2ND EDITION

Mastering Malware Analysis

A malware analyst's practical guide to combating malicious software, APT, cybercrime, and IoT attacks

ALEXEY KLEYMENOV | AMR THABET

Mastering Malware Analysis

Second Edition

A malware analyst's practical guide to combating malicious software, APT, cybercrime, and IoT attacks

Alexey Kleymenov

Amr Thabet

BIRMINGHAM—MUMBAI

Mastering Malware Analysis

Second Edition

Group Product Manager: Mohd Riyan Khan
Publishing Product Manager: Mohd Riyan Khan
Content Development Editor: Adrija Mitra
Technical Editor: Nithik Cheruvakodan
Copy Editor: Safis Editing
Project Coordinator: Ashwin Kharwa
Proofreader: Safis Editing
Indexer: Manju Arasan
Production Designer: Ponraj Dhandapani
Marketing Coordinator: Ankita Bhonsle

First published: June 2019
Second edition: September 2022

Production reference: 1010922

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

978-1-80324-024-4

www.packt.com

1

Cybercrime, APT Attacks, and Research Strategies

Our modern world relies more and more on IT systems of various kinds. Being able to control them, as well as the information they may contain and process, is a strong power that attracts various types of criminals.

In this chapter, we are going to discuss the evolution of the cybercrime landscape up until now and the role of malware analysis in fighting it. Then we will dive into various types of attacks and associated malware to get an idea of possible attack stages and the logic behind them. In addition, we will learn different research strategies and approaches universal to all platforms that help malware analysts do their job, from collecting relevant telemetry and samples to performing **Reverse Engineering** (**RE**) tasks and answering specific questions.

In this chapter, the following topics will be covered:

- Why malware analysis?
- Exploring types of malware
- The MITRE ATT&CK framework explained
- APT and zero-day attacks and fileless malware
- Choosing your analysis strategy
- Setting up the environment

Why malware analysis?

Cyberattacks are undoubtedly on the rise, targeting governments, the military, and the public and private sectors. The actors behind them may have numerous motivations, such as exfiltrating valuable information as part of espionage campaigns, gaining money by various means such as demanding ransoms, or damaging assets and reputations as a form of sabotage.

The growing dependency on digital systems, which accelerated immensely during the COVID-19 pandemic, also led to a massive increase in malware and particularly ransomware-related incidents in recent years.

With adversaries becoming more and more sophisticated and carrying out increasingly advanced malware attacks, being able to quickly detect and respond to such intrusions is critical for cyber security professionals, and the knowledge, skills, and tools required to analyze malicious software are essential for the efficient performance of such tasks.

In this section, we will discuss your potential impact as a malware analyst in fighting cybercrime by responding to such attacks, hunting for new threats, creating detections, or producing threat intelligence information to get your and other organizations better prepared for the upcoming threats.

Malware analysis in collecting threat intelligence

Threat intelligence (aka cyber threat intelligence, commonly abbreviated as threat intel or CTI) is information, usually in the form of **Indicators of Compromise** (**IoCs**), that the cybersecurity community uses to identify and match threats. It serves multiple purposes, including attack detection and prevention, as well as attribution, allowing researchers to join up the dots and identify current and future threats that might originate from the same attacker. Examples of IoCs include sample hashes (most commonly MD5, SHA-1, and SHA-256) and network artifacts (primarily, domains, IP addresses, and URLs). There are multiple ways in which IoCs are exchanged within the community, including dedicated sharing programs and publications. **Indicators of Attack** (**IoAs**) are also commonly used to describe anomalous behavior very likely associated with malicious activity. A good example is a machine in a **demilitarized zone** (**DMZ**) that suddenly starts communicating with multiple internal hosts. As we can see, unlike raw IoCs that require additional context, IOAs more often reveal the intention behind the attack and can therefore be easily mapped to particular **tactics, techniques, and procedures** (**TTPs**).

Malware analysis provides a very accurate and comprehensive list of IoCs compared to other methods such as log analysis or digital forensics. Some of these IoCs may be very difficult to identify using other digital investigation or forensics methods. For example, they might include a specific page, post, or an account on a legitimate website, such as Twitter, Dropbox, or others. Tracking down these IoCs can eventually help in taking down the corresponding malicious campaign faster.

Malware analysis also adds invaluable context as to what each IoC represents and what it means if it is detected within an organization. Understanding this context may help in prioritizing the corresponding events.

Malware analysis in incident response

Once an attack is detected within an organization, an incident response process is kicked off. It starts with containment of the infected machines and a forensic investigation aimed at understanding the cause and impact of malicious activities to follow the right remediation and prevention strategy.

When malware is identified, the malware analysis process starts. First, it generally involves finding all the IoCs involved, which can help discover other infected machines or compromised assets and find any other related malicious samples. Second, malware analysis helps in understanding the capabilities of the payload. Does the malware spread across the network? Does it steal credentials and other sensitive information or include an exploit for an unpatched vulnerability? All this information helps evaluate the impact of the attack more precisely and find appropriate solutions to prevent it from happening in the future.

Apart from that, malware analysis may help in decrypting and understanding the network communications that have occurred between the attacker and the malware on the infected machine. Some enterprise network security products, such as **Network Detection Responses** (**NDRs**), can record suspicious network traffic for later investigation. Decrypting this communication may allow the malware analysis and incident response teams to understand the attacker's motivations and more precisely identify the compromised assets and stolen data.

So, as you see, malware analysis plays an important role in responding to cyberattacks. It can involve a separate team within the organization or an individual within the incident response team equipped with the relevant malware analysis skills.

Malware analysis in threat hunting

In contrast to incident response, threat hunting involves an active search for IOAs. It can be more proactive, taking place before the security alert has been triggered, or reactive, addressing an existing concern. Understanding possible attackers' tactics and techniques is crucial in this case as it allows cybersecurity professionals to get a higher-level view and navigate the potential attack surface more efficiently. A great advancement in this area was the creation of the MITRE ATT&CK framework, which we are going to cover in greater detail later.

Malware analysis knowledge helps cybersecurity engineers to be more professional threat hunters who understand the attackers' techniques and tactics on a deeper level and who are fully aware of the context. In particular, it helps understand how exactly the attacks may be implemented, for example, how the malware may communicate with the attacker/**Command and Control** (**C&C**) server, disguise itself to bypass defenses, steal credentials and other sensitive information, escalate privileges, and so on, which will guide the threat-hunting process. Armed with this knowledge, you will better understand how to hunt efficiently for these techniques in the logs or in the systems' volatile and non-volatile artifacts.

Malware analysis in creating detections

Multiple companies across the world develop and distribute cybersecurity systems to protect their customers against all types of threats. There are multiple approaches to detecting malicious activity at different stages of the attack, for example, monitoring network traffic, exploring system logs and registry entries, or checking files both statically and during the execution. In many cases, it involves some sort of rules or signatures to be developed to distinguish malicious patterns from benign ones. Malware analysis is irreplaceable in this case as it allows security professionals to identify such patterns and create robust rules that don't generate false positives.

In the next section, we will discuss how malware can be classified depending on its functionality.

Exploring types of malware

In this section, we are going to discuss why malware exists in general, what makes it different from other computer programs, and what different varieties we can encounter in the wild.

A short history of malware development

Before the rise of personal computers, only a very limited number of software developers existed. Their goal was to make maximum use of the hardware available at that time to make people's lives better, whether it was software for accounting, sending a man into space, or gaming. Rapidly developing networking connected multiple machines to each other and enabled machines and people to communicate over long distances. Around the same time, with the further spread of computers, making them more affordable to the general public, the first hacking communities started evolving around the globe. However, it was the academic sector where one of the most infamous incidents of malware with significant impact emerged – the Morris worm. It was capable of propagating via networks to other machines exploiting several vulnerabilities, mainly in the `sendmail` and `fingerd` software. However, the worm wasn't checking whether the targeted machine was already infected or not and this way spawned multiple copies of itself on each machine, quickly consuming all the victim's system resources and making them unusable. Created just for the sake of pure interest, it showed the world what consequences several lines of code could bring and led to the first-ever conviction for malware development. Many other types of malware began to emerge after this. The main goal of the authors at that time was to demonstrate their skills within the community.

Later, the focus slowly started shifting toward making money. Programming became more and more popular, being taught at schools and universities, and the creation of new high-level programming languages made it easier for less experienced people to start writing their own code, including malicious code. Finally, professional cybercrime gangs began to emerge with a clear separation of responsibilities, making malware development a very lucrative organized illegal activity. These groups utilized all possible ways of money laundering available including, at first, money mules and later switching to cryptocurrencies to avoid tracing and subsequent arrests. These groups are generally called financially motivated actors.

In the last few years, the focus of financially motivated groups gradually shifted from attacking the consumers to attacking big organizations and making big money in a single place. The most common example is the use of ransomware to encrypt victims' files before demanding a ransom to restore access. In many cases, a double-extortion scheme is used, where the criminals also threaten to release sensitive materials to the public.

Governments also started looking for possibilities to use malware for cyber espionage and sabotage purposes. It was the Stuxnet attack that really brought the public's attention to its existence and its initial devastating capabilities. The malware-developing groups involved in this process are generally state-sponsored. Apart from this, there are companies that openly develop and sell advanced surveillance malware to governments. Examples include NSO Group, selling the Pegasus threat; Hacking Team with Da Vinci and Galileo platforms; and Lench IT Solutions (part of Gamma Group), selling FinFisher spyware.

It is no surprise that malware follows the most commonly used platforms to have the best coverage possible. Therefore, it is Windows-based malware that is still most prevalent for workstations. In the mobile market, Android remains the market leader and thus is targeted by the biggest number of malware families. Finally, **Internet of Things** (**IoT**) malware is also on the rise, targeting historically less-protected smart devices (mostly Linux-based). And of course, it doesn't mean that if a platform is less common it is more secure and malware-free.

Malware categories

Malware categories are generally defined by either an impact or a propagation method. Different antivirus companies may use slightly different logic in defining or naming them. Here are some of the most common examples:

- **Trojan:** The most universal malware category, simply defined by its performing of malicious activities in the unaware user's environment, named for the legendary Trojan Horse used to conquer the city of Troy:
    - **Downloader:** The main goal here is to download and somehow execute the external payload (either explicitly or by adding it to autorun).
    - **Dropper:** Here, additional payloads are not downloaded but extracted from the Trojan's body.
    - **Backdoor**, as known as **Remote Access Trojan** (**RAT**): In this case, the malware may receive remote commands to perform a range of actions.
    - **Ransomware:** Here, attackers prevent users from performing their daily activities and demand a ransom to restore them. This can be done by various means, usually by either locking the whole system or locking access to particular files within it. Another common scenario when targeting individuals is accusing them of some criminal deed and demanding a "fine" to be paid, threatening escalation or public announcement in the case of non-compliance.

  - **Infostealer, aka Password Stealer** (**PWS**): The main goal here is to steal sensitive information, such as saved credentials of any kind (from other machines, financial organizations, social networks, email and instant messenger accounts, videogames, and so on).
  - **Spyware**: While spyware's purpose is quite similar to infostealer's, this category is broader and may also include video and audio recording capabilities or tracking the victim's location with GPS.
  - **Banker**: This category may commonly fall into the infostealer one but has a narrower purpose and bigger scope of potential functionality. Here, malware may be strongly focused on gaining access to money, so it can also support intercepting one-time tokens sent by the bank as part of **two-factor authentication** (**2FA**), modifying financial information to redirect payments, or injecting scripts to intercept entered banking credentials.
  - **DoS**: The main goal here is **Denial of Service** (**DoS**), making the target system or service unusable; it is commonly used for sabotage, hacktivism, or vandalism purposes.
  - **Wiper**: Here, malware is used to delete information that is either sensitive or critical to the system's operation, making it another tool for a DoS attack.
  - **DDoS**: In this case, a **Distributed Denial of Service** (**DDoS**) attack is launched, where multiple bots attack the victim via the network.
  - **Spammer, aka spambo:** This threat can send spam on behalf of the victim.
  - **Clicker**: Here, attackers may simulate real user clicks to get money from advertisements, search engine poisoning, or promoting fake accounts.
  - **Miner**: In this case, the unwitting victim's machine is used to mine cryptocurrencies, spending the machine's precious resources.
  - **Packed**: Not referring to the actual purpose of the associated threat, this detection name generally means that the corresponding sample is protected with some malicious packer.
  - **Injector**: Not referring to the actual purpose of the threat, it means that the corresponding sample uses process injection for some reason (see the dedicated *Chapter 5*, *Inspecting Process Injection and API Hooking,* for more information about potential use cases).

- **Worm:** This category of threat is defined by the ability to self-propagate between different machines. There are multiple variants of worms depending on the protocol (for example, IRC) or media (instant messenger, email, and so on) they utilize to propagate.
- **Virus**: Unlike worms propagating between machines, the main goal of a file infector is to propagate within the current system by infecting other executables and documents. In this case, when the victim opens/launches a legitimate file, control is also given to the malicious code. There are several variants of how it can be used, from actually writing malicious code and data into executables and adding macro templates to documents to simply replacing victim files with their own body and storing a copy of an original file elsewhere to execute it later.

- **Rootkit**: Nowadays, this name doesn't have a single definition. Originally used to define tools elevating privileges (giving root access), it is most commonly used now to define threats that are either used to hide other ones or simply operate in the kernel mode. More information can be found in *Chapter 7*, *Understanding Kernel-Mode Rootkits*.
- **Bootkit:** Such threats insert themselves into the booting process (for example, by modifying the boot sector or boot loader) to gain access before the operating system.
- **Exploit:** Here, malware abuses a vulnerability in the victim software to achieve its goal (elevate privileges, access sensitive information, perform **arbitrary code execution** (**ACE**), and so on). See *Chapter 8*, *Handling Exploits and Shellcode,* to get more information about exploits.
- **FakeAV:** This category of threats shows users various warnings about allegedly critical problems with their systems and aggressively demands that the "full version" of itself is bought to remediate it.
- **Hoax**: Usually created as a joke or an act of hooliganism, this category of threats aims at simply scaring the user about some "critical" but actually non-existent problem.
- **PUAs:** Standing for **Potentially Unwanted Applications**, these threats generally involve less devastating but still annoying activity, such as silently installing legitimate but unrequested applications.
- **Adware:** Here, the threat displays non-requested advertisements to victims, in many cases aggressively and without an easy way to remove them.
- **Hacktool**: This is a big category involving multiple tools that can be used by both attackers and cybersecurity professionals, for example, for red teaming purposes.
- **Dual-use tools**: In this case, the corresponding tools can be used by both attackers and legitimate users, such as system administrators. Examples include the `psexec` tool by Sysinternals, which can be used to execute commands on remote machines, and various remote administration tools.

In many cases, samples fall into multiple categories. For example, one sample can propagate as a worm by stealing credentials and downloading additional payloads, while another sample may execute custom commands like a backdoor; the list of commands will include infostealing capabilities, elevating privileges by using an exploit, and organizing DDoS attacks. The choice of the final single category is generally dictated by each antivirus company's policy, where some categories are prioritized over others, usually based on the potential impact.

Sometimes, the software may fall into the so-called grayware category. In this case, it may not be completely clear whether this software is legitimate or malicious. Examples are some forms of PUAs and adware software or FakeAV-style security programs offering extremely little benefit compared to the price demanded. Usually, it is up to each antivirus company to decide what should be detected as a virus.

Naming conventions

Unfortunately, the cybersecurity community has not agreed on a single universal convention to name malicious samples and each antivirus vendor is free to use its own notation. Generally, the detection name will include the targeted platform, the malware category and family, and sometimes the version and the detection technology. Here are the detection names used by different vendors for the same malware sample 9e0a15a4318e3e788bad61398b8a40d4916d63ab27b47f3bdbe329c462193600 based on **VirusTotal** results:

- Avast: *ELF:CVE-2017-17215-A [Expl]*
- DrWeb: *Linux.Packed.1037*
- Kaspersky Lab: *HEUR:Backdoor.Linux.Mirai.b*
- Microsoft: *Trojan:Win32/Ceevee*
- Sophos: *Linux/DDoS-CI*
- Symantec: *Trojan.Gen.NPE*

As we can see here, different vendors commonly assign different names to the same malware family. Moreover, many companies have default names that they assign if identifying or creating the malware family name is too expensive or simply not worth it; examples are Agent, Generic, Gen, and others. In many cases, the situation also becomes complicated when the source code of some threat is leaked to the public, exchanged between hacker groups, or re-used in another project by the same author, resulting in the creation of threats that combine the code and functionality of multiple malware families. To choose a malware family name, follow the policy of your company or consider using the MITRE ATT&CK notation, if you want something vendor-agnostic.

The MITRE ATT&CK framework explained

As we have mentioned before, different cybersecurity vendors commonly give different names to hacker groups and malware families. Therefore, knowledge exchange becomes more complicated, eventually affecting the performance of the community. The MITRE ATT&CK framework was created to address this and other similar issues and let security experts speak the same language. This is a vendor-agnostic global knowledge base on various attack techniques grouped into tactics, which also provides examples of the attackers and malware utilizing them, giving the tactics widely accepted names.

Basic terminology

Here are some of the most important terms used in this field:

- **Tactic**: Represents a high-level goal of the attacker, a reason why the corresponding action is performed

- **Technique:** The practical way in which the defined high-level goal is achieved
- **Sub-technique**: A more detailed and granular description of how exactly a certain action is conducted
- **Procedure**: An actual implementation of the technique/sub-technique
- **TTPs**: Stands for tactics, techniques, and procedures: a summary of the methods used by attackers with an explanation of what is achieved by utilizing them
- **Group:** Represents a set of related adversarial activities likely to be performed by a single entity known under this name
- **Mitigation**: Technology and concepts that are used to circumvent or prevent an attack
- **Software**: Code that can be used to conduct adversary actions, combining both publicly available tools and malware
- **Matrix:** A combination of TTPs related to a particular industry secto

There are several matrices within the framework for the enterprise, **Industrial Control Systems** (**ICSs**), and mobile sectors. The most commonly used one is the Enterprise Matrix, so let's talk about it in greater detail.

Enterprise Matrix

At present, the Enterprise framework defines the following tactics:

- **Reconnaissance**: This stage involves collecting relevant information about the victim to perform a successful attack, for example, about some organization's infrastructure and personnel.
- **Resource development**: Here, attackers establish all the required dependencies based on the collected information. This can be achieved by various means: buying/renting, creating, or stealing the prerequisites (for example, hosting or software).
- **Initial access**: At this stage, attackers attempt to establish the first foothold within the victim's environment. One of the most common examples of this tactic is sending spear-phishing messages (mainly emails).
- **Execution**: Here, attackers execute code of any kind within the victim's environment to achieve their goals.
- **Persistence**: Includes everything attackers do to maintain their presence within the compromised environment. Common examples include adding malicious code to autorun or adding SSH keys to the list of authorized entries.
- **Privilege escalation**: As the initial access is in many cases achieved by compromising low-access accounts, here, attackers attempt to gain higher-level permissions to have more control over the affected environment.

- **Defense evasion**: The main goal of the attackers here is to avoid being detected until their objective is achieved. Examples include obfuscating malicious code or marking related files as hidden.
- **Credential access**: This tactic involves stealing credentials to misuse them later. Some of the most common techniques here involve dumping saved credentials and intercepting them, for example, by logging pressed keys.
- **Discovery**: Here, attackers collect information on the internals of the victim's environment, starting with the network and the local systems. This information is generally used to facilitate other tactics, such as lateral movement.
- **Lateral movement**: At this stage, attackers propagate upward to other machines until the systems of interest are reached.
- **Collection**: Involves collecting various information of interest from the affected systems. Common examples include stealing proprietary source code and documents.
- **Command and control**: This tactic covers the various ways attackers may remotely communicate with compromised systems.
- **Exfiltration**: Techniques that attackers may utilize to actually move sensitive information out of the compromised environment.
- **Impact**: Finally, this tactic describes other ways attackers may have a negative impact on compromised systems. Common examples include the manipulation, interruption, or destruction of critical systems and data.

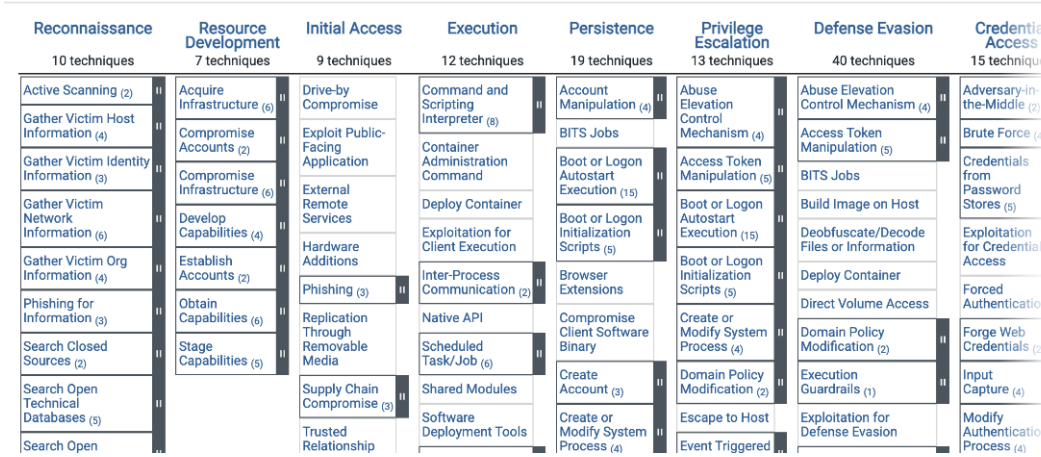

Figure 1.1 – Web representation of the MITRE ATT&CK's Enterprise Matrix

It is worth mentioning that the framework is not static and constantly evolves, incorporating users' feedback and addressing the new challenges the industry faces. Each version of the framework is shipped with a **Structured Threat Information Expression** (**STIX**) representation of itself: `https://github.com/mitre-attack/attack-stix-data`. It allows efficient integration with various software products and makes it possible to combine stability and efficiently oversee any changes introduced. STIX is a versatile format that is also commonly used by the cybersecurity community to exchange IoCs, where version 1 is XML-based and version 2 is JSON-based.

APT and zero-day attacks and fileless malware

Here, we are going to explain the meaning of some terms commonly found in whitepapers and news articles related to malware.

APT attack

APT stands for **Advanced Persistent Threat**. Generally, malware receives such a title if the actors tailored it to target a particular entity, whether it was an organization or a particular individual. This means that the attackers chose a specific victim and won't simply give up and go away if one approach doesn't work. In addition, the threat should be relatively advanced – for example, it should have a complex structure, use non-standard techniques or zero-day exploits, and so on.

Re-using IoCs for detection purposes in many cases is useless for APT malware as attackers register new network infrastructures and re-compile samples for each victim.

In reality, there are no strict objective criteria to evaluate how advanced a particular threat is. As a result, news outlets and affected organizations often tend to overuse this term to make attacks look more sophisticated than they actually are. This way, pretty much anything that is either relatively new or has led to a successful breach can be called an APT.

Zero-day attack

Many attacks involve the use of exploits targeting certain vulnerabilities to achieve particular goals, such as gaining initial access or performing privilege escalation. Usually, once the vulnerability becomes known to the public, the software vendor addresses the issue and releases a patch so that end users can update their systems and be protected against it. Zero-day attacks involve the use of zero-day exploits, which target vulnerabilities that were not previously known, thus defining a "day zero" upon which it happened. What that means for end users is that there is no solution for them to update the vulnerable systems and thereby address the threat. In this case, users are usually offered some partial workarounds to temporarily minimize the potential impact until the patch is ready, but they commonly have various drawbacks that affect the performance of the systems used.

Fileless malware

There are many reasons for malware to stay below the radar. First, it assures that malware will successfully land in the victim environment and perform all the necessary attack stages. Second, it will complicate the detection and remediation process, prolonging the infection and increasing the chances of success.

Incident Response (**IR**) engineers use all possible places where malicious activity may be recorded to build up a full picture, efficiently eliminate the threat, and prevent the incident from happening again. The data science that this comprises is called digital forensics. As part of this, the analysts will collect various indicators throughout the system, including file artifacts.

So-called fileless malware has emerged to prevent malicious activity and to bypass traditional antivirus products strongly focused on detecting malicious samples in the form of files. The idea here is that malicious code has no independent sample to detect and delete. Instead, the shell and inline script commands are used. An example of such a threat is Poweliks, which stores a malicious command in the registry key that provides autorun capabilities.

With all the important terminology now clear, it is time to talk about how to approach new reverse-engineering tasks.

Choosing your analysis strategy

Reverse engineering is a time-consuming process, and in many cases, there aren't the resources available to allow engineers to dive as deep as they would like to. Prioritizing the most important things and focusing on them will ensure that the best result is produced within the allocated time every time. Here is some advice that may help in this challenging task.

Understand your audience

Depending on who is going to use the result of your work, the actionable deliverables may be very different. Examples of the potential use cases for reverse engineering include the following:

- **Threat intelligence**: Here, the focus will be mainly on obtaining IoCs, such as hashes, filenames, and network artifacts. Therefore, extracting embedded payloads and downloading remote samples, as well as finding other related modules involved and extracting C&C information from all of them, will likely be the top priority.
- **AV detection**: In this case, the focus will be on anything unique enough to create a robust detection that doesn't produce **false positives** (**FPs**). Examples are distinctive pieces of code and strings related to the malicious functionality and any custom encryption algorithms used. Understanding the main logic will help choose the right category, and code and data similarity will lead to assigning the malware family.

- **Technical article or conference presentation**: Here, the most important part will be interesting novel technical details related to functionality, similarities with other malware families, and actor attribution.
- **Article for the general public**: For non-technical people, it is common to provide a high-level description of functionality without many technical details, focusing mainly on impact.

Answer your audience's questions

It's very important to answer the main questions your audience is asking. Make the answers clear and easy to find in your analysis report.

Here is a list of possible questions your audience might need an answer to in your report:

Target Audience	Questions to Answer
Chief Information Security Officer (CISO)	What's the impact of this attack? What are the assets that were compromised or exposed to the attack?
	What are the weaknesses in the existing security protocols in the organization?
	What is the remediation plan to stop similar attacks in the future?
Security Operations Center (SOC) Team	What are the IoCs?
Incident Response Team	What happened?
	What are the attackers' TTPs?
	What actions should be taken? What's the remediation plan?
Vulnerability Management Team	What are the vulnerabilities that were used in this attack to gain initial access, escalate privileges, or exploit other machines in the network?
	Are there any zero-day vulnerabilities involved? Does the vendor know and can they suggest a workaround for this issue?

As long as this part is clear, we can start prioritizing particular topics.

Define your goals

Once the audience is confirmed, define your goals carefully based on the resources available: first, time and skillset. After this, prioritize the selected goals and focus on the most important ones first. It is very easy to get lost in assembly when doing static analysis, so having a checklist of what needs to be done and in what priority will help you get back on track.

Avoid unnecessary technical details

Regardless of who is going to consume the result of your work, having too many extra details won't show your level of expertise but will simply complicate the understanding of the work and result in wasted time. Common examples include executed instructions, WinAPIs used, standard registry keys accessed, or mutexes created. Therefore, you should do the following:

- Choose the level of detail required depending on the target audience.
- If some fact doesn't help the reader, avoid elaborating on it.
- Don't just mention technical details – explain their high-level purpose and why the attackers had to explicitly use them.

Finally, make sure that the most important sections are covered in detail and are definitely correct. Never attempt to make statements based purely on gut feeling or prior knowledge without any material facts related to the current sample. You can always use the appropriate wording for something that you have spotted but don't have time to dig deeper into (for example: "there are indications that... but more work is required to confirm it").

Example structures

Here are some of the details that are generally included in the resulting work, depending on its format and the audience.

Technical article

In most cases, the following information will be useful:

- Sample(s) details:
  - Hashes (MD5, SHA1, SHA2)
  - Compilation timestamps
  - File types and sizes

  - **In-the-wild** (**ITW**) filenames
  - AV vendors' detections
- Modules' relationships (if there are several involved)
- For each module:
  - A description of the main functionality
  - Persistence mechanisms
  - Network communications:
    - Protocols
    - Encryption algorithms and keys
    - C&C details (IP addresses, domains, URLs, unique whois details, host countries, and so on)
  - Anti-reverse engineering techniques used
- IoCs
- Detection rules (YARA, Snort, and others)

General-public article

- High-level functionality description with a focus on the impact
- The scale of the attack
- Victim profile:
  - Types of organizations targeted
  - Victims' geolocation
  - Loss estimates
- Actor attribution:
  - Sample similarity
  - Matched IoCs (hashes, network artifacts, filenames, and so on)
  - Language codepages and strings used
  - Compilation timestamps

Typical analysis workflow

Now that we know what to focus on, the next question is: how do we organize the work to produce the best possible result in a timely fashion? The following steps are suggested for you to follow:

- **Triage**: Here, collect the maximum amount of easily available information on the sample:
  - Analyze the PE header.
  - Check whether the sample is likely to be packed or not (high-entropy blocks).
  - Check public resources for known IoCs (hashes, network artifacts, AV detection names, and so on).
- **Behavioral analysis**: Most of the information will be obtained from file, registry, and network operations. This way, we will have an idea about the capabilities of the potential sample.
- **Unpacking (if necessary)**: Static analysis is impossible before the sample is unpacked as the actual malware's code and data are not readily available yet.
- **Static analysis**: Performed with the help of disassemblers and decompilers:
  - Start from available strings and commonly misused WinAPIs.
- **Dynamic analysis**: Performed with the help of debuggers. May be quite expensive to set up and perform, so use it only when needed:
  - Confirming certain functionality
  - Handling string/APIs/embedded payloads/communications encryption

Setting up the environment

Being able to safely analyze malicious samples is a prerequisite for any engineer performing reverse engineering, whether it is a one-time task or a daily routine. Usually, for this purpose, **Virtual Machines** (**VMs**) are used because it is easy to make copies of them, apply any changes, and save snapshots to restore some previous state of the machine. Another option is to have dedicated physical machines separated from critical networks; in this case, some backup software is generally used to quickly restore the previous state of the machine. In this section, we are going to talk about setting up a safe environment for malware analysis and the most important steps to focus on.

Choosing the virtualization software

When you are ready to create a new VM, the first task is to choose what software will be used for this purpose. Generally, the top choices of reverse engineers are the following:

- **VMware**: A very popular commercial solution that also provides a free player to run already existing VMs
- **VirtualBox**: A free fully functional alternative that allows both the creation and running of VMs

Both of the preceding options provide similar end-user-oriented functionality and features such as snapshot management, emulation of shared ports, devices, folders, a clipboard, and network access.

QEMU is another option here, but the project has historically been more focused on emulation than virtualization, and its **user interface** (**UI**) might be less user-friendly for daily reverse engineering work. Other projects worth mentioning here include the **Kernel-Based Virtual Machine** (**KVM**) virtualization module, commonly used together with QEMU, and the Xen and Hyper-V hypervisors.

Regardless of what software you choose, the corresponding VM images can generally be converted from one type to another. However, each virtualization software has its own guest tools that make it possible to use features such as shared clipboards – in this case, they will need to be installed and set up separately.

Finally, there are pre-built VM images with a set of RE tools already pre-installed:

- **FLARE VM**: A free, open source, Windows-based solution supported by Mandiant/FireEye
- **REMnux**: A free, open source, Linux-based distribution that also provides pre-built VMs

Safety features

Here are the top safety features that should be respected when creating an RE-oriented VM lab:

- **Disabled network**

  As we know, many malware categories may misuse the network for malicious purposes. Whether it is sending spam, propagating to other machines, or stealing engineers' proprietary licenses, the rule of thumb here is to disable the network by default. There are plenty of techniques and pieces of software that can be used to simulate a network connection for analysis purposes, such as INetSim and FakeNet.

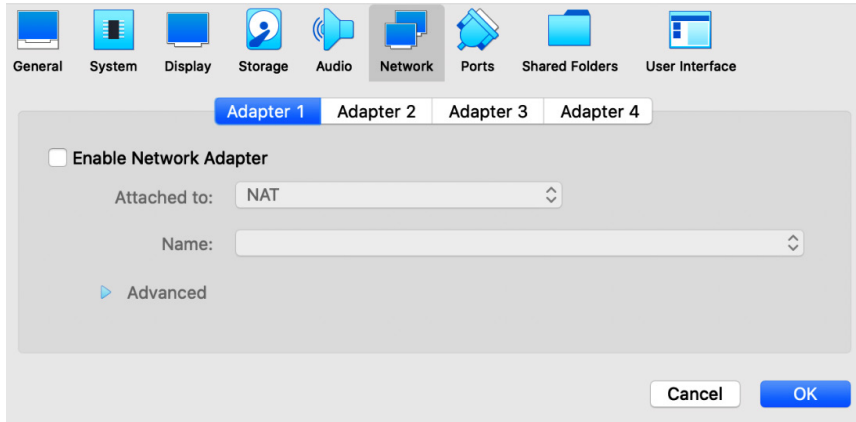

Figure 1.2 – Disabled network in the VirtualBox VM's settings

- **No shared devices**

  Many forms of virtualization software, by default, link connected peripheral physical devices to the VM. This can be extremely dangerous, for example, in the case of USB drives. In this case, malware can propagate there and this way escape the secure environment. Therefore, all such devices should be disabled.

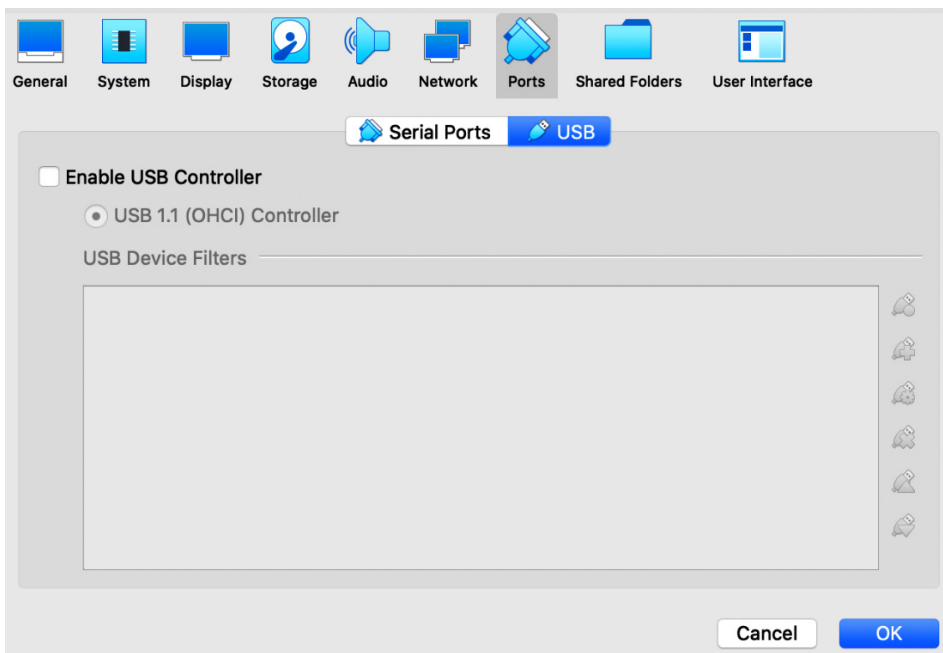

Figure 1.3 – Disabled USB controller in the VirtualBox VM's settings

- **Be careful with shared folders**

  Shared folders map some folders present on the host machine to folders mapped on the guest (virtual) machine for easy file transfer. The main concern here is that viruses can infect files located there (namely, executables or documents) or replace existing files with malicious ones. And just like that, the malware has found a way to the host machine. So, shared folders should always be used with care. One way this can be done is to avoid storing any files there longer than necessary: once the files are copied there on the host machine, take them out of there on the guest VM and leave the folder empty until the next task. Making the shared folder read-only for the guest machine is another option.

Once we have prepared our lab VM, the next question is – how can we copy our malicious samples there for analysis? There are multiple ways this can be done:

- **Private network**: Ideally, this should be avoided as malware running on the guest machine may also have network access to the host machine.
- **Shared folders**: As just discussed, use with care.
- **Shared clipboard**: One of the safest solutions. Requires guest additions to be installed on the VM in order to work.

As for moving files back from the VM to the production PC, the rule of thumb here is to exercise extreme caution. Consider doing it only for text files containing the result of your work and similar cases. If it is absolutely necessary to transfer anything containing malicious code and data (including memory dumps and network PCAPs), consider using password-protected archives to store them, which shouldn't be extracted on the host machine.

Summary

In this chapter, we have become familiar with various types of modern threats and shed some light on important terms used within the cybersecurity community. We discussed the MITRE ATT&CK framework, provided an overview of its capabilities, and highlighted some of its important features. We also provided instructions on how to set up a safe environment to analyze malware. Finally, we provided recommendations on how to organize work when dealing with malicious samples by various means.

In the next chapter, we are going to cover the basics of various assembly languages, which will give us the fundamental knowledge required to understand malware functionality and perform static and dynamic analyses of various types of threats.