Malware Analysis Framework

Table of Contents:

Introduction to Malware Analysis

Why Malware Analysis?

Malware Analysis is becoming more and more an important part of digital forensics and incident response (DFIR) for any kind of organization. The more the work is shifting to use computers to gather, process and store data and the more these systems are connected, the bigger is the attack surface to interrupt regular operation of an organization. Especially the topic of ransomware has shown how vulnerable some organizations were and how easily they could be attacked. To protect your organization from such attacks and to detect an attack before having a big impact on your organization, any security-, SOC- or CSIRT Team should have some knowledge about malware analysis to be able to identify or prioritize the incidents accordingly. This framework helps interested people to start in the field of malware analysis but is far from being complete. The Malware Analysis Special Interest Group from FIRST started this framework to enable interested people to get a first footstep into the topic and to find useful resources to support their learning process.

Advantages of Malware Analysis

- Learn how attackers try to attack your Organization / Network
- Learn how attackers where able to enter your Organization / Network
- Improve your monitoring and defense mechanisms

Success Stories

- The WannaCry Kill-Switch domain

General questions

- Does my CSIRT Team need malware analysis know-how?
- If yes, how much know-how is needed, which steps in Phase 1/2/3 shall we be able to do by our self?
- Can we outsource malware analysis?

Collaboration with other SIGs / Teams

The Malware Analysis SIG started this framework and is recommending to see what other related SIGs are doing, as malware analysis is not a separated field of interest but has a lot of interaction with other topics. To protect our organization we don't just need to understand how the attacker or malware is functioning, but we also need to be able to adapt our monitoring systems or share information with other teams or peers.

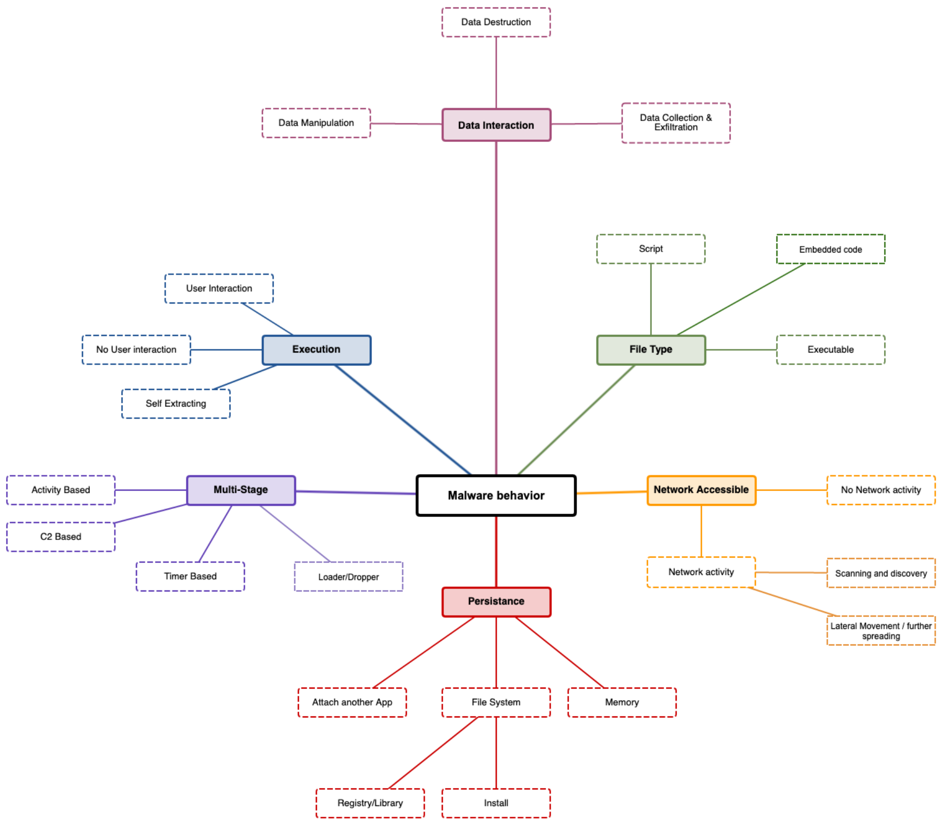
- Cyber Threat Intelligenge CTI SIG: sharing analysis results, TTPs etc.
- Information sharing SIG: sharing analysis results, TTPs etc.
- Red Team SIG: sharing analysis results, TTPs etc.
- Vulnerability Reporting SIG: reporting new detected vulnerabilities

Phase 1 - pre-Analysis

Questionnaire / Decision Tree

Before starting, there are different questions which should be answered to decide the strategy how to tackle the analysis of a malware sample. Maybe we're interested in how the malware is behaving on the system or we might be interested to find out how we can identify the malware on a system or in the network. As long as we don't know what we're looking for, malware analysis is like searching a needle in a haystack. But with a set of questions we want to be able to answer, we can set the direction in which we plan to go and we can speed up the analysis process to find the most important answers to this questions as fast as possible.

Questions about the sample



Data Interaction

How does the malware interact with data?

- Data manipulation: data will be manipulated, we can no longer trust the data we have on the affected system
- Data destruction: data will be destructed and will no longer be usable (e.g. Ransomware)
- Data collection & exfiltration: data will be collected and extracted, sensitive information will be stolen and might be published without consent of the data owner (e.g. Ransomware) or might be used for further attacks (e.g. Usernames / Passwords)

File Type

Which form does the malware have

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

Network Accessible

Does the malware use the network of an infected host?

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

Persistence

Does the malware try to survive a reboot or to stay permanently on the host?

- Attach an other app: the malware attaches itself to an other application (aka. host-app) to stay hidden on the system. As long as the host-app stays installed on the system, the malware might be able to execute the intended steps
- File system: the malware installs itself somewhere in the file system
    - Registry/Library: the malware adds commands to the system's registry to be automatically executed during a system start or installs itself as a library and might be started when the library is called
    - Install: the malware installs itself on the host like any other app
- Memory: the malware only resides in memory, a power loss would be fatal to the malware and all traces would be lost. There are malwares to be known to use a combination of such techniques. For example the are installed on the system and started via a registry key, as soon as the malware is running and loaded into memory, the malware deletes itself from the file system to be invisible to anti virus scanners. During a proper shut down procedure of the host system, the malware would reinstall itself on the file system and reactivate the registry key to be functional again after the next reboot. A loss of power would destroy the malware from memory and it could not save a copy of itself on the filesystem.

Multi-Stage

Is the malware the complete package or are there more stages to be installed after the infection?

- Activity based: based on the activity of the host system or the user, the malware might load additional modules
- C2 based: after contacting a command and control server via the internet, the malware gets more commands or modules loaded to be able to offer further functionality
- timer based: based on time algorithms, the malware might be 'awaken' after some activity on the host system and perform it's tasks
- loader / dropper: the malware might only be a first step of the infection and load or drop additional malware on the host system

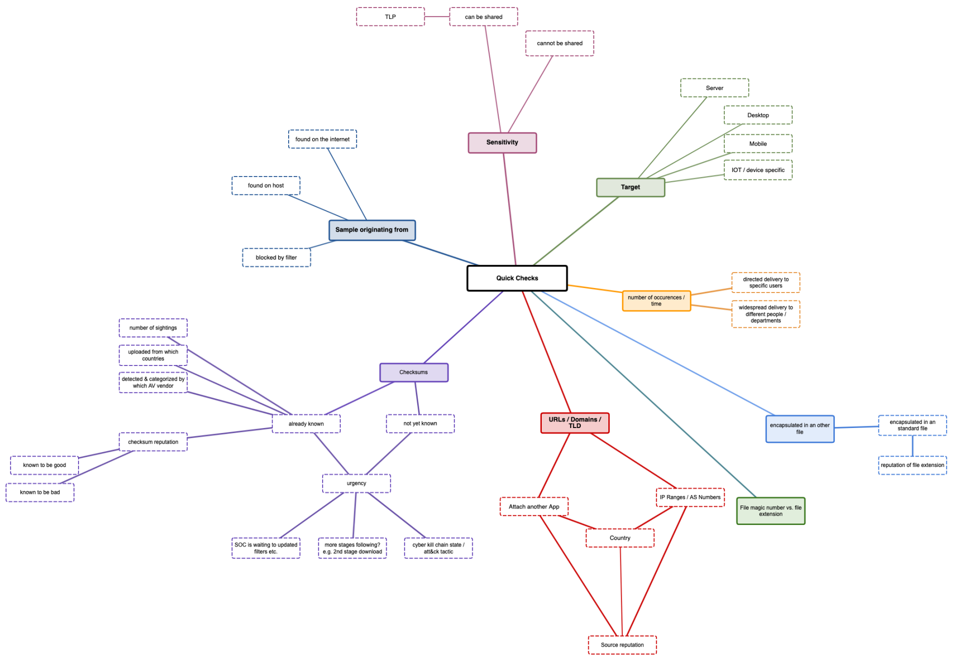
Execution

Does the malware need some requirements to be fulfilled or some specific user interaction to start?

- User interaction: the user needs to actively interact with the malware like clicking on a link in an infected office document
- no user interaction: as soon as the malware is started, the processes are started without any user interaction
- self extracting: some malware families are distributed in a compressed format and first need to be extracted to be functional. The extracting processes are part of the malware and as soon as the process is started it will extract itself on the host system and start it's main processing steps

Quick checks

On a usual CSIRT working day, there are more malware samples found than a team usually can analyze. Triage is one of the most important steps to reduce the number of sample which should be analyzed to a level the CSIRT team can handle. Analyzing a sample which you found 3 month ago might be interesting for you as an analyst, but it might be to late for your organization to be able to react to the threat. Time is often a critical component of malware infections and as sooner it it known what a sample is doing, the better are the options to fight the threat.


TLP
can be shared
cannot be shared
Sensitivity
Server
Desktop
Mobile
IOT / device specific
Target
found on the internet
found on host
Sample originating from
blocked by filter
Quick Checks
directed delivery to specific users
number of occurences / time
widespread delivery to different people / departments
number of sightings
uploaded from which countries
detected & categorized by which AV vendor
Checksums
already known
not yet known
checksum reputation
known to be good
known to be bad
urgency
SOC is waiting to updated filters etc.
more stages following? e.g. 2nd stage download
cyber kill chain state / att&ck tactic
URLs / Domains / TLD
Attach another App
IP Ranges / AS Numbers
Country
Source reputation
encapsulated in an other file
encapsulated in an standard file
reputation of file extension
File magic number vs. file extension

Target

What is the target platform the malware should running on?

- Server: the malware will run on a server, mostly without any user interaction
- Desktop: the malware will run on a desktop / notebook, mostly with some kind of user interaction
- Mobile: the malware will run on mobile devices like tablets or smartphones
- IOT / device specific: the malware will run on specific devices with specific CPU platforms like ARM, RISC or on specific IOT devices like routers, network connected cameras (e.g. MIRAI)

Number of occurrences / time

How many samples did you find in our environment and who are the intended recipients?

- the malware is sent to to a small number of recipients in specific positions (e.g. Finance, Management) - this could be a hint the malware was sent for a specific reason and with a specific intend to attack your organization
- widespread delivery to a huge number of recipients in different departments or regions - this could be a hint the malware is more general and is just trying to infect as many victims as possible

encapsulated in another file

The malware itself is encapsulated in another file, e.g. a macro in a standard office document. Often such files are sent out with strange file extensions (e.g. an ISO File as a Mail attachment). Many of these attachments are often not the standard files used within your organization and might not have the best reputation.

file magic number vs. file extension

Malware is often transferred and the attacker tries to circumvent typical filtering mechanisms. Some filters just check the file extension and do not check the file itself. For example an EXE file could be renamed to HTML and the filter would let this file pass. On the target system, the file will still be a fully functional EXE file and can be started without any problems. If a file extension and the file magic number are not correct, this could be a hint that someone tries to attack your organization.

URLs / domains / TLD

From where was the sample sent?

Malware is often sent from all over the internet, but maybe a lot of sources are very unusual in your organizations daily routine

- Why do I get an strange email from South America if my organization only as relations in Asia?
- Is the sending Top Level Domain known to be often misused?
- Is the domain, ASN or corresponding URL already listed to be malicious on well known reputation lists?
- Is the source well known or has a bad reputation?

Check sums

Instead of sharing filenames (which can be changed without any problems), malware samples are often compared via checksum like MD5, SHA1 or SHA256 hashes. This check sums only change, when the content of the file changes and are independent from the filename or file extensions. Like this, they are a very handy quick check as you can calculate, compare and share this check sums with others

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

Sample originating from

Where did you get the sample from?

- you found it on a host in your environment and think it should not be there or it was detected by an anti virus software
- it got blocked by a filtering mechanism (web proxy, mail gateway)
- someone found it somewhere on the internet and sent it to you for further analysis

Sensitivity

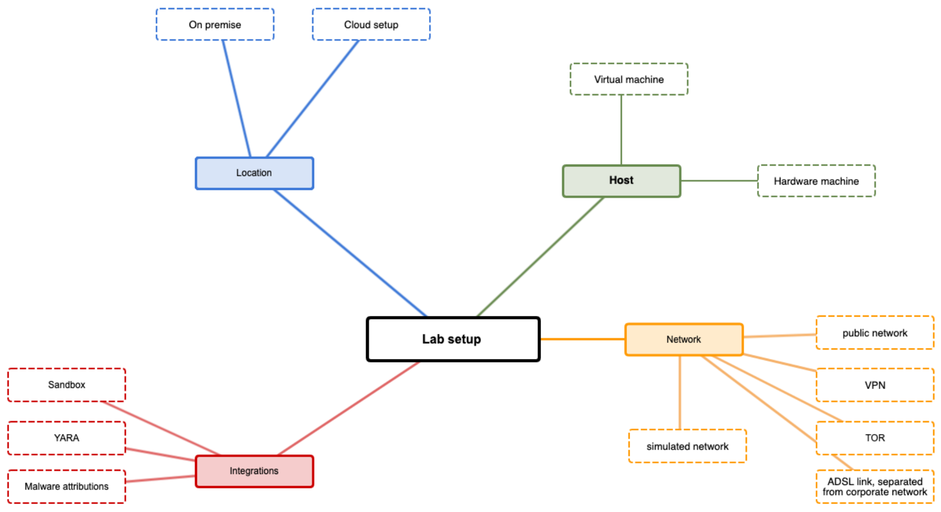
Can the file be shared without disclosing any sensitive information?

- Office files with malicious macros in it, can still be valid office documents and contain sensitive information about your organization. As soon as you share such a file with other peers or public platforms like VirusTotal, the information in this file is public. Some information should not be shared with others and you might not be allowed to upload such a file to any public platform
- if the file does not contain sensitive information, you can share it with others
    - Is there a TLP level, that does allow you to share this sample with others or not?

Phase 2 - the actual Analysis

Lab Setup

The setup of your malware analysis lab can have different setups and different components. Depending on the questions you're trying to answer you might not need or use all of this components. Depending on the maturity of the malware analysis capabilities of your team or your available budget, the lab might also differ from other team's lab setup. Important to know is, there is not one solution that fits all needs and there's nothing like a 'perfect setup'. Your setup has to cover your needs and your capabilities. The most fancy and most expensive setup will not help if you don't have the time to use it to the full extent.



Host

Usually you use separate hosts to infect them with a malware sample and you don't use your daily PC for this task. Such hosts can either be virtual running on VirtualBox, VMWare, etc. or dedicated hardware.

- virtual machines have the advantage of being flexible, mobile (you can move them between different virtualization hosts), offer snapshots to revert them to the same stage before starting an analysis. Some malware families might use 'anti-reversing-techniques' to prevent them infecting a researcher's host and might check for connected peripheral devices like sound cards, keyboard, mice, ...

- physical analysis machines are a bit more complicated to revert to a 'clean stage' before starting an analysis, but are more flexible for longer analyses, where you check the behavior of a sample for a longer time frame. For example, if you observe a malware sample for multiple days or weeks to see how they change the C2 communication servers or what additional modules are loaded, you might want to let them running without interruption, while you'll be able to shutdown or de-connect your PC or Notebook from the network.

In both cases you should keep an eye on 'anti-reversing-techniques' where malware samples check the host for specific conditions like number of stored documents on the users desktop, connected peripheral devices etc. The more your analysis host resembles to an actual used host, the higher is the chance to get a successful infection.

Architecture

Malware is developed for specific tasks or platforms. Commodity malware will be developed for the most used platforms. As more and more new platforms are available on the market (ARM, IOT, ...) more and more malware will be created for this platforms as well. Make sure you'll have the tools and simulators available for all different architectures you want be able to analyze.

Network

The network setup is one of the most crucial and important points to be clarified before starting an analysis.

- simulated networks: you don't connect the infected host to a real network, but use a completely separated and closed network (e.g. a host network on a virtualization host). Instead of having an internet connection, you use a internet simulator like inetsim on this network to simulate the internet. With this setup, no connection can leave the network and there is no possibility any information can leave or enter your environment. You might not be able to get a 2nd stage module, but you might be able to see which C2 connections a malware is trying to establish.
- ADSL link, separated from corporate network: one of the most important aspects of malware analysis is to protect your working environment from the malware. Some teams use separated Internet links (e.g. via ADSL) to connect the malware hosts to, to offer them full internet access but being separated from the rest of the corporate network. If an attacker should keep an eye on which victims are 'calling home', he'll just see that the connection is coming from a SoHo-ADSL link and will not be able to trace you back to the actual company you're working for.
- VPN: instead of using separated ADSL links, you could setup a VPN connection from your lab to some VPN provider and use this connection for all analysis tasks. It's also a very good possibility to separated all analysis traffic from your corporate network, but might be the less costly option than separated internet links. For multinational organizations this has also the advantage to be able to switch to different VPN outbreaks in different countries in case you expect the malware to use some kind of Geo-fencing techniques or only infect hosts in a specific country. Some malware families are known to not infect a host if the language or IP can be resolved to specific countries.
- TOR: like VPN, you can use a connection to the TOR Network to route all your malware connections over this connection
- public network: you could also use you're company's public network to connect your malware lab to the internet, but with this setup you really should make sure the malware can under no circumstances reach any of the company's network segments. This setup is very risky and not advised to be used!

Integrations

Often the analysis process consists for multiple steps going back and forth. You see some detail or behavior from your sample, you check this with other tools and sources, decide to go a step further with your analysis and so on. For example, a packed malware sample will not be detected by a well known Y ARA rule. As soon as the sample is unpacked, this YARA rule will be able to detect the malware and you'll know which malware family it belongs to. Such integrations can be very helpful in your analysis and you might be able to integrate them as good as possible in your lab setup.

Sandboxes: Malware analysis sandboxes are often used to get a first impression about a sample. This might be one possibility to see which samples are already detected or known in the public and which are interesting to dig into deeper.

Malware attributions: not every malware is developed from scratch, programmers reuse code all the time and every programmer or group has it's own style of coding. Malware attributions is the process of trying to identify similar part of codes and attribute them to a specific actor or group of actors.

Location

Depending on your possibilities and budget, you might not be able to have all equipment and tools in your own environment but might be able to outsource some of them to the cloud. Sandboxes are usually quite expensive to have them on premise but offer often quite good prices for their cloud versions. Like this, you could use a cloud version of such a sandbox to run some quick analysis first and identify the samples which are worth to spend some more work in your on-premise lab.

Storage

Analyzing malware can be a storage intensive taks, especially if you want to keep the analysis results for later comparison to other samples or for legal reasons. Memory, disk or network dumps can take up to multiple gigabytes for one single case. Make sure you have enough storage available for the analysis itself and also for long term storage of artifacts, dumps and so on. Cheap USB Disks may be a simple way to store data, but during an analysis data throughput might ask for a better solution or searching over multiple differen USB disks might can unnecessarily complicate the work. Legal requirements might also ask for specific solutions like write blockers to ensure an suspected device was not changed during the analysis.

Static Analysis

Static analysis is the analysis of a malware sample without executing the sample. Like this you can already get a lot of interesting details about a file, without actually running it and risking an infection or any other damage. A lot of the quick checks from the pre-analysis phase 1 are actually steps from static analysis.

File type

- What is the file type of the sample?
- Does it comply with the file extension?

Splitted files

- Is the sample in a single file or splitted over multiple files?

Hashes

To identify a file, you can compare the files size, name, timestamps and so on. The filename for example can be changed, without changing the file content and you wouldn't be able to identify a copy with a different filename as the same file. For this cryptographic hash functions are used to calculate the file hash and compare it. These hashes will stay the same as long as the file content does not change. MD5, SHA1 and SHA256 hashes are often used to identify and report files.

AV scanning

Is the file already recognized by Antivirus engines? Before you invest too much time into analyzing a file, you should check if this file is already known by AV solutions. One possibility is to check the file hash on VirusTotal to see what most of the AV vendors know about this file.

extracting strings

Depending on the file type (e.g. a script), most of the file is more or less human readable. The more compressed the file is, the less readable it gets. Even compiled EXE or DLL files can still contain some readable parts, as not all compilers try to minimize the file size and pack or encrypt as much as possible. Checking if some strings can be extracted from a malware sample might already give you some hints like URLs, domain names or IP Addresses which can be extracted and checked if they might be dangerous or not.

file obfuscation

Some malware samples as heavily packed and encrypted to make it as hard as possible getting detected by AV solutions or other checks. Compiled files and be packed or scrambled even more and end up in a self extracting format. This makes it even harder to analyze the file, as it must be de-packed or decrypted before being able to identify strings or other details. On the other hand, regular EXE files are usually not packed and encrypted beyond recognition, so a heavily packed or encrypted file already seems suspicious. There are a lot of tools which can identify the packing algorithm and some of these algorithms are known to be used by malware programmers.

PE Header information

Files, especially executable files, have to follow specific rules to be conform to a format defined by the operating system. Windows executables must conform to the PE/COFF (portable executable/common object file format) to be accepted and executed by the operating system. The format is called 'portable' because it contains information, resources and references to dynamic-linked libraries (DLL) that allows the operating system to load and execute the machine code on any suitable machine. The PE file is a series of structures and sub-components that contain the information required by the operating system to load the application into memory. When an application is compiled, the compiler will add all the needed header information, which describes this structure. When executed, the operating system loader reads this information and loads the binary content from the file into memory. Examining the PE header yields a load of information about the binary and it's functionality.

file dependencies and imports

Usually every malware interacts in some way with the local system it is running on. And as programmers do not want to reinvent the wheel and every operating system offers functions to interact with the system (file access, network, registry and so on), malware often depends on the functions exposed by the operating system. Windows for example exports most of its functions, called Application Programming Interfaces (API), in dynamic link libraries (DLLs) that provide different functionality. Such functions are 'imported' by the malware and are called imported functions or imports.
If a malware sample for example wants to create a file on disk, on Windows it can use the API CreateFile(), which is exported in *kernel32.dll*. To call the function, the malware first has to load *kernel32.dll* into its memory and then call the *CreateFile()* function.
Inspecting a malware sample and checking which function it imports, gives you a good overview about it's functionality. Every malware that want's to interact with the network, will have to use some imports related to network functionality (DNS lookups, TCP/IP communication, ...). A DLL can also import functions from other libraries to perform system operations.

exports

Exports are the inverse of imports. An executable or DLL can export functions, which can be used by other programs. Usually DLLs export functions that are imported by the executable. A DLL cannot run on its own and depends on a host process for executing its code. An attacker can try to create a DLL that exports functions containing malicious functionality. Inspecting the exported functions can give you a quick idea of the DLL's capabilities.

PE Section table and sections

The actual content of the PE file is separated into sections. This sections represent either data or code, have in-memory attributes such as read/write and are immediately followed by the PE header. This section representing code contains instructions that will be executed by the CPU, whereas the section containing data can represent different types of data, such as program data (global variables), import/export tables, resources and so on.

compilation timestamp

The PE header contains information that specifies when the binary was compiled. This field can give an idea of when the malware was created. This can also help in compiling a timeline of different versions of the same malware or malware family. Attackers often also try to modify the timestamp to prevent the analyst from knowing the actual timestamp and can try to set this timestamp to a future date.

PE resources

The resources required by the executable such as icons, dialog, menu and strings are stored in the resource section (.rsrc) of an executable. Often attackers store information such as additional binary and decoy documents, configuration data in this section. So examining this section might reveal valuable information about the executable.

comparing and classifying the malware

- fuzzy hashing: classic file hashes change as soon as one bit in a file is changed. This makes it very difficult to identify "similar" files, where a bit portion is equal and only some small parts differ. Fuzzy hashing is what helps in this case. It's a great method to compare files for similarity instead of equality. Like this it is quite easy to detect samples belonging the the same malware family or the same actor group, as this files are similar. ssdeep is one example of a fuzzy hashing tool which is often used for this purpose.
- digital signature / authenticode: operating systems try to make it harder to install software from unknown sources. Digital signatures like authenticode for Windows are used to verify the software was created by a trusted source. Malware authors try to fake or re-use signatures to make their software look as trusted.
- import hash: import hashing is another technique to identify related samples and the samples used by the same threat actor group. Import hash (or imphash) calculates a value based on the library/imported function (API) names and their particular order within the executable. If the files were compiled from the same source and in the same manner, their import hash values would tend to be equal. If you come across samples that have the the same imphash values, it means that they have the same import address table and are probably related.
- section hash: similar to import hashes, every section of a PE file can be analyzed and a corresponding hash value can be calculated. Equal sections in different samples will have the same hash value.
- YARA: A malware sample can contain many strings or binary indicators. Recognizing the strings or binary data that are unique to a malware sample or malware family can help in malware classification. YARA is a powerful malware identification and classification tool. Malware researches create and share YARA rules based on textual or binary information contained withing the malware specimen. Be aware, YARA usually does not work for packed samples, unless you have a rule to identify the packer itself.

Behavioral Analysis

Behavioral Analysis covers the steps of actually running a malware sample in an isolated environment and looking on different aspects of its behavioral like network connections, reading or writing files on disk, adding registry keys etc. Most of this can't be seen or found using static analysis or the behavior cannot be understand by only looking at the results from the previous steps. Behavioral analysis is like watching an animal in a zoo and studying its habits for hunting, eating and so on.

process inspection

Modern operating systems treat every running application as a set of processes with a hierarchical ordering. You can inspect this processes, how and by whom they were created and inspect their properties. Like this it is quite easy to identify the malicious binary and which other process started it's execution. To identify malicious processes on a server for example, you can take a snapshot of the known running services from a clean state (aka 'the known good') and compare a snapshot from a later moment. If there are new processes running, this might be a hint that someone interfered with the server.

Logging system activities

Every application usually interacts with the system. It reads or writes files, it sends or receives data over the network and so on. As applications usually use system functions offered by the operating system and do not interact directly with the network interface to send data for example, these interactions can also be monitored. Like this you can identify with process interacts with the filesystem, network, registry and so on and what they are trying to achieve. Every malware will leave some traces on the system they are attacking and running on, so we just need to find this little details.

capturing network traffic

The advantage of analyzing the network is, we can do it either on the system itself or on some central network point where every connection has to go through. Like this, a malware cannot identify if there might be some local capturing process running and will not be able to identify the central inspection point. Otherwise, if the connections are encrypted, we might not be able to see what the malware is sending out via the network. But just by knowing where a malware connects to, we already have a hint and might see other hosts using the same connection endpoints and can identify them as infected too. If we really want to understand what the malware is sending via the network, we might have to dig into the code to get to the functions which trigger the connections or we might have to try to use a man-in-the-middle attack to see what a process is communication over the network. How deep we want to dig into the network traffic, is one important question we should have answered in phase 1. Do we only want to identify that there is something strange going on or do we want to know all the details about what is going on.

simulating services

In the lab setup section we discussed the different options like full network, VPN, isolated network and so on. On an isolated network we can inspect all the connections from a running malware, but there will not be much going on if the malware cannot resolve DNS Queries and so on. By simulating such services, we can simulate a connected network and can at least identify some of the details like DNS queries and IP addresses if they should be hard coded into the malware.

A good example is the WannaCry case, where the researcher Marcus Hutchins saw the malware he was analyzing tried to resolve the kill-switch domain www.ifferfsodp9ifjaposdfjhgosurijfaewrwergwea[.]test. When he saw the domain was not registred, he registred it and added it to a sinkhole network to see how the malware would interact with this domain. Surprisingly the malware stopped encrypting the local host as soon as the DNS query was answered with a valid IP address and the WannaCry outbreak was stopped. The reason for this kill-switch domain is still unknown, but the simple step of identifying this domain name, registering and activating it stopped the malware from spreading further and encrypting more systems.

Dynamic-Link library (DLL) Analysis

A Dynamic-Link library (DLL) is a module that contains functions (exported functions or exports) that can be used by another program. Windows contains many DLLs that export various functions called application programming interface (APIs). The functions contained in these DLLs are used by the processes to interact with the file system, process, registry, network and the graphical user interface (GUI).

Attackers will distribute their malicious code often as DLL instead of executable files. Advantages of using a DLL instead of an executable are

- a DLL cannot be executed by double clicking it, a DLL needs a host process to run. Like this, a malicious DLL can be loaded into any process, even a legitimate one like Explorer.exe or winlogon.exe and will be "hidden" by the legitimate process
- injecting a DLL into an already running process provides the capability to persist on the system
- when a DLL is loaded by a process into it memory space, the DLL will have access to the entire process memory space, including the ability to manipulate the process's functionality. For example, a malicious DLL loaded into a browser process will have access to stored credentials
- analyzing a DLL is not straightforward and can be tricky compared to analyzing an executable

Most malware samples download or drop a DLL and then load this DLL into the memory space of another process and the dropper/loader component deletes itself. Like this you'll only find the DLL on the system and the dropper/loader has disappeared.

Code Analysis

Code analysis is the next step in malware analysis, where you try the analyze the actual code of the sample. Usually this step is called 'reversing' and means you run the sample in an code debugger and step through the program's sequence.

Anti reversing techniques

While we as malware analysts are interested in as many information about a sample as possible to understand the malware and its intent, the creator of the malware wants to prevent this as much as possible to make our life harder. He's trying to hide information, make it invisible (at first glance), split it into different parts or hide it somewhere he thinks we're not going to look. Anti reversing techniques are different approaches to make a sample as inconspicuou as possible but still capable to fulfill its intent. As malware analysts and researchers we must understand this techniques to be able to detect and identify them and being able to circumvent the obstacles the creator wants to throw at us.

eliminating symbolic information

When a program is compiled, source code written in a high level programming language like C++, Go, .. will be translated into machine readable assembler code which can be processed by the CPU. In this process, the compiler will add some symbolic information which are not needed by the CPU to run the sample, but which would be helpful to debug the code in case of problems. Eliminating this symbolic information makes the debug process harder, but does not impact the program flow. On the other hand it is a hint, the author of the program wants to hide something.

code encryption

Code or parts of the code can be encrypted to make it impossible to analyze the sample without running it. One of the first steps of the program would be to load the encrypted part and decrypt it and than start the execution of this decrypted part. Some samples use code encryption to just encrypt settings or part of settings like URLs, Domains, IPs so they are not visible in the sample for a simple string analysis. During the execution, this encrypted variables would be decrypted before using them in the program flow.

active anti debugger techniques

Malware authors try a lot of things to protect the malware from being detected and analyzed. Often they check in which environment the malware is running to see if it's a valid environment with a high probability of success or if it's in a lab environment with almost no probability to end in a successful infection. There are anti debugger techniques to check, if the running host is a regular PC or a researcher device or a sandbox.

- IsDebuggerPresent: is a simple function to check, if a debugger is present on the running host. If there is one, the malware will most likely finish execution to not reveal any more details
- CheckRemoteDebuggerPresent: checks if a process is being debugged by another parallel process
- Breakpoints are used to halt the process, if breakpoints exist on a system it might be used for analyzing software
- System Flags can be used to set global settings for the Windows operating system. Checks if this flags are present and/or set may indicate if the host might be used for research only

This are only a few samples, but they all might be used within a malware to check if it is running on a valid host or not.

confusing disassemblers

An disassembler is the opposite of an compiler. It takes assembler code and tries to rebuild the corresponding source code. Unfortunately this step is very complicated as the same assembler code can originate from different source functions and the reverse step is not unambiguous. By trying to confuse a disassembler the malware will still offer the intended functionality, but is much harder to reverse.

code obfuscation

Code obfuscation is a simple technique to make code difficult to be understood by humans. Plain text strings can be transformed into base64 strings or can be changed by simple XOR function. Other strings might get encrypted like we discussed in the section 'code encryption'. We as humans will have problems to understand this obfuscated code blocks, but the malware will still run without any problems. Some malware authors also add junk functions or routines with not real functionality, just to throw off analysts.

Memory Analysis

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

Phase 3 - post-Analysis

We started the process by selecting the questions we want to answer and decided how to answer them. The end of the process or phase 3 is to bring this all together and summarize our findings, decisions and steps we took and what we've learned on the way. We self might be satisfied by what we've discovered and learned, but other teams and especially our management might want a bit more information. A structured report helps to record everything for other teams or upcoming analysis which might profit from everything we've learned. Also you should think about your documentation and what should be updated or changed for upcoming analysis. The end of one analysis is most likely the start of the next pass of the same process.

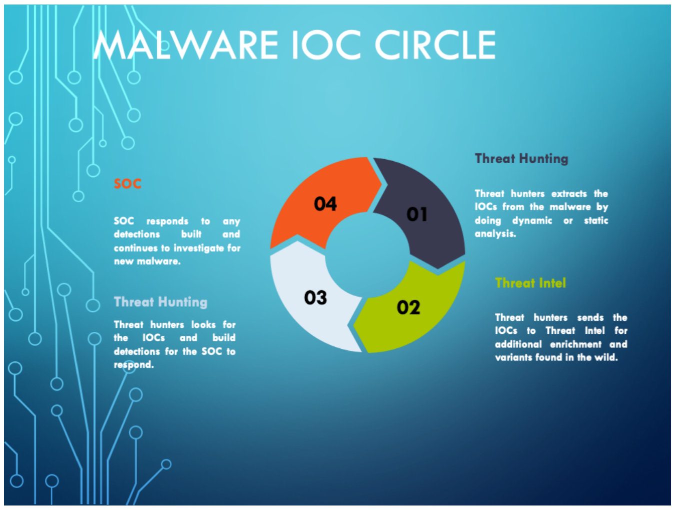
Especially learnings where you've lost a lot of time and did not find anything interesting might help you to prevent the same issue in another investigation.

The Report

Your report should contain at least the most important details about the sample, to identify the sample as good as possible

- filename
- filetype
- hashes

Also try to think what other teams might be interested in to know about the sample. A SOC team might be very interested in IOCs like C2 IPs or domains, while management can profit more from a classification and high level description of the basic capabilities of the sample. Also don't forget to add why you decided to analyze this exact sample, what was it that caught your eye and made it interesting to you. Supporting figures like log files and analysis result should be stored for some time as well, maybe there will be a similar sample in the upcoming hours, days, weeks and you want to be able to compare them to the behavior of a new sample.



Threat Hunting

Threat hunters extracts the IOCs from the malware by doing dynamic or static analysis.

Threat Intel

Threat hunters sends the IOCs to Threat Intel for additional enrichment and variants found in the wild.

Threat Hunting

Threat hunters looks for the IOCs and build detections for the SOC to respond.

SOC

SOC responds to any detections built and continues to investigate for new malware.

Example Report

CMAR Template 2...alware SIG.docx

External references

MITRE ATT&CK Framework

A lot of organizations nowadays use the MITRE ATT&CK framework to describe and compare malware. Referencing your report or parts of your report to external sources will help others to better understand what details you have discovered and how they can use your results to protect their environment from the same kind of attack.

In this version of our framework we don't got into more details, as the ATT&CK framework is quite well described by it's creators and would go beyond the scope of the malware analysis framework.

Other sources

Don't forget to link to other resources as well. E.g. if your following a threat actor group or malware family for a longer time, you might know them as well as you might know some of your best friends and for you it might be absolutely clear why something is the way it it. Other teams or even other members of your team might not have the same knowledge about this topic, so add links to all kind of external resources like other reports, blog posts and so on.

Sharing your report

TLP

While creating the report you should also think about who else might be interested in knowing about the work you've just done. Maybe you can help other organizations or communities in sharing your report or parts of it with them. Maybe you are not allowed to share all of the details, but you might be able to share parts of it or just have to anonymize some parts of your report to be able to share it with peers. You'll see sharing such reports (or parts of it) will help others to protect their environment through the results you gathered and they will also be willing to share their results with you!

TLP is one way to classify your report and who others can use the report in their environment.

IOC Sharing

As you can see in the Malware IOC Circle, a report is not an isolated document but is part of a circle of different teams working together and using each others results to improve their own results. Nowadays there are sharing communities everywhere who often use MISP or other tools to share analysis results in an efficient way to learn from each other and to import this IOCs into your monitoring environment. Automated IOC sharing is also a way to share without exposing to much information to other organizations. You might be able to share the hash of a malware sample without disclosing the exact filename or from where you got the sample. C2 IPs or domains are not secret and usually can be shared without any difficulties, but you might not be able to share internal details like your network setup and so on.

Addendum

Malware Classes

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

Authors / Contributors

Olivier Caleff, James Potter, Raja Jasper, Andreas Mühlemann

Reviewers

Per Morten Sandstad, Hiroki Kuzuno