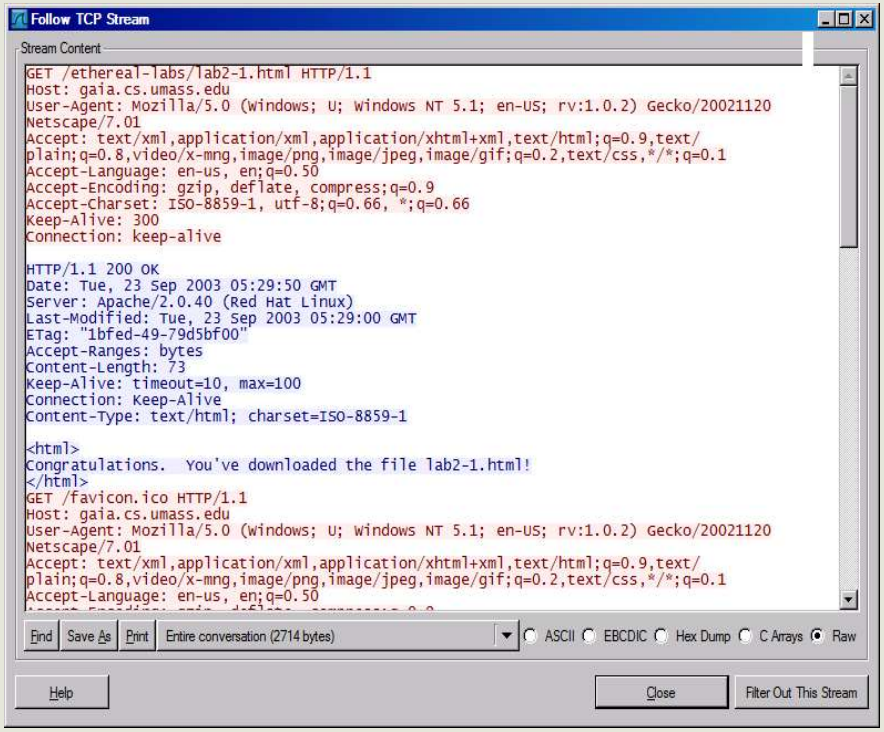
→ **Logical expressions**:

| English format: | C like format: | Meaning: |
|---|---|---|
| and | && | Logical AND |
| or | \|\| | Logical OR |
| xor | ^^ | Logical XOR |
| not | ! | Logical NOT |

# Wireshark

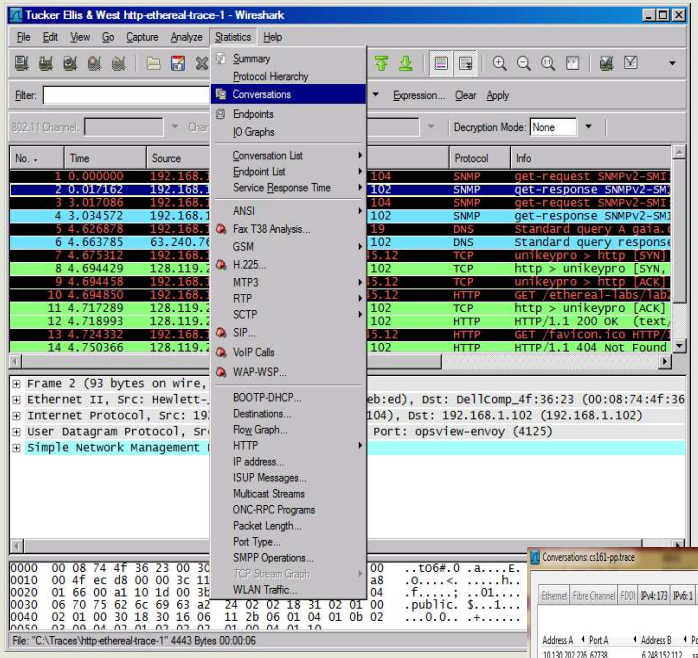
**Follow TCP Stream**

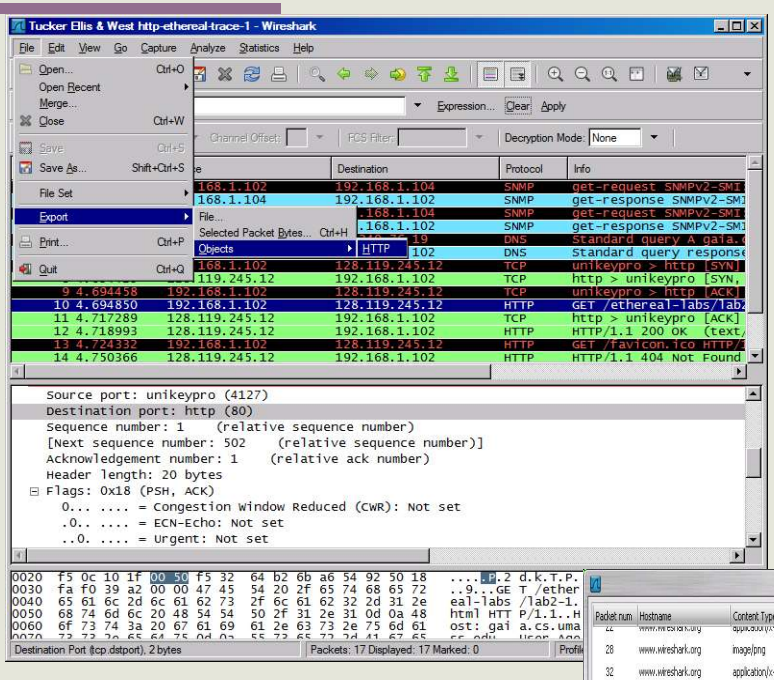


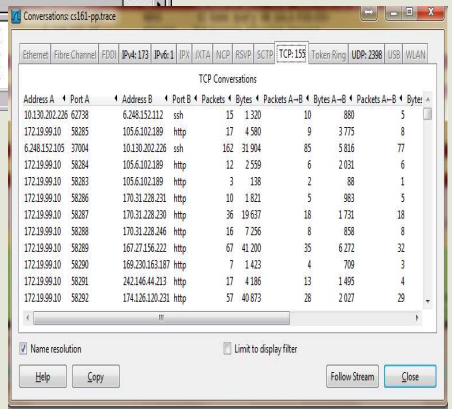



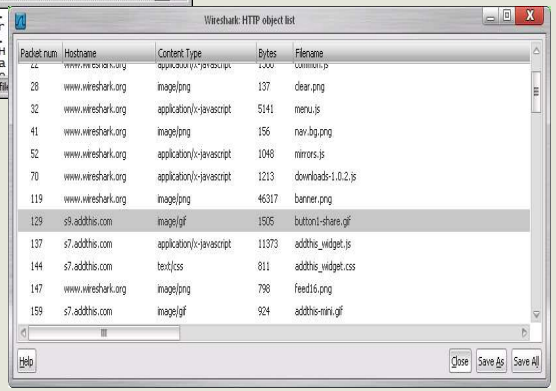

**red - stuff you sent**
**blue - stuff you get**

# Wireshark


Tucker Ellis & West http-ethereal-trace-1 - Wireshark
File Edit View Go Capture Analyze Statistics Help
Summary
Protocol Hierarchy
Conversations
Endpoints
IO Graphs
Conversation List
Endpoint List
Service Response Time
ANSI
Fax T38 Analysis...
GSM
H.225...
MTP3
RTP
SCTP
SIP...
VoIP Calls
WAP-WSP...
BOOTP-DHCP...
Destinations...
Flow Graph...
HTTP
IP address...
ISUP Messages...
Multicast Streams
ONC-RPC Programs
Packet Length...
Port Type...
SMPP Operations...
TCP Stream Graph
WLAN Traffic...
Frame 2 (93 bytes on wire,
Ethernet II, Src: Hewlett-
Internet Protocol, Src: 19
User Datagram Protocol, Sr
Simple Network Management
File: "C:\Traces\http-ethereal-trace-1" 4443 Bytes 00:00:06


Tucker Ellis & West http-ethereal-trace-1 - Wireshark
File Edit View Go Capture Analyze Statistics Help
Open... Ctrl+O
Open Recent
Merge...
Close Ctrl+W
Save Ctrl+S
Save As... Shift+Ctrl+S
File Set
Export
Print... Ctrl+P
Quit Ctrl+Q
File...
Selected Packet Bytes... Ctrl+H
Objects
HTTP
Source port: unikeypro (4127)
Destination port: http (80)
Sequence number: 1 (relative sequence number)
[Next sequence number: 502 (relative sequence number)]
Acknowledgement number: 1 (relative ack number)
Header length: 20 bytes
Flags: 0x18 (PSH, ACK)
0... .... = Congestion Window Reduced (CWR): Not set
.0.. .... = ECN-Echo: Not set
..0. .... = Urgent: Not set
Destination Port (tcp.dstport), 2 bytes
Packets: 17 Displayed: 17 Marked: 0


Conversations: cs161-pp.trace
TCP Conversations
Name resolution
Limit to display filter
Help
Copy
Follow Stream
Close


Wireshark: HTTP object list
Packet num
Hostname
Content Type
Bytes
Filename
www.wireshark.org
image/png
clear.png
menu.js
nav.bg.png
mirrors.js
downloads-1.0.2.js
banner.png
s9.addthis.com
button1-share.gif
s7.addthis.com
addthis_widget.js
addthis_widget.css
feed16.png
addthis-mini.gif
Help
Close
Save As
Save All

# Snort

What is Snort?

Snort is a multi-mode packet analysis tool

- Sniffer
- Packet Logger
- Forensic Data Analysis tool
- Network Intrusion Detection System

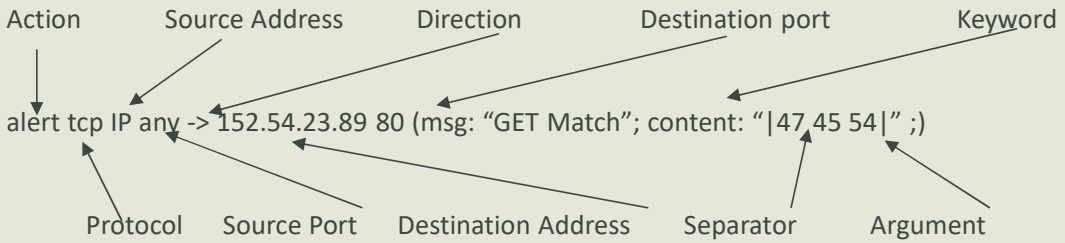
Where did it come from?

Developed out of the evolving need to perform network traffic analysis in both real-time and for forensic post processing

- Snort rules have two logical parts
  - Rule Header

| Action | Protocol | Source Address | Source Protocol | Direction | Destination address | Destination Protocol |
|---|---|---|---|---|---|---|

  - Rule Options
- Rule Options follow the Rule header and they are enclosed in closed braces
- Example of a simple rule



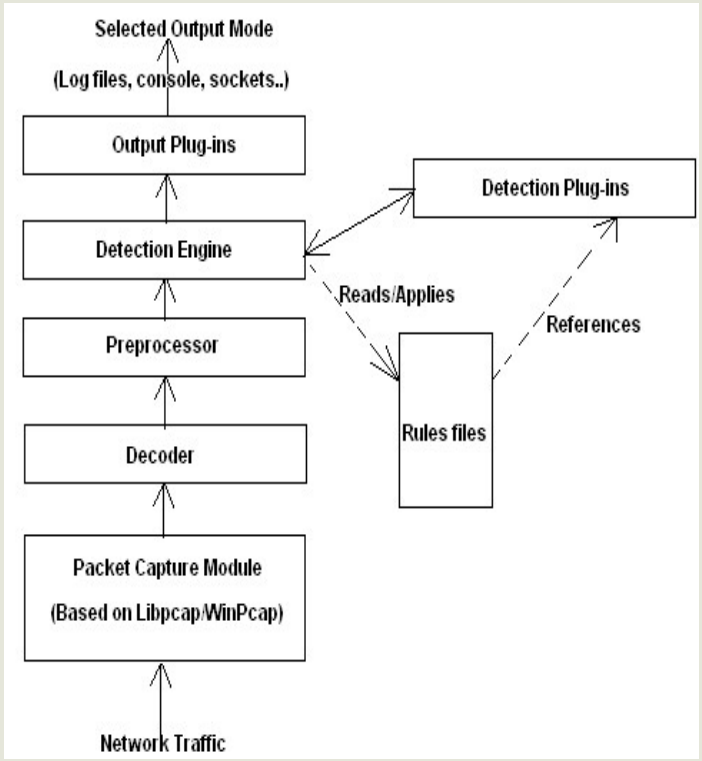
# Arhitectura Snort

*1. Decoder-* fits captured packets into data structures and identifies link level protocols. Then it takes the next level, decodes IP, and TCP/UDP to get information about port addresses.

Snort alerts for malformed headers, unusual TCP option.

should be checked later in Detection Engine module (like suspicious connection attempt to some TCP/UDP port or too many UDP packets received during a port-scan).

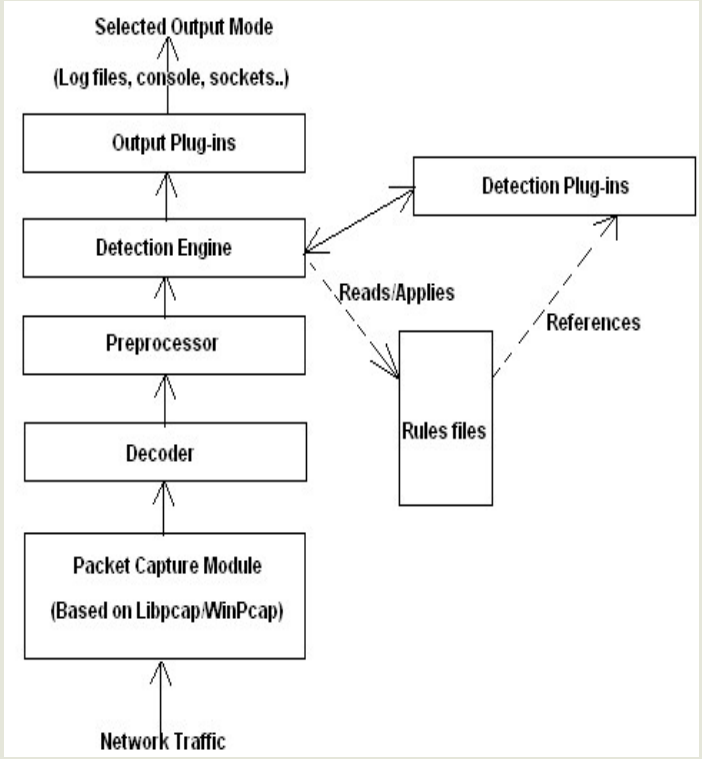
3. *Rule files:* Text files with rule sets written with a known syntax.
4. *Detection Plug-ins:* Those modules referenced from its definition in the rule files, and they are intended to identify patterns whenever a rule is evaluated.
5. *Detection engine:* Making use of detection plug-ins, it matches packets against rules previously charged into memory since snort initialization.
6. *Output plug-ins:* Alerts, logs, extern files, databases.



# snort.conf setup

Set the variables for your network

```
var HOME_NET [10.1.1.0/24,192.168.1.0/24]
var EXTERNAL_NET any
var DNS_SERVERS $HOME_NET
var HTTP_SERVERS $HOME_NET
var HTTP_PORTS 80
var ORACLE_PORTS 1521
var RULE_PATH /etc/snort/rules
```



# Action

Purpose of this field is to show what action will be taken when rule conditions are true

There are five predefined actions

Pass: Tells snort to ignore the packet

Pass icmp any any -> 192.324.3.23 any (msg:" pass example"; sid:123938;)

Log: Used to log a packet

Log tcp any any -> 192.324.3.23 any (msg:" log example"; sid: 102938;)

Alert: An alert message is generated when the rule conditions are met

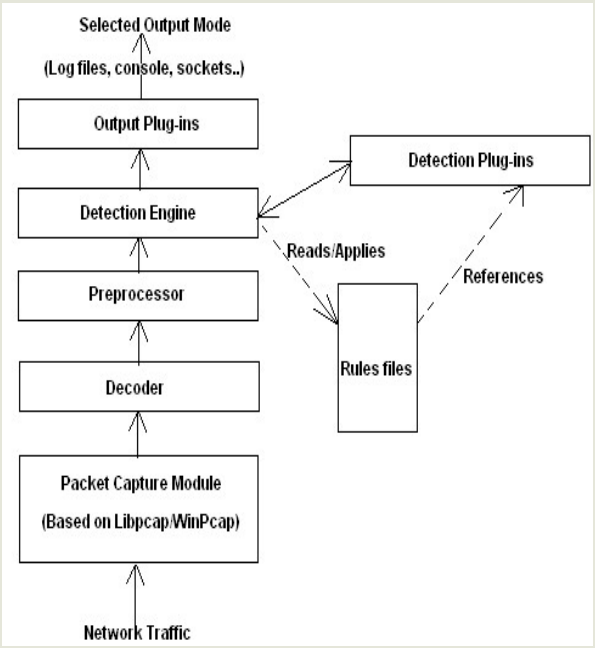
Activate: Used to create an alert and then to activate another rule for checking more conditions

Dynamic: These are invoked by other rules using activate action

Activate tcp any any ->any 143 (msg:"IMAP buffer overflow";content:"\bin";flags:PA;activates:1;)

Dynamic tcp any any -> anyh 143 (activated_by:1; count:50)

In addition to these actions you can define your own actions



# Protocol

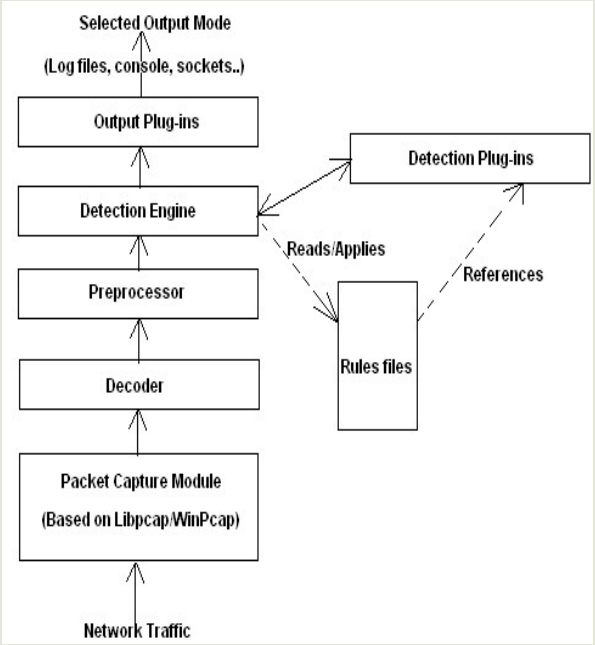
This part shows on which type of packet the rule will be applied.

These are the protocols used in snort,

- IP
- ICMP
- TCP
- UDP

Log udp any any -> any any (msg:"udp packet detected"; sid:1234534;)

Alert tcp any 21-> 152.39.23.4 any (msg: "Tcp packet"; sid:1245323;)



# Address

There are two address parts in a snort rule

The address may be single IP address or a network address. Keyword “any” is used for all addresses

You can also specify list of addresses in a Snort rule

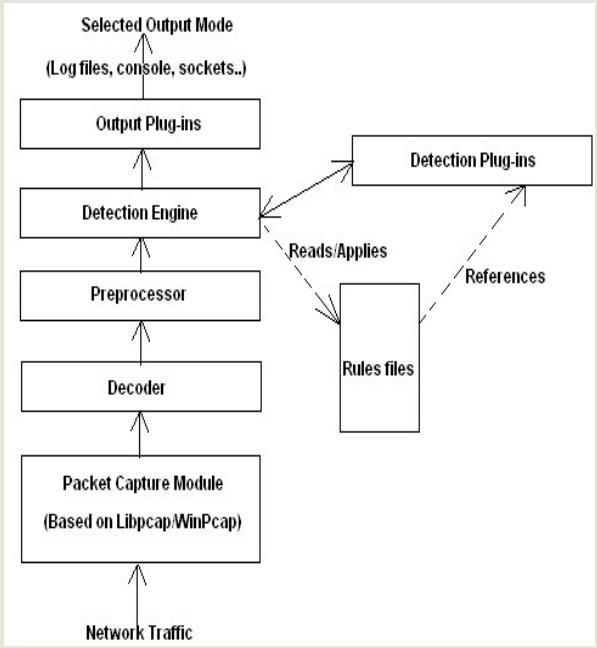
These addresses can be separated by comma included in square brackets

These specifications are applied to both source and destination addresses

Negation symbol is used to exclude the address

Alert tcp !152.8.39.48 any -> [192.4.23.3,31.3.2.1] any (msg: example for range of address”; sid:129845;)

Alert ip 192.168.1.0/24 any -> 192.168.0.0/16 any (msg:”example for netmask”; sid:198749;)



# Port Number

Port number is used to apply a rule on packets that originate from or go to a particular port or a range of ports

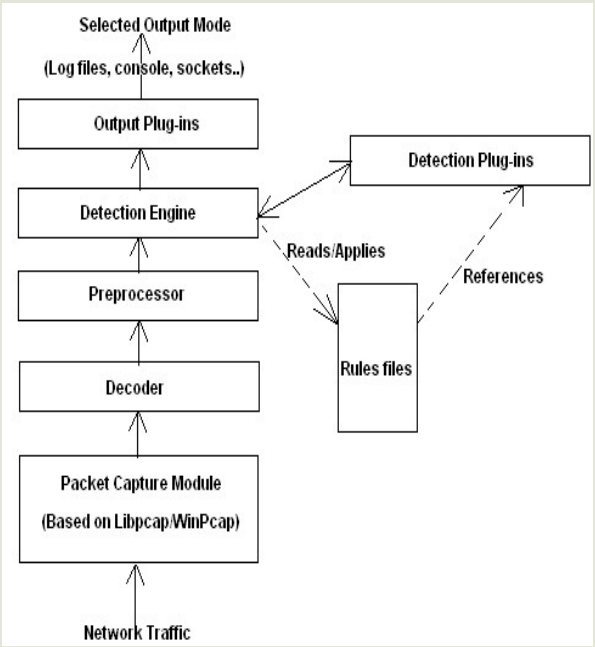
Keyword “any” is used to apply the rule on all packets irrespective of the port number

Range of ports is specified by using a colon to separate starting and ending port numbers

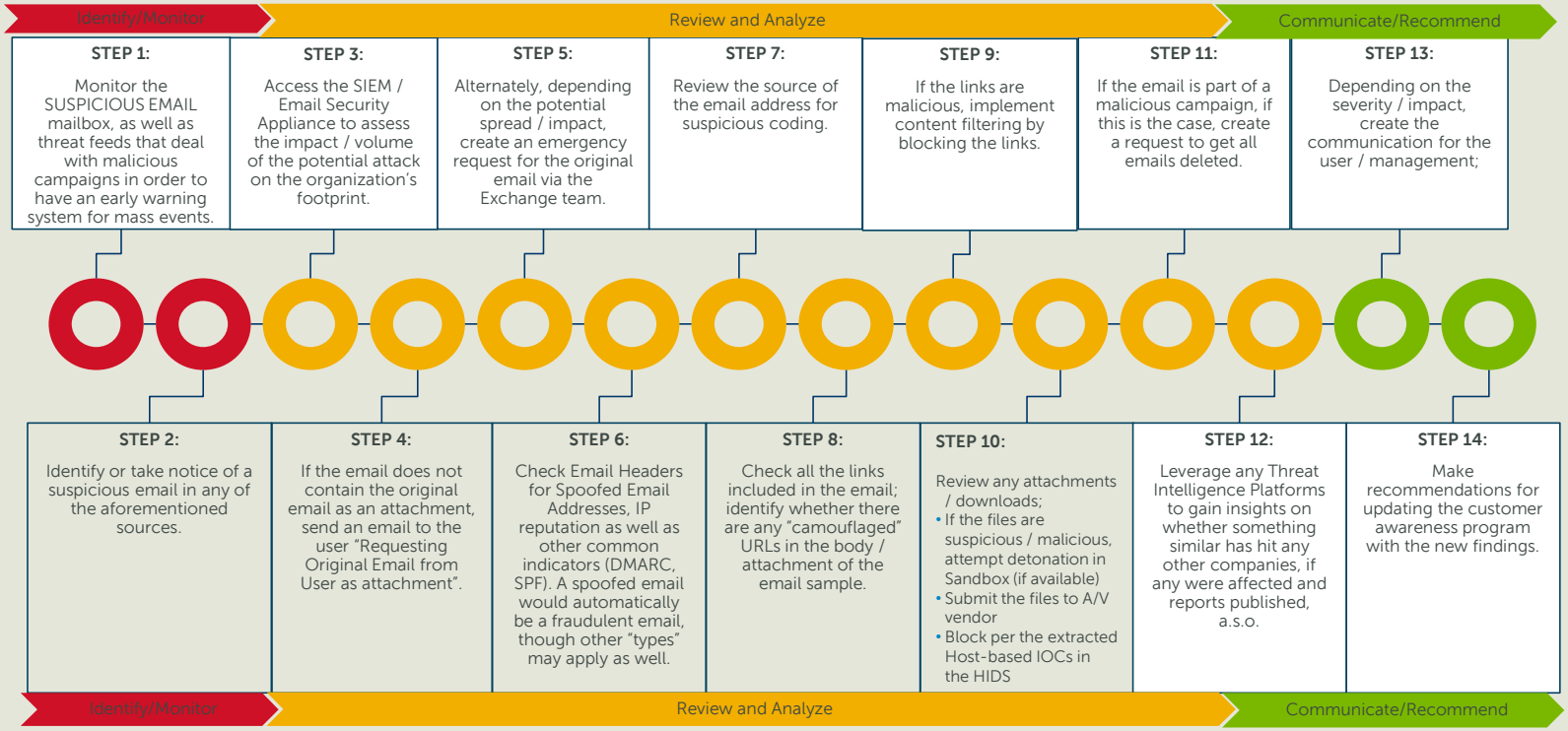
Log tcp any any ->152.8.38.254 1:1024 (msg:”port range example”;

sid:124353;)

Log tcp any any-> 143.5.3.1 80: (msg:”another example”; sid:1253422;)



# Scenario #1: Suspicious Email



# Scenario #2: Host Infection