Contents

INTRODUCTION TO MALWARE

What malware is:

Malware (combination of words malicious and software) is kind of software intentionally designed to cause damage to a computer, server or computer network. It can take the form of an executable, script, code or any other software and perform malicious actions. Attackers use it to steal information spy or take control of the system. Usually it works without users' permission and sometimes even without their knowledge.

Who creates malware and why:

At the beginning, malware was created by students and used only for scientific purposes only. With the flow of time and computer development some people started trying to make a profit by using malicious software. It led to creation of black market and many other illegal things. Attackers started stealing things such as: personal data, credit cards information, gaming accounts and sell them illegally. Also, they started taking control over victims' computers or destroying them.

How malware penetrates systems:

To start stealing and gathering information, attackers must spread their malicious software across as many computers or mobile devices as possible. This can be achieved through one of two ways:

a) Social engineering – it uses the weakest point of security system which is human. It lures an unwary user into launching an infected file or opening a link to an infected site.
b) Infecting system without user's knowledge – cybercriminals can use techniques that covertly introduce malicious code into a system by exploiting vulnerabilities in the security functions of the operating system and in the software.

Popular methods of spreading malware:

a) By social engineering:
   - Impersonate an organization administrator and sending links to websites which look like official ones
   - Links to infected websites or infected files, hidden in e-mail messages
   - Using peer-to-peer connections
b) Using exploits:
   - In the software or operating system
   - On websites (after entering an infected website, proper script causes download of malicious software)

Bypass anti-viruses:

Cybercriminals to fully achieve their goal must either force user to launch an infected file or penetrate the system through vulnerability that evades any antivirus filtering. Common techniques to do so:

a) code packing and encrypting – in order to detect such malware, antivirus programs must add either new unpacking and decoding methods, or signatures for each sample of a malicious program. This inevitably leads to decrease in detection rates as no antivirus company can possibly have a sample of every piece of malware at its disposal

b) code mutation – is mixing the malicious code with "spam" instructions, resulting in the change of its appearance, whilst retaining basic functionality

c) stealth techniques – the so-called rootkit technologies are used to intercept and substitute system functions which make the infected file invisible to the operating system and antivirus programs. Sometimes even the registry branches where the malicious software is registered, along with other system files, are hidden

d) blocking antivirus programs and antivirus database update systems – malware searches for antivirus programs in the list of active applications and try to block them, damage their database and block their update processes

e) masking of the code on a website – in order to combat antivirus scanning, a webpage can be modified, thus if the request was sent by an antivirus company, a non-infected file is downloaded instead of infected one

f) a "quantity" attack – is the generation and distribution of large quantities of new malicious software versions across the Internet within a short period of time. As a result, antivirus companies receive huge amounts of new samples and it takes time and effort to analyze each one, which gives the malicious code an additional chance to successfully penetrate user's computers

Classification:

The need to classify detected objects arose with the advent of the first antivirus program. Even though viruses were few and far between at that time, they still needed to be distinguished from each other. The first attempts to regulate the classification process were taken in the early 1990s by the CARO (Computer AntiVirus Researcher's Organization).

There are many classification systems, one of the most widespread and used as the basis for classifications by several many antivirus vendors is system used by Kaspersky Lab. Malicious programs detected by Kaspersky Lab projects are represented below:
viruses and worms:

a) Email-Worm
- IM-Worm
- IRC-Worm
- Net-Worm
- P2P-Worm
- Virus
- Worm

b) trojan programs:
- Backdoor
- Exploit

- Rootkit
- Trojan
- Trojan-ArcBomb
- Trojan-Banker
- Trojan-Clicker
- Trojan-DDoS
- Trojan-Downloader
- Trojan-Dropper
- Trojan-FakeAV
- Trojan-GameThief
- Trojan-IM
- Trojan-Mailfinder
- Trojan-Notifier
- Trojan-Proxy
- Trojan-PSW
- Trojan-Ransom
- Trojan-SMS
- Trojan-Spy

c) suspicious packers:
- MultiPacker
- RarePacker
- SuspiciousPacker

d) malicious tools:
- Constructor
- DoS
- Email-Flooder
- Flooder
- HackTool
- Hoax
- IM-Flooder
- SMS-Flooder
- Spoofer
- VirTool

Viruses and worms[1] are malicious programs that self-replicate on computers or via computer networks without the user being aware. Each subsequent copy is also able to self-replicate. The main characteristic used to determine whether a program is classified as a separate behaviour within the Viruses and Worms subclass is how the program propagates.

Most known worms are spread as files sent as email attachments, via a link to a web or FTP resource, via a link sent in an ICQ or IRC message, via P2P file sharing networks etc. Some of them spread as network packets; these directly penetrate the computer memory, and the worm code is then activated.

Viruses can be divided in accordance with the method used to infect a computer:

- file viruses
- boot sector viruses
- macro viruses

1 https://encyclopedia.kaspersky.com/knowledge/viruses-and-worms/

- script viruses

Any program within this subclass can have additional Trojan functions.

Trojans [2]are malicious programs that perform actions which are not authorized by the user: they delete, block, modify or copy data, and they disrupt the performance of computers or computer networks. Unlike viruses and worms, the threats that fall into this category are unable to make copies of themselves or self-replicate. Trojans are classified according to the type of action they perform on an infected computer.

Suspicious packers[3] occurs after using a variety of methods combined with file encryption in order to prevent reverse engineering of the program and to hinder analysis of program behaviour with proactive and heuristic methods.

Malicious tools[4] are malicious programs designed to automatically create viruses, worms, or Trojans, conduct DoS attacks on remote servers, hack other computers, etc. Unlike viruses, worms, and Trojans, malware in this subclass does not present a direct threat to the computer it runs on, and the program's malicious payload is only delivered on the direct order of the user.

Three types of software which cannot be fully assigned to malware are listed below:

a) RiskWare
   - Client-IRC
   - Client-P2P
   - Client-SMTP
   - Dialer
   - Downloader
   - FraudTool
   - Monitor
   - NetTool
   - PSWTool
   - RemoteAdmin
   - RiskTool
   - Server-FTP
   - Server-Proxy
   - Server-Telnet
   - Server-Web
   - WebToolbar

b) PornWare
   - Porn-Dialer
   - Porn-Downloader
   - Porn-Tool

c) Adware

Riskware[5] covers legitimate programs, which can cause damage when they fall into the hands of malicious users (and are used to delete, block, modify, or copy data, or disrupt the performance of computers or networks). These programs include remote administration utilities, IRC clients, dialler programs, file downloaders, software for monitoring computer activity, password management utilities, and numerous Internet server services such as FTP,

2 https://encyclopedia.kaspersky.com/knowledge/trojans/
3 https://encyclopedia.kaspersky.com/knowledge/suspicious-packers/
4 https://encyclopedia.kaspersky.com/knowledge/malicious-tools/
5 https://encyclopedia.kaspersky.com/knowledge/riskware/

web, proxy and telnet. Riskware is not malicious by nature although they do have functions that can be used for malicious purposes.

PornWare [6]can be installed on a user's computer by malicious users by exploiting operating system or browser vulnerabilities, or by using Trojans such as Trojan-Downloader and Trojan-Dropper. This is usually done to push advertisements for fee-based pornographic websites and services that a typical user might otherwise not be aware of.

Adware[7] covers programs designed to display advertisements, redirect search requests to advertising websites, and collect marketing-type data about the user in order to display customized advertising on the computer. In addition to displaying advertisements, many advertising systems also collect data about the computer and the user, such as:
the computer's IP address
the operating system and browser version
a list of the most frequently visited sites
search queries
other data that may be used to conduct subsequent advertising campaigns
If Adware does not notify the user that it is gathering information, then it is classified as a malicious program, specifically covered by the Trojan-Spy behaviour.

The classification described above is based on the propagation methods used, whereas other malicious programs are classified according to what actions they perform. However, these characteristics are often not enough for analysts to identify a trend in malware evolution, which is why additional factors are also used to classify objects.

Alternative classification:

- Crimeware
- Spyware
- Ransomware
- Bot-clients

Crimeware[8] is any computer program or set of programs designed expressly to facilitate illegal activity online. Common types:
pishing kit – a collection of tools assembled to make it easier for people with little technical skill to launch a pishing exploit. It usually includes web site development software, complete with graphics, coding and content that can be used to create convincing imitations of legitimate sites and spamming software to automate the mass mailing process
keylogger – surreptitiously installed to gather information illegally, it will record everything that is entered at the keyboard, including passwords and other privileged information

Spyware[9] is software that is installed on a computing device without the end user's knowledge. Such software is controversial because even though it is sometimes installed for relatively innocuous reasons, it can violate the end user's privacy and has the potential to be abused. Spyware that is installed for innocuous reasons is sometimes referred to as tracking software. In the workplace, such software may be installed on corporate laptops to monitor employees' browsing activities. In the home, parents might install a keystroke logger to

6 https://encyclopedia.kaspersky.com/knowledge/pornware/
7 https://encyclopedia.kaspersky.com/knowledge/adware/
8 https://searchsecurity.techtarget.com/definition/crimeware
9 https://searchsecurity.techtarget.com/definition/spyware

monitor their children's activity on the internet. Or an advertiser might use cookies to track what webpages a user visits in order to target advertising in a contextual marketing campaign.

Ransomware[10] is a subset of malware in which the data on a victim's computer is locked, typically by encryption and payment is demanded before the ransomed data is decrypted and access returned to the victim. The motive for ransomware attacks is nearly always monetary and unlike other types of attacks, the victim is usually notified that an exploit has occurred and is given instructions for how to recover from the attack. Ransomware malware can be spread through malicious email attachments, infected software apps, infected external storage devices and compromised websites. A growing number of attacks have used remote desktop protocol and other approaches that don't rely on any form of user interaction.

Bot-clients[11] cover malicious programs used to unite infected computers into a botnet (bot network, aka zombie network). This gives malicious users centralized control over all infected machines without the users knowing. For instance, botnets may be created to mass mail spam and conduct DDoS attacks

History of malware:

History of malicious software is long and shows how it has changed with a flow of time and how it may look like in the future.

First computer virus is controversial topic. The idea of a computer virus extends back to 1949, when early computer scientist John von Neumann wrote the "Theory and Organization of Complicated Automata", a paper that postulates how a computer program could reproduce itself.

In the 1950s, employees at Bell Labs gave life to von Neumann's idea when they created a game called "Core Wars." In the game, programmers would unleash software "organisms" that competed for control of the computer.

The early years of true malware[12]:

Early malware was primitive, often spreading entirely offline via floppy discs carried from computer to computer by human hands. As networking and the Internet matured, malicious software authors were quick to adapt their code and take advantage of the new communication medium. Below is a representative list of some significant early versions of malware and how they impacted the world:

- **1971 Creeper:** An experiment designed to test how a program might move between computers.
- **1974 – Wabbit:** A self-replicating program that made multiple copies of itself on a computer until it bogs down the system to such an extent that system performance is reduced and eventually crashes. Researchers named this virus "wabbit" (rabbit) because of the speed at which it was able to replicate.
- **1982 – Elk Cloner:** Written by a 15-year-old, Elk Cloner is one of the earliest widespread, self-replicating viruses to affect personal computers. Elk Cloner displayed

10 https://searchsecurity.techtarget.com/definition/ransomware
11 https://encyclopedia.kaspersky.com/knowledge/alternative-classifications/
12 https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/#_edn1

a friendly little poem on the infected system: “It will get on all your disks; It will infiltrate your chips; Yes, it’s Cloner!”

- **1986 – Brain Boot Sector Virus**: Generally regarded as the first virus to infect MS-DOS computers. Its origin stems from two brothers in Pakistan who created it to test loopholes in their company’s software.
- **1986 — PC-Write Trojan**: Malware authors disguised one of the earliest Trojans as a popular shareware program called “PC-Writer.” Once on a system, it would erase all of a user’s files.
- **1988 — Morris Worm:** This worm infected a substantial percentage of computers connected to ARPANET, the forerunner of the Internet, essentially bringing the network to its knees within 24 hours. Its release marked a new dawn for malicious software. The author, Robert Morris, became the first malware author convicted for his crimes.
- **1991 — Michelangelo Virus**: It was so named because the virus was designed to erase information from hard drives on March 6th, the birthday of the famed Renaissance artist. The virus was at the centre of a wild media storm with panicked reporters claiming that it had infected millions of computers and that the world would see dire consequences on March 6th. In reality, the virus only impacted about 10,000 systems, but the hype significantly raised public awareness of computer viruses.
- **1999 — Melissa Virus**: Generally acknowledged as the first mass-emailed virus, Melissa utilized Outlook address books from infected machines, and mailed itself to 50 people at a time.

Development of malware[13]:

Between 2000 and 2010, malware grew significantly, both in number and in how fast infections spread. At the start of the new millennium, Internet and email worms were making headlines across the globe. Later, we witnessed a dramatic increase in malware toolkits, including the now infamous Sony rootkit, which was instrumental in malware authors including rootkits in most modern malware. Crimeware kits aimed specifically at websites also rose in popularity, and the number of compromised websites escalated correspondingly. SQL injection attacks became a leading threat, claiming popular victims such as IKEA.

Here’s a summary of some of the significant malware released between 2000 and 2010:

- **2000 – ILOVEYOU Worm:** Spreading by way of an email sent with the seemingly benign subject line, “ILOVEYOU,” the worm infected an estimated 50 million computers. Damages caused major corporations and government bodies, including portions of the Pentagon and British Parliament, to shut down their email servers. The worm spread globally and cost more than $5.5 billion in damages.
- **2001 – Anna Kournikova Virus**: Emails spread this nasty virus that purported to contain pictures of the very attractive female tennis player, but in fact hid the malicious malware.
- **2003 – SQL Slammer Worm:** One of the fastest spreading worms of all time, SQL Slammer infected nearly 75,000 computers in ten minutes. The worm had a major global effect, slowing Internet traffic worldwide via denial of service.
- **2004 – Cabir Virus:** Although this virus caused little if any damage, it is noteworthy because it is widely acknowledged as the first mobile phone virus.

13 https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/

- **2005 – Koobface Virus:** One of the first instances of malware to infect PCs and then propagate to social networking sites. If you rearrange the letters in “Koobface” you get “Facebook.” The virus also targeted other social networks like MySpace and Twitter.
- **2008 – Conficker Worm:** A combination of the words “configure” and “ficker”, this sophisticated worm caused some of the worst damage seen since Slammer appeared in 2003.

Sponsored and profitable malware:

After 2010 malware evolve again. Some people found malicious software profitable, so they started sponsoring development teams. Such workgroups also evolve today, developing advanced malware with evasion tactics that outsmart many conventional anti-malware systems. Infiltrating factories and military systems became a common reality what led to cyber wars and other serious troubles.

Malware which had a major impact between 2010 and today[14]:

- **2010 – Stuxnet Worm:** Shortly after its release, security analysts openly speculated that this malicious code was designed with the express purpose of attacking Iran's nuclear program and included the ability to impact hardware as well as software. The incredibly sophisticated worm is believed to be the work of an entire team of developers, making it one of the most resource-intensive bits of malware created to date.
- **2011 — Zeus Trojan:** Although first detected in 2007, the author of the Zeus Trojan released the source code to the public in 2011, giving the malware new life. Sometimes called Zbot, this Trojan has become one of the most successful pieces of botnet software in the world, impacting millions of machines. It is often used to steal banking information by man-in-the-browser keystroke logging and form grabbing.
- 2**013 – Cryptolocker:** One of many early ransomware programs, Cryptolocker had a significant impact globally and helped fuel the ransomware era.
- **2014 – Backoff:** Malware designed to compromise Point-of-Sale (POS) systems to steal credit card data.
- **2016 – Cerber:** One of the heavy-hitters in the ransomware sphere. It's also one of the most prolific crypto-malware threats. At one point, Microsoft found more enterprise PCs infected with Cerber than any other ransomware family.
- **2017 – WannaCry Ransomware:** Exploiting a vulnerability first uncovered by the National Security Agency, the WannaCry Ransomware brought major computer systems in Russia, China, the UK, and the US to their knees, locking people out of their data and demanding they pay a ransom or lose everything. The virus affected at least 150 countries, including hospitals, banks, telecommunications companies, warehouses, and many other industries.

Even today malware is evolving. These days not only computers are at risk but also anything with a microprocessor. Researchers demonstrated that targets like automobiles, water system or airliners can also be infected. According to CNBC, cyberattacks are the fastest growing crime in the United States. They are evolving with every day and new tactics are being developed.

Newest malware attacks[15]:

The way that users are interacting with their networks, data, and the world around them has changed–and with these changes have come several new malware infection vectors. The Internet of Things has created a variety of new endpoints, which may connect

14 https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/
15 https://www.lastline.com/blog/malware-attack-vectors/

directly to a corporate network in the form of anything from a network printer to a “smart” light bulb.

Here are some of the ways in which old attack vectors are changing and some of the new vectors that have emerged recently:

- **Drive-by Downloads:** With a drive-by download, users simply need to view a website to trigger a malware download, which can occur without their knowledge. Drive-by downloads take advantage of vulnerabilities within a web browser, using JavaScript and other browsing features to inject malicious code. Nearly every application out there has security holes, including those that are used to browse the web. Having outdated software solutions, such as old browsers or old operating systems, can significantly increase the risk of drive-by downloads. But drive-by downloads are also evolving, by now including more advanced attacks –such as sophisticated ransomware packages–that can propagate swiftly across a network once downloaded.
- **Ad-Based Malware:** Malware embedded in advertisements has always been common, as third-party networks are often the driving force of many ads. Simply viewing a malicious ad could be enough to inject malicious code into an unprotected device. These ads are often distributed and viewed on large, trusted websites, which makes many users potentially vulnerable. Malicious ads can also be directly embedded into apps and served through apps that are otherwise trusted. In recent years, many of these malicious ads have been used to mine cryptocurrency (in other words, to generate digital currency that can then be used by hackers). The popularity of cryptocurrency has bolstered the spread of ad-based malware, which in turn takes advantage of the user’s computer resources and may impact the stability of their system.
- **Mouse Hovering:** A fairly new technique, mouse hovering takes advantage of a vulnerability in some well-known solutions such as PowerPoint. When a user hovers over a link to see where it goes, shell scripts can be automatically launched. This is especially insidious as many users have been taught to hover their mouse over links rather than click them to make sure the link is safe.
- **Scamware Disguised as Malware Protection:** Scamware disguised as malware protection is not new, but the techniques being used are growing more advanced. In the past, this attack vectors generally targeted users who were not computer savvy, showing ads that indicated their computer had been compromised, and demanding payment to unlock their computer. This was a prototypical form of modern ransomware. But in 2017 we also saw vulnerabilities that were introduced by CCleaner, a popular computer cleanup utility that was also used to fight malware. Hackers were able to inject malware into the product so effectively that the product was distributed by the original manufacturer in this format. Millions of users were impacted by this attack, which came from a theoretically trusted resource.

MALWARE INJECTION

Introduction

These days malware is nothing new and people know that we must defend from it. Creators of malicious software must think about new ways to go around security systems and how to inject their code. Common method is to hide malware is to inject its code to another process which is not suspicious. Such way is mostly used with writing exploits, but in case of malware, some mechanisms are similar. However, the purpose of malicious software is different. Its target is to hide behind trusted process and to continue its mission under cover.

Injection techniques

Necessary thing to make malware works, common for all techniques, is to inject a code to remote process and execute it. To do this, a loader is used. It's a designed application for this task. The way it works is simple:

- find existing process or create new in suspended state
- create handle for found process
- allocate memory to inject malicious code
- inject code
- forward entry address to the code
- wake up the thread containing injected code

Injecting code to the existing process:

Unstable program may stop working so that's something unwanted. Executing malicious code in new thread is the best way so adding it must be done carefully to leave program stable. First thing to do is to find a handle to the existing process.

Creating a handle:

- find running processes
- check how many processes were found
- check found processes
  - if process meets requirements create a handle to it

Injecting code to the newly created process:

Newly created process must be suspended so it will be mapped to the memory, but it won't execute yet. In that case, malicious code can be added. Creating such process:

- prepare structure, which will be filled with data
- use function to create new process

Putting code to the remote process:

To execute malicious code in another process it must be copied to memory space, allocated for this process. Allocating additional memory space or creating new section must be done. Examples to do so:

a) writing directly in remote section and providing handle to the chosen process:
   - allocate memory in remote processes
   - copying buffer to allocated memory

b) adding section to existing process and changing its mapping to remote process:
   - create new section
   - map this section to the memory of local processes
   - map the same section to the memory of remote processes
   - unmap section from the memory of local processes
   - return address of section

Forwarding to the injected code:

There are many methods of forwarding to the injected code and attackers still invents more and more. The most popular ones:

- executing added code in new thread
- adding to the existing thread
- overwriting entry point of process
- injection to the Tray window
- PowerLoader

Executing added code in new thread:

It is the easiest method. New thread can be created as suspended or to immediate executing. Some functions like *CreateRemoteThread* are commonly used so they are monitored by security systems. That's why malware creators usually use some alternatives. This method works both for injecting code to existing process and newly created. The difference is that during execution of new process, main thread can't be activated because it may lead to detection of malware. That's why this thread stays suspended. Usually, new thread in which injected code will be executed is created and later activated.

Adding to the existing thread:

Each thread has APC queue, which contains addresses of codes to execute when it receives a signal. Existing queue can be flushed, or new procedure can be added to it. To use this method simple steps must be done:

- add entry point of for example shellcode to the queue
- resume the thread

Overwriting entry point of process:

This method is complicated but very effective. If process is suspended it means that execution of its code wasn't started yet. Entry point wasn't reached yet so existing code can be overwritten and after resuming the process automatically executed. It also can be done with file on hard drive. Description of this method:

- search for headers with PE format in memory
- add information about results to the structure
- read gathered information (*Process Environment Block* should be gathered)
- read address of remote module (PEB + 8) – it is address where PE file was loaded
- read PE headers
- read entry point for code of PE file
- overwrite the code at entry point
- resume the process

Similar method much easier is overwriting the process context. This informs process that entry point is in malicious code. As before process must be suspended and not executed.

Steps to do this:

- take entry context of the process
- take entry point address from EAX register
- overwrite original address with address to malicious code
- set context with changed value

Injection to the Tray window

This method is complicated and sometimes not stable. It overwrites callback, which occurs every time when signal receives the window. Forwarding must have specific form so besides code, two additional addresses must be added to memory. Another problem of this method is that injected code will be executed every time the signal receives the window. It can be stopped by restoring the original state after first execution, but it's another complication.

Example of procedures for this method:

- find running process with Tray window
- take PID of the process with window
- save current address to restore it later
- create handle
- insert shellcode and right addresses to remote buffer
- set address in window
- send signal to the window to execute injected code

PowerLoader[16]

It's much different and more advanced method than others. It uses trusted process which is "explorer.exe". It gets the malicious code into the explorer by opening an existing, shared section already mapped into it. This fact allows to skip allocating heap space or overwriting the process memory. This section is not executable so to use this method ROP

16 https://www.malwaretech.com/2013/08/powerloader-injection-something-truly.html

(Return Oriented Programming) technique had to be used. It's also usually used with exploits. PowerLoader uses many tricks and advanced techniques so researchers are still developing it.

PE files injection methods

Except injecting code, malware creators also inject whole PE files. Like with code, there are many methods to do it. They vary by efficiency and implementation difficulty. The main thing is to avoid typical system function because they are usually monitored by security systems.

Below some injecting methods are listed:

- DLL Injection
- RunPE
- ChimeraPE
- Reflective DLL injection (Reflective Loader)

DLL Injection

The easiest method to inject PE file is classic DLL injection. There is no need to know PE format or to implement loading of its elements because all of this is done by system function. It forces process to load a dynamic-link library. From there, the entry point of the DLL will be executed by the operating system once it's loaded. It can be used for existing processes and newly created ones. This method is commonly known and easy to detect so it's not used too often.

How to make it works:

- allocate memory in remote processes
- add DLL path to the allocated memory
- get address of function which loads the libraries
- add new thread in remote process, which will execute loading libraries function

RunPE

This method is also called as **Process Hollowing**, because it lets replace module already loaded to memory. The idea is to create a new process in a suspended state, then unmap its memory and replace with malicious PE file.

It is easy to implement but also easy to detect. By comparing header of PE file loaded to memory and header of this file on disc, security system can detect differences and when they appear it is known that another process is using it.

ChimeraPE

It is like RunPE method but more advanced. The idea is to connect two ore more executable files for one process. Original PE file remains untouched so it can't be detected like RunPE. It can also infect memory of already running processes, not only newly created. This method is efficient but hard to implement as well. Import table must be filled by attacker so it's not easy task and additional restrictions occur because of that.

Creating Chimera:

- create new process or create a handle to existing one
- allocate memory in target process and actual process
- fill import table

- inject PE file to remote processes
- free the local buffer
- take entry point of injected PE file
- run injected payload in new thread

Reflective DLL injection

Very popular method among malware creators. A special scheme was created by Stephen Fewer (available on his GitHub[17]), which can be easily modified. This scheme allows creation of DLL file, which can be injected to other processes without additional work. This method allows the sourcing of the DLL in the form of its raw data. Because of that, to inject data into the process, the binary must be manually parsed and mapped into virtual memory. The process of reflective DLL injection:

- allocate memory in remote processes
- copy the DLL into the allocated memory
- rebuild import table
- parse base relocation table to fix addresses
- the mapped DLL is then written into the target process

Conclusion

There are a lot of techniques and methods to inject code or PE files. What is more, they are still developed. Many schemes are available on the Internet and can be easily modified. Learning about it all not only helps to analyse malware but also to understand how operating system executes and works with code.

17 https://github.com/stephenfewer/ReflectiveDLLInjection

MALWARE REVERSE ENGINEERING

Introduction

To create or analyse malware, extracting information from its binary code is needed. Reverse Engineering allows such analysis of program's components and behaviour. However, to understand all of this, knowledge about assembler is required.

In 2005 Eldad Eilam proposed two possible fields for reverse engineering:

- software development – where the source code is available but it's poorly documented
- software security – source code is not available, and reconstruction of analysed software is needed

Malware Analysis is an attempt to understand how it behaves and it has few stages:

- manual code reversing
- interactive behaviour analysis
- static properties analysis
- fully-automated analysis

This analysis has two main techniques:

- static analysis
- dynamic analysis

Static analysis

It's safe technique, because there is no need to run malicious code. It is about capturing notes during analysing a disassembled code. It's possible to see all the possible instructions and specific functions inside the code.

The problem with this technique is that analysing the binary all variables or data structures are not available. Also, malware creators try to hinder this kind of analysis.
Some useful ways of analysis

a) file comparison – fingerprint the file which will be examined and compare it after analysis

b) virus scan – investigated file may be already discovered and documented in virus scanner documentation

c) strings analyse – sometimes suspicious strings may appear and reveal the malicious code. However, drawing conclusions must be careful because it can be planted by attacker to deceive the analyst.

Dynamic analysis

It is a deeper form of analysation to understand hidden functionality not understood statically. More important is "what" malware is doing than "how" it's accomplished. It may be not that safe because to observe and analyse malicious code there is a need to run it.

During this technique, main things to observe are:

- registry activity
- file activity
- process activity
- network traffic

Closer look to malicious code

The main thing with malware is that it can't be terminated during its execution. Creators of malware use reverse engineering to manage memory addresses which are very important. If code face the memory address which can't be reached it will terminate.

Absolute memory address causes many problems. Address where malicious code is going to be injected usually is random, so if it won't be injected to proper base address, the absolute addresses won't have any sense.

Another problem maker are imported functions. To increase functionality of malware, external function must be used. To do this, library must be mapped to memory, where the address is known and can be accessed.

Shellcode

Shellcode is often used by creators of malware as a part of their payload. It usually used to import or run another malware's components. It can't be terminated.

Knowledge of assembler and reverse engineering is a key to create or analyse shellcode. It usually needs an information about the address where it was injected. Understanding registers like *EAX*, structures like *PEB* and instructions like *CALL*, *POP*, *PUSH*, *MOV* allows to obtain addresses, read headers or load something to registers and use it.

PE files

A lot of possibilities can be obtained with PE files. Most of malware is copied to other programs so it can work without being listed on processes list.

By default, Windows loader maps PE file which is run and add all dependencies. In case of manual mapping, all procedures must be implemented. What is more, if mapping is manual, much more advanced formats can be injected.

To avoid problems, injected payload should be as simple as possible. All Data Directory tables must be filled properly, so unnecessary ones shouldn't be added. Also, dependency from external libraries should be minimized. To make less calls from import table, everything what is possible to link, should be linked statically.

Another thing to minimize import table is late loading of libraries. It can be done from code level. Library *kernel32.dll* contains many functions which allows useful options, so the best way is to load this library and load them from code level.

Next problem is address of payload which is variable. Payload should contain relocation table to always find proper address. All DLL files contain it, but in EXE ones, it's optional.

Manual loading of PE file

As mentioned in previous section, more possibilities are obtained when PE file is loaded manually. However, during mapping it to memory some basic procedures must be done:

- conversion of raw image to virtual one
- dealing with relocation problems
- filling an import table

Again, to deal with this problem, knowledge about reverse engineering is needed. Understanding *PEB (Process Environment Block)* and *TIB (Thread Information Block)* is very important because they keep information about threads and processes which are already running.

From raw to virtual image

During memory allocation, the least operable size is one page (0x1000 bytes), raw size on the other hand is more compressed. Although content is the same, it may be different because of that fact. PE loader take information how to map sections from section header, that's why it's important to understand it.

Each section has own rights. Some are used to write, some to read or execute. The easiest way is to allocate one big memory area with all privileges and then copy sections to that area.

Tables addresses

Mapping dependencies requires dealing with tables. The main two types of them are import table and relocation table. Pointer to them can be found in PE header in *Data Directory* list.

Relocation table

Because of addresses randomization, some of them must be relocated to make payload working. They are listed in relocation table. All DLL files have such table, but in PE files case it's optional. Lack of it causes restrictions in injection so payload should contain it.

There are few methods of relocation but with most PE files, a bit field is used. It has 32-bit and 64-bit version. This format adds page address to shift address so RVA *(Relative Virtual Address)* of relocation can be calculated.

Import table

PE files needs dynamic libraries for proper working and import table contains list of them. Each library exports functions which are used in program. To call such functions a pointer with proper addresses are needed. Because manual loading of PE file requires filling this table it can be done in two ways: usage of payload loader or creating a payload which fills it manually. Second method is much harder because knowledge about reverse engineering must be much higher.

First way uses the fact that one libraries like *kernel32.dll* and *ntdll.dll* are always imported under the same address, even when program doesn't need them. Because of that, payload loader can contain those addresses because they don't change.

Second way gives more possibilities but is harder to implement. PEB *(Process Environment Block)* contains list of all modules loaded to memory. It can be read and search in order to find important addresses. This technique allows creating a file without import table.

Forwarding to injected PE file

Contrary to shellcode, entry point of PE file isn't at the beginning of allocated memory. It must be read from header and then forwarded to proper address. However, when it will be able to load it's all dependencies, its injection will be the same as in shellcode case. Again, it can be done in two ways: parsing PE file headers in order to find entry point by loader side or by modifying payload.

Conclusion

Reverse engineering is very important thing in analysing malware. Combining static and dynamic analysis provides a lot of information about malicious code. Debugging shows exactly what happens inside the program. It allows to understand how it works, how it was made and also how to modify it. Having knowledge about reverse engineering for sure will help to understand how to defend from malware as well as how to create it.

FUZZING

Introduction

Fuzz testing (fuzzing) is an automated technique for testing software input validation by providing invalid, unexpected or random data input. It has few associated tasks such as: target identification, crafting inputs, system automation, and monitoring results. It is used to find out exceptions like crashes, failing built-in code assertions, potential memory leaks or other bugs.

Background

Fuzzing has a long history and occurred to be effective for finding bugs. In 1988 it was used at the University of Wisconsin to test UNIX system utilities for faults. In modern information security it developed a lot and serves as a way for security professionals and developers to audit the input validation of software.

Main advantage of this technique is its automated nature. Fuzzer can be running and various other tasks can be done during that time. Further, developing a simple fuzzer does not require much time. Also, because of automatic process, bugs that can be overlooked during manual review may be found.

Successful Fuzzing

There are four main steps which must be done to achieve successful fuzzing:

1. Identifying a target
2. Generating inputs
3. Input delivery
4. Crash monitoring

First task is the most important and three others are dependent on it. After target has been selected, input generation can be achieved in different ways. Then it must be delivered to the target software depending on the chosen attack type and surface. Finally, to check if security issue was found, monitoring test results must be done.

Examples of Fuzzers:

Generating inputs is the basic of fuzz testing, it determines efficiency of fuzzer. Creating different data samples may be done in three main ways:

a) Alter existing data samples to create new test data – very simple method, usually small changes to existing inputs are done, so input may still be valid but exercise new behaviour.
b) Constructing test cases from RFC or specification, which describes the file format or network protocol, it performs better than previous method
c) Generating test cases from an existing one, the most successful fuzzer but it requires detailed knowledge of tested protocol.

Fuzz Testing Tools:

Different machines require different tools, some of them may be same but in general, computer tools are different than mobile one. Below is a list of some of them:

a) **Burb Suite** – written in Java and used for testing and fuzzing Web applications, that's why it can be used for PC and mobile security testing. It processes attacks to detect vulnerabilities such as buffer overflow, SQL injection, cross-site scripting etc.
b) **American Fuzzy Lop** – uses genetic algorithms to efficiently increase code coverage of the test cases. It has many features and is popular among security community. Program requires sample command that runs the tested application and at least small example input file. When it detects crash, special report is generated for further inspection.
c) **Peach Fuzzer** – more like fuzzing platform/framework. Powerful tool which provides a way to define the format of data that should be generated as well as how and when the fuzzed data should be generated. It enables to find known and unknown threads.
d) **Appium** – open source, cross-platform tool for testing mobile applications. There is no need to recompile app or modify it, due to use of standard automation APIs on all platforms. Any WebDriver-compatible language and testing framework can be used.