Algorithms of Information Security: Different types of malware and principles of analysis

Faculty of Information Technology
Czech Technical University in Prague

November 24, 2020

What is malware?

- The term malware is short for MALicious softWARE, and it refers to any software that does something that causes detriment to the user.
- Malicious activities may include disrupt a computer operation, gather sensitive information, or unauthorized access to a computer system or network.
- The problem with each definition of malware is that it is very broad and not rigorous.
- There is no definition of what precisely malware is.

Motivation

- If we do not know what is malware then is it possible to create some malware detection models with 100% of accuracy?
- Malware classification models are evaluated using labeled data from test set.
- To achieve perfect accuracy (for each user of some antivirus program), there must be a consensus among all users on the definition of malware.
- Since there is no such exact definition, the evaluation results of each malware detection model is only relative to the labels of samples from test set.
- An example of programs where consensus is not achieved is adware, which is sometimes referred to as malware and sometimes as benign files (someone may like an advertisement)

Types of malware

- Malware classification can take into account the purpose of malware or its functionality.
- However, classification based on malware's functionality may not be possible since malware can have many functionalities.
- For example, malware can behave like a worm (i.e., scans the network and exploits vulnerabilities) and can download another malware component such as backdoor.
- As a result, the following types of malware are not mutually exclusive, and they are solely intended to familiarize the reader with the various types of malware.

Types of malware

- **Virus** is a malware that is attached to a host file. Virus depends on human activity and when such an infected host is run then virus is activated and performs some malicious activity and spread itself to other computers.
- **Worm** is similar to virus, however, it does not need a host file nor human activity to activate itself. Worm is able to self-replicate and can spread itself to other computers.
- **Trojan horse** disguises itself as benign software, however, perform malicious activities in the background.

Types of malware

- **Spyware** retrieves sensitive data from a victim's system and sends them to the attacker. Such sensitive data may include passwords, credit card numbers, history of visited web pages, emails, and various documents. An example of spyware is keylogger that captures keystrokes and transfer them to the attacker. Another example of spyware is a sniffer that monitors internet traffic.
- **Rootkit** is used to modify the operating system so that an attacker can keep administrator privileges. The characteristic of rootkit is to conceal its presence or the presence of other malicious programs.

Types of malware

- **Ransomware** is type of malware that locks victim's screen and encrypts chosen types of files (such as popular .xsl, .docx, .txt, .sql, .jpg, .cpp, .mp3, and many others). The AES, RSA, and Elliptic curve cryptography are the most common encryption transformations. The attacker then demands a ransom (usually paid in bitcoins) for providing the decryption key.
- **Downloader**, also called dropper, is designed to download and install additional malware or malicious components.

Types of malware

- **Bot** is a malware that allows the attacker (called bot master) to control the compromised system remotely. A Group of interconnected bots remotely-controlled by a bot master using command and control (C&C) software is called a botnet. Usual malicious activities of a botnet are DDoS attacks, spyware activities, sending spam emails, or distributing other malware.
- **Backdoor** is classified as a Trojan horse and it is designed to enable the attacker to get access to a system and execute commands.

Potentially unwanted program

- Potentially unwanted program (PUP) are specific type of software that is considered as a gray area among malware and benign files.
- The intent of PUP may be unclear, or for some user the benefit of PUP may outweigh the potential risk.
- Antivirus vendors deal with PUPs as a lower-risk category. Some AVs allow users to decide whether PUPs will be detected or not on their system.
- An example of PUP is an adware that displays advertisements, often in the form of pop-up messages.
- Another examples of PUP are toolbars, extensions, or plugins installed on the user's browser.
- Note that adware is often considered as malware; however, this statement is dependent on the definition of malware, which is not fixed.

Malware families

There are many malware families. For illustration, here are five prevalent ones:

- **Allaple** – a polymorphic network worm that spreads to other computers and performs denial-of-service (DoS) attacks.
- **Dinwod** – a trojan horse that silently downloads and installs other malware on the compromised computer.
- **Virlock** – a ransomware that locks victims' computer and demands a payment in order to unlock it.
- **Virut** – a virus with backdoor functionality that operates over an IRC-based communications protocol.
- **Vundo** – a trojan horse that displays pop-up advertisements and also injects JavaScript into HTML pages.

Static versus dynamic analysis

- static analysis
  - analysis of the file format without actually running the program
- dynamic analysis
  - it aims to examine a program which is executed in a real or virtual environment
  - not feasible when dealing with a high number of malware samples
- hybrid analysis
  - it combines both static and dynamic approaches

Signature-based detection

- Most AV rely primarily on a signature-based detection technique which is relatively simple and efficient rule-based method for detecting known malware.
- Signature (sequence of bytes, hash values, strings) of the malware is extracted and added to the database.
- The antivirus engine compares the contents of a file with all malware signatures in its database and if a match is found, the file is reported as malware.
- Signature should not be too generic nor too specific.
- A good signature must capture malware with a minimal false positive probability.

Machine learning-based detection

- data extraction (static and/or dynamic)
- data preprocessing and normalization
- feature selection
- machine learning algorithms (classification or clustering)
- evaluation of performance

Detection techniques - properties

- signature-based detection
  - relatively simple and efficient rule-based method
  - low FP probability
  - inability to detect obfuscated and zero-day malware

- non-signature detection techniques based on ML algorithms
  - able to detect previously unknown or obfuscated malware
  - does not need human to create any kind of signature
  - prone to have high false positive rate

Types of data for malware analysis

- **Byte sequences** - (static analysis): bytes sequences represent the malware's binary content. By treating a file as a sequence of bytes, byte n-grams are extracted by looking at the unique combination of every n consecutive bytes as an individual feature.
- **API & system calls** - (static or dynamic analysis): Application Programming Interfaces (API) functions and system calls are related to services provided by the operating systems such as networking, security, file management, and so on. The invocation of particular API functions provides key information to represent the behavior of malware.

Types of data for malware analysis

- **opcodes** - (static analysis): Opcodes (or operational codes) can act as a predictor for detecting obfuscated or metamorphic malware. We can extract the opcode sequences and their frequency of appearance and use some opcode-based similarity measure.
- **network** - (dynamic analysis): Detecting malicious traffic on a network can uniquely provide specific insights into the behavior of malicious programs. As soon as malware infects a host machine, it may establish communication with an external server to obtain the commands to execute on the victim or to download updates, other malware or to leak private and sensitive information of the user/device.

- **PE file characteristics** - (static analysis):
  - A PE file consists of headers and sections that encapsulate the information necessary to manage the executable code.
  - The PE file header provides all the descriptive information concerning the locations and sizes of structures in the PE file to the loader process.
  - The header of a PE file consists of the DOS header, the PE signature, the COFF file header, the optional header, and the section headers.
  - The optional file header is followed immediately by the section headers which provide information about sections, including locations, sizes, and characteristics.
  - Sections divide the file content into code, resources, and various types of data.

Types of data for malware analysis

- **Strings** - (static analysis): String analysis refers to the extraction of every printable string within an executable or program. The information that may be found in these strings can be, for instance, URLs that the program connects to, file locations or filepaths of files accessed/modified by the program, names of the menus of the application, etc.
- **Entropy** - (static analysis): files with segments of code that have been compressed or encrypted tend to have higher entropy than native code. The entropy of a bytes sequence reflects its statistical variation. High entropy value indicates that the chunk consists entirely of distinct values - it detects the presence of encrypted or compressed segments of code.

Types of data for malware analysis

- **Image** - (static analysis): visualization of the malware's binary content as a gray scale image. This is achieved by interpreting every byte as one pixel in an image, where values range from 0 to 255 (0:black, 255:white). Afterwards, the resulting array is reorganized as a 2-D array. Drawback - techniques like encryption and compression might completely change the bytes structure of a binary program.
- **Function call graph** - (static analysis): A Function Call Graph is a directed graph whose vertices represent the functions of which a software program is composed, and the edges symbolize function calls. A vertex is represented by either local functions implemented by the programmer to perform specific tasks or external functions provided by the OS and external libraries. Function call graphs are generated from the static analysis of the disassembly file.

Types of data for malware analysis

- **Control flow graph** - (static analysis): A Control Flow Graph (CFG) is a directed graph in which the nodes represent basic blocks and the edges represent control flow paths. A basic block is a linear sequence of program instructions having an entry point (the first instruction executed) and an exit point (the last instruction executed). A CFG is a representation of all the paths that can be traversed during a program's execution.
- **Memory** - (dynamic analysis): The behavior of a computer program can be represented by the values of the memory contents at runtime. In other words, values stored in different registers while a computer program is running can distinguish benign from malicious programs.

Types of data for malware analysis

- **Instruction Traces** - (dynamic analysis):
  - A dynamic instruction trace is a sequence of processor instructions called during the execution of a program.
  - Contrary to the static instruction trace, dynamic traces are ordered as they are executed while static traces are ordered as they appear in the binary file.
  - Dynamic traces are a more robust measure of the program's behavior, since code packers and encrypters can obfuscate and hinder the code instructions from static analysis.

Malware Obfuscation Techniques

- Malware authors often use various techniques to modify malware to make it more difficult to detect them.
- These kinds of techniques are called obfuscation techniques, and they are used to protect the malware from malware analysts and reverse engineers.
- Obfuscation Techniques make difficult to extract strings from binary.
- These techniques successfully avoid signature-based detection. Obfuscation can be applied on several layers, such as code, a sequence of instructions, or binary.
- Among the most popular obfuscation techniques belong packing, encryption, polymorphism, and metamorphism.

Malware Obfuscation Techniques

- **Packing** is a process that uses compression (one or more layers) to obfuscate an executable file. As a result, a new executable file with obfuscated content is created. The unpacking process consists of a decompression routine that retrieves the original file (or code).
- **Encryption** is similar to packing; however, encryption is used instead of compression. Encrypted and packed malware must contain a decryption module, which can be leveraged for signature-based detection.

Malware Obfuscation Techniques

- **Polymorphism** use encryption, and besides, the decryptor module is morphed, and as a result, exhibits no signature. However, polymorphic malware still needs to be decrypted, and the original (non-obfuscated) code can be used for signature/based detection.
- **Metamorphism** is the most advanced obfuscation technique and also the most challenging for malware authors. Metamorphic malware changes internal structure while maintaining its original functionality.

Polymorphic and metamorphic malware have the ability to change their code with each new generation. Once such kind of malware is executed, it can be obfuscated again to avoid signature-based detection.

Infection Vectors

- **Exploiting Vulnerable Services over the Network** - network services running on a server provide shared resources and services to clients in a network.
- **Drive-by Downloads** - Drive-by downloads usually target a victim's Web browser. By exploiting a vulnerability in the Web browser application, a drive-by download is able to fetch malicious code from the Web and subsequently execute it on the victim's machine.
- **Social Engineering** - All techniques that lure a user into deliberately executing malicious code on her machine, possibly under false pretenses.