FUNDAMENTALS OF STATIC MALWARE ANALYSIS: PRINCIPLES, METHODS AND TOOLS

Andrej FEDÁK, Jozef ŠTULRAJTER

Abstract: Nowadays, the security of all systems connected to the public network is severely tested. Most users try to protect themselves against many abusive practices by using many security tools to keep their privacy safe. Information technology security involves many branches that address the prevention and protection against malicious software. One of those branches is the analysis of malicious files, specifically we will focus on the static analysis of malware. In static analysis, a suspicious sample is not executed and observed as in dynamic analysis, but many tools and methods are used to extract meaningful character strings from sample, data from the header of executable file format, information about the type of compression, the type of compiler used to create the file, and last but not least the application code. This work provides an initial insight into the complex subject of static analysis.

Keywords: Forensic analysis; Static analysis; Malware; Portable executable; String; PE header; Extractor; Obfuscation; Compression.

1 INTRODUCTION

The aim of this work is to describe the basic tools and methods used in the analysis of malware. Malware analysis is a large part of Information Technology (abbr. IT) security that is aimed at preventing the spread of malicious software. It analyses individual components of malware as well as the behaviour of malware in the infected computer. The main task of the analysis is to find out what functionality a given malware has, i.e. what it does and can do under what conditions. It is able to prevent on-coming computer attacks by detecting the way how malicious code get in your computer. For this purpose, analysts are offered a large number of methods and tools to analyse samples of malicious software. An important part of the analysis is also obtaining a sample of specific malware, which is usually in the form of an executable file (in the Windows operating system we can talk about files in Portable Executable format). Finally, a report of complete analysis should contain all the relevant information about the malware that was collected during the analysis [4].

2 MALWARE ANALYSIS

Methods of analysing malware can be divided into two main branches - static and dynamic methods. While both types of analysis have the same goal of finding out how a given malware works, the tools, time, and experience needed to perform the analysis are different. The basic difference between these methods is that in a static analysis, a given sample is not executed, whereas in a dynamic analysis this is necessary. Detailed static analysis of the program involves the use of a disassembler to allow subsequent analysis of the internal logic of the software using the exposed code. Dynamic analysis executes malicious code in a controlled environment that closely monitors its behaviour. When performing analysis of malicious software, pieces of information from static and dynamic analysis complement each other and help to get a complete picture of the malware.

Generally, static analysis is the analysis of computer software that is performed without the need of executing programs. This analysis describes the data structure of the program or the process of analysing the code. Thanks to this, it is possible to determine some functions of the analysed software. Some of the static analysis methods are considered to be the primary analysis of malware. Basic static analysis provides information on whether a file is considered to be harmful, processes data from the file header (e.g. date of its creation) or provides a list of strings used in the code (from libraries to Internet Protocol addresses), but of course only if those parts of the code are not obfuscated. This analytical method is very fast and straightforward and quickly helps us to get familiar with the basic functionality of the file [4].

After the basic analysis, the acquired base of knowledge can be further expand using methods and tools of advanced analysis. The most detailed, complex and time consuming method is the analysis of the program code itself. This method consists of decompilation the machine code of application into the lowest-level programming language - assembly language, or in some cases a higher programming language or pseudocode, but there is a major problem with the code reconstruction, because the high level of abstraction sometimes makes the code unintelligible. Because of their nature, languages using intermediate representation (Java, C#) allow decompilation to a much simpler form. There are also some simplifications for other languages (e.g. IDA with HexRays decompiler). This is followed by an analysis of the program code, estimation of its functionality and extraction of additional data from the program code. Even though the analyst has an executable program or any part of it, he does not have the source code, so he sees only what is going to be executed at the processor level, but not the high-level concepts that author of the code actually used [3], [4].

One of the main differences between static and dynamic analysis is that static analysis is somewhat safer than dynamic one because it does not directly execute malicious code. Therefore, we do not need to worry too much about becoming a victim of dangerous malware techniques. The risk of accidental execution of malware can be further reduced by using a virtual machine (VMware, VirtualBox, etc.), by analysing malware on an operating system for which it was not made or by increasing the level of User Account Control (confirmation is required to run the program). Another advantage of the static analysis is the possibility to detect potential functions of malicious software that may not be found during dynamic analysis. Although the static analysis is more thorough, it is also more time-consuming. Many methods used in static analysis increase the time required to analyse code. Nowadays, almost every malware is obfuscated, which means that parts of the program are replaced by another functionally equivalent parts that are encoded, compressed or intentionally extended with random or confusing code. Because of this, security teams do not use such detailed analysis when dealing with a large number of incidents. Limited capabilities, resources and time do not allow each incident to be resolved by slow methods and therefore security teams tend to use automatic, partially less informative methods. And even after a comprehensive analysis of the code, they may not be able to identify all the functions that the software could potentially perform (for example external communication with websites, servers or receiving encryption keys from the environment) [4], [5].

3 ONLINE ANTIVIRUS SCAN

The first step in analysing files is to make sure that sample is perceived as malicious code using available antivirus (abbr. AV) tools. Online multi-AV scanners provide a quick and clear picture of an unknown file that can be potentially dangerous for us. In many cases the use of these services is very easy because of the intuitive and user-friendly interface. Some online scanners allow their services to be used with their own tools and scripts that allow the user to automate and speed up repetitive tasks.

Before we begin, the risks associated with using these services should be understood. False positives and false negatives will always be a problem. Even if 100 % of antivirus products indicate that a file is safe, that doesn't necessarily mean the file is safe. This can also be applied the other way around. In addition, if a private instance of the service does not start, files that have been uploaded to public websites may be automatically shared with other resellers and third parties. This is generally good because the vendors need samples to build new signatures. However, targeted malware may contain hard-coded usernames, passwords, domain names, or Internet Protocol addresses (abbreviated IP) of internal systems that should not be distributed to suppliers and possibly to the public. [1]

Probably the best known online tool for analysing malware is VirusTotal. This tool allows you to upload a dangerous file, check a suspicious Uniform Resource Locator (abbr. URL), search for an already uploaded file using a hash and so on. Then it can perform automatic forensic analysis on the uploaded file using more than 60 antivirus engines as shown in Fig. 1. The result of such scan are simple pieces of information quickly obtained by many methods of static and dynamic analysis. On the other hand, the disadvantage of this tool is its closed source code. Similar features are provided by other online scanners such as VirSCAN and Jotti [3], [6].

As previously mentioned, the use of multi-AV online service is quite simple. All you need to know is the URL of a specific online tool (eg Www.virustotal.com, www.virscan.org or virusscan.jotti.org) and after opening the website, the suspicious file only needs to be dragged to the website or the full path to the file which will be recorded and scanned. These online scanners provide sophisticated scripts and custom applications for their faster use and automation of certain tasks. The script called virt.py created by Xiaokui Shu was used to illustrate the use of VirusTotal service. By modifying the registry in Windows, this auxiliary batch script has been added to the right-click context menu:

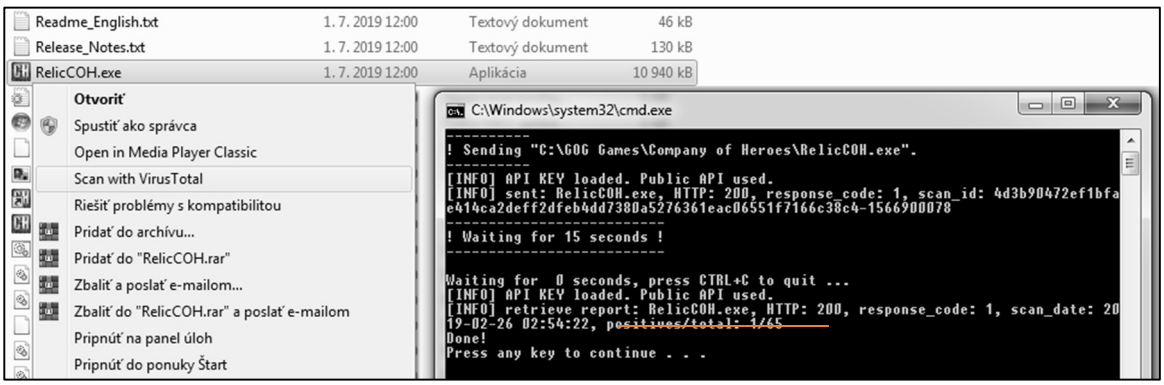

Fig. 1 Using the online VirusTotal service with the script
Source: [3]

```
REG ADD "HKEY_CLASSES_ROOT\*\shell\Scan with VirusTotal"
REG ADD "HKEY_CLASSES_ROOT\*\shell\Scan with VirusTotal\command" /t REG_SZ /d "\"%CD%\script_check_file.bat\" \"%%1\""
```

The code of the auxiliary batch script looks like this:

```
REM Enter the directory which contains our scripts
cd /d "%~dp0"
REM Execute the script with the parameter -s (send file) and the input data
python.exe virt.py -s %1
REM Wait for the online scanner to process the file
timeout /T 15 /NOBREAK
REM Execute the script with the parameter -r (retrieve report) and the input data
python.exe virt.py -r %1
```

A community that uses online multi-AV scanner services is raising its global level of IT security by sharing results of scanned malicious files and URLs. However, such openness to the community is also a major disadvantage of the online scanner, what makes it useless in some cases. Specifically, the biggest problem is the fact that all users may retrieve a report of any sample at any time. Authors usually modify their malware to have a unique hash fingerprint (no sample with that fingerprint has yet been analysed by VirusTotal). And when the analyst uploads the sample to VirusTotal, the author of malware immediately learns that his malware was found and is being analysed by a forensic analyst. Because of this, an attacker may change the behavioural strategy, turn off the sample and so on. Although the tool provides helpful features and integrates many analytical methods and tools, it is not advisable to use VirusTotal during an analysis conducted by a security team such as Computer Security Incident Response Team (abbr. CSIRT) [1], [3].

4 EXTRACTION OF STRINGS

Extraction of strings (a sequence of Unicode and ASCII characters - American Standard Code for Information Interchange) from the suspicious software is another method used by analysts when analysing malicious files. This extraction is probably the simplest method by which it is possible to reveal some features of the program. This method tries to find meaningful text strings in binary files that create a sequence of bytes with values in the range of printable characters ending with the byte of zero value. It is basically a trivial data mining from the binary files that can often be quite effective. It is a source of a huge number of artefacts, some of which may be crucial for forensic analysis. Such crucial artefacts include various strings such as IP addresses and URLs with which the malware is able to communicate, registry keys with values, commands that malware uses for communication over the Internet (for example the Internet Relay Chat protocol is easily recognized by its text commands), file names and file paths that the malware works with, or decryption keys for the encrypted parts of the code. Although strings do not give a clear picture of the purpose and capability of a file, they can give a hint about what malware is capable of doing [4].

This approach will not work with encrypted strings and the output may additionally contain a significant amount of strings that do not represent any meaningful information. Malware authors often use tools and methods to prevent reverse engineering and encoding or compression to make the analysis and detection more complicated. A software without malicious code almost always contain a large number of strings, while compressed malware has only a few. Therefore we know that if we encounter a software containing a small number of strings, it is probably compressed and may contain a malicious code. Then the extraction of strings can be used again after the hidden part of the code is unpacked.

4.1 Tools Strings, HexDive or BinText

Specialized software such as Strings, HexDive or BinText can be used to search for strings stored in the program. All of these programs search for Unicode or ASCII characters and list all strings with a pre-set length. Strings from Windows Sysinternals is a basic tool that implements string extraction and its main advantage is a great compatibility. Once downloaded, it is a good idea to copy this tool to a directory which is included in the environment variable named Path (the content of the variable can be displayed by executing the command “set“ or “echo %Path%“) or add the path to the variable in order to run Strings from the command line [3], [7].

If you want to list strings of seven or more characters from a suspicious file, use the following command (Fig. 2):

```
strings -n 7 -o input_file > C:\output.txt .
```

HexDive is an intelligent extractor that speeds up the analysis of strings obtained from executable files. This is achieved by displaying only the relevant strings for malware analysis (its output is about two-thirds smaller than the output of Strings) [5], [8].

Finally, BinText provides useful information about strings in an intuitive graphical interface with the options to search, filter and store the output data in the table as depicted in Fig. 3. In the Windows operating system the shortcut to this application or the application itself may be copied to the folder C:\Users\<username>\AppData\Roaming\Microsoft\Windows\SendTo (in Explorer also accessible via the address shell:sendto), that way, it'll be always available for quick use [9].

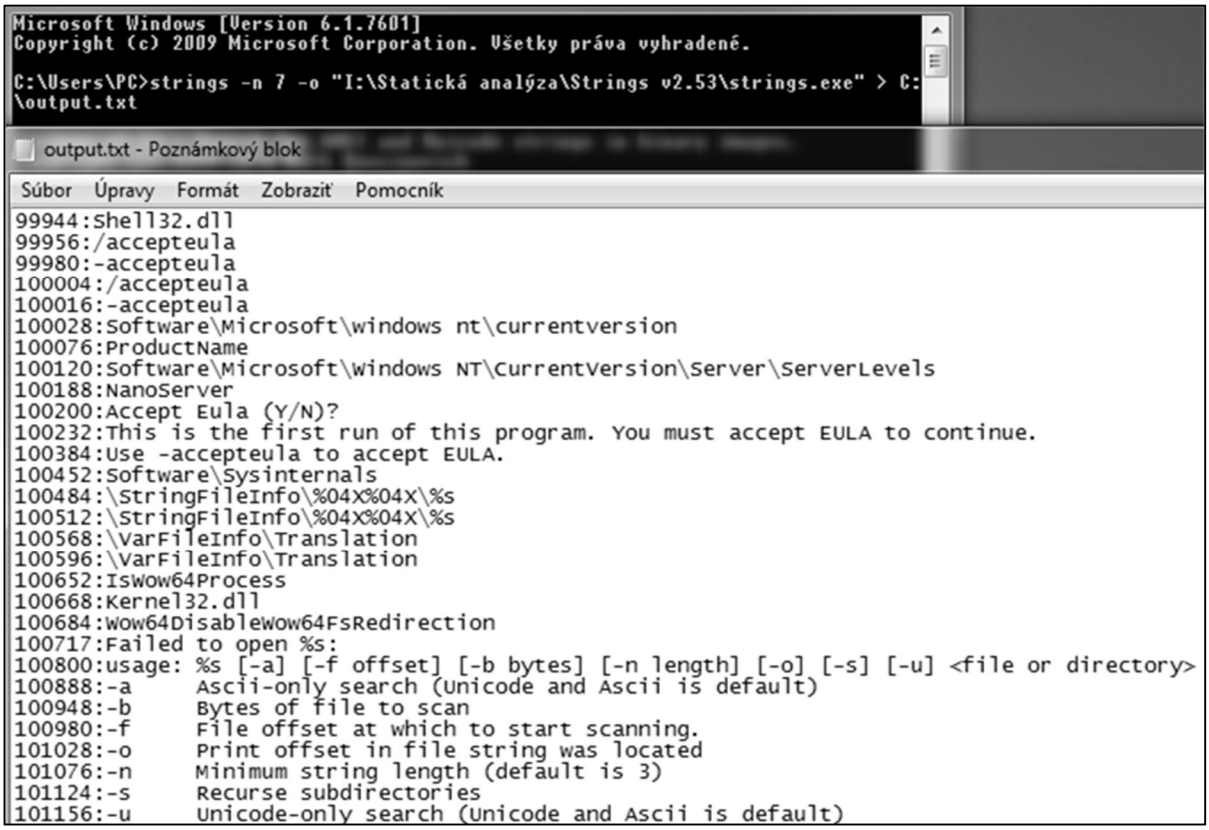

Fig. 2 Using the tool called Strings
Source: [5]

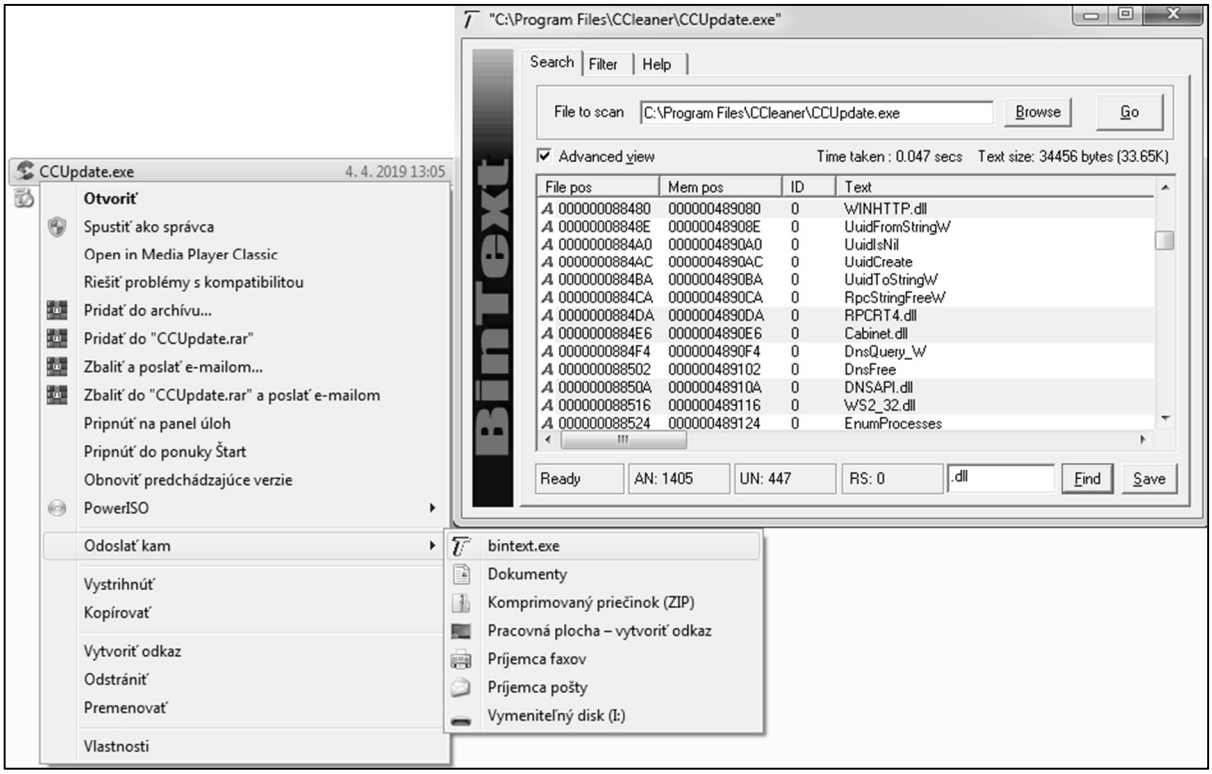

Fig. 3 Using the tool called BinText
Source: [3]

5 PORTABLE EXECUTABLE FILE FORMAT

In static analysis other very useful pieces of information can be obtained from the headers and sections of the Portable Executable (abbr. PE) file format such as the list of all Dynamic-link Libraries (abbr. DLL) and functions that the file imports. Binary executable files (usually with extensions like exe, dll, sys, acm, mui and others) used in all versions of Windows operating system (abbr. OS) are nowadays mostly in PE file format (rarely some legacy file formats are used) which is defined by the exact data structure. Data structure of PE file format contains the information necessary for the Windows OS loader to manage the wrapped executable code. As the name implies, the Portable Executable file format is portable between all versions of Windows OS regardless of the way the processor carries out the instructions of a computer program. Therefore the PE file can be executed on 32-bit systems as well as 64-bit systems [3].

The data structure of the PE file format apart from the actual application code and application data also defines the header where you can find detailed information about that program. Excluding the program code itself, the file header is one of the main sources of information in the static analysis, mainly because the header is available immediately at the start of the analysis and it can provide a first insight into the parameters and features of the analysed malware. Fig. 4 shows the structure of PE files which begins with a header containing information about the code, the type of application, the required library functions, the required disk space, the creation date and many more. Just the list of used libraries and function calls can reveal many features of the program [5].

A PE file consists of a number of headers and sections. To maintain compatibility with the old Microsoft Disk Operating System (abbr. MS-DOS), each PE file begins with a header programmed for that system. This header is known as IMAGE_DOS_HEADER. In most cases it only contains the message "This program cannot be run in DOS mode." Especially the first e_magic field is interesting from an analyst's point of view because it is always at the beginning of each executable file and it has the fixed value of two characters (MZ). Therefore, if the analyst knows that this is an executable file, but there are no MZ characters at the beginning of the file, it is possible that the file is encrypted. Additionally these characters, together with that MS-DOS message, can help us find out the encryption key and decrypt the program because the values of both these fields of data are known [3], [4].

The IMAGE_FILE_HEADER (PE Header) structure contains basic information about the file, such as the date and time the file was created (TimeDateStamp), the number of sections which immediately follow the headers (NumberOfSections), the processor architecture (Machine) for which the program is intended etc. Such pieces of information are very important in the

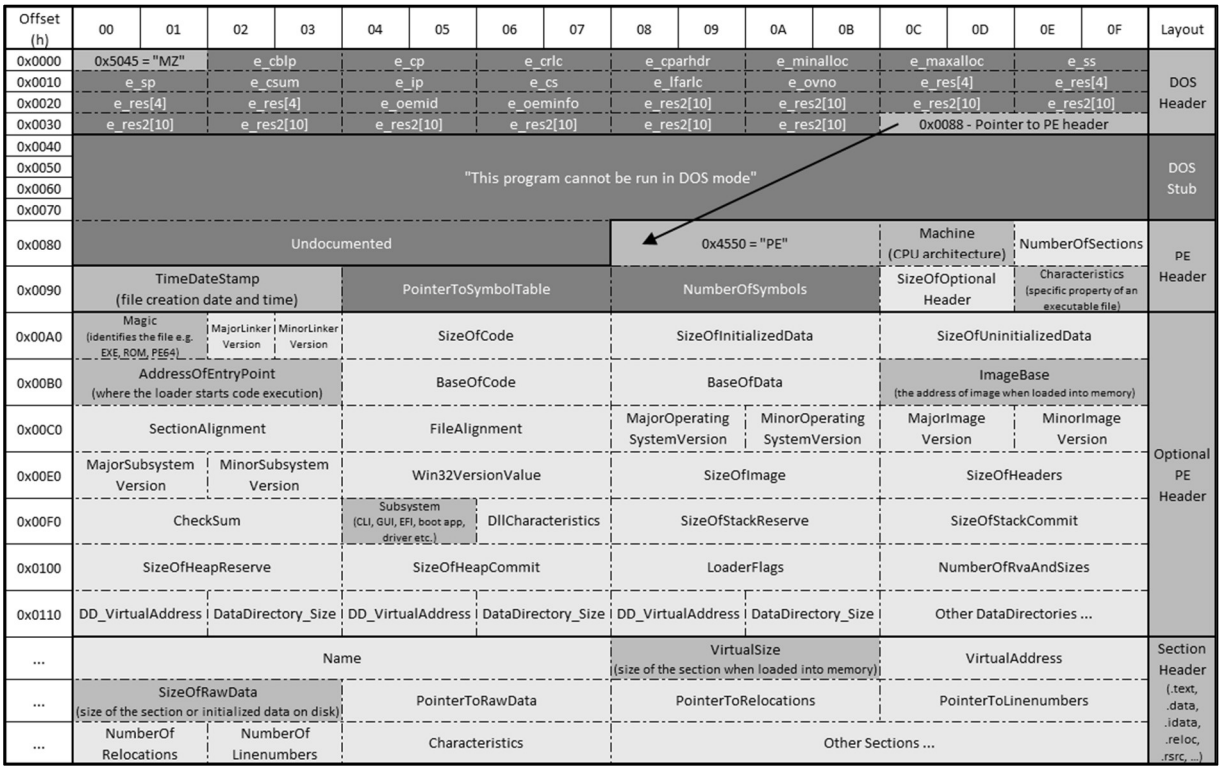

Fig. 4 Structure of a portable executable file format
Source: [2]

static analysis. For example, the creation date of the file will determine whether it is an old sample or a new one that has not yet been scanned by an antivirus technology. Also a value stored in TimeDateStamp could not make any sense at all (referring to the future or the distant past). This artefact usually deepens our suspicions that the file may be malicious [4].

Moreover the header of PE file includes a structure called IMAGE_OPTIONAL_HEADER (Optional PE Header), which contains additional pieces of information for static analysis. There is an important field called AddressOfEntryPoint that contains the address of the entry point at which the program execution starts. The ImageBase field is also essential. It determines at which address in the memory the image of the program should be placed. Its default value is always 0x00400000 (for the DLL it is 0x10000000) and, as with TimeDateStamp, another value can be a sign of something potentially malicious.

The headers are followed by a table of sections and sections themselves which are an excellent source of information for forensic analysis. Here we will be interested in the sizes of individual sections. The virtual size (VirtualSize) specifies how much space should be reserved for the section when loaded into memory. The field named SizeOfRawData contains the size of the section or the size of the initialized data on disk. These sizes should be with small variations approximately the same. If the virtual size is much larger than the size of raw data, it might indicate that the file has been compressed [4].

One of the most useful pieces of information that we can gather about an executable is the list of functions that it imports. Imports are functions used by one program that are actually stored in a different program, such as code libraries that contain functionality common to many programs. Code libraries can be connected to the main executable by linking. Programmers link imports to their programs so that they don't need to re-implement certain functionality in multiple programs. The information we can find in the PE file header depends on how the library code has been linked. Code libraries can be linked statically, at runtime, or dynamically [2].

Static linking is the process of copying the entire code of imported functions directly into the body of the program what may result in a huge increase of the file size. Because of this impractical fact, static linking is not very used nowadays. In the field of malware analysis, dynamic and runtime linking is crucial [4].

When dynamic linking is used, program imports functions during its compilation. The code of the function is not stored directly in the program but it is stored only as a reference in the header of PE file. The .idata section contains the import directory table which includes a list of entries for every DLL which is loaded by the executable. In the first stage of the analysis, thanks to the import table the analyst can figure out some of the application functions such as the feature to connect to the Internet or work with other files or resources. Based on this we can search for other artefacts such as IP addresses, domain names, file or application paths and so on [3].

The specialty of malware developers is the runtime linking. In the runtime linking, the functions are called during the program execution when a specific function is directly requested. Functions are neither imported at the time of compilation nor embedded directly into the program code. These functions are often called using the system functions (known as system API) such as LoadLibrary, GetProcAdress, LdrLoadLibrary, LdrGetProcAddr or using a serial number (each function has an assigned number). Then those system functions can be found in the import directory table. Runtime linking is usually used in the programs that are encoded, compressed or encrypted, and their code is used as a malware loader that extracts or decrypts the code of the application itself, which then loads the required libraries at runtime. Malware authors take advantage of the compression or the encryption to hide program functionality but it can be sometimes found in legitimate applications as well [2], [4].

5.1 Tools CEF Explorer, GT2

The tool called CFF Explorer allows you to extract the metadata from the PE file header as can be seen in Fig. 5. Additionally it offers the basic translation of machine language into assembly language, but in practice it is preferred to use specialized tools. The main advantage of this tool is that it presents the results completely and precisely, including the offset values, hexadecimal values with their meaning and other values a field might contain. On the other hand, the program expects that the user will be professionally experienced and able to correctly interpret the listed values. Therefore it does not inform the user about any anomalies and does not present any special results by their significance in forensic analysis, but only completely presents the results in the order in which they were found out [10].

GT2 is a command line program which is able to identify most of the executable files and archives by their binary signatures. So it is different from standard Windows filetype detection since it does not consider the file's extension by default. In addition, it can also read and analyse the metadata obtained from the file header [5], [11].

The frequently used tool called Dependency Walker can list all DLL libraries used in executable programs (this feature is also included in the tools mentioned above). Dependency Walker also displays a recursive tree of all the dependencies of the executable file (all the files it requires to run) which is evident in Fig. 6 [12].

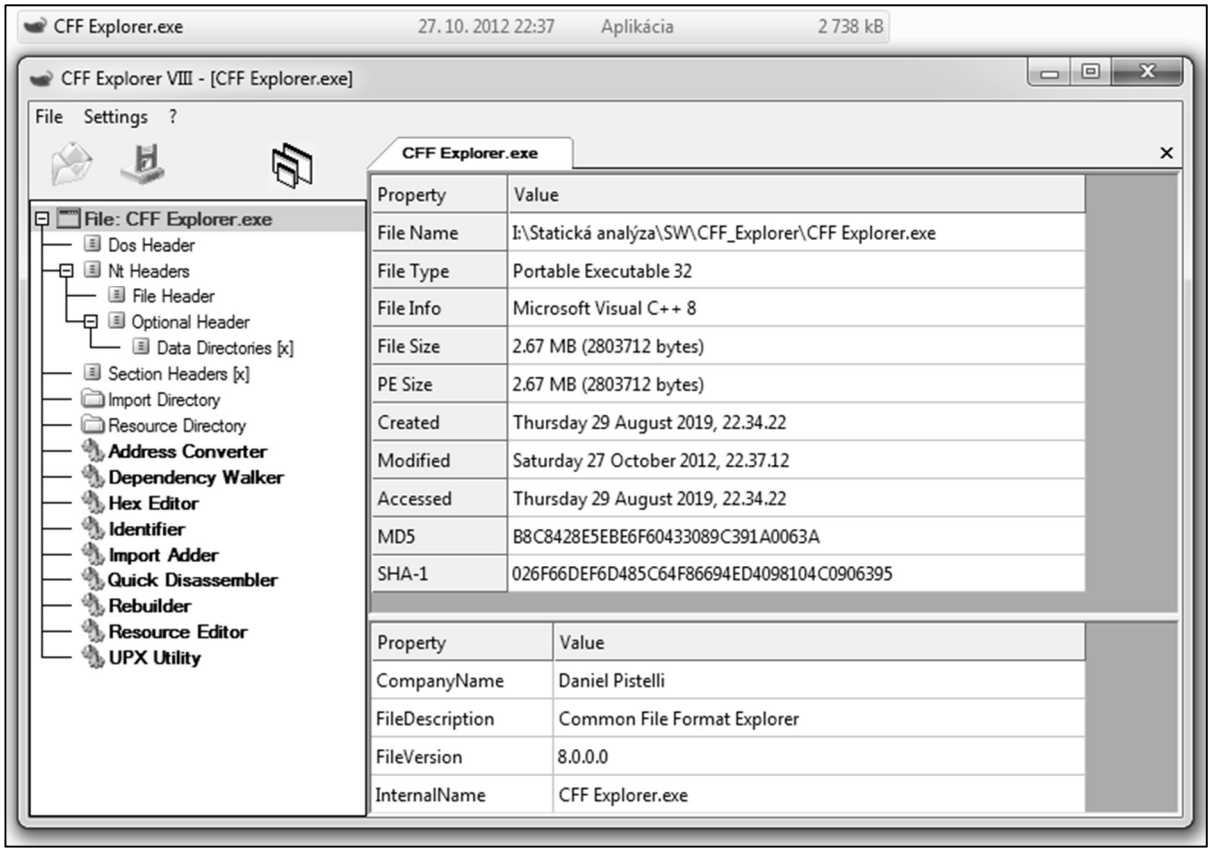

Fig. 5 Using the tool called CFF Explorer
Source: [5]

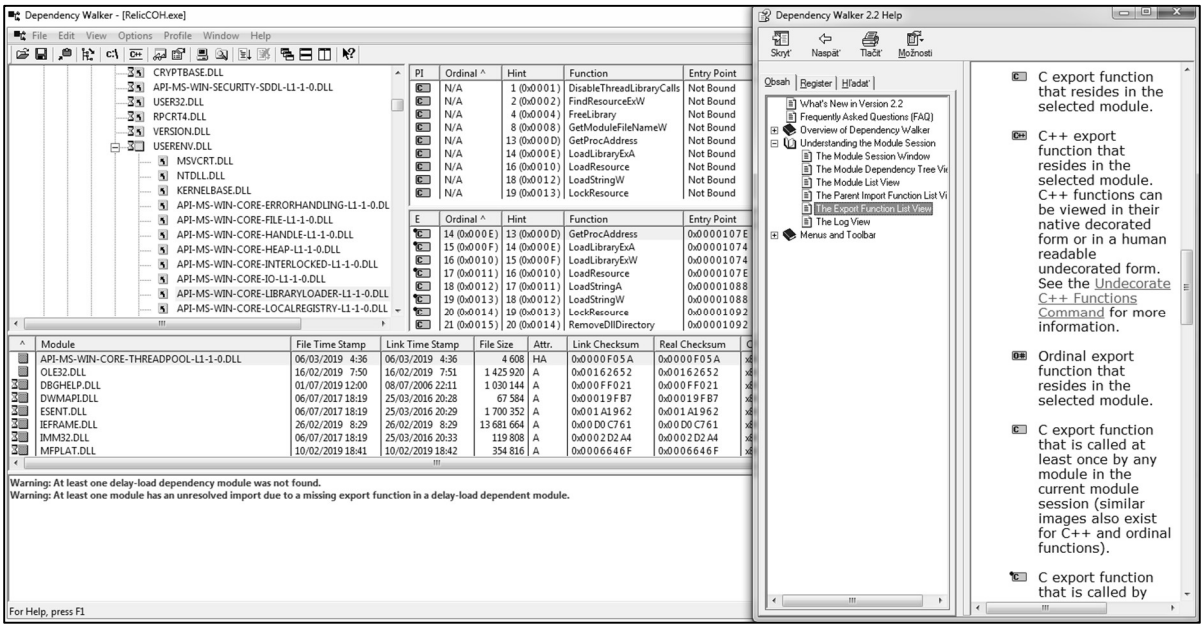

Fig. 6 Using the tool called Dependency Walker
Source: [12]

6 COMPRESSION OF MALWARE

De-obfuscation is the process of turning unintelligible information into something that you can understand. De-obfuscation is an undeniable requirement for malware analysis. Decoding, decryption, and packing are classified as forms of obfuscation. Although these terms differ slightly in a technical sense, they're all methods that attackers use to keep eyes off certain information. Without de-obfuscation techniques, your understanding of malware and its capabilities will be limited [1].

Malware compression is a very popular method for encrypting malicious programs because there are a lot of free and easy-to-use utilities that can do it. Compressed malware is smaller in size, difficult to detect by antivirus programs and difficult to analyse. The principle of the compression is to transform the binary code of the executable file into another form. As a result of this change, malware can escape the attention of antivirus programs when detecting signatures, because after each use of the compression tool, a new, unique sample is created that the antivirus databases do not recognize. Even several compression tools can be used on a single sample what may reduce the chance of successful detection [4].

When trying to statically analyze packaged malware, an extreme lack of information is evident. No interesting strings were found, the list of imported functions will be minimal (usually LoadLibrary and GetProcAdress) and all program instructions will be encrypted. The purpose of unpacking is to remove the layer of confusion applied to the program when it was packaged. There are many different methods for unpacking programs, most of which can be classified as manual or automated methods. Automated unpackers can definitely save you time, but they don't always work [2].

There are many special tools that automate and greatly simplify the detection of the packer (software for compression of malware). One of the most used are the veteran PEiD and frequently updated Exeinfo PE. Both applications provide the user with an intuitive graphical interface.

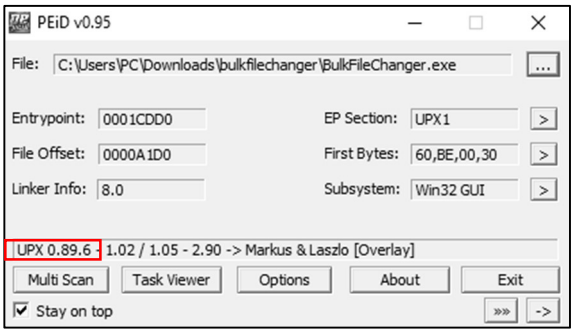

Fig. 7 Using the tool called PEiD
Source: [13]

6.1 PEiD and Exeinfo Pe

The most popular and most widely used program for basic analysis of malware is PEiD which can extract essential information from the file header and identify the type of compression or a compiler used to create the malware as shown in Fig. 7. It can detect over 470 different signatures in PE files. Official support and development of this tool has ended, but it is still often used in the analysis, mainly because it is still possible to add a new signature to the database based on which the compression methods are identified [13].

Exeinfo PE is another free portable application for extracting pieces of information about compression tools from executable files (Fig. 8). The latest version of the application (v0.0.5.6) is capable of detecting more than 1040 specific signatures of PE files. An external database containing approximately 4500 additional signatures (may not be reliable) is included with the application. Furthermore, the user may use other 490 signatures of files that are not executable. This application also provides a lot of information which is able to extract from the PE file header. Its functionality can be further extended by downloadable add-ons [14].

After successfully identifying the packer used in the creation of malware we can proceed further with its decompressing and disassembling (translating machine language into assembly language - assembler) with a number of specific tools. Then the analyst is free to closely analyse the malicious code.

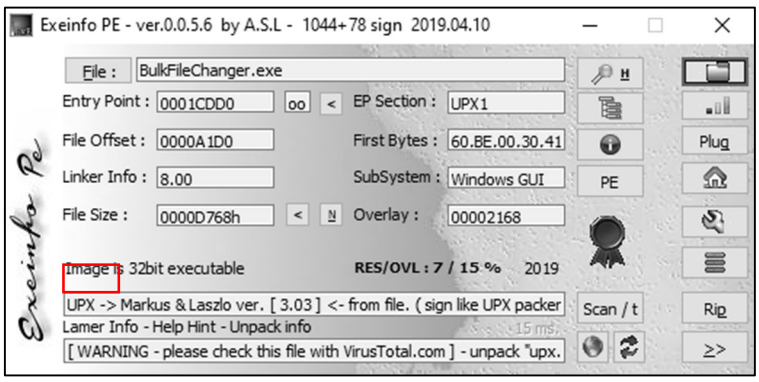

Fig. 8 Using the tool called Exeinfo PE
Source: [14]

7 CONCLUSION

One of the goals of this work is to get yourself familiar with the malware analysis, specifically with the complex subject such as static analysis. This includes clarifying what the analysis is and what it is used for. This paper presents a brief overview of basic methods and tools used in static analysis. Another contribution of this paper is further description of these analytical tools and methods. Finally, after testing the analytical tools, it is advised to combine them into a single package and automate their functions to reduce the time required to perform a static analysis, while ensuring that the resulting malicious file report contains all the necessary information.

References

[1] LIGH, M. H., ADAIR, S., HARTSTEIN, B., RICHARD, M.: *Malware Analyst's Cookbook.* Indianopolis : Wiley Publishing, Inc., 2011. s. 746. ISBN 978-0-470-61303-0.

[2] SIKORSKI, M., HONIG, A.: *Practical Malware Analysis.* San Francisco : No Starch Press, Inc., 2012. s. 802. ISBN-10: 1-59327-290-1.

[3] KRÁL, B.: *Forenzní analýza malware*. Brno : Vysoké učení technické v Brne, 2018, s. 63.

[4] DANILOV, M.: *Metody a nástroje malwarové analýzy.* Praha : Vysoká škola ekonomická v Prahe, 2016. s. 85.

[5] FUJTÍK, O.: *Zjišťování podobnosti malware.* Brno : Masarykova univerzita, 2014. s. 72.

[6] VirusTotal. [Online]. [accessed 20. July 2019]. Retrieved from: <https://www.virustotal.com/gui>

[7] Strings – Windows Sysinternals. [Online]. [accessed 20. July 2019]. Retrieved from: <https://docs.microsoft.com/en-us/sysinternals/downloads/strings>

[8] HexDive 0.6. [Online]. [accessed 20. July 2019]. Retrieved from: <http://www.hexacorn.com/blog/category/software-releases/hexdive>

[9] BinText. [Online]. [accessed 20. July 2019]. Retrieved from: <https://www.aldeid.com/wiki/BinText>

[10] Explorer Suite. [Online]. [accessed 25. July 2019]. Retrieved from: https://ntcore.com/?page_id=388

[11] GT2 0.34. [Online]. [accessed 25. July 2019]. Retrieved from: <http://www.helger.com/gt/gt2.htm>

[12] Dependency Walker 2.2. [Online]. [accessed 25. July 2019]. Retrieved from: <http://www.dependencywalker.com>

[13] PEiD. [Online]. [accessed 1. February 2020]. Retrieved from: <https://www.aldeid.com/wiki/PEiD>

[14] Exeinfo PE“ [Online]. [accessed 1. February 2020]. Retrieved from: <https://www.softpedia.com/get/Programming/Packers-Crypters-Protectors/ExEinfo-PE.shtml>

Lt. Dipl. Eng. Andrej **FEDÁK** (PhD. student)
Department of Computer Science
Armed Forces Academy of General M. R. Štefánik
Demänová 393
031 01 Liptovský Mikuláš
Slovak Republic
E-mail: andrej.fedak@gmail.com

Prof. Dipl. Eng. Jozef **ŠTULRAJTER**, CSc.
Armed Forces Academy of General M. R. Štefánik
Department of Computer Science
Demänová 393
031 01 Liptovský Mikuláš
Slovak Republic
E-mail: jozef.stulrajter@aos.sk

Andrej Fedák - was born in Žiar nad Hronom in 1994. He received his engineering degree from the Armed Forces Academy of General M. R. in the field of Military Communication and Information Systems. Nowadays, he is an officer of aeronautical ground information systems - Air Force Headquarters. His research is focuses on computer networks, information systems, information and cyber security.

Jozef Štulrajter works as a professor at the Department of Informatics, Armed Forces Academy of General M. R. Štefánik in Liptovský Mikuláš. He graduated (Ing.) at the Military Technical College in 1974. He obtained the degree of CSc. diploma in Theoretical Electrical Engineering - Theory of Circuits and Systems of the Military Academy in Liptovský Mikuláš in 1992. His research interests include Information and Communication Technology (ICTs), computer architectures, image coding, computer security.