# Malware Analysis Framework

**-+Table of Contents:**



# Introduction

## Overview

The Malware Analysis Framework, developed by FIRSTs Malware Analysis Special Interest Group (SIG), is a document aimed to help CSIRTs establish their own malware analysis workflow(s). It provides step-by-step guidance in all workflow phases on how to develop malware analysis capabilities within CSIRTs. This document also lists supporting resources that can further assist in understanding how malware analysis procedures can be carried out efficiently.

## Motivation

Nowadays, most cyberattacks have a malware component enabling threat actors to succeed in their mission. Whether to gain initial access into an organization, steal confidential information or encrypt sensitive data with ransomware – threat actors are dedicated in developing new malware and delivering it to organizations worldwide to cause damage. As such, to efficiently and promptly prevent and manage potentially impactful incidents, CSIRTs must understand how their constituents can become infected with malware and develop strategies that enables responders to prioritize, analyze, and remediate malware-related incidents.

Malware Analysis is an important part of digital forensics and incident response (DFIR) for all types of organizations. The more work is done by digital assets - which collect, process and store data and are interconnected with each other - the greater the attack surface becomes. To protect your organization from such attacks and to detect an attack before having a significant impact on your organization, any security function such as SOC and CSIRT Teams should have malware analysis capabilities to be able to identify and prioritize incidents accordingly.

### Success Stories

The importance of running malware analysis processes inside CSIRTs can be further justified with the following examples, demonstrating how malware analysis have been critical to be able to prevent major global cybersecurity incidents:

- In February 2024, Korea Internet and Security Agency published a decryption tool for Rhysida, a ransomware that was used in attacks against many organizations, including ones operating in government, information technology and healthcare sectors. Analysis of this ransomware helped researchers find fatal mistakes the malware developers made in code responsible for file encryption. Thanks to the presence of these mistakes, researchers have become able to recover files encrypted with Rhysida and thus eliminate the need to pay ransom to attackers to restore valuable data
- In May 2022, several cybersecurity companies that have been working in collaboration with CSIRTs of Industrial Control Systems companies discovered a previously unknown malware called Incontroller (also known as PipeDream) that was developed to sabotage industrial processes. What is notable is that this malware had been detected before it was deployed in the wild. Due to that, a prompt analysis of this malware allowed security teams of industrial organizations to be better prepared for deployments of Incontroller and minimize impact it could make to critical infrastructure.
- In 2017, during the outbreak of the WannaCry ransomware, the security researcher Marcus Hutchings found out that the ransomware had a kill switch mechanism inside it. It checked whether a specific domain name consisting of random letters (namely ifferfsodp9ifjaposdfjhgosurijfaewrwergwea[.]test) was registered and stopped working if so. Hutchings was able to register this domain name, and this in turn made it possible to terminate operations of this ransomware and significantly slow down spreading of WannaCry.

## Outcomes

The Malware Analysis Framework intends to provide generic and high-level guidance on how malware analysis workflow(s) can be performed as part of CSIRT operations. These workflows can be customized to each team based on current maturity, needs and resources available. It is expected that studying information provided in this Framework will help CSIRTs decide, if they require malware analysis capabilities, and if so, which types of capabilities should be developed by the CSIRTs themselves and which capabilities should be outsourced.

As a result of developing malware analysis strategies, CSIRTs will eventually become able to:

- Understand which methods are commonly used to enter digital assets and infrastructure of its constituents
- Identify which techniques threat actors use to stay undetected, once they gain access to the infrastructure of its constituents
- Improve monitoring and defense mechanisms by providing stakeholders with facts about observed tactics used by threat actors

### Collaboration

Malware analysis is a vast domain. Results produced during malware analysis can be useful to multiple different teams working in cybersecurity. Specifically, information about how particular attackers or a particular malware function can be used to further adapt the constituencies monitoring and detection systems. Sharing this information is vital to effective incident response, and FIRST, as a global organization with global outreach, has several Special Interest Groups (SIG) for whom this information is helpful:

- The Cyber Threat Intelligence SIG can be interested in techniques, tactics and procedures (TTPs) used by threat actors, as well as technical details justifying attribution of malware to a particular actor
- The Information Sharing SIG can provide guidance on organizing information exchange of data such as indicators of compromise (IOCs)
- The Red Team SIG can research tools used by threat actors to conduct initial access and lateral movement operations
- The Vulnerability Reporting SIG can further study vulnerabilities exploited by threat actors and provide guidance on vulnerabilities based on trends

Additionally, information about the inner workings of malware and mitigation strategies may also provide useful for information hubs, such as:

- Information Sharing and Analysis Center (ISAC) that provide a central resource for gathering information on cyber threats (in many cases to critical infrastructure) as well as allow two-way sharing of information between the private and the public sector about root causes, incidents and threats, as well as sharing experience, knowledge and analysis
- Information Sharing and Analysis Organization (ISAO) that gathers and analyzes critical cyber and related information in order to better understand security problems and inter-dependencies related to cyber systems, so as to ensure their availability, integrity, and reliability

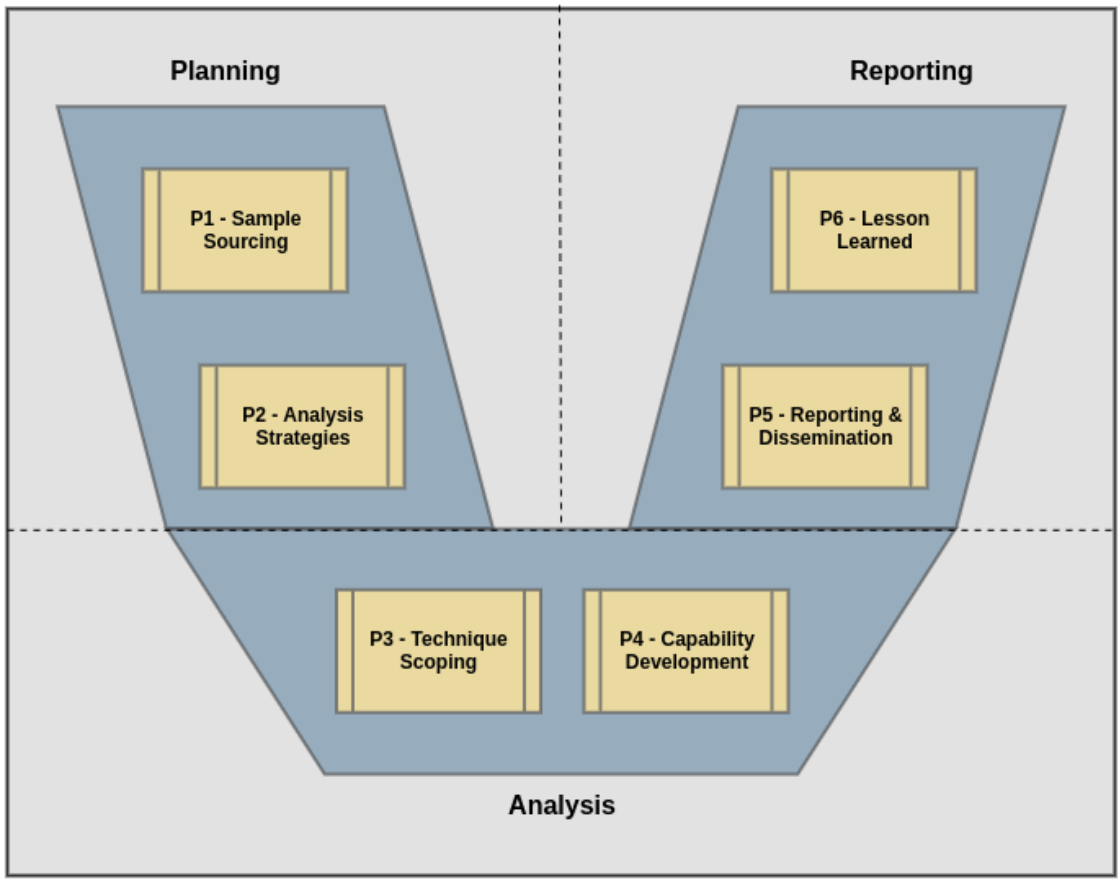
# Phases Overview

The Malware Analysis Framework is structured into six phases with various scopes:

- The **planning scope** involves establishing a robust framework that ensures the timely collection, analysis, and prioritization of malware samples to safeguard the constituents of a CSIRT. This requires identifying and defining reliable sources of malware samples, creating strategies for prioritizing malware analysis, and developing procedures for monitoring and improvement of malware analysis capabilities.

- The **analysis scope** involves determining the depth of investigation required for each malware sample, defining the goals of the analysis, and using various methods to extract actionable insights. This process helps CSIRTs understand the threat posed by the malware and how to respond to it effectively.
- The **reporting scope** involves documenting the findings from the malware analysis in a clear, structured, and actionable format. Reporting is crucial for sharing insights with various stakeholders, from technical analysts to business leaders, and guiding future defense strategies. Additionally, lessons learned activities include a review of the outcome of the malware analysis workflow based on defined goals, and a list of observables that can be considered for next iteration.

This is known as the V-model as illustrated below:




# Phase 1 – Identifying Malware Collection Sources

In order to make sure that a CSIRT is able to timely receive malware samples that currently pose a threat to its constituents, it is important to understand which sources can be used to get malware samples for analysis activities. The type of available sources in this list will depend on the type of CSIRT, their maturity level and telemetry sources.

For CSIRTs that provide endpoint security services to constituents, reports of endpoint protection solutions installed can become extremely valuable in obtaining malware samples. A vast majority of malware attacks, both mass-spread and targeted ones, start with infection of endpoints, for example via phishing. When collecting malware from endpoints, it is additionally beneficial to gather basic information detailing origins of the malware sample. For example, in the case of malware downloaded from a website, it can be useful to identify the link used for downloading, as well as how this link was accessed (e.g. by clicking on a malicious advertisement in a search engine). Or, if the malware was found attached to an email, it is possible to retrieve the headers and body of the malicious email message.

Another location to search for malware samples are servers used by constituent organizations. Infecting servers by using exploits is another popular way for attackers to breach internal perimeters of organizations. As such, it is possible to detect attempts to deploy these exploits in network traffic and extract malware sample payloads from them for further analysis.

To quickly determine which endpoints and servers inside constituent organizations are likely to be attacked with malware, it is possible to consult asset inventories that typically is used by organizations to keep track of used devices, such as a CMDB. If these inventories are available, they should be regularly checked for updates.

More mature CSIRTs can additionally use external sources, such as threat data feeds or threat intelligence reports, to find malware samples for analysis. For instance, if a CSIRT notices an increase in attacks targeting organizations within a specific industry or region and seeks to understand the techniques used, it can leverage external feeds to identify the malware commonly involved in these attacks and conduct further analysis.

# Phase 2 – Developing Analysis Prioritization Strategies

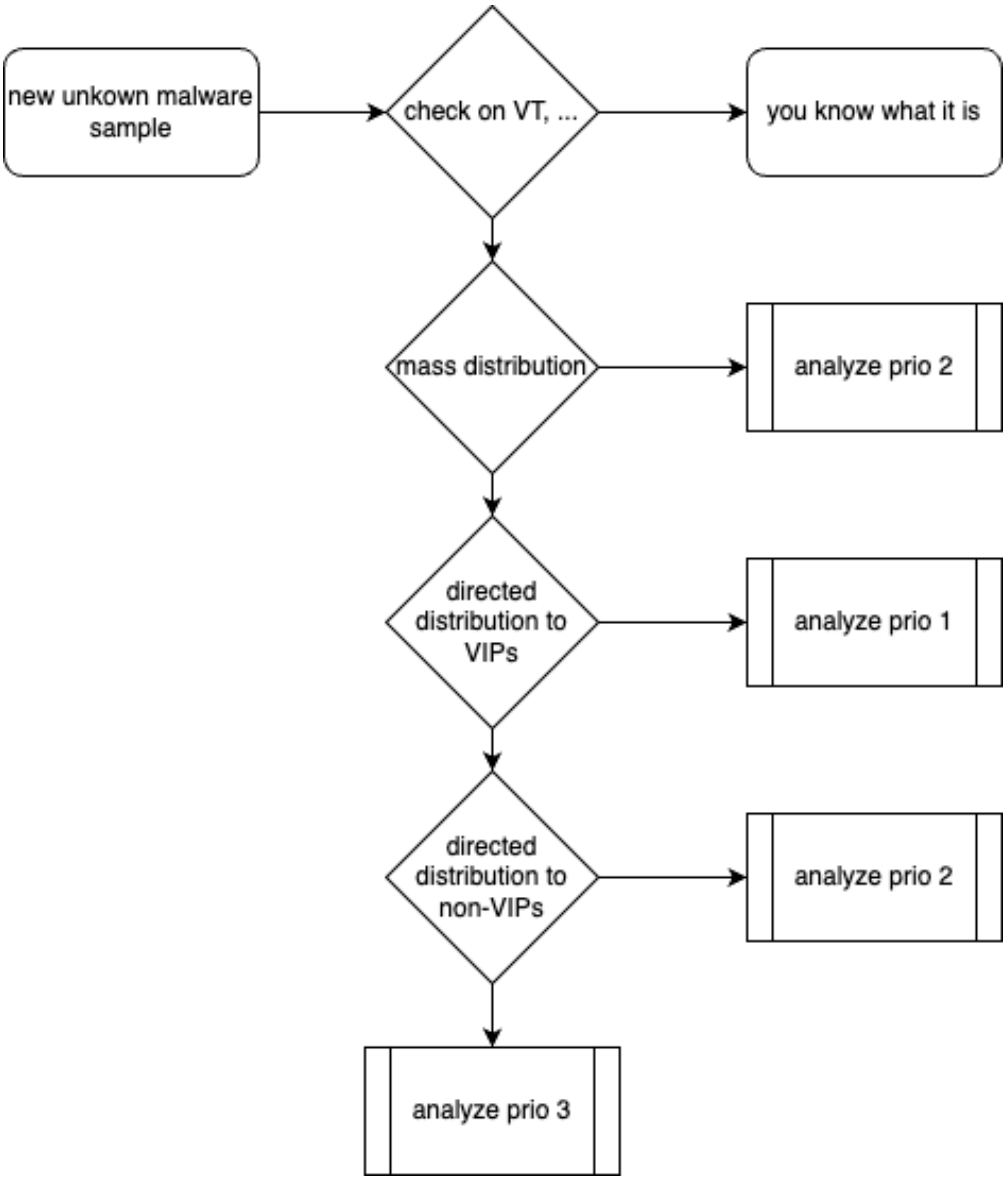
Once a CSIRT starts receiving malware samples from its constituents, it will be necessary to rank their processing order and resource allocation. Malware analysis can be a very resource-demanding task and a CSIRT Team will likely not be able to process all samples received within the same working day.

The context about the received malware samples that was discussed in the description of Phase 1 can be key to successfully prioritizing incoming malware samples. It is possible to use information about the samples, such as the URL where it was downloaded or the email address from which it was sent, to carry out quick open-source checks by using search engines and malware multi-scanning services. These checks should take minimal time and may identify that a sample is already known to the wider cybersecurity community. This pre-existing knowledge may in some cases be sufficient to triage the sample and be used as a decision point on possible further actions. Examples:

- Previously described in a public threat intelligence report
- Previously dissected by a reverse engineer in open (or partnered) sources
- Previously uploaded to a multi-scanner platform and is detected by the majority of security solutions

Once established that a sample is unknown, it is beneficial to check if it is used in a targeted attack and which asset it originated from to further determine how urgently it needs to be analyzed. If the malware appears to be generic, it may not be worth further investigating it because mass-distributed samples are usually very quickly detected by security solutions. On the other hand, if the malware was deployed via a spear phishing attack, i.e. in an email mentioning information of the target organization, which indicates a targeted attack, and thus should be handled with higher priority. Furthermore, if the identified malware is specifically targeted against a high-profile person (e.g. business leader) or a critical department inside the target organization (e.g. finances or R&D), the malware should be handled with especial urgency.

The following graph provides an example of a possible prioritization workflow to determine next analysis steps:




**Important takeaways:**

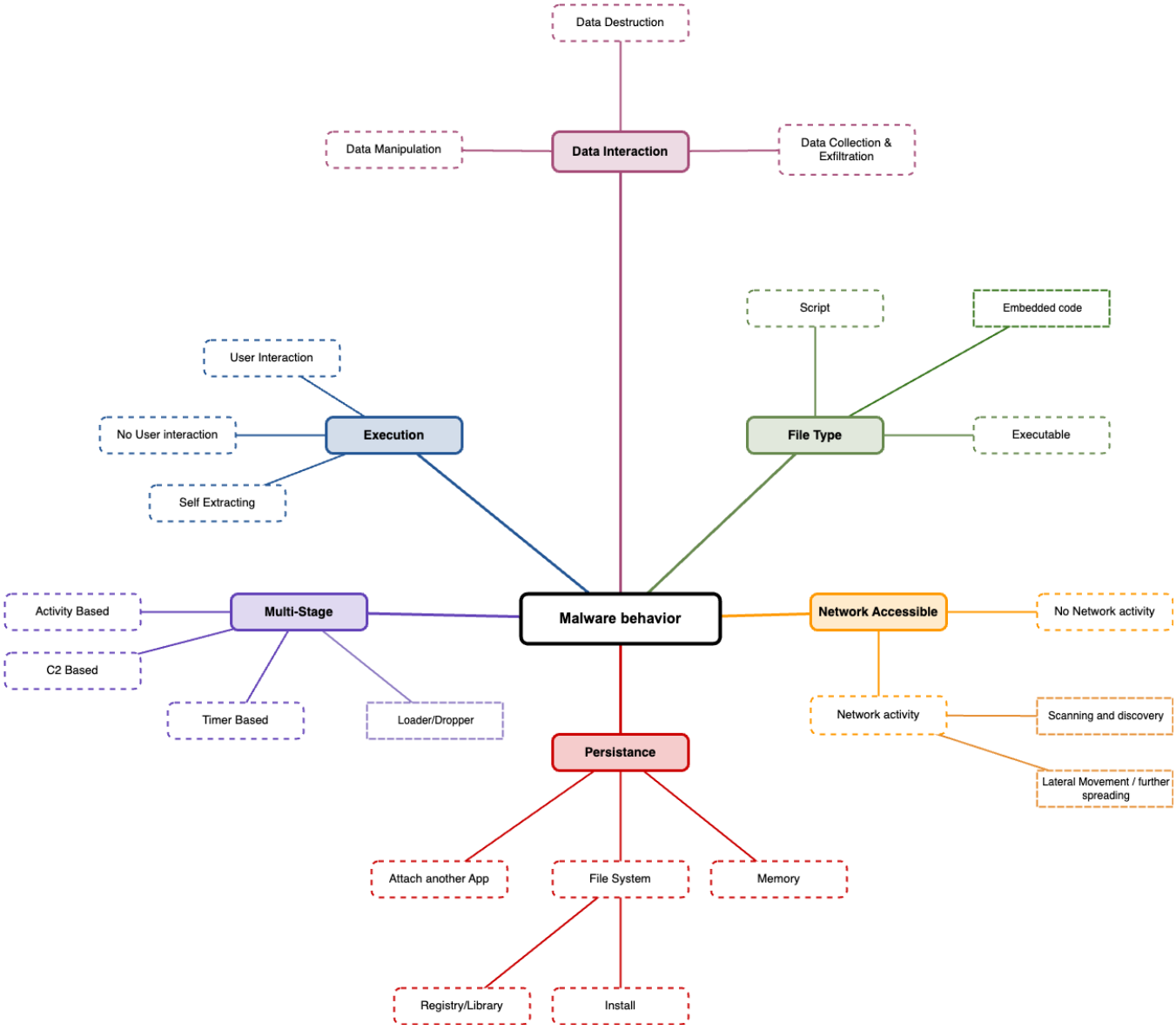
- Don't duplicate work: what has been analyzed before, doesn't need to be analyzed again

- Mass-spreading malware is often less interesting, as security engines (e.g. AV, Proxies, EDRs) usually update their threat coverage quite frequently
- The type of entry point, exposure channel or target may be an indication of the threat actors motive:
    - Directed distribution to VIP employees (e.g. C-Suite, finance, R&D) can be an indication of a threat actor looking for something specific (e.g. intellectual property) or is using some specific malware type (could be tailor-made for this attack)
    - Directed distribution to non-VIP employees could be an attempt to get a foothold into your environment and continue through lateral movement

# Phase 3 – Defining Malware Analysis Goals

Once decided that a malware sample should be analyzed, it is important to determine how deep the analysis should be. Modern malware can consist of multiple stages and a full detailed analysis of all layers may take significant time. Allocating all malware analysis resources on a single sample for multiple days may not be possible nor even optimal during an incident. Also, in many cases it may not be necessary to conduct such a full-fledged analysis. For example, if the identified malware is known to connect to a command-and-control (C2) server and the goal is to find out whether any machines inside a network have been communicating with it, it will suffice to check if any devices accessed the C2-server by inspection the network traffic.

Typically, the depth of analysis carried out depends on the set of questions that need to be answered about a particular malware – the more questions there are, the longer the analysis will take. Below is a list of possible questions that can be used to define the goals of the analysis and how in-depth it should be.




## Q1 - What is the format of the analyzed malware?

Analyzed malware can come in different forms. Most malware observed are executables - files that are compiled for the target hosts and that can work independently (in the case of standalone executables, such as .exe files) or as part of another application (in the case of .dll files that require to be loaded in a standalone executable to function). However, it is also common for malware to be embedded into non-executable files, such as malicious documents containing macros, or come in the form of a script that is run via a runtime engine (e.g. PowerShell or AutoIT malware).

- Script: the malware is written in some kind of scripting language and needs the corresponding run-time engine to be available on the system to be executed (e.g. JavaScript, .NET, Shell scripts)
- Embedded code: the malware is embedded within a legitimate file type like an Office Document and might be automatically executed when the file is opened or requires some interaction by a user (e.g. clicking a link, enabling a macro, ...)
- Executable: the malware is a fully functional application compiled for the target host and can be started without the need of a specific run-time engine or host application

## Q2 - What platforms and systems are targeted by the analyzed malware?

Malware targets one or multiple types of digital devices, examples of classifications:

- Endpoints (e.g. requires human interaction for manual execution)
- Servers (e.g. deployed to exposed server machines with vulnerabilities)
- Mobile devices (e.g. phones or tablets through malicious apps)
- IoT devices (e.g. run on less popular architectures, such as ARM, MISP or RISC-V. Notable example is the Mirai botnet)
- Industrial control systems (e.g. crafted to execute on specific configurations in an OT network. Notable example is Stuxnet)

It should also be noted that some malware may be intended to run on multiple platforms at once (i.e. platform independent). An example of such malware can be malicious browser extensions that are commonly coded in cross-platform scripting languages (e.g. JavaScript).

## Q3 - How is the analyzed malware intended to be executed for the first time?

Execution of malware samples can be triggered in many different ways. Most malware deployed to endpoints require user interaction – such as launching an executable file with the malware itself or enabling macros in an infected document. Conversely, malware deployed to servers is usually installed without any manual interaction – as mentioned above, threat actors usually find vulnerabilities affecting software run on software and exploit them.

Additionally, some malware is intended to be manually executed by threat actors. This is usually the case with various red team tools that threat actors use to move laterally inside networks of infected organizations. Such tools are typically launched via other malware already installed on an infected machine.

- User interaction: the user needs to actively interact with the malware like clicking on a link in an infected office document
    - This can also include malware families that are distributed in a compressed format and first need to be extracted to be functional. The extracting processes are part of the malware and as soon as the process is started it will extract itself on the host system and start it's main processing steps
- No user interaction: as soon as the malware starts, another process starts without any user interaction
    - This can also include malware families that exploit zero-click vulnerabilities in web-browsers, email-clients, applications and similar.

## Q4 - Is the malware configured with persistence mechanisms to survive a reboot?

During the process of malware installation, it is common to configure deployed implants to start when the operating system boots up, which makes it possible for an infection to stay permanently on the host device. For example, on Windows devices a very popular way to configure persistence is to drop the malware on disk and then store the path to the malicious file in registry, for instance in the Run key. Another common method of making the malware infection persist after reboots is to design the malware to work as a plugin of some host application. Examples of such malware can be IIS server extensions or plugins for commonly launched software such as office tools.

In the case of more complex attacks, especially those affecting large enterprise networks, it may be possible that attackers may not configure all deployed malware samples to persist. Instead, they could use tools allowing to automatically scan an infected network for newly powered on machines and further deploy malware to their memory. In such a situation, deployed implants are not stored on disks of infected machines, which makes it more difficult to detect a compromise with security solutions.

- File system: the malware installs itself somewhere in the file system. An example is where the the malware adds commands to the system's registry to be automatically executed during a system start or installs itself as a library and might be started when the library is called
- Memory: the malware only resides in memory, which means a computer shutdown could be fatal to the malware and all traces would be lost. There are malware to be known to use a combination of such techniques. For example these are installed on the system and started via a registry key, as soon as the malware is running and loaded into memory, the malware deletes itself from the file system to be invisible to AV-scanners. During a proper shut down procedure of the host system, the malware would reinstall itself on the file system and reactivate the registry key to be functional again after the next reboot.
- Bundled with an app: the malware attaches itself to an other application (aka. host-app) to stay hidden on the system. As long as the host-app stays installed on the system, the malware might be able to execute the intended steps

## Q5 - How does the malware use network communications?

Malware may require access to a network to function properly. For example, it is popular for backdoor malware to regularly connect to an attacker-controlled server (C2-channel) and request commands from them. Once these commands are executed, the malware communicates with the server once again to upload the execution results to the threat actor.

It is also common for malware to interact with the internal network of an infected organization. This can be the case when the malware is performing port scanning or host discovery inside the internal network or propagating the infection to other hosts located inside the network.

However, it may also be possible for a malware sample to not perform network communications at all. Examples of such samples are wipers that perform data destruction or infostealers that collect sensitive data from infected machine and stage it on the filesystem for further exfiltration by a separate malware.

- No network activity: the malware does not use any form of network communication and works stand-alone (e.g. data destruction)
- Network activity: the malware does use available networks for further steps. Usually this includes
    - Home-calling to a command and control server (C2-channel) via the Internet to retrieve further commands or to exfiltrate data
    - Scanning and discovery: the malware scans the network or use discovery techniques to identify other hosts which could be attacked as well
    - Lateral movement / further spreading: the malware can directly attack other hosts on the network and infect them as well or use other techniques to move through the network

## Q6 - How does the analyzed malware interact with data during execution?

Answering this question is typically required to understand the impact that a malware sample can have on an infected organization. Typically, malware performs one or several of the following actions with data:

- Data collection and exfiltration: typical for attacks involving infostealers. In this case, retrieved data may be published without the consent of its owner or stored inside the threat actor's internal systems, if the attack goal is cyber espionage. In addition, collected data, such as usernames and passwords, may be used for further attacks.
- Data manipulation: generally results in losing trust in stored data. For instance, such manipulations are commonly carried out by financially motivated threat actors that interact with financial systems to transfer funds to their accounts. Data manipulation can also be used by actors attacking industrial control systems: by modifying configurations of programmable logic controllers, they are able to disrupt production processes or deal damage to industrial equipment.
- Data destruction: intended to make data unavailable by encrypting it with ransomware or deleting it with wipers.

## Q7 - Is the malware the complete package or does it use additional components?

A malware might not be the complete package, but only a single step in a multi-stage infection. A dropper might infect the system and get the first foothold on it. Afterwards an additional step will download more modules or applications and place them on the system to start the next step in a multi stage infection.

- Activity based: based on the activity of the host system or the user, the malware might load additional modules
- C2 based: after contacting a command and control server via the internet, the malware gets more commands or modules loaded to be able to offer further functionality
- Loader / dropper: the malware might only be the first step of the infection and loads or drops additional malware on the host system

## Q8 - How has the malware analyzed previously been used?

In order to extract as much information from malware analysis as possible, each sample analyzed should not be considered in isolation. Instead, it is paramount to use all the information known about a sample (e.g. filenames or addresses of command and control servers) to retrieve additional context about it. For example, it is beneficial to find out, if there have been different versions of the analyzed malware used in recent attacks, or other versions of the malware deployed previously. Such investigations often enrich information available about the analyzed malware, as they help reveal additional tactics and techniques used by the threat actor behind it.

In addition, while analyzing the history of how a particular malware was previously deployed, it may come to light that it is used to specifically target one particular industry. In such cases, it can be worthwhile to discuss operations of this malware in a special collaborative group, such as an ISAC/ISAO.

## Q9 - Which threat actor could have developed the analyzed malware?

Apart from identifying previous use cases of the analyzed malware, it is also beneficial to determine which threat actor is deploying it, i.e. perform its attribution. As many threat actors conduct their attacks with the use of same or similar methods, establishing the particular actor behind a malware can reveal additional techniques used by this actor. In turn, this will make it possible to further study these techniques, implement better defenses against them and thus become more resilient to other attacks conducted by the identified threat actor.

Typically, attribution of malware can be performed by identifying specific unique features of a malware and then searching existing threat intelligence reports and malware descriptions for use cases of these features. Examples of such features include:

- Unique combinations of tactics, techniques and procedures
- Names of files and directories
- Addresses of command and control servers previously observed in other attacks
- Unique code functions
- Text strings indicating that the malware developer is fluent in a particular language.

While performing malware attribution, it is important to remember that threat actors may deliberately insert "false flags" into their malware that can make researchers attribute the malware to a wrong actor. For instance, this was the case with the OlympicDestroyer malware developed by the APT28 threat actor and used during the 2018 Olympic Games. In an attempt to prevent this malware from being correctly attributed, this actor implanted the code of OlympicDestroyer with artifacts observed in attacks of APT3, APT10 and Lazarus actors.

# Phase 4 – Developing Analysis Capabilities

After determining the set of questions that need to be answered about a malware sample, the next step is to answer them through malware analysis. There are different analysis approaches to do this:

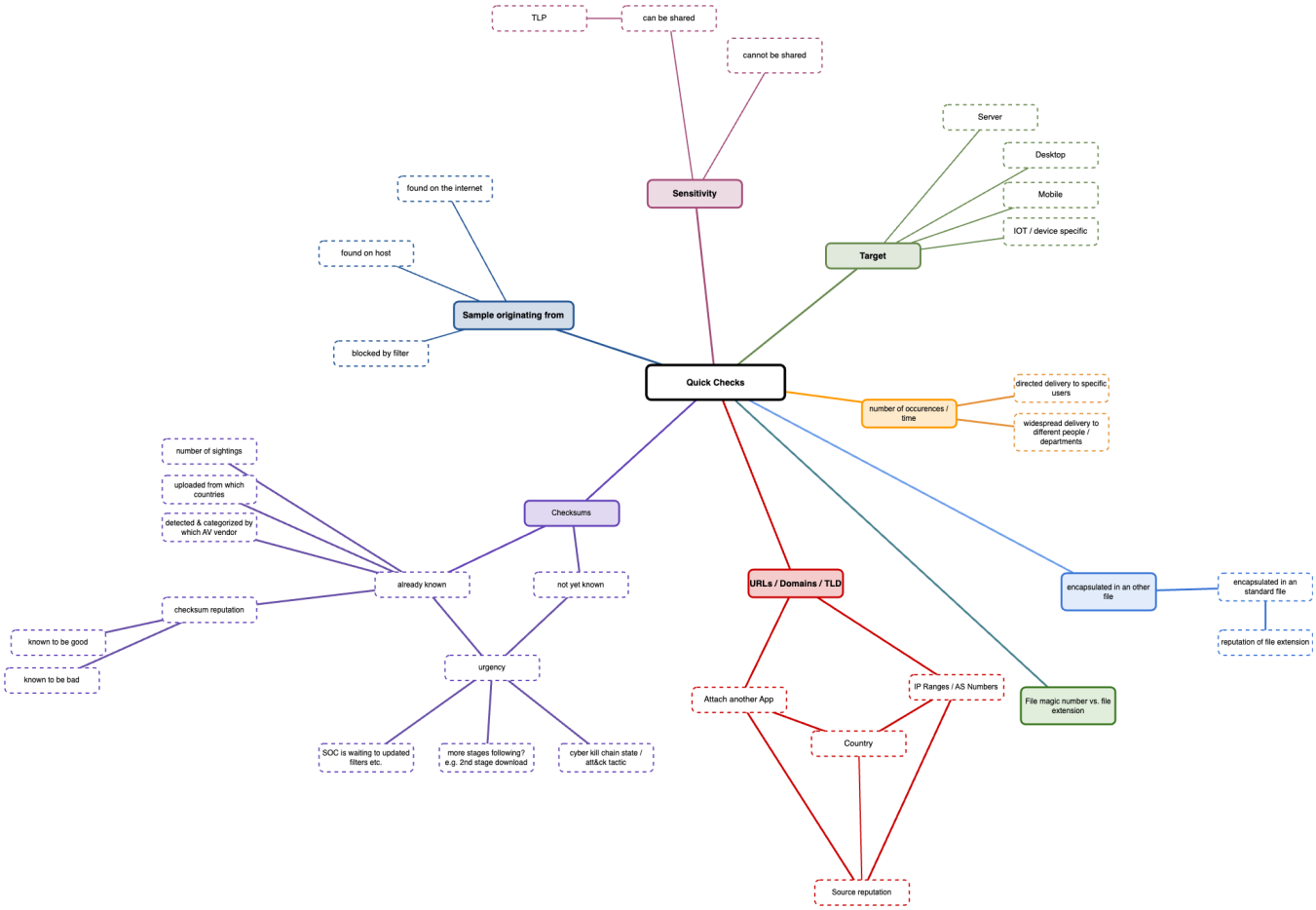
- Static analysis: initially conducted analysis that involves examining basic properties of the analyzed malware sample to identify how complex the analysis process is expected to be and outline the strategy of further, in-depth analysis
- Behavior analysis: conducted to watch what malicious activities the analyzed sample conducts at runtime
- Code analysis: the most time-consuming type of analysis used to obtain an accurate description of a malware's code, typically by examining it with reverse engineering instruments.

For maturing a malware analysis team, its essential to learn how to conduct all three types of analysis mentioned above, as sometimes a combination is needed to adequately answer the questions.

## Static Analysis

Static analysis (sometimes referred to as triage analysis) typically starts with identifying the type of the malicious file by examining its first bytes. These bytes typically contain magic sequences that are unique to each file format, such as 'MZ' (0x4D 0x5A) for Windows executable files. It is always worth checking whether the identified file type matches the file extension, as attackers may use techniques such as right-to-left override masquerading (i.e. inserting a special character that reverses symbols in a string to, for example, make the file name "xcod.scr" display as "rcs.docx") to tamper with file extensions.

After determining the file format (extension), it is common to use specific tools intended to display metadata of that file. Files of different formats store various information in their headers, and depending on the format, this information may or may not be useful to an analyst. One particular kind of files that stores many details important for malware analysis are executable files – and as such, when dealing with analysis of these files, it is worth using executable file header parsers that can extract useful information out of malicious files.




### Evaluating sensitivity

At every stage of the static analysis, it is also important to note whether the malware contains any sensitive information. For example, this may happen if the analyzed sample drops a lure document that can be used to identify the name of the targeted organization. Information about such samples should be assigned an appropriate TLP level, and it should be discussed whether it can be shared with partners. Additionally, sensitive samples should be handled extremely carefully during further analysis. They cannot be uploaded to public platforms like multi-scanners and cloud sandboxes, as it would allow others outside of the constituency to download the malware.

As a result of completing static analysis, it will be possible to draw conclusions about the analyzed sample that will help decide how to proceed with its further analysis. Specifically, while assessing triage results, it can be useful to discuss how complex the malware sample is and how much time it can take to analyze it. If understanding how a sample works will likely take an unreasonable amount of time, it may be better to outsource its analysis to an external malware analysis team. Otherwise, if it is possible to proceed with further analysis of the sample, it is crucial to discuss further analysis strategy. For example, if the sample is developed using a scripting language that is easy to analyze, it may be possible to find out its capabilities by reading its code. However, if the code of the sample is difficult to understand, it will be required to analyze the sample using behavior analysis, which will be discussed next.

### Identifying known or similar samples

Malware samples are often compared via checksum (e.g. MD5, SHA1, SHA256), as filenames can be changed easily and is not accurately identifiable for a malware sample. The checksum only changes when the content of the file changes and is very handy for quick cross-checks which you can compare and share with others

- The checksum is not known - this could be a new sample or you could be the first to detect it
- The checksum is already known and listed on well known reputation lists - others have seen the exact same sample before you
    - Number of sightings can give a hint this sample is part of a widespread delivery and many other organizations got it as well
    - Uploaded from which country: platforms like VirusTotal register from which country a sample was uploaded and can give you a good overview from where the attack might have started
    - Detected and categorized by which AV vendor: every AV vendor uses his own detection and naming scheme. Some malware gets tagged as 'generic malicious' while others classify it with a specific malware family name. There might be some AV Vendors you trust more than others regarding their detection and naming results.
    - Reputation: checksums can be generated for legitimate files, not only malware. Also legitimate files like system libraries can be checked and shared to be sure the version installed on your system is the one the vendor has distribution and not a modified version. There are 'known to be good' and 'known to be bad' reputation lists available, which can be checked for the known state of a file.

### Analyzing sections

When looking at headers of an executable file, one type of objects to be studied in the first place are file sections. These sections typically represent either code or data and have attributes specifying whether they are readable, writable, or executable. When examining sections, particular emphasize should be on:

- Size of the code section: the larger it is, the more code the malware contains, and the more time-consuming it will likely be to examine the file via code analysis
- Size of sections storing data: if there is a section reaching hundreds of kilobytes in size, it may be possible that the malware contains an embedded payload that needs to be extracted and subsequently analyzed
- Contents of sections storing data: they can contain strings revealing indicators of compromise such as file paths or address of malicious servers
- Entropy of sections: if the code section of the analyzed file has a high entropy, it usually means that the code of the file is obfuscated. Additionally, if a section storing data has a high entropy, the data in it can be encrypted – for example, in the case of encrypted strings being present. An analyst should keep in mind that decrypting this data can reveal answers to many questions asked about the analyzed file.

### Examining function imports and resources

Executable file headers can also contain information about capabilities of the operating system used by the malware. For example, Portable Executable (PE) files used on Windows contain an import directory, which stores names of the operating system API functions used by the malware. Based on names of these API functions, it is often possible to guess what the type of the analyzed malware is. For instance, malware using functions performing file reading (e.g. CreateFile, ReadFile and CloseFile) and encryption (e.g. CryptAcquireContext and CryptEncrypt) can likely turn out to be a ransomware, while malware using keyboard interaction functions (such as GetAsyncKeyState or SetWindowsHookEx) can implement keystroke logging functionalities.

It is noteworthy that as analysis of an executable file imports provides an easy way for analysts to identify the behavior of a malware sample. That is why, many threat actors nowadays disguise names of used API functions by using techniques such as Dynamic API Resolution. In case such techniques are used, the import table of an analyzed sample may be empty or filled with names of functions that the sample will not use during runtime. From the perspective of an analyst, this means that examining such a malicious file will be a less straightforward task.

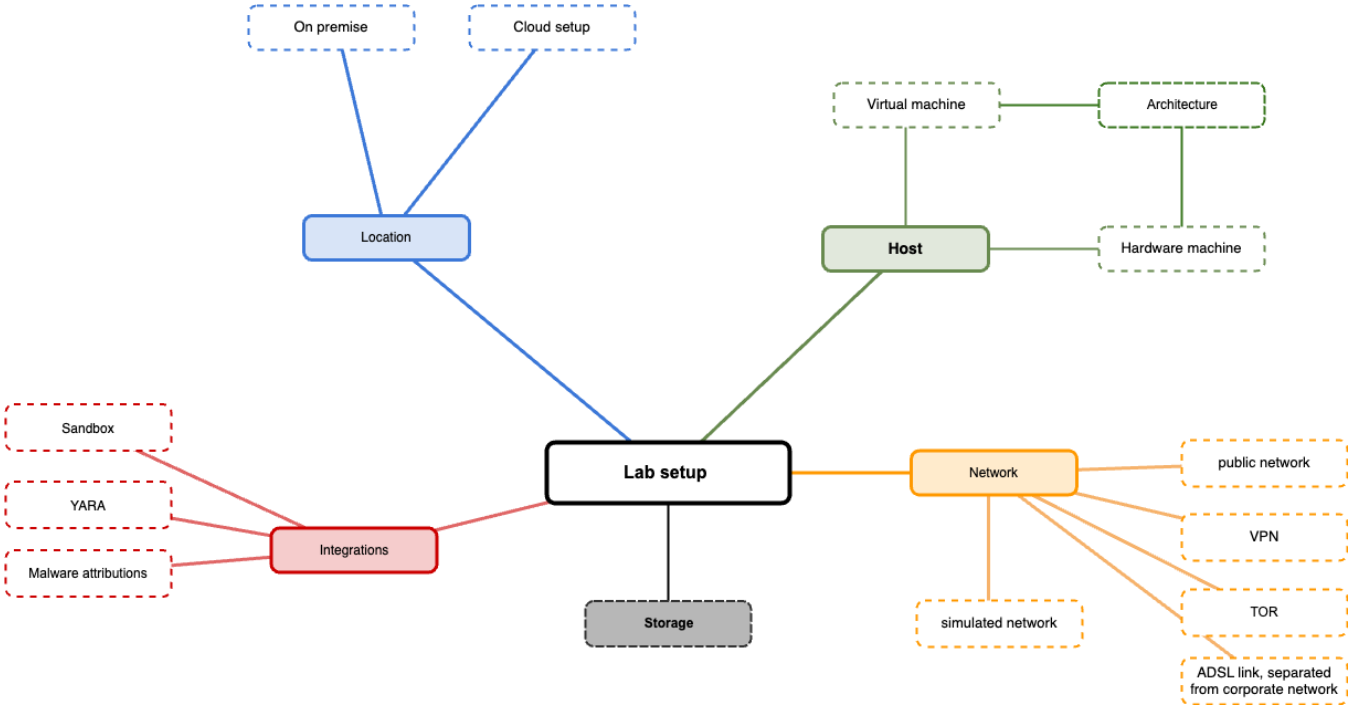
Resources of an executable file can be another trove of information useful for analysis. Normally, resources are used by benign executables to store data such as icons or window outlays, however, in the case of malware resources can contain additional (possibly encrypted payloads), decoy documents or configuration parameters. As such, examining resources of an executable files can reveal valuable information about its behavior.

### Estimating time for analysis

In addition to examining metadata from headers while performing triage, it is also useful to perform basic checks on the code of the malware using tools such as disassemblers and decompilers. During a quick analysis, it is possible to check the programming language used to develop the malware to determine how difficult it will be to analyze the code. Usually code produced from compiled languages such as C, C++, Golang or Rust is less readable and thus more difficult to analyze, while code written with interpreted languages like Java and Python is well-readable, which makes it simple to understand. It is additionally a good idea to check whether the malicious code is obfuscated as analysis of such code typically requires more time.

## Behavior analysis

As mentioned earlier, behavior analysis involves launching the malware sample being analyzed and examining activities performed by it, such as reading or writing files or performing network communications. The setup of your malware analysis lab can have different setups and different components. Depend ing on the questions you're trying to answer you might not need or use all of this components. Depending on the maturity of the malware analysis capabilities of your team or your available budget, the lab might also differ from other team's lab setup. Important to know is, there is not one solution that fits all needs and there's nothing like a 'perfect setup'. Your setup has to cover your needs and your capabilities. The most fancy and most expensive setup will not help if you don't have the time to use it to the full extent.




## Types of behavior analysis environments

While performing behavior analysis, it is paramount that the malware analyzed is launched in a secure and isolated environment. Executing malware on computers used for daily tasks can lead to detrimental damage - sensitive data may get stolen or wiped, or the infection may spread to other machines in the network. That is why, before doing behavior analysis for the first time, it is important to configure a proper analysis environment that would allow to run malware without causing damage. Usually, such analysis environments are created in one of the following ways:

1. Using virtual machines
2. Using physical machines with dedicated hardware
3. Using dedicated sandbox-solutions using a combination of the former

## Virtualized environments

Using virtual machines is usually the quickest and cheapest way to configure a behavior analysis environment. This method typically involves installing hypervisor software on a physical machine and then using it to configure one or multiple virtual environments. Apart from being cost-effective, using this method also is beneficial because it allows to run multiple virtual machines at the same time. As such, this makes it possible to, for instance, configure a single physical machine for analyzing Windows, Linux and Android malware at the same time. In addition, virtual machines are very easy to revert to an uninfected state after finishing the analysis, namely by using the snapshot feature present in modern hypervisor software.

However, using virtual machines for behavior analysis has a significant drawback – the malware analyzed can be coded to detect virtualized environments and refuse to launch if such an environment is detected. This is known as anti-analysis techniques. There are many methods that can be used to detect if the malware is run in a virtual analysis environment, and it can be challenging to make your environment resilient to these techniques.

## Physical environments

An alternative to using virtual machines for behavior analysis is to dedicate separate physical machines for running malware on it. The most significant benefit of this approach is that it is much more difficult for threat actors to detect a physical analysis environment than a virtual one. However, this approach is more time-consuming and potentially more costly, as opposed to virtual machines, the process of reverting a physical computer to an non-infected (clean) state requires much more effort.

Over the course of developing malware analysis processes in a CSIRT, it is beneficial to configure both virtual and physical labs for behavior analysis. In order for malware analysis to be as efficient as possible, virtual environments can be used to examine how a malware sample behaves over a short period of time (such as a few minutes or hours), while physical environments are more suitable for long-term analysis (days or even weeks).

## Anti-analysis techniques

Regardless of how a malware analysis lab is configured, it is important to be mindful of techniques used by threat actors to detect analysis environments. To make sure that an analysis environment is as resilient to these techniques as possible, it is important that it resembles an actually used endpoint or server as much as possible – the more the resemblance is, the higher is the chance for the analysis to be successful.

While configuring analysis environments, it is also worth considering the location where they will be hosted. Depending on resources and budget available, it may not be possible to host all analysis machines in a dedicated environment. That is why it may be at first more beneficial to deploy analysis environments in the cloud, and migrate them to dedicated machines over the course of malware analysis processes maturing.

### Monitoring functionalities

In order for analysis environments to be fully efficient, it is also crucial to equip them with software allowing to monitor malicious behavior. Normally, a full-fledged malware analysis lab will have the following capabilities:

- Tracking malicious processes: once a malware is started, it can use various processes to perform its activities. Malware samples can either create new processes (for example, the powershell.exe command line interpreter to execute shell commands) or tamper with existing ones (for example, to inject malicious code into a trusted processes). As such, it may not be possible to get a complete picture of how a malware operates without knowing what processes are involved in malicious activities.
- Logging system activities: just like legitimate applications, malware commonly performs various operations with the operating system. For instance, it can read and write files stored on disk, perform operations with registry or interact with installed services. It is important for malware analysts to pinpoint such activities, as information about them can be used to identify indicators of compromise (such as file paths or registry keys) for the examined malware, as well as better understand what the malware is used for.
- Ability to capture network traffic: over the course of operating many malware samples send data to attacker-controlled servers, and addresses of these servers also commonly serve as indicators of compromise. Apart from identifying addresses of these servers, it is also beneficial to examine traffic exchanged with them. For instance, traffic data examination can help identify commands implemented inside the malware, data sent to it by attackers as well as additional modules deployed by attackers.

It should also be taken into account that modern malware typically performs command and control server communications using secure protocols such as HTTPS. That is why while setting up capturing of network traffic, it is also important to deploy tools allowing to decrypt captured traffic, such as tools performing man-in-the-middle attacks over encrypted traffic.

### Network communication capturing

Another important aspect of setting up a malware analysis environment is configuring its network configuration. As mentioned earlier, a malware analysis lab cannot be connected to an enterprise network, as in this case it becomes possible for the deployed malware to perform a scan of this network and possibly propagate to other machines inside it. In addition, it is also of benefit to conceal the IP address of the analysis machine to make sure that attackers become unable to remotely attack it.

When it comes to securely configuring network communications inside a malware analysis lab, there are multiple strategies to choose from. Below are the most popular of them:

- Using a simulated network: it is possible to configure software (a popular used example is inetsim) to simulate various network services such as DNS or HTTP. With this setup, no connection will reach the external network, and as such malware operators will be unable to find out that their implants are being analyzed. However, due to malicious server responses being simulated, the malware may not work correctly (this may happen if these responses do not have the format expected by the malware) and it will be impossible to retrieve any second-stage payloads possibly deployed by attackers.
- Using a public network: it is possible to use a distinct internet link to connect analysis machines to the Internet and at the same time entirely separate it from the corporate network. Due to the infected machine being connected to a public network, an attacker performing checks on its IP address will not be able to associate the machine with any organization.
- Using a VPN: a dedicated VPN server can be rented to route all traffic coming from analysis machines. This method is quite cost-effective, as nowadays VPS servers can be rented cheaply. In addition, it is possible to rent multiple VPN servers located in different countries, which can come useful when analyzing malware that uses geofencing (infecting machines located in a specific country, region or organization).
- The TOR network can be used as a free alternative to renting a VPN server. However, in this case the connection speed will be much slower, which can be a disadvantage when downloading large payloads from attacker-controlled servers.

Finally, it is possible to further enrich behavior analysis by integrating analysis labs with various systems and data sources. Examples of these integrations are as follows:

- It can be useful to scan random access memory of analysis machines with YARA rule collections. For example, a packed malware file may not be detected by a well-known YARA rule, but the same rule could be able to detect its unpacked version residing in memory. The same can be done with Sigma rules to better detect malicious events occurring inside an analysis lab.
- MISP instances can be used to conveniently export indicators of compromise from analysis environments.
- Malware collection systems can be used to lookup similar malware to the one being analyzed and potentially associate it with an already known actor.

Overall, the process of setting up a full-fledged behavior analysis environment can be characterized as quite complex. Building fully functional malware analysis labs usually takes lots of time, thus it is most beneficial for maturing malware analysis teams to create primitive versions of such labs and then improve them over time as skill-sets improve. For additional information on creating malware analysis environments, it is possible to refer to the “Malware Lab Examples” Annex to this Framework, which provides detailed technical descriptions of both simple and complex analysis labs.

## Code analysis

Code analysis is a method that involves retrieving the code of a malware sample and then studying it to determine capabilities of the analyzed malware. It is the most complex and time-consuming method of malware analysis, and mastering it usually requires significant effort. For these reasons, it is usually used only if static analysis and behavior analysis methods fail to achieve desired results. This may be case is the malware analyzed is sophisticated (such as a multimodular backdoor) or uses complex anti-analysis techniques.

### Working with packed malware

The exact strategy of how code analysis is performed depends on specific details obtained during triage. Code analysis usually starts with establishing whether the analyzed malware is packed – this is often done by calculating entropy of data stored inside the sample. In case of a packed malware, it is further required to establish the name of the packer used. This can be usually done by checking the sample with a collection of YARA rules identifying popular packers.

If a packer has been successfully identified, then it is possible to consult further guidance on how to handle it. There are ready-made tools available for commonly used packers and using them often saves a lot of analysis time.

However, many malware developers nowadays use their own, custom packers, and there will likely be no available unpacking guidance. In such situations it will be likely necessary to use debugger software to dump the unpacked executable from memory. As an alternative, it is possible (but more time-consuming) to reverse engineer the packer code, determine how the packed payload is stored in the analyzed executable and then perform its extraction.

### Establishing tools used for further analysis

After obtaining an unpacked version of the analyzed malware, the next step in code analysis is usually to download and set up tools enabling to read the code of the malware sample. In the case of malware written in languages such as PowerShell or JavaScript, such tools will not be required as it will be sufficient to use a text editor to read the code. For interpreted languages like Java or C#, it will be necessary to download decompilers for these languages (e.g. jd-gui or ILSpy) to be able to read the code. Finally, the code of compiled languages such as C++ or Rust can be viewed with general-purpose disassemblers and decompilers (e.g. IDA, Ghidra, Binary Ninja).

### Dealing with obfuscation and conducting further analysis

In turn, reading the code will reveal whether it is obfuscated. Examples of obfuscation can be code filled with redundant operations or encryption of strings. The workflow of removing code obfuscation is usually similar to the one used when removing packers: it includes attempting to determine the name of the obfuscator used, and then using an already available deobfuscation tool or developing one from scratch.

For a sample clear of obfuscation, it becomes possible to conveniently study its code, while referring to necessary documentation (such as the one for operating system API functions), and document actions conducted by the analyzed malware.

## Memory Analysis

Every software that runs on a host, leaves some traces in the memory of this host. Memory blocks are read, added, removed, overwritten etc. Also there are some malware families which install them completely into the host's memory when started and delete itself from the storage device to be 'invisible' to anti virus scans of this storage device. Like this, a lot of information about a malware can be found in memory (e.g. decrypted information, encryption keys). Memory analysis is one of the most complex and difficult analysis steps, but can be very rewarding in terms of understanding the inner workings of malware. When we get a memory image from an infected system with a malware running, we'll find all interesting details in the memory, such as command & control endpoints, encryption keys and so forth.

Memory analysis is very resource- and knowledge intensive and usually performed by malware analysts. As a CSIRT Team you might want to have the knowledge to take a memory snapshot of a machine, but analyzing it in full detail and look for malware traces might be a task for a later stage, when the team has gained more experience.

# Phase 5 – Creating Reporting Guidelines

After finishing the analysis process, it is important to document the findings obtained while researching analyzed samples. To do that, it is necessary to establish guidelines determining which information each report should include, as well as create a template for reports. Specific details about the report contents usually depend on its the target audience.

A good starting point for a report is to provide an executive summary of the researched malware. It is meant to be understood by a broad range of readers, from business managers to technical analysts and can include the following information:

- How the analyzed sample was discovered
- The type of malware with a brief description of its functionalities
- What makes the analyzed samples interesting and in what way it can be dangerous to constituent organizations
- Proposals on how the sample may be disarmed or other means in which renders it ineffective

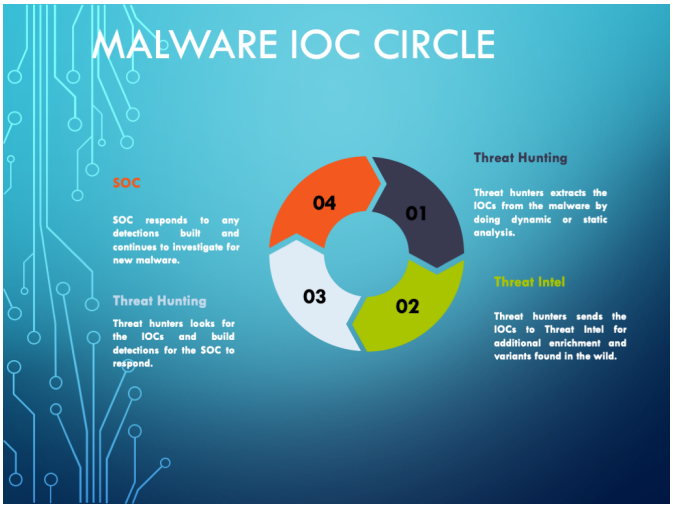
The introduction can be further followed by a description of analyzed malware sample, such as including:

- Sample filenames and hashes (e.g. MD5, SHA1, SHA256)
- Sample type and classification (see Appendix A - Malware Classes)
- Description of key information about the sample, such as:
    - How it is deployed to machines
    - How it achieves persistence
    - What anti-analysis techniques it uses and what tools or methods were used to tackle them
    - What its capabilities are

Technical data about discovered samples can be, if possible, further followed by a description of the threat actor using it. Such information can be useful for other analysts who have much less knowledge about the relevant actor. It is additionally beneficial to supplement the description of the threat actor with links to external resources such as other reports or blog posts.

It is also helpful - if this is a defined goal for the analysis - to provide recommendations on what measures to take to better protect against the analyzed malware, and include a summary of observed tactics, techniques and procedures (TTPs) by providing MITRE ATT&CK mapping.

A malware analysis report should additionally be complemented with a list of indicators of compromise (IoCs) that can be used to detect the described malicious activities. This list should be composed very carefully and it should be checked for potential false positives, such as legitimate utilities (e.g. living off the land binaries). The list of IoCs should also be as comprehensive as possible. As evidenced by the Malware IoC Circle below, multiple teams use IoCs for different purposes. As such, the more accurate and contextual the IoCs are, the more efficiently these teams will be able to work.




An example of a report following the guidelines described above is provided below in Appendix D - Supporting Documents.

## Establishing Information Sharing Processes

Malware analysis reports pose interest to a wide range of people working in cybersecurity, and sharing these reports often helps others to better protect their environments. That is why it is also important to discuss which external entities may be interested in knowing about the conducted analysis. It is then worthwhile to set up sharing agreements with these entities. Such agreements are usually most efficient when data sharing is handled automatically, for example through automation tools such as MISP.

However, some details about the analyzed malware can be extremely sensitive, and thus it may not be always possible to externally share all information mentioned in reports. Because of that, each report should be assigned a TLP classification. To do that, it is important to develop internal standards detailing how reports should be classified.

It is also beneficial to create guidelines outlining how information provided in reports should be anonymized. This will in turn allow to share sensitive reports with peers without risk of revealing confidential data.

- As an example, Office files with malicious macros in it, can still be valid office documents and contain sensitive information about your organization. As soon as you share such a file with other peers or public platforms like VirusTotal, the information in this file can be considered public. Some information should not be shared with others and you might not be allowed to upload such a file to any public platform
- If the file does not contain sensitive information, you can share it with others. However, sharing of the report should be reflected by its TLP classification, e.g. a TLP: Amber+Strict should only be distributed to constituence and selected recipients, not the wider community.

In order to effectively share information it is recommended to identify potential stakeholders and expected outcome. For example, a SOC would be interested in knowing the specific IOCs of the malware and observables that may support a triage process. A vulnerability manager would be interested in knowing the weaknesses the malware exploits and potential mitigation controls. A detection engineer would be interested in knowing the malware's process lineage and artifacts created. Altogether, these stakeholders can be internal as exemplified, or external via various trust groups and ISACs/ISAOs.

# Phase 6 – Lesson Learned and Reviewing Performance Results

To make sure that all malware analysis procedures are conducted efficiently, it is important to regularly (for example, every quarter) to assess activities conducted by a malware analysis team. During these assessments, it is helpful to review reports produced over a certain period of time and identify important trends out of available information. One example of  this can be a stable increase in the use of a specific infection technique. Such trends can highlight what defense measures organizations should better invest in to become more protected. Moreover, an increase in observed malware deployed through phishing can mean that it is beneficial for relevant organizations to deploy more advanced email security solutions, or further tighten its configuration to reduce exposure.

It can also be helpful to evaluate the overall performance of the malware analysis team by discussing its successes and shortcomings. This will make it possible to conclude what particular steps (such as buying equipment, taking malware analysis courses or recruiting more analysts) can be taken to make malware analysis workflows more efficient in the future.

One approach to evaluate the success of a malware analysis workflow, is to synthesize all analysis/reports produced in a given period and identify gaps (technical, procedural or skill-sets) that can be improved until next iteration. Here is an example based on the questions taken from phase 3 - Defining Malware Analysis Goals.

| Questions | Samples (success) | | | | Potential improvement area(s) |
|---|---|---|---|---|---|
| | | | | | |

| | Sample1 | Sample2 | Sample3 | Sample4 | |
|---|---|---|---|---|---|
| **Succeeded in identifying targeted platforms and systems?** | ✅ | ❌ | ⚠️ | ✅ | ▪ Unable to detonate samples XYZ in analysis environment due to lack of platform support. |
| **Succeeded in determining data interaction during execution?** | ✅ | ❌ | ✅ | ✅ | ▪ Unable to fully execute samples XYZ due to sandbox evasion techniques. |
| **Succeeded in identifying network behavior and C2 channels?** | ✅ | ⚠️ | ⚠️ | ⚠️ | • Unable to dissect samples XYZ due to unknown obfuscation techniques. |
| **Succeeded in acquiring the full malware payload chain and -stages?** | ❌ | ❌ | ✅ | ✅ | • Unable to fetch payload samples XYZ due to victim profiling or geofencing by threat actor. |

# Appendix

## A - Malware Classes

| Class | Description |
|---|---|
| **Virus** | Code that propagates (replicates) across systems with user intervention |
| **Worm** | Code that self-propagates/replicates across systems without requiring user intervention |
| **Bot** | Automated process that interacts with other network services |
| **Trojan** | Malware that is often disguised as legitimate software |
| **Ransomware** | Malware that holds the victim's data hostage by cryptography or other means |
| **Rootkit** | Masks its existence or the existence of other software |
| **Backdoor** | Enables a remote attacker to have access to or send commands to a compromised computer |
| **RAT** | Remote Access Trojan, similar to a backdoor |
| **Infostealer** | Steals victims information, passwords, or other personal data |
| **HackTool** | Admin tools or programs that may be used by hackers to attack computer systems and networks. These programs are not generally malicious |
| **Hoax** | Program may deliver a false warning about a computer virus or install a fake AV |
| **Dropper /Downloader** | Designed to "install" or download some sort of malware |
| **Adware** | Automatically renders advertisements in order to generate revenue for its author. |
| **PUP/PUA** | Potentially Unwanted Program, sometimes added to a system without the user's knowledge or approval |

## B - Tooling

Malware Tools Overview

## C - Training

Training Materials

## D - Supporting Documents

### D.1 Example Report


## E - Glossary

### Version

V2.0 - October 2024

V1.0 - April 2022

### Authors / Contributors

Olivier Caleff, James Potter, Raja Jasper, Andreas Mühlemann, Georgy Kucherin Andreas Bråthen

### Reviewers

v2.0

Sarah Plocher Andreas Bråthen

v1.0

Per Morten Sandstad, Hiroki Kuzuno