MASTER THESIS

"Dynamic analysis of **malware** in Windows operating systems from previously captured network packets"

Mentor:
Assoc. Prof Dr.
Mentor Hamiti

Candidate

BELKIJA RESHITI

Abstract

This research gives basic introduction about malicious software applications by describing the infection vectors. Here, I will define the layers of malware analysis with distinction between static and dynamic analysis. Specifically I will consider performing dynamic analysis over previously captured network packets over Windows XP Operating System by building a software platform or safe environment for malware analysis with required software tools and resources. In this research I tried to determine a working practical solution how the malware has to be analyzed with its constraints, which standard operational procedures shall be implemented and at least how a security expert should deal to minimize the security problems infected by malicious software codes.

List of tables and figures

Table of Contents

Introduction

Malware or Malicious software is a collective term for all kinds of threats including viruses, worms and Trojans.

- A computer **virus** is designed to infect machines and travel autonomously from computer to computer. This is often is triggered by a user's action, such as opening an infected e-mail attachment.
- A **worm** also spreads automatically. However, instead of writing its code to multiple objects on a disk, it installs itself once and then looks for another computer to infect. Some worms for example e-mail worms require the action of an individual in order to spread. But others, such as network worms, spread without the need for human interaction.
- **Rootkits** (Root UNIX/Linux term equivalent to Administrator in Windows OS and Kit as software component) hides (itself) objects like files, process and registry and results the infected machines to be spied or monitored.
- **Trojans** are named after the mythical 'Trojan horse'. This is because historically, they were often malicious programs that masqueraded as something benign, or even useful. Some Trojans still work this way. Someone might download and run them in the expectation of the file performing a useful function. But instead it carries out a harmful operation on their computer, without their knowledge or consent. A Trojan may also be installed silently on a computer when the victim visits a web page that has been compromised and contains malicious code. This code runs automatically when they view the page and is referred to as a 'drive-by download'. Trojans are distinct from viruses and worms because they don't self-replicate, by rely on the connectivity that provide internet. There are several different kinds of Trojan each, designed to carry out a specific purpose.

  For example:

- **Backdoor Trojans** permit system access by an uninvited party potentially allowing remote administration of a system. Often they include a keylogger that records every key pressed, in the hope of finding out the victim's password or some other piece of confidential data. Other types of Trojan include banking Trojans, which are

designated to steal money from a victim's bank account, and Trojan downloaders, that download update code to the computer. More recently we have seen the emergence of hybrid threats that combine the functionality of virus, worm and Trojan in one package providing cybercriminals with greater flexibility in their method of attack.

Once a cybercriminal is in control of a computer they can do pretty much anything from collecting confidential data, to sending spam the list is almost endless. The first they will typically however is connecting the computer with other infected computers effectively creating an infected network, commonly known as a **botnet**. This infected group of machines can be instructed by the criminals or by whoever is in control of the **botnet**, to do any number of thins from sending out thousands of spam messages to launching targeted attacks on specific organizations. One example of this would be a Distributed Denial of Service attack. This is where thousands of machines send small amounts of data to one target, to interrupt the normal running of a website, email server or any other business system. When malware first emerged, it served a simple purpose to cause damage with no financial gain, commonly referred to as 'cyber-vandalism'. This could be deletion of files renaming of data or the erasing of data storage media. Some were designed to do nothing at all, although they could have unintended side-effects. While viruses might not have been visibly running, a victim could sometimes 'feel' them, perhaps through a sluggish machine or slow internet connection. Nowadays, the overwhelming majority of malware is created to make money illegally, often by collecting confidential data from the victim's computer. To do this malware is designed to install as discreetly as possible running without disturbing the victim and ensuring that the machine is up and ready to be used. A damaged offline machine is no value to cyber criminals, but an infected machine is a powerful asset, able to perform any number of tasks. There has always been lots of speculation about the financial impact of cybercrime. If you search online for 'costs of cybercrime' you will find estimate ranging from millions to hundreds of billions. But the illegal nature of cybercrime activities makes it impossible to give an accurate figure of how much it costs. One thing is for sure; the growing volume of attacks makes it clear that its highly profitable for those involved in the 'dark market' that is cybercrime. The threat landscape has been dominated for almost a decade by random, speculative attack on anyone unfortunate enough to be infected.

However the number of targeted attacks is growing. Such attacks are normally aimed specifically at one business. The **motives** can vary. Attackers may want to steal confidential business or customer data, damage a company's reputation, sabotage the normal running of an organization, or even make a political statement. Targeted attacks are highly sophisticated but they often originate by tricking individuals into disclosing information that allows the attacker to access corporate systems. The widespread use of social networks, and the vast amount of data that we all post online, makes it easy to set up such attacks. Since 2003, malware has been used for criminal purposes targeting both businesses and consumers. Commonly known as 'cybercrime' it is effectively 'the use of malware as profit.
The malware employed in cybercrime typically has some simple, well known objectives. Make money by stealing sensitive information.

1. The first is to make money by stealing sensitive information such as online banking logins, credit card numbers or intellectual property. This is identity theft stealing the victims' online credentials and using these to impersonate them. Cybercriminals can access accounts and use them in a number of ways including simple theft, digitally laundering money, or selling on the data to other criminals.
2. Another objective of cybercrime is to extort money. This is often achieved by encrypting the data with a password and asking for money to decrypt it. This method is known as **ransomware**, and can be very lucrative given the high value that an individual or business places on digital information.

Extortion of money can also take the form of what is known as a fake anti-virus scam. These scams revolve around making someone believe they do not have adequate protection. The bottom line is that the victim is asked to download and pay for removal of malware that isn't actually on their computer. The way they work is very simple the online victim may see pop-up windows or an inescapable barrage of warning messages that seem to indicate the presence of malware. Criminals even manipulate search engine results, so that their 'adverts' appear at the top of search list. Not only has the victim paid to remove something that doesn't exist, but the criminals now have their credit card details.

There are several ways in which malicious code can spread. Someone may be infected just by visiting a seemingly harmless website, as cybercriminals look for security loop holes in web servers. These serves may host more than one website. Criminals hide their code in pages stored on the server and when someone views on of those pages, malware is transferred automatically to their computer, hidden inside the rest of the content they were expecting. This is often referred to as 'drive-by download'.

Software vulnerabilities are exploited in order to run malware. As well as being served from infected we pages, malware my also spread via e-mail, typically via attachments or links. Malicious links can also spread rapidly through social networks like Twitter, Facebook, YouTube etc. Malware can spread through traditional storage media such as CD's or USB memory sticks. Of course since its physical media, the spread of malware is significantly slower. Malware doesn't always relay on user interaction to spread. In fact, it often takes advantage of holes in software also known as vulnerabilities to infect other devices. These vulnerabilities or bugs can be found within the operating system, or in widely used software such as Java, Adobe PDF Reader, Microsoft Office or other applications. These flaws are not uncommon and cybercriminals exploit them in order to run malware. With an ever increasing malware range of malware being created a multiple ways of it entering corporate networks, organizations need to ensure their businesses are protected.

Mostly of the time AV companies analyses hundreds of thousands analyses malware samples every day. The bulk of these are modifications of existing viruses, commonly referred to as 'variants'. Historically AV companies have worked by searching for snippets of code that identify a known virus, worm or Trojan that are commonly referred to as 'signatures'. This includes heuristic analysis, sandboxing, application whitelisting, behavioral analysis and more.

Malware Analysis Pre –requisites

The learning objective of this topic is to give basic introduction about Operating Systems by underlining Windows XP in which later is used as guest OS to test and analyze malware specimen. In addition there is classification of malicious codes by describing malware infection routine over operating systems. Moreover malware installation is followed by describing processes, auto-start mechanism, registries, PE files, packed versus non packed files as pre-required info for malware analysis. Finally there is basic definition but sure and safe malware handling policy before to go to malicious specimen analysis.

1.1 Operating Systems and Windows XP

An operating system is that enables services for software applications to run on a computer. An important task of an operating system is taking care of the communication between the software applications and hardware devices attached to your computer. Operating systems are large programs consisting of thousands of functions, which provide services of various kinds. Often called by events in the system, the functions perform a service when needed. (Krogh, 2015, p. 19)

It is hard to pin down what an operating system is other than saying it is the software that runs in kernel mode – and even that is not always true. Part of the problem is that operating system perform two basically unrelated functions: providing applications programmers (and application programs, naturally) a clean abstract set of resources instead of the messy hardware ones and managing these hardware resources. (S.Tanenbaum, 2007, p. 3)

How do you begin to describe an operating system? One way is to look at its major characteristics or key features. The major characteristics of Windows NT are that it implements:

- Multithreading

- Pre-emptive multitasking
- Demand paged virtual memory, which utilizes a single, global common cache
- Multiprocessing
- A processor-independent architecture
- An internal OS structure based on a modified Microkernel model
- Integrated networking
- Multiple operating system emulation (G.Viscarola & W, 1998, p. 9)

Windows XP is a *computer operating system* and *graphical user interface* (GUI), which enables you to work with a wide variety of programs on your computer, often simultaneously. Windows XP is itself a special computer program that communicates your instructions to the actual computer hardware, and displays the results. (Introduction to MS Windows XP, 2009, p. 3).

Launched in August 2001, Windows XP has been the most popular version of Windows based on the number of copies sold. (Krogh, 2015, p. 16)

According to *Spiceworks' Network and Endpoint Security* report, one third of businesses still operate at least one device running Windows XP, which reached the end of its extended support cycle way back in 2014. (Interestingly, the final variant of XP, *Windows Embedded POSReady 2009*, only came out of support in April 2019. This variant of XP was used as a point-of-service operating system by – for example – shops. However many XP Home and Professional users were using a hack to receive security updates intended for this lesser known XP variant. But alas, this has now come to an end.) Of course, Windows XP doesn't get updates anymore. Including security updates. This means as crooks learn about vulnerabilities in XP, there is nothing to stop them using those exploits against XP users. As such, using Windows XP in 2019 is really one of the biggest security faux pas' you can commit. (Charles, 2019)

According to the NetMarketShare a web analytics company known for its global market share statistics for Internet Technologies, Desktop operating system market share of Windows operating systems counts about %86.31 of market share.

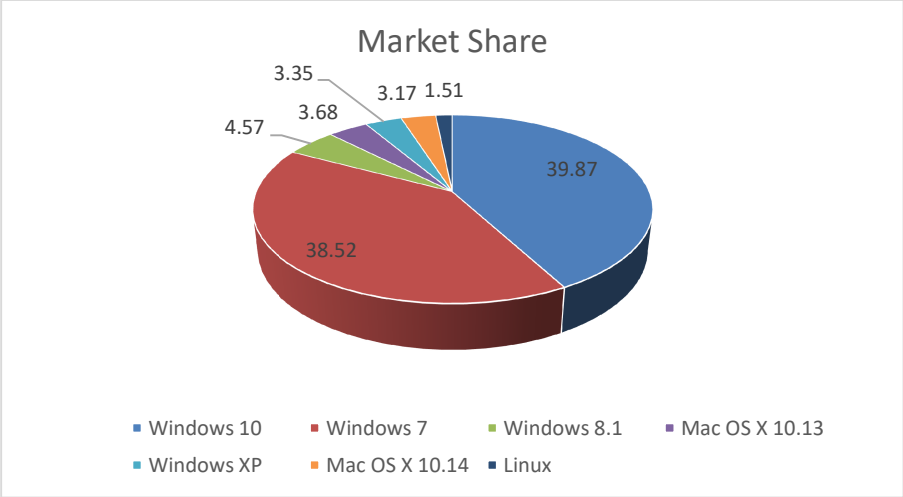


Figure 1 Operating Systems market share[1]

Popularity is one of the main reason why Windows OS is a big target. Malware coders with malicious applications infects average computers users and target Windows OS because that's where the most users are. Mobile OS such Android allows users to install software from outside Google Play and Linux based OS allows its users to install software from outside their software repositories, the majority of the application for Android and Linux installation comes from trusted centralized repository. On the other hand Windows desktop OS, allows to make web search for an application download and install it easily.

Because of high popularity, easy and user friendly GUI, and high hit from malicious codes Microsoft Windows XP will be the host operating system for researching and analyzing malicious specimen in this paper.

[1] Exported report from NetMarketShare.com for Desktop operating system market share in a June 2018 to May 2019 , [Date accessed April,11 2019].

1.2 Classification of Malware

Malware mainly is classified in two categories:

First one is based on **how it spreads**, propagates to get the targets and second one is action or **payloads** performs once a target is reached. For instance, parasitic codes such as Viruses need a host program. Worms, Trojans and Bots are independent or self-contained malicious programs. Malicious malware such as Trojans or spam e-mail do not replicate itself. Viruses and worms are self-replicate but do not infect files two or three times because file becomes bigger and bigger and this may result binary file change which causes curiosity to be obvious and discovered from AV's and firewalls.

1.3 Malware infection routine

Malicious applications infection routine consists three different classes in which one by one draws the path how machines get hit by malicious codes. All classes has their elements and attributes in which defines prospective layout about malware behavior in machines.

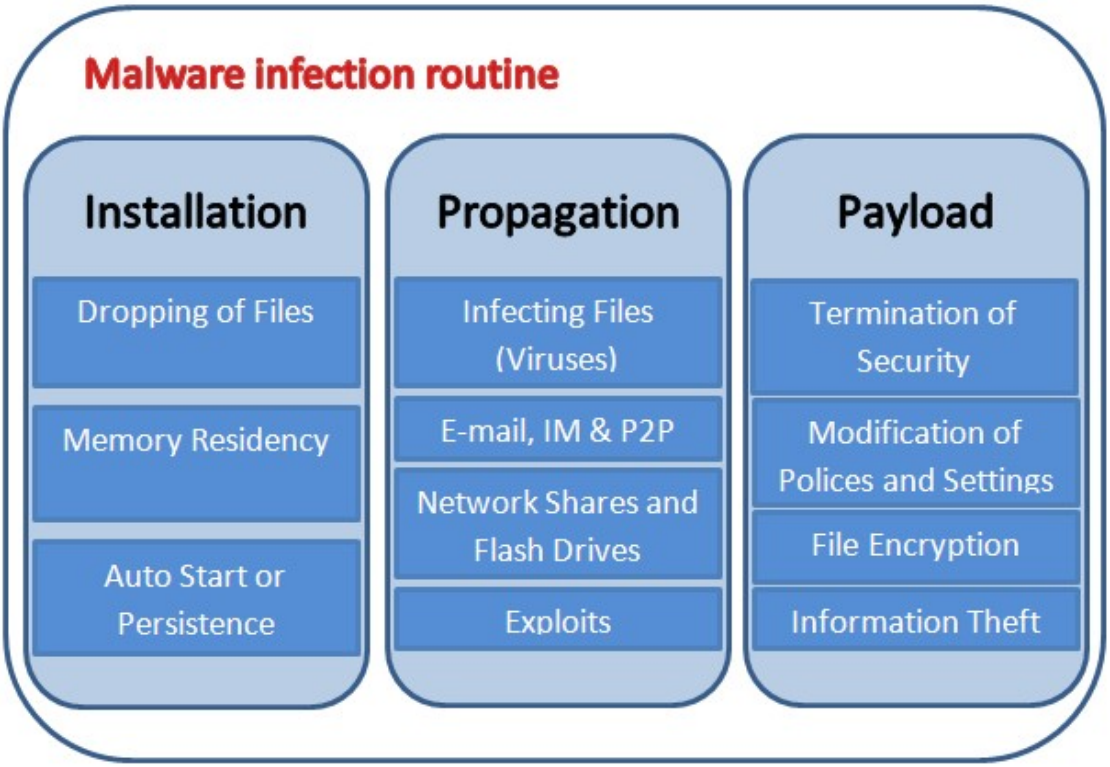


Figure 2 Malware infection routine[2]

[2] Figure 2, Malware infection routine. [Taken from training materials with the topic Malware Analysis Training organized by the Interpol, Cybercrime Directorate held by Trend Micro Inc. in period 05-09 September Lyon].

As shown above in *Figure 2*, malware infection process starts from dropping copies of them to be executed during startup/reboot, to be executed in a certain trigger condition and then uses to propagate. Some malware drop several components to help accomplish malicious activities such as make them properly to execute, make them to intact in the system or make them harder to remove and get memory residency. Some malware drop other malware to act as a carrier for the said malware or be an accomplice in carrying out malicious activities. After one machine is infected malicious code performs propagation through subsequently spread to the other machines via infected files, e-mails, per to peer communication, network shares, flash drives or exploits. Finally malware payload actions by termination of security, implementing modifications over policies and settings, corrupting system or data files, encrypting files, theft of information from the system using key logging.

1.4 Malware Installation

Malicious specimen installation could be traced through the analysis of processes, Windows XP operating system auto-start mechanism, file systems and the registries. All mentioned concepts here is represented in following next coming contents.

1.4.1 Processes

While executing each application there would have its equivalent Process. Actually the process it is the image of file (application) running in the memory. There is distinguish between processes between **Parent process** and child **process**.

- **Parent process** is a process that has created one or more child processes;
- **Child process** in computing is a process created by another process (the parent process).

Investigating process activity can tell you whether the malware spawned additional processes

1.4.2 Auto-Start mechanism

Windows Auto-start its mechanism that triggers the malware to be executed automatically at the startup or boot of system. Mostly of the time malicious codes targets Windows XP using Auto-start mechanism to keep malware always execute.

Common methods of targeting Windows XP Auto –start mechanism are:

- Windows Startup Folder;
- Registry modification;
- AUTORUN.INF Modification.

Windows Startup Folder is location which contains tools making Windows XP efficient. Also it can be described as folder automatically configuring Windows XP desktop. Windows XP installed machine looks in this file during the boot process and launches anything it finds there, including everything from starting applications to opening files and folders.

Default Windows Startup Folder locations in Windows XP are as following:

- C:\Documents and Settings\(USER)\Start Menu\Programs\Startup
- C:\Documents and Settings\All User\Start Menu\Programs\Startup

Malware coders has been known to use this method to load malicious application at the default startup desktop configuration which makes malware not easily to stop.

1.4.3 The Registry

The registry is a hierarchical database that contains information about the various settings used by operating system. The registry contains system settings used at the Window startup. The registry plays an important role in setting up and controlling the Windows operating system it keeps track of system settings user setting and what hardware and applications are on the computer. The registry contains information about the applications installed on machine, in addition to information the processor, memory drivers network and similar aspects. (Krogh, 2015, p. 42)

Analyzing and investigation the registry operations is one of the jobs of malware analyst in which he can look, track and report how malicious specimen infected the system. I did describe and demonstrate registry in third topic with title "Dynamic Malware Analysis on previously captured SMTP packets"

Here important is to mention Windows XP Auto-start the registry key modification locations, which malware specimen writes its path and file name to be executed during startup services.

Figure 3[3]

As you realize from *Figure 3*, malware specimen default infection on common Windows XP Auto-start registry locations are:

- HKEY_LOCAL_MACHINE hive that contains about currently detected hardware and device drivers, information for the software installed, as well as for the Windows operating system itself.
- HKEY_CURRENT_USER hive that contains configuration information for Windows and software specific to the currently logged in user. Various registry values in various registry keys located under the HKEY_CURRENT_USER hive control user-level settings like the installed printers, desktop wallpaper, display settings, environment variables, keyboard layout, mapped network drives, and more.

Basic Auto-start Sequence in *Figure 4* represent the states Windows boot up, logon prompt and startup folders of machine and the registry keys affected during the execution of the states. This gives better picture for malware analyst where to get data information to analyze and dissect information about the malicious activities

[3] Figure 3, Common Windows XP Auto-start registry locations. [Taken from training materials with the topic Malware Analysis Training organized by the Interpol, Cybercrime Directorate held by Trend Micro Inc. in period 05-09 September Lyon].

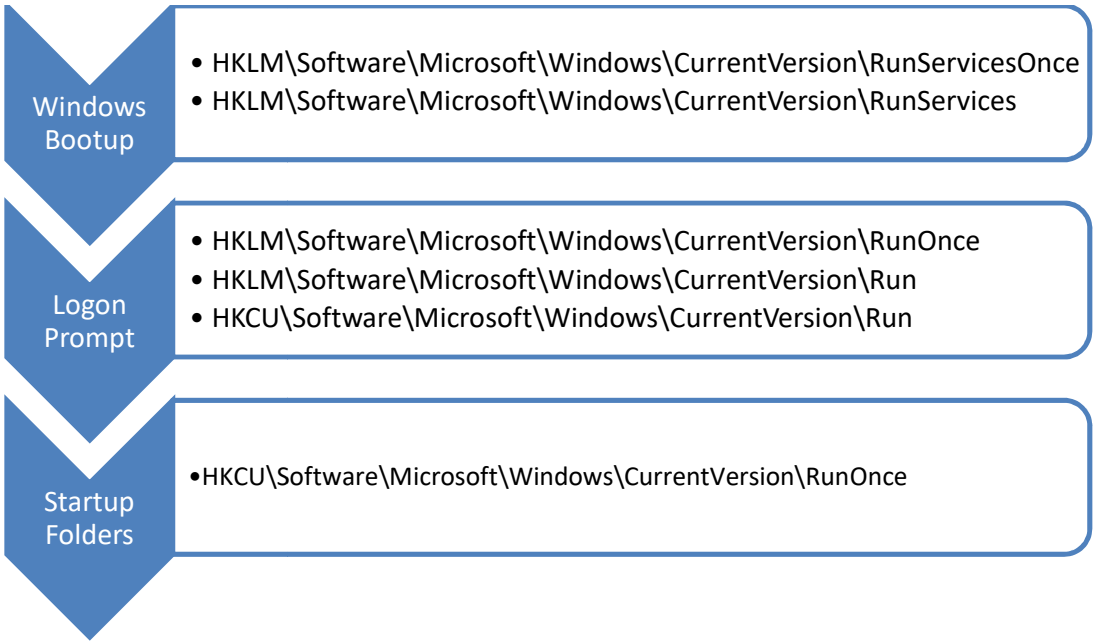


Figure 4 Basic Auto-Start sequence[4]

1.4.4 File system

Exploring file system interaction can show all files that the malware creates or configuration files it uses. Malicious files are involved in nearly every major data breach and is causing growing problems for business and personal users. Wide host-based anti-malware products is obliged, they are not conducting the work done alone. The tendency of ever changing malware still flows over the walls of protection and into computer systems. Once the malicious files have implanted themselves, the challenge falls on the incident responders and forensics experts to identify, contain, and eliminate those attacks.

Suspicious code analysis is extracting information from files through physical file analysis. Most malware specimen it's obfuscated from their programmer using so called packers. Before to go to malware analysis there is need to be distinguished a packed or non-packed file. If malicious file is packed than there is need to be unpacked in order to extract more information while static malware analysis.

Microsoft Windows XP and other Microsoft Windows family of operating systems executable file format is co called **PE (Portable executable)** file. Depending on the processor

[4] Figure 4, Represents Basic Auto-Start sequence and the registry entries of Windows XP operating systems.

PE file format could be 32 bit or 64. Common files in Windows operating systems are the following extensions are .exe, .dll, .ocx, .sys.

The Portable Executable (PE) file consists from four important section described below:

- **Headers** - defines ho to execute a Win32 Portable Executable (PE) file;
- **Sections** – holds information about how a PE file is organized internally;
- **Entry Point** – is the starting point of program execution;
- **Import Table** – contains the list of functions imported by the program from dynamic-link libraries;

The PE file standard is created by Microsoft and openly available online at web page http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx (Microsoft Windows, 2019), which contains additional information for commonly used section names, entry points and import tables. Now here is the breaking point. Even though this specification is spelled out by Microsoft, compilers/linkers chose to ignore some parts of it. To make things even worse, the Microsoft loader doesn't enforce a good portion of this specification and instead makes assumptions if things start getting weird. So even though the specification outlined Microsoft Windows standard that says a particular field is supposed to hold a certain value, the compiler/linker or even a malicious actor could put whatever they want in there and the code will continue to run as a normal application.

For instance Microsoft Windows uses MZ headers (“MZ” are the initials of Mark Zbikowski on of leading developers of MS-DOS) in DOS while executing .EXE files. This means that file can be identified by the ASCII string "MZ" so called “magic number” (hexadecimal: 4D 5A) at the beginning of the file.

Depending on the compiler, a Windows application is usually separated in different sections corresponding to program code, data and miscellaneous items (e. g icon, relocation data, etc.). Commonly used section names in PE File Sections are .text, CODE, .data, .idata, .bss, .rdata, .edata, .reloc, .rsrc.

Mostly of the time PE Files sections have pre-defined names, in which they are created while file is compiled. The most common ones are:

- .text – Which contains the code of the file
- .rdata – Read only data

- .data – Other data.
- .rsrc – The resource section, contain icons, menus, images etc.

1.4.5 Packed vs Non Packed binary code

Malware programmers by using packers (kind of compression the files) however will alter these sections, and different packers do it in different ways. For example the ASPacler always creates sections with .aspack in the name as represented in Figure 5 below.

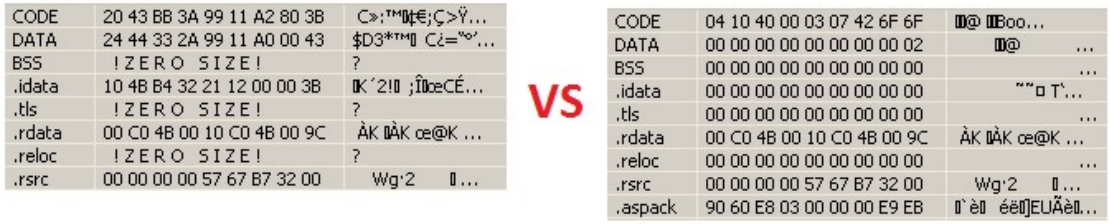


Figure 5 [5]

The graph on the right in *Figure 6*, shows the code flow of an unpacked program– with each line representing a path through the code. On the left we have the same file after it has been passed through a packer.

As you can see it changes quite a lot - the packed file is much more complex. There are numerous other features that commonly show up in packers. Some packers will have identifiable strings in them that can be used to identify what packer you are dealing with. Other packers can be identified by what imports they use. In general packed files will have much less imports than standard files, although some obfuscators deliberately add more imports to make analysis more complex.

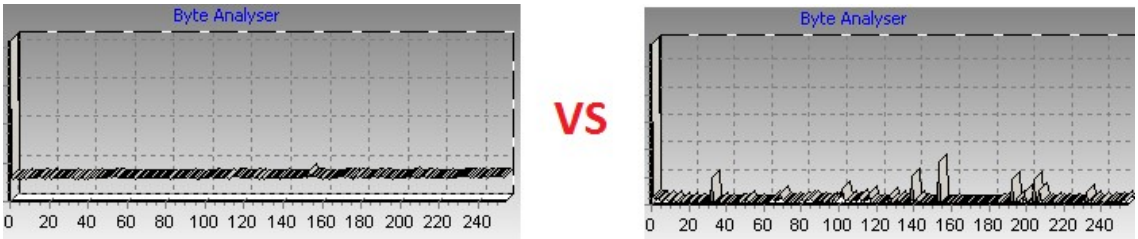


Figure 6 [6]

[5] Figure 5, Section .aspack.
[6] Figure 6, Packed vs Unpacked Graph.

Sections in Packed Files also have what is known as “High Entropy”. Entropy is essentially a mathematical formula to measure randomness or predictability for instance a fair coin toss has an Entropy value of 1. A rigged coin toss would have lower entropy, as it is more predictable. A String of AAAs in a row has Entropy of 0 – in other words it is completely predictable. The English language has an Entropy between 1.0 and 1.5. Think about English -> Th is normally followed by one of eoui, Q is normally followed by U – in fact there are only 12 words that don’t that rule. Different Languages have different entropy.

Packed data looks like encrypted, so it has HIGH entropy, it can be seen in the packed section of code when tested as in *Figure 7*, with the PEiD tool I got the entropy value 8.00 (Packed)

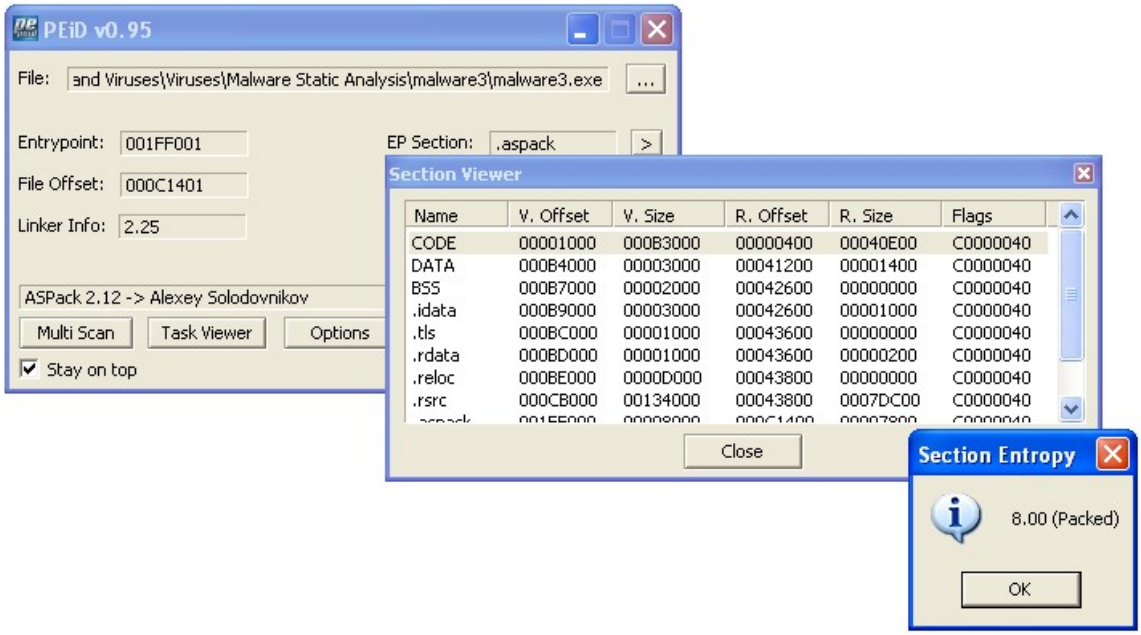


Figure 7[7]

1.5 Safe Malware Handling Policy

Before to start to analyze any malicious specimen there is need to build safe malware handling policy or at least to list some procedures. These procedures rises the consciousness dealing with infected specimen because it is very common while working with them you can be hit and infected. So in this topic I listed the required safety procedures of malware handling:

[7] Figure 7, Entropy of the packed file using PEiD tool.

- All samples should be compressed and password protected or encrypted.
- Physically transfer the Portable Device to an “Isolated Stand Alone System” for analysis.
- Once the Samples are loaded onto isolated Stand Alone System, you may decompress the files and remove the password protection or encryption.
- You may then analyze the Samples
- Once you have competed your analysis, you must delete the Samples from the Isolated Stand Alone System by holding the “Shift” button and then pressing Delete or emptying the recycle bin or “deleted items” folder to ensure the completion of the delete process.
- Then verify that the Samples stored on the Portable Device are still compressed and password protected or encrypted, and delete file from the Portable Device.
- Rebuild or revert the Isolated Stand Alone System.

1.6 Article 251-a, Criminal code of Republic of North Macedonia

According to the Criminal code Republic of North Macedonia, making and importing computer viruses Article 251-a:

(1) A person who will or will take another computer virus with the intention of entering it into another computer or computer network, shall be punished with a fine, or with imprisonment of up to one year.

(2) A person who, using a computer virus, will cause harm to another computer, system, data or program, shall be punished with imprisonment of six months to three years.

(3) If the offense referred to in paragraph 2 causes more damage or if the crime has been committed in the composition of a group created for such an offense, the perpetrator shall be punished with imprisonment of one to five years.

(4) The attempt for the crime from item 2 is punishable.

(5) If the crime referred to in this Article is committed by a legal entity, it shall be punished with a fine.

Note that the samples of malware analyzed here are available online and I used them just for demonstration purpose.

Static Malware analysis

Static malware analyze is analysis of malicious specimen without running or executing the code. This is usually done by determining the obfuscation of malware specimen if it is packed or not packed. If it is packed it is decompressed by various tools and techniques. Decompression leaves backdoor on malicious code in which by various string search follows dissecting detailed information about commands, IP addresses, e-mails, messages etc, whither happens during execution flow of specimen. Additionally there is digital signature calculation of hash value, which later can be compared with AV data hash sets.

Learning objectives of this topic is to give basic inputs on static malware analysis by building up an environment with required tools for malware analysis Lab. Moreover there is demonstration techniques on examining information from dead binary code. Unpacking malicious specimen, finding dynamic link libraries on Import table will fortify the executive procedure on intercepting malicious code. After I conducted some string search over malicious sample to dissect useful data footprints over commands that has on execution flow. Finally by hashing file digital signature I did online search to get additional information about my specimen in which got benefit on validation of my findings.

2.1 Building up an environment and tools for malware analysis Lab

VirtualBox v4.3

A virtual machine is taken to be an efficient, isolated duplicate of the real machine. (Popek & Godberg, July 1974, p. 413)

Reliable access to analyze malware requires infecting a machine and system with malicious software piece and then using applied software monitoring tools to analyze its behaviors. To do and test malware pieces, here I used Windows XP installed as guest operating system to an Oracle VirtualBox virtualization environment.

VirtualBox is cross-platform virtualization software from Oracle Corporation. VirtualBox can run multiple operating systems simultaneously. Main advantages of using VirtualBox are:

1) VirtualBox allows users to run more than one operating system simultaneously. So users are able to run window software on Linux or on Mac operating system simultaneously without rebooting the system;
2) It can run on multiple operating systems e.g. on Windows, Mac OS, Linux and Oracle Solaris Systems.
3) No special hardware is required for running VirtualBox.

Oracle **VirtualBox** is free and the suitable method to set a laboratory or a system that involves a virtualization of software, which allows in one single computer hosting a multiple virtual systems, each running a different operating system. Another useful feature of using VirtualBox is that can make snapshots and still to have system state before we infect it, and then return the default environment after the finishing the malware analysis. (Singh, September 2018)

While performing malware analysis as a first point I considered and followed a method for isolation of laboratory system from production and reproduction of malware Main aim here is to disable and stop my test Lab communicating any another machine through network or accessing to the internet as shown in Figure 8 below, in which I will be sure %100 percent sure not for unconsciousness infections.

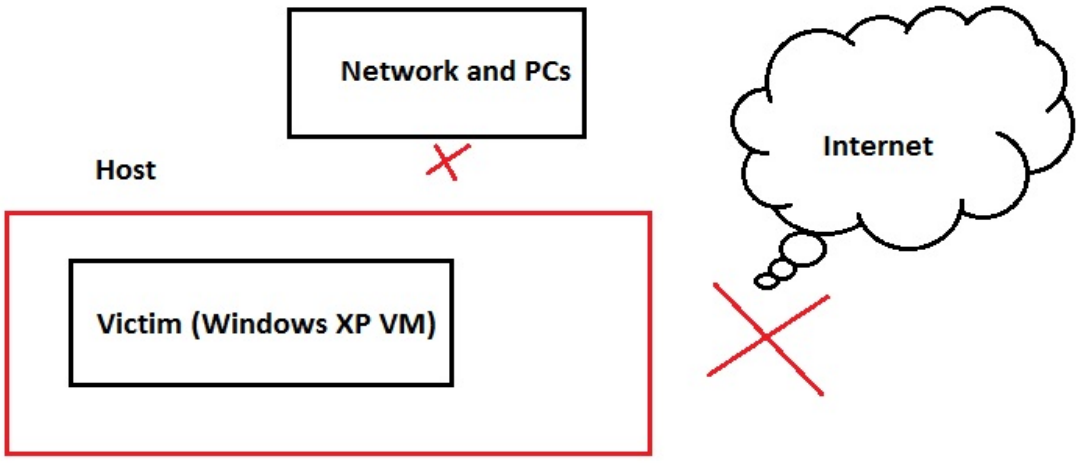


Figure 8. Host & Virtual Machine [8]

[8] Figure 8, Host and Victim Virtual machine interaction.

2.1.1 Tools for Lab

Here I listed the tools that I used to perform stating and dynamic analysis. With the help of these tools I used monitor and track modifications, behavioral changes, dissecting detailed information caused by malware samples. All the tools used here are open source and free that could be found online.

Autoruns v.12

This utility, which has the most comprehensive knowledge of auto-starting locations of any startup monitor, shows you what programs are configured to run during system bootup or login, and what extension load into various Windows processes, including Explorer and Internet Explorer. It reports the image timestamp of executables, the last-modified timestamp of other file types, and the last-modified timestamp of the autostart locations. A "Hide Signed Microsoft Entries" option helps you to zoom in on third-party auto-starting images that have been added to your system. Autoruns works on Windows XP and higher, including 64-bit Windows.

Process Explorer v.16.2

Process Explorer is an advanced process management utility that picks up where Task Manager leaves off. It will show you detailed information about a process including its icon, command-line, full image path, memory statistics, user account, security attributes, and more. When you zoom in on a particular process you can list the DLLs it has loaded or the operating system resource handles it has open. A search capability enables you to track down a process that has a resource opened, such as a file, directory or Registry key, or to view the list of processes that have a DLL loaded.

SysTracer v1.0

SysTracer is a system utility tool that can scan and analyze your computer to find changed (added, modified or deleted) data into registry and files. SysTracer can scan your system and record information about:

- changed files and folders

- modified registry entries
- installed programs
- system services
- system drivers
- applications that are configured to run at computer startup
- running processes
- loaded dlls
- opened files, folders and registry
- opened TCP and UTP ports

PEid v.095

PEiD detects most common packers, cryptors and compilers for PE files. It can currently detect more than 600 different signatures in PE files. Also it identifies the characteristics of PE Files, uses signature based file-type identification

Bin Text v.3.0.3

Bin Text v.3.0.3 is extracting ASCII and Unicode strings from files. It is used to identify possible file behavior based on string. Moreover is used to identify malware variants based on string patterns

HashmyFiles

- Enables to get the hash given Files, Files in the folder, Files of for a given process
- Gives out hash values by MD5, SHA1, CRC32

VirusTotal

- Online analyze suspicious files and URLs to detect types of malware, additionally search of URL, IP address, domain or file hash.

InstallRite

- Tool used for keeping track of changes prior to the installation / execution of a program
- Monitors changes (additions, deletions and modifications) made on files and registry entries.

WireShark v.1.10.14

- Originally known as Ethereal
- Open-source GUI-based packed analyzer
- Cross-platforms : Windows, Linux, OSX
- Tool used to capture packets and browse network traffic
- Allows examination of the data sent

B64 Tool

stand-alone (command-line) utility for encoding /decoding files based .b64 UPX Tool

AspackDie 1.41

Unpacker for PE files (EXE, DLL) which is aspack compressed.

2.2 Dissecting information through static malware analysis

Static malware analyze is analysis of malicious specimen without running or executing the code. This is usually done by determining the obfuscation of malware specimen if it is packed or not packed. If it is packed it is decompressed by various tools and techniques. Decompression leaves backdoor on malicious code in which by string search there is dissecting information about executed commands, IP addresses, e-mails, messages etc which happens while execution flow of infected specimen. Additionally there is digital signature calculation of hash value, which later can be compared with AV data hash sets.

For demonstration purpose of static malware analysis in this section I used a sample with the file name "*malware1.exe*".

I go and right click to file properties of "*malware1.exe*" and get the *Figure 9* shown below.

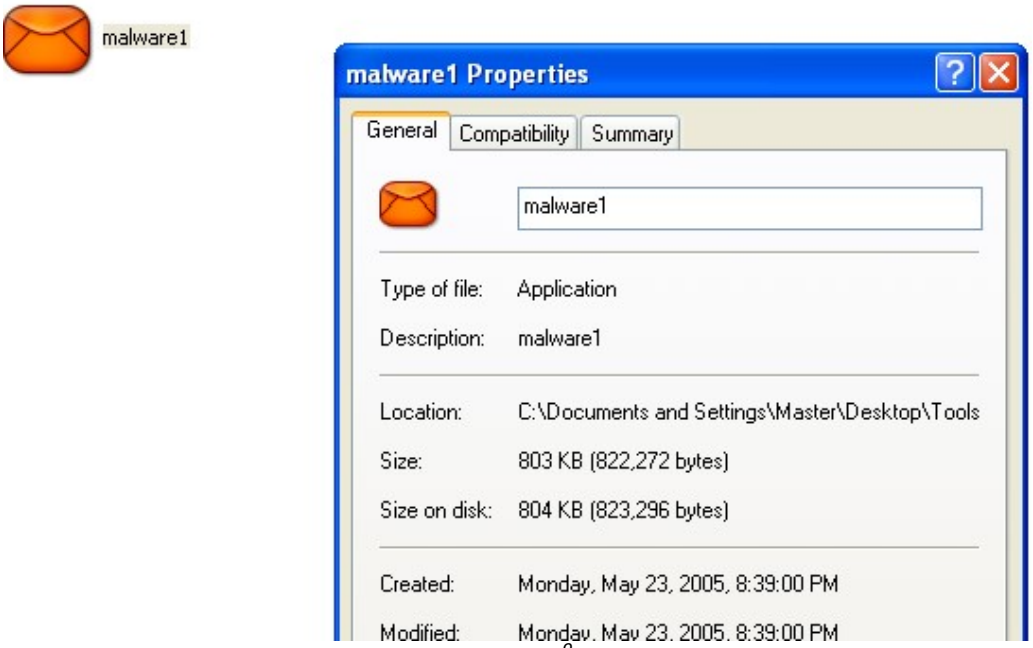


Figure 9[9]

According to the *Figure 9* I know that malicious file **malware1.exe** size of file is 803 KB (Kilobytes), additional information about the file type that is application, file description, location of file, and information when file is created. From this perspective of view I know that this file is malicious but I don't know what behavior of sample has and I don't know what malicious activities does over my system. So what I need to do is to get or extract more information from file. With the help of tool **PEiD** I can do more physical file analysis and file identification process. When I implement the tool I get the information shown *Figure 10*, below:

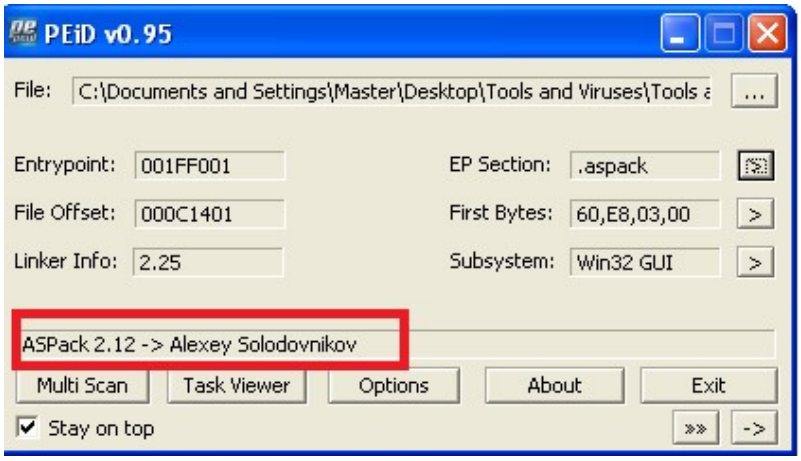


Figure 10[10]

[9] Figure 9, malware1 sample properties

Information extracted over malicious sample that I got is very useful and interesting indicator for the file itself. Redline block in image shows that this sample is packed with ASPack packer.

Most malware authors want their malicious applications to be protected from most antimalware products, and this can be done through the packers. With packers, it is sometimes difficult to reverse engineer the malicious sample to deduce its functionality. However as well all know that whenever a compressed, encrypted or packed malware needs to run, it will in decompressed or in decrypted form while in memory. When I see packed malware specimen I notice that it will reveal very limited information about itself, for instance, there is fewer strings that can be visible, very limited number of Import Address Table (IAT) calls, etc.

2.3 Unpacking malicious specimen

There are various packers available out there. ASPack is the famous among them and is commonly found as the packer of files. In the next section, I tried various tools that can help me to conclude that a specimen under investigation is packed. Note here that not all packed executables are malicious. Packing legitimate executables is a common practice.

Before to go unpacking my malware specimen I represented some additional information about packed files. With the help of PEiD I open my sample hexadecimal view as shown in *Figure 11* below.

[10] Figure 10 Analyzing malware sample with PEiD tool.

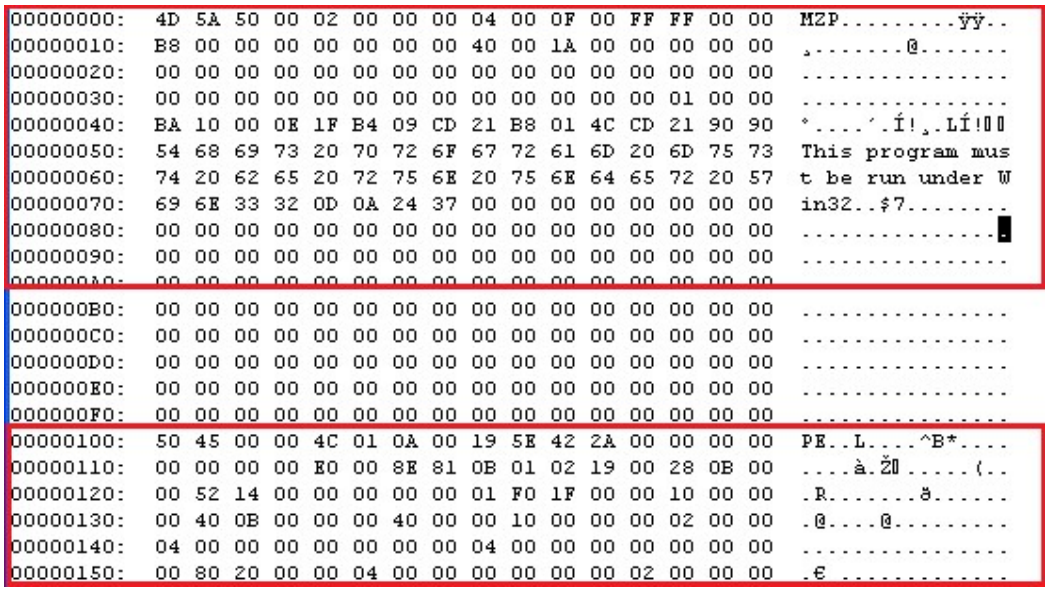


Figure 11 malware1.exe [11]

The leftmost one is the offset address pane; each number in the pane shows the address of the first byte of the corresponding line. The next is the Hex pane, which displays the file contents as an array of hex bytes. The rightmost ANSI pane shows the file contents as characters. Note that all three data panes show different representations of the same data. First red block data represent the MZ headers used in DOS. Second red block is PE header in which Portable Executable (PE) files can be identified by the presence of it.

Depending on the compiler, Windows, application is usually separated into different sections corresponding to program code, data and miscellaneous items (e.g icon, relocation, data, etc) as represented in *Figure 12*.

Name	V. Offset	V. Size	R. Offset	R. Size	Flags
BSS	000B7000	00002000	00042600	00000000	C0000040
.idata	000B9000	00003000	00042600	00001000	C0000040
.tls	000BC000	00001000	00043600	00000000	C0000040
.rdata	000BD000	00001000	00043600	00000200	C0000040
.reloc	000BE000	0000D000	00043800	00000000	C0000040
.rsrc	000CB000	00134000	00043800	0007DC00	C0000040
.aspack	001FF000	00008000	000C1400	00007800	C0000040
.adata	00207000	00001000	000C8C00	00000000	C0000040

Figure 12 PE file [12]

Commonly used section names or very default section names are: .text, CODE, DATA, .idata, .bss, .rdata, .edata, .reloc, .rsrc, as shown in figure above. About the PE file and its sections

[11] Figure 11, malware1.exe in PEiD hexadecimal view
[12] Figure 12 PE (Portable Executable) file sections.

we mentioned above and described them what data they carry. In addition, what I can note here is to mention about flags which are defining characteristics of the sections, raw offset is physical address of the section start, raw size is physical section size in hard disk, virtual offset is actual address when program is loaded in RAM by the windows PE loader and virtual size is actual size of section when it run in memory. In my case *.aspack* (as you realize it's not common used section name) virtual size is = 8000 (hex number) and raw size is = 7800 (hex number), that means virtual size is bigger than raw size because always there is a file alignment which is = 200 (hex number).

Malware specimen that was analyzed with *PEiD*, *EntryPoint* is hex value 1FF001 which is the starting point of program execution. But if I click to plugins of the tool and try to get generic original EntryPoint of program, I get the hexadecimal value 483644. This is another symptom or vector of packed malicious sample.

Before to go in unpacking my sample I described manual checking technique of packed samples as following:

- Renamed sections
- Compact contents
- Few APIs on its import table
- Big difference between the virtual size and the raw data size of each section (except the section where decompression routines lie)
- EntryPoint is not on the first section (for some samples)

If there is renamed sections, compact contents, few API's on its import table, big difference between (the virtual size and raw data) size of each section and EntryPoint it's not on the first section then it's very natural that the file is packed.

After representing the elements of packed files on manual checking for packed samples with help tool AspackDie vs 1.41. (a very easy and usable tool), I did left mouse clicking in which I selected the file and clicked OK button where packed specimen is decompressed to default name unpacked.exe. Here after output default name I renamed the specimen to

"*unpacked_malware1_aspackdie.exe*" to be better distinguished while comparison in further.

Actually there so many tools to unpack samples. Unpacking can be done while it is executed in memory or it can be done while it is in disk. Unpacking in memory means that malicious sample is in running state and it decompresses itself into memory with aim performing all its functionalities and in some situation it's a state where malicious specimen has weakness to be captured by AV products. Running state of malware and its analysis is represented in next topic of this work.

To continue on further analysis I did right mouse click to the file properties of "*unpacked_malware1_aspackdie.exe*" and get the *Figure 13*, as shown below.

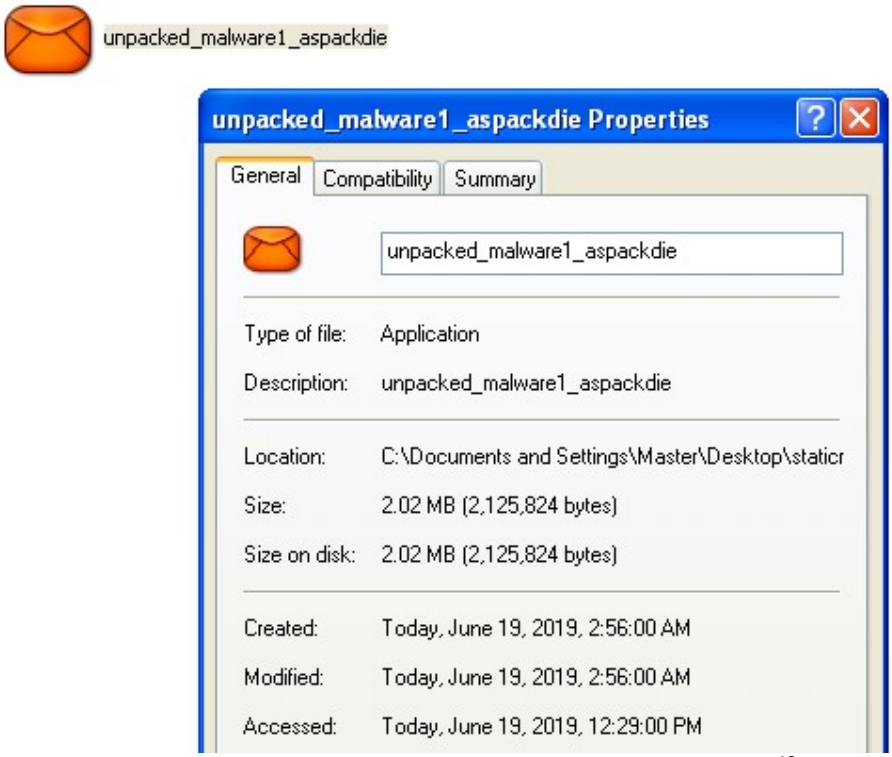


Figure 13 unpack_malware1_aspackdie.exe[13]

Information from properties tells me that this sample is application file, it has default Windows XP file description, location of file, size of file which is 2.02 MB, and some additional information when file is created etc.

[13] Figure 13 unpack_malware1_aspackdie.exe file properties.

In comparison unpacked file "*unpacked_malware1_aspackdie.exe*" vs packed file "*malware1.exe*", I find out that unpacked file with size 2.02 MB is bigger than packed file and in fact it has more user readable information for malware analysis purpose.

Using the tool **Hiew.exe** below its shown effects of packed file versus unpacked file.

Figure 14, shows compressed file content which is very compact and *Figure 15*, shows uncompressed file which has very loose content.

Packed content

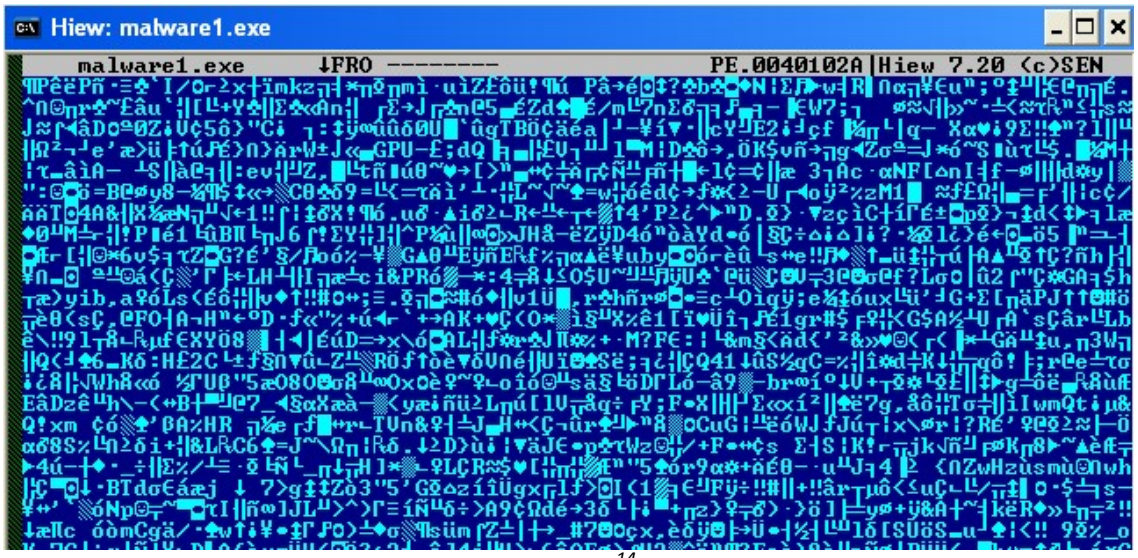


Figure 14[14]

VS

[14] Figure 14, packed file content

Unpacked content

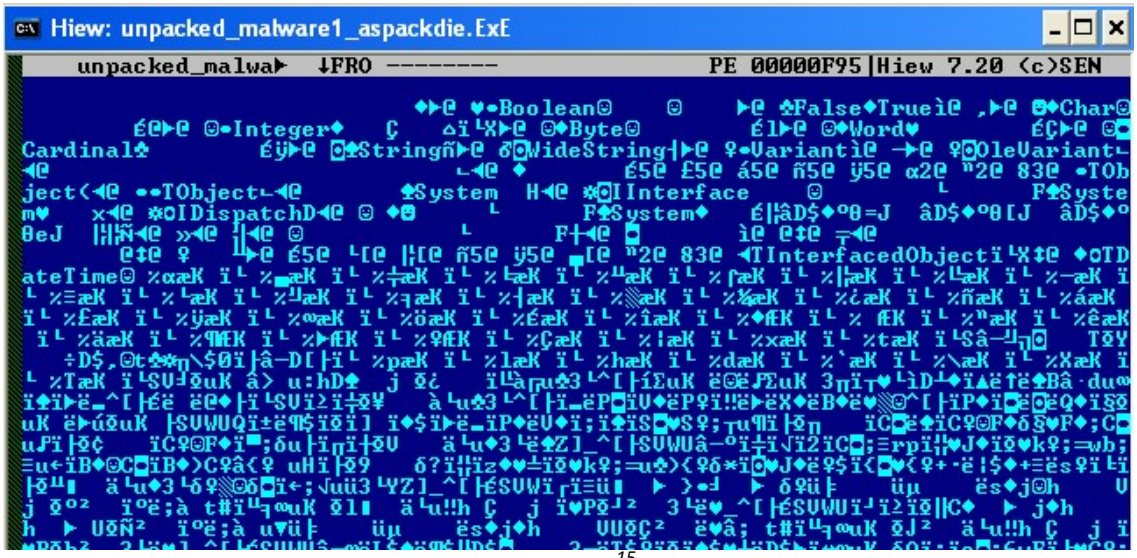


Figure 15[15]

2.4 Malware specimen Import Table

Next very important case while malware analysis is **Import Table** which contains the list of functions imported by the program from dynamic-link libraries. As I mentioned before packed samples shows few APIs on its **Import Table** than unpacked specimen.

Static malware analysis is dissecting information from died executable file. By looking and examination of imported functions I can gather more precise and flag positive information about malware itself.

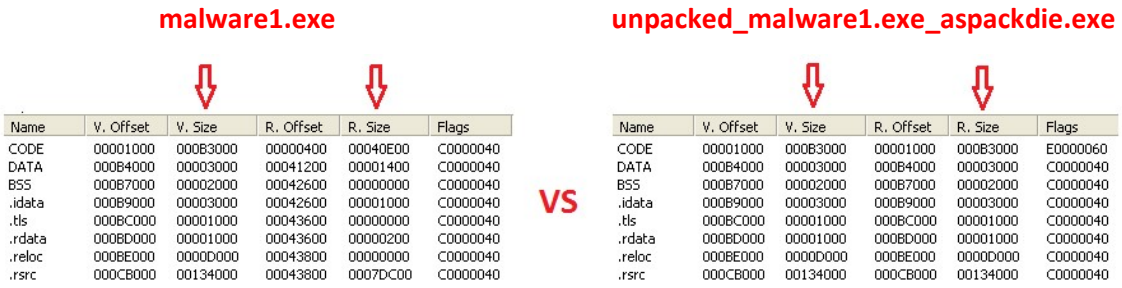
With the help of **PEiD** tool I just extracted data information and verified that my malicious specimen **malware1.exe** imports the following 13 (thirteen) dynamic-link libraries (DLL's) such as: kernel32.dll, user32.dll, advapi32.dll, oleaut32.dll, advapi32.dll, version.dll, gdi32.dll, user32.dll, oleaut32.dll, ole32.dll, oleaut32.dll, comctl32.dll, user32.dll with additional 15 (fifteen) functions.

My unpacked specimen **unpacked_malware1.exe_aspackdie.exe** imports 16 (sixteen) modules of dll's such as following kernel32.dll, user32.dll, advapi32.dll, oleaut32.dll,

[15]Figure 15, unpacked file content.

kernel32.dll, advapi32.dll, kernel32.dll, version.dll, gdi32.dll, user32.dll, kernel32.dll, oleaut32.dll, ole32.dll, oleaut32.dll, comctl32.dll, user32.dll, with additional 449 functions Note that there is repeating dynamic-link libraries but they exports different functions in repeating instances. I will not enter more about the imported DLL's details but there is online resources with detailed explanation about DLL's such as https://xpdll.nirsoft.net/ site built by scanning all DLL files located in system32 directory of Windows XP.

In further according to the fourth point of manual checking there is big difference between the virtual size and the raw data size of each section (except the section where decompression routines lie).



Name	V. Offset	V. Size	R. Offset	R. Size	Flags
CODE	00001000	000B3000	00000400	00040E00	C0000040
DATA	000B4000	00003000	00041200	00001400	C0000040
BSS	000B7000	00002000	00042600	00000000	C0000040
.idata	000B9000	00003000	00042600	00001000	C0000040
.tls	000BC000	00001000	00043600	00000000	C0000040
.rdata	000BD000	00001000	00043600	00000200	C0000040
.reloc	000BE000	0000D000	00043800	00000000	C0000040
.rsrc	000CB000	00134000	00043800	0007DC00	C0000040

Name	V. Offset	V. Size	R. Offset	R. Size	Flags
CODE	00001000	000B3000	00001000	000B3000	E0000060
DATA	000B4000	00003000	000B4000	00003000	C0000040
BSS	000B7000	00002000	000B7000	00002000	C0000040
.idata	000B9000	00003000	000B9000	00003000	C0000040
.tls	000BC000	00001000	000BC000	00001000	C0000040
.rdata	000BD000	00001000	000BD000	00001000	C0000040
.reloc	000BE000	0000D000	000BE000	0000D000	C0000040
.rsrc	000CB000	00134000	000CB000	00134000	C0000040

Figure 16[16]

As you can realize from the *Figure 16* above packed sample **malware1.exe** in each section has bigger virtual size than raw data size. In the other hand **unpacked_malware1_aspackdie.exe** nearly all the sections has similar virtual size and raw size. One of the main characteristics of packed malware is the big difference between virtual size and raw data size.

As stated above packed **malware1.exe** sample, starting point of execution **EntryPoint** hex value is **1FF001** and original **EntryPoint** of program in which with help of the tool I got packed sample has hex value **483644**. Do not forget that **PEiD** tool do not gets always the original EntryPoint number maybe there is need to implement additional disassembler tool and analyze more detailed on instruction byte code to get this data more accurate.

[16] Figure 16, Virtual size and Raw size difference.

Actually main spot here is as you can realize that **malware1.exe** has entry point outside the code (as specified in the PE header). This means that this file is self-extracting or self-modifying and its one the famous characteristic of malware.

When I check unpacked specimen **unpacked_malware1_aspackdie.exe** with PEiD tool as shown in figure *Figure 17*, I realize that the file was compiled in Borland Delphi version 6 or 7, additionally there is **EntryPoint** in hex value **83644** and original **EntryPoint** hex number hex value **483644**, which is not outside the code.

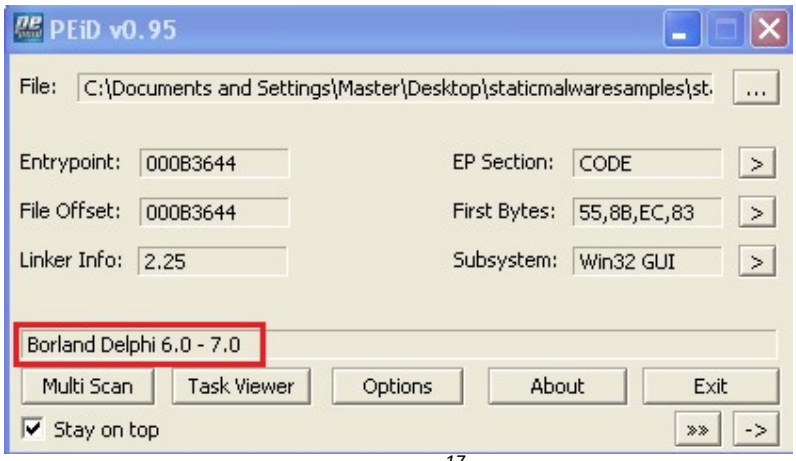


Figure 17[17]

The Original Entry Point is a concept typically referred to in reverse engineering for an executable that has been modified by some means such as being compressed (or encrypted) by a packer or infected with malware. Prior to modification, the entry-point of an executable is the original entry point (OEP). When an executable has been modified, such as to include a stub of code that runs prior to the original code, the entry-point of the executable is changed to point to the new code. The stub then references the old entry-point when it is done. So once the stub runs, it will transfer control to the address of the original entry point so the modified program still works (or appears) to work as normal. (byteptr, 2017)

Malware applications using the availability of entry point modifications can embed or link itself to the Windows XP processes and conduct modifications over the system.

[17] Figure 17, unpacked specimen **unpacked_malware1_aspackdie.exe** with PEiD tool.

2.5 Searching strings over malicious sample

Following step that I got string values of unpacked specimen with the purpose to dissect some useful information about malware code itself. There is many tools you can do this, but here I used **BinText** application. This tool extracts text from any kind of file and includes the ability to find plain ASCII text, Unicode (double byte ANSI) text and Resource strings, providing useful information for each item in the optional "advanced" view mode. Its comprehensive filtering helps prevent unwanted text being listed. Just to remind that previously we got information that unpacked specimen **unpacked_malware1.exe_aspackdie.exe** imports 16 (sixteen) modules of dll's such as following kernel32.dll, user32.dll, advapi32.dll, oleaut32.dll, kernel32.dll, advapi32.dll, kernel32.dll, version.dll, gdi32.dll, user32.dll, kernel32.dll, oleaut32.dll, ole32.dll, oleaut32.dll, comctl32.dll, user32.dll. with 449 functions.
Here I made analysis by picking out some of the imported functions from the list which are more used and have more relevance to malware/the detections from AV vendors.

Before to go in next step I need to state that there are acknowledged processes containing legitimate DLL's, acknowledged processes containing malicious DLL's, malicious processes containing legitimate DLL's and malicious processes containing malicious DLL's. Beyond deep analysis I can't select out functions like **ExitProcess** and conclude that it tries to exit or close an external process (since this function is being used by default for the current program to ends the calling process and all its threads). Alike for the function **ReadFile** , I cannot just undertake that the sample reads data from the specified file or input/output (I/O) device, because this had nothing to do with malicious specimen code but it was relating to the auto-generated code which was built with the compiler for the pre-after code execution. Therefore, if I cannot reach the required findings I should consider these functions for future and further analysis (static analysis techniques or dynamic analysis technique), with the goal to import of these functions not to be skipped false fact findings.

Static malware analysis its analyzing malicious code on a disk or can be called offline analysis. Malicious code is not executed to track its activities on disk or in memory. But it's just studying information as more as possible without executing. After the search with

BinText tool, from dozens of data I extracted (shown in the Table 1) some of them for further analysis which helped me to find digital footprints of a malicious specimen.

File hex position	Memory hex Position	Text
0000000BA210	0000004BA210	CreateFileA
0000000BA1DA	0000004BA1DA	DeleteFileA
0000000BA22E	0000004BA22E	CopyFileA
0000000BA15E	0000004BA15E	FindFirstFileA

Table 1[18]

Selected strings and information on *Table 1* about krernel32.dll library and its imported functions with previously knowledge that not all applications has ability to create, delete or add a file, I consider such functions as suspicious and I consider that these won't be added in automatically by the compiler therefore I conclude that it must have been added by the programmer.

Here I know that with function **CreateFileA,** malware specimen tries to create or open a file or I/O device during execution flow. Function **DeleteFileA** deletes an existing file that malware do not want to be seen or tries to damage any system file. Following function **CopyFileA** tries to copy an existing file to a new file, which is malicious characteristic of malware specimen while reproducing itself to be persistent in disk. Finally function **FindFirstFileA** malware searches a directory for a file or subdirectory with a name that matches a specific name (or partial name if wildcards are used), if there is no match malware reproduces itself as much as many more locations by combining the commands.

Funcion **MessageBoxA** which is Win32 API function (an import from user32.dll library), during the execution flow displays a modal dialog box that contains a system icon, a set of buttons, and a brief application-specific message, such as status or error information. Malware sometime try to block users with repeating popups.

Next I can see that there are many registry functions imported: RegCreateKeyExA, RegCloseKey, RegQueryValueExA, RegFlushKey, RegSetValueExA, RegOpenKeyExA and others. This tells me that the sample most likely interacts with the registry for whatever purpose at some point during execution flow. The functions RegCreateKeyExA(from Advapi32.dll library) will allow the sample to create a new key in the registry. Based on first

[18] Table 1, Extracted strings using BinText tool.

view I would suspect that the sample would be using this to allow itself to auto-start at boot, although without further analysis I can only assume. Based on some of the functions I listed above with relevance to registry operations, I know that the sample is capable of: deleting registry keys, querying info from registry keys, modifying the value of existing registry keys, and closing the handle to a registry key it had opened. You can find more information about Windows API registry functions at the following page online (https://docs.microsoft.com/en-us/windows/desktop/api/ or you can check at Apendix A, Important Windows Functions page 453, Practical Malare Analysiss, Apendix A, Michael Sikorski and Andrew Honig).

Next interesting string that I found is **svchost.scr**. Malicious specimen during the execution flow will execute file that has been identified as a program and is very famous in malware community. It's known that this startup entry is started automatically from a Run, RunOnce, RunServices, or RunServicesOnce entry in the registry. Gathered additional data about my specimen tells me that in some point has changed registry entries and auto-start locations, in which makes me consider as a suspicious or malicious activity.

Very important string that I found during **BinText** search is **smtp.bhi.com.br** which tells me malware during execution flow uses SMTP (Simple Mail Transfer Protocol) and tries to send an e-mail to manyall@mail.com account in which I got extracted during **BinText** search.
All information gathered by BinText above is alluding that my malware specimen **unpacked_malware1.exe_aspackdie.exe** can be classified as Backdoor Trojan which tries to capture keyboard or monitor activities and sends data using SMTP mail protocol to the unwanted locations or e-mail address.

Important issue to mention while static malware analysis is hashing. Hashing is getting a digital finger print of file such as using MD5 or SHA-1 hashing. Before to start to investigate any file in computer systems, digital forensics expert creates exact forensic copy of original one. Beyond to create forensic copy of a hard disk, forensic image or file, the expert first task is to calculate hash value of original data and then create forensic copy and calculate the hash value of copied image and if it is match the data integrity is guaranteed. So this is what AV companies and tools does. They collect data sets hashes of infected files and then

compare to every file hash value that their engines has. If there is matching hashes the result is flag positive and automatically take the action to quarantine or delete the infected file. That's the Anti-Virus companies does.

Here by using the tool **HashMyFiles** I just got the MD5 hash values of packed **malware1.exe** specimen and unpacked **unpacked_malware1_aspackdie.exe** sample as represented following *Table 2*.

Filename	MD5 hash
malware1.exe	fa4b88391e7129e8f2e08c61cefe517e
unpacked_malware1_aspackdie.exe	6e90de67b76a4e220c1f6e070d94313c

Table 2[19]

One of the best tools in the market is VirusTotal which aggregates many antivirus products and online scan engines to check for viruses that the user's own antivirus may have missed, or to verify against any false positives. VirusTotal.com is a free online scanning web service and allows file uploading scan, URL scan, search for IP address, domain or file hash checking done with more than 40 antivirus solutions.

Beyond this I conucted hash search of **malware1.exe** specimen using MD5 value *fa4b88391e7129e8f2e08c61cefe517e* and VirusTotal give me feedback that 52 engines of 67 has detected this file as malicious and mostly of them categorize this specimen as Trojan.

The tab relations gives information about two IP addresses. First one is **149.56.69.2011** company name 16276 –OVH SAS with country origin Canada and second IP **50.97.52.2019** , 36351 – SoftLayer Technologies Inc, with company orgin USA. Additionally there is one domain contacted **smpt.bhi.com.br** which actually I found above using BinText search.
Next, I did MD5 hash search with the value *6e90de67b76a4e220c1f6e070d94313c* of unpacked specimen **unpacked_malware1_aspackdie.exe**, and VirusTotal says no matches found. This is because last hash value of my specimen is not in VirusTotal data hash set. When I unpacked my specimen actually I created new file and new files always has new

[19] Table 2, hashing malicious specimens.

hash values, Finally I uploaded unpacked malicious file into VirusTotal and did file search. As result I got hit flag positive of 52 engines from 71, in which my specimen was classified as Trojan malware.

2.6 Automatization of Static Malware Analysis

Every day, the AV-TEST Institute registers over 350,000 new malicious programs (malware) and potentially unwanted applications (PUA). These are examined and classified according to their characteristics and saved. Visualization programs then transform the results into diagrams that can be updated and produce current malware statistics. (The Independent IT-Security Institute Magdeburg Germany, 2019)

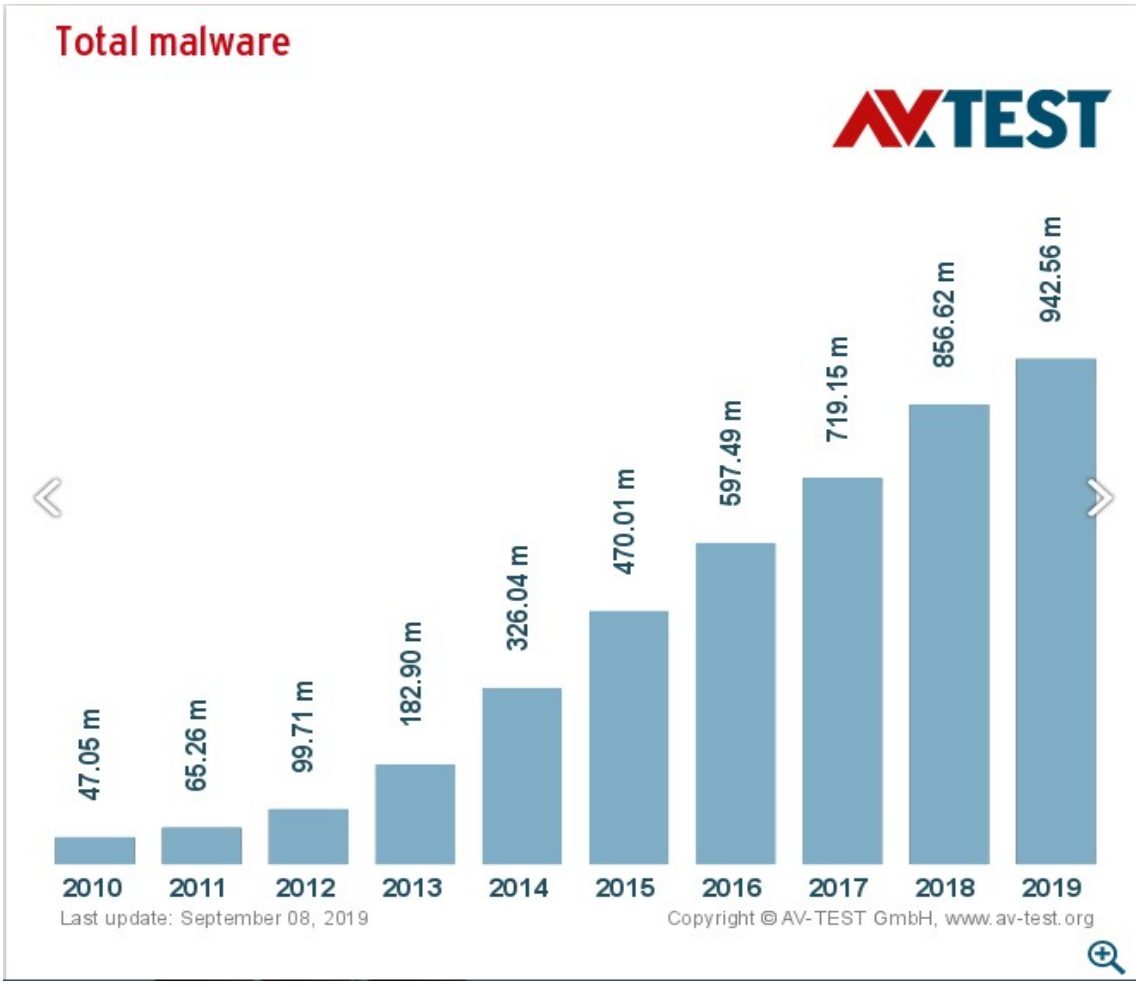


Figure 18 Malware statistics[20]

[20] Figure 18, Total Malware statistic from 2010 – 2019, according to Independent IT-Security Institute, Magdeburg Germany. [Image is take from web page https://www.av-test.org/en/statistics/malware/ at 09.09.2019 year]

As presented according to Independent IT-Security Institute Magdeburg, Germany they register every day 350,000 malware specimens. Additionally according to the Figure 18 yearly statistic from period 201-2019, shows visual rising diagram of how malware numbers are increasing drastically. Without looking the statistics how many malware is created for Windows, Android, IOS or any another operating system, implementing static analysis for every created specimen one by one is impossible.

Mostly AV companies develop hash set database of found malware, and automatic classifying of malicious specimen according to their characteristics and according to the different vectors.

Here I write out a simple automation procedure or a flowchart of static malware analysis in which it can be used to combine, write methodology and create algorithm to perform automatic static malware analysis.

Flowchart is listed in **Appendix A** in this work and actually is summary of five basic points on static malware analysis could be implemented with automated and previously described steps, a gathered intelligence from practice.

1. File properties checking because most malware :
   - No presents of a Digital Signature,
   - Incomplete version of Information,
   - Uses similar icons to office applications, folders or any icons that will make the malware look.
2. By knowing that common packers such as (upx, aspx upack etc) and common compilers such as (Borland C++, C#, Visual C++/.NET), there is need to be check if is packed or unpacked.
3. If it is packed infected specimen could be unpacked using three ways
   - Using the unpacker tools
   - Dump the process while in memory
   - Tracing sample using debugger tools
4. After unpack use BinText toolto get a hint or initial analysis of what the file sample do such as:
   - Filenames;
   - Registry keys;
   - IP addresses;

- E-mail subject and body;
- API imports.

5. Validate findings:
    - Using hashmyfiles tool get digital foot print or hash value of file;
    - Perform hash checking for flag positive using online tools such VirusTotal.com or any another AV Vendor tools.

Dynamic Malware Analysis on previously captured SMTP packets

Dynamic analysis is the process of executing malware in a monitored environment to observe its behaviors. This technique can quickly yield information such as created files, created registry keys, contacted websites, and so on. If you're not an experienced IDA Pro user or simply don't have time to perform a thorough static analysis of the code, you can use dynamic analysis to get a quick initial perspective of the malware's capabilities. (Ligh, Adair, Harstein, & Richard, 2011, p. 283)

Dynamic malware specimen analysis is main aim of the topic. Dynamic or as I call live malware analysis is process of executing malicious specimen in sterilized digital lab environment, to track behaviors of malicious specimen and malicious impacts onto the system. Behavioral analysis has very sensitive nature which explains malware installation on the machines via file system, registry and process. While Static analysis is looking for suspicious strings and commands, Dynamic analysis is process of looking for the suspicious files, looking for the suspicious processes to find suspicious files, and looking for suspicious registries to obtain suspicious file.

The scenario here is analyzing previously captured network packet with the file name **smtp.cap** and hash value **aff1528eaee1ea8d948192fac16d8db1** as a unique digital signature of file. Very comprehensive network traffic capture tool is WireShark. With the help of this tool I browsed my sample to demonstrate extracting malicious attachment from an e-mail SMTP traffic. After I extracted the malicious attachment sample I run it to obtain suspicious files, processes and registries. Finally at the end with the data gathered I did classification of my malware specimen.

3.1 Extracting the E-mail from Network Packet

I right click with mouse on my demonstration specimen **smtp.cap** and open it in WireShark tool to investigate it. The "Packet List" pane of the tool shows 474 captured packets. All packets has number, timestamp, source the address where this packet is coming from, destination the address where this packet is going to, the protocol name, length of each packet and additional information about the packet content. When packet is selected the "Packet Details" pane show more detailed form of packet such as protocols and protocol fields. Additionally the "Packet Bytes" pane with tabs shows data reassembled from multiple packets or decrypted data as shown in *Figure 18*. WireShark tool has so many features, but given details is enough on understanding this demonstration.

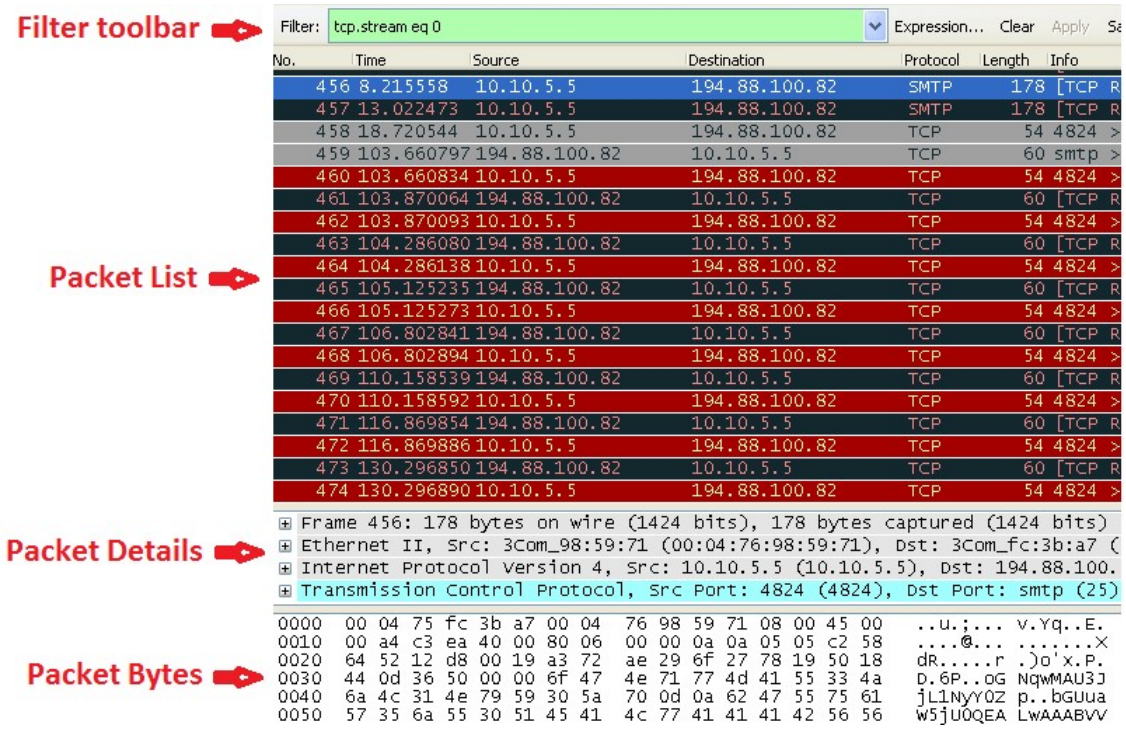


Figure 19[21]

As I said above the tool gives me 474 captured packets which I need to look for. When I started to analyze one by one all the packets I realized my suspicious communication between two IP addresses:

[21] Figure 19, WireShark tool panes.

1. IP address 10.10.5.5
2. IP address 194.88.100.82

I know that my host IP address is 10.10.5.5 a private IP address from Class A, which is used as a separate set of address from companies, governments etc. Second IP address 194.88.100.82 is out of range from private IP addresses so I investigate it online doing Whois search. The result of the search tells me that this IP address location is United States, Walnut issued to "Noop Llc" company. During cybercrime investigation IP address information is very important because it's legal issuer where you can request data preservation for further ingestion.

Second important information about the communication of these to IP's is that they are using TCP (Transmission Control Protocol) and SMTP (Simple Mail Transfer Protocol). First three packets show established connection between client IP address 10.10.5.5 send SYN (Synchronization) flag to the server IP address 194.88.100.82. Server send SYN-ACK (Synchronization – Acknowledged) flag and client responds again ACK flag.

After connection established, from packets obviously I found that there is an e-mail exchange between the client and server through SMTP (a protocol responsible for delivery of mail). Because of that I filter out "SMTP" protocol only and click apply. In WireShark you can do various type of filtering or advanced filtering to the packet for further analysis but I will stop here because it's enough for my purpose. After the filtering I receive 337 packets or 71.1% of total communication between the client and server.

Next I right click on the "SMTP Frame" and Select "Follow TCP Stream" as presented following *Figure 19*. I choose only one direction of the conversation, the client with IP address 10.10.5.5 to the server with IP address 194.88.100.82, because I want to check only the sent packets my client (infected host machine) to server machine.

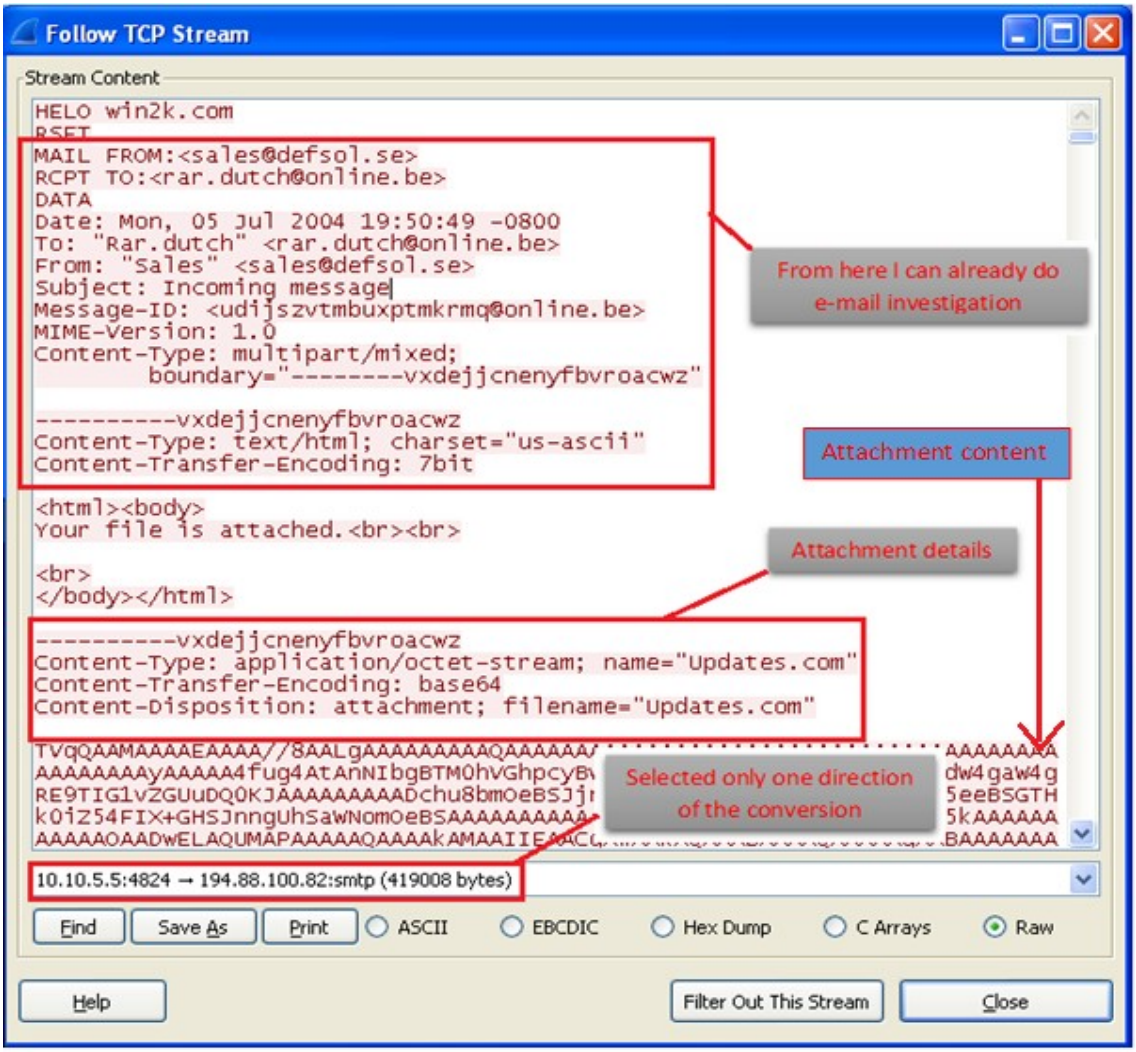


Figure 20[22]

By looking the e-mail header I got some additional information and I already do an e-mail investigation. If you are not familiar with email header analyzing you can use online email header analyzers.

> *"MAIL FROM:<sales@defsol.se>*
> *RCPT TO:<rar.dutch@online.be>*
> *DATA*
> *Date: Mon, 05 Jul 2004 19:50:49 -0800*
> *To: "Rar.dutch" <rar.dutch@online.be>*
> *From: "Sales" <sales@defsol.se>*
> *Subject: Incoming message*
> *Message-ID: <udijszvtmbuxptmkrmq@online.be>"*

[22] Figure 20, E-mail extraction from Network Analysis.

Above headers tells me there is an email exchange between the victim e-mail address sales@defsol.se as a sender and suspicious e-mail adress rar.dutch@online.be, as a receiver. E-mail exchange happened with Subject “Incoming message” on 05 July 204 19:50:49 and time zone -0800. Significant to note that email header data is very important while cybercrime incident investigation because, using the mutual legal assistance we can get personal information about the owner of suspicious e-mail (receiver in my case).

If I go forward and check the body content of exchanged e-mail, there is information about attachment details as shown below.

> “----------vxdejjcnenyfbvroacwz
> Content-Type: application/octet-stream; name="Updates.com"
> Content-Transfer-Encoding: base64
> Content-Disposition: attachment; filename="Updates.com"”

Reading the attachment details tells that there is an application sent which is encoded bas64 (binary to text encoding scheme), and there is file name “Updates.com”. To extract attachment from e-mail I just select the attachment data content (see *Figure 19*) on TCP stream box and save raw data as a new file using “extrack_email.**b64**”, “.b64” as the file extension. In order to decode attachment I used b64 tool. In command-line I executed following code: “b64 –d extract_email.b64 update.exe”, in which I decoded extract_email.b64 file to binary file update.exe.

3.2 Executing the malicious binary code

In this stage I am ready to execute the malicious code on the Lab by following the safe malware handling policy mentioned in second section. The tool that I will use before to run the malware specimen Installrite 2.5c. This tool used for keeping track of changes prior to the installation and the changes after execution of a program. Additionally monitors changes (additions, deletions and modifications) on files and registry entries. I click the button ‘Create a manual “Snapshot” of your PC and I get the current picture of my system. Then I run a malware specimen and click a button ‘Perform an “Analysis” of the changes since the last “Snapshot”’. At the end I click the ‘Review Installations’ button and I get following data as a result:

The installation performed the following activity:
6 files added
2 files deleted
4 files updated
7 registry entries added
0 registry entries deleted
5 registry entries updated

Installed 6/24/2019 12:18:10 PM

3.3 Technical Details of malicious specimen “update.exe”

After execution of my malware specimen I monitored the following changes of my isolated system.

Created following files:

C:\Documents and Settings\Master\Local Settings\Temp\Perflib_Perfdata_530.dat

C:\WINDOWS\Prefetch\LOADER_NAME.EXE-384A43FB.pf

C:\WINDOWS\Prefetch\UPDATE.EXE-34320991.pf

C:\WINDOWS\system32\loader_name.exe

C:\WINDOWS\system32\loader_name.exeopen

C:\WINDOWS\system32\loader_name.exeopenopen

Deleted following files

C:\System Volume Information

C:\WINDOWS\SoftwareDistribution\DataStore\Logs\tmp.edb

Added following Registry entries

ID	Key	Value	Data
1	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{75048700-EF1F-11D0-9888-006097DEACF9}\Count	HRZR_EHACNGU:P:\Qbphzragf naq Frggvatf\Znfgre\Qrfxgbc\QlanzvpZnyjnerFnzcyrf\QlanzvpZnyjnerFnzcyrf\hcqngr.rkr	hex:01,00,00,00,06,00,00,00,40,31,f2,83,c1,2a,d5,01,
2	HKEY_CURRENT_USER\Software\Microsoft\Windows\ShellNoRoam\MUICache	C:\Documents and Settings\Master\Desktop\DynamicMalwareSamples\DynamicMal	"update"

		wareSamples\update.exe	
3	HKEY_CURRENT_USER\Software\Microsoft\Windows\ShellNoRoam\MUICache	C:\WINDOWS\system32\loader_name.exe	"loader_name"
4	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\kmixer\Enum	0	"SW\{b7eafdc0-a680-11d0-96d8-00aa0051e51d}\{9B365890-165F-11D0-A195-0020AFD156E4}"
5	HKEY_USERS\S-1-5-21-1202660629-706699826-1343024091-1003\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{75048700-EF1F-11D0-9888-006097DEACF9}\Count	HRZR_EHACNGU:P:\Qbphzragfnaq Frggvatf\Znfgre\Qrfxgbc\QlanzvpZnyjnerFnzcyrf\QlanzvpZnyjnerFnzcyrf\hcqngr.rkr	hex:01,00,00,00,06,00,00,00,40,31,f2,83,c1,2a,d5,01,
6	HKEY_USERS\S-1-5-21-1202660629-706699826-1343024091-1003\Software\Microsoft\Windows\ShellNoRoam\MUICache	C:\Documents and Settings\Master\Desktop\DynamicMalwareSamples\DynamicMalwareSamples\update.exe	"update"
7	HKEY_USERS\S-1-5-21-1202660629-706699826-1343024091-1003\Software\Microsoft\Windows\ShellNoRoam\MUICache	C:\WINDOWS\system32\loader_name.exe	"loader_name"

Modified following entries

ID	Key	Value	Data Before	Data After
1	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{75048700-EF1F-11D0-9888-006097DEACF9}\Count	HRZR_EHACNGU	hex:01,00,00,00,11,00,00,00,f0,85,fb,5d,c0,2a,d5,01,	hex:01,00,00,00,12,00,00,00,40,31,f2,83,c1,2a,d5,01,
2	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\RNG	Seed	hex:f0,bd,f5,8b,f8,07,1d,55,cc,b0,f5,e0,2f,b8,92,df,8f,6a,cb,bf,f2,10,46,41,61,58,ff,c0,80,f8,0d,c3,76,49,e1,79,92,c2,6b,d9,78,6b,1f,b8,0e,6d,53,88,ca,d2,a2,f8,cb,d8,17,e5,a4,55,55,fc,85,3f,78,89,3a,61,3e,83,2b,b6,a0,23,ea,c3,26,e7,41,df,74,a1,	hex:58,19,a8,98,30,41,98,f9,5c,d3,03,23,e8,f3,75,ae,e4,ec,a6,de,a8,c4,8a,f0,0c,d3,b3,bd,f9,99,31,03,d4,5e,ab,6f,ce,11,71,ca,0b,8c,71,1f,ae,8c,09,2b,98,02,f3,46,0c,b7,e4,3b,86,c9,a4,37,49,8c,15,68,2c,56,1d,e1,69,69,7e,a9,bb,cb,64,cd,84,ce,29,c1,
3	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\kmixer\Enum	Count	dword:00000000	dword:00000001

4	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\kmixer\Enum	NextInstance	dword:00000000	dword:00000001
5	HKEY_USERS\S-1-5-21-1202660629-706699826-1343024091-1003\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{75048700-EF1F-11D0-9888-006097DEACF9}\Count	HRZR_EHACNGU	hex:01,00,00,00,11,00,00,00,f0,85,fb,5d,c0,2a,d5,01,	hex:01,00,00,00,12,00,00,00,40,31,f2,83,c1,2a,d5,01,

As you can see after execution I got the behavior of my malicious binary code. It resulted several changes where many files added, updated, deleted and some registry entries changed. While installing or executing application, file adding in some cases could be normal but file deletion without prompting to user can't be classified as normal Windows application behavior. Likewise in every installation in Windows systems there is registry changes related to the application, but there is no registry entry adding, deleting and updating not related to the prior installation.

The following files

C:\WINDOWS\system32\loader_name.exe

C:\WINDOWS\system32\loader_name.exeopen

C:\WINDOWS\system32\loader_name.exeopenopen

are added to very critical part of the Windows XP operating system location *C:\WINDOWS\system32* where important system file are stored. Malicious file "**loader_name.exe"** spreads itself by expanding with "**open**" string after the "**.exe**" to the disk memory.

Modified and added registry entries are so suspicious. It seems that malicious specimen through registry manipulation in Windows XP tried to read Internet Explorer history and cookies. Additionally It ads malicious file and path to the registry.

I know my testing tool works well but sometimes malicious code knows to hide itself in registry entry values such as file-less malware, roots etc, and some data changes could not be tracked just with one tool. Additionally different type of malware acts different in

isolated environment than client server environment. Moreover in digital data investigation preferable way is to use second tool implementation in order to verify results at least from two different sources.

To get better picture I tested my specimen tool SysTracer v1.0, in which enables me to run "update.exe" as new process and trace the changes into the system. The following behavioral findings or technical details are gathered after execution:

Creates the following file to windows system folder location

C:\WINDOWS\system32\loader_name.exe

Added following registries

ID	Registry Key	Operation	Value	Data
1	HKCU\Software\Microsoft\Windows\CurrentVersion\Run	set registry value	reg_key	C:\WINDOWS\system32\loader_name.exe
2	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders	set registry value	Personal	C:\Documents and Settings\Master\My Documents
3	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\{07dba273-15b8-11e8-b769-806d6172696f}	set registry value	BaseClass	Drive
4	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\{07dba271-15b8-11e8-b769-806d6172696f}	set registry value	BaseClass	Drive
5	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders	set registry value	Common Documents	C:\Documents and Settings\All Users\Documents
6	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders	set registry value	Desktop	C:\Documents and Settings\Master\Desktop
7	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders	set registry value	Common Desktop	C:\Documents and Settings\All Users\Desktop
8	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap	set registry value	UNCAsIntranet	0
9	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap	set registry value	AutoDetect	1
10	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders	set registry value	Cache	C:\Documents and Settings\Master\Local Settings\Temporary Internet Files
11	HKCU\Software\Microsoft\Windo	set registry	Cookies	C:\Documents and

	ws\CurrentVersion\Explorer\Shell Folders	value		Settings\Master\Cookies
12	HKCU\Software\Microsoft\Windows\ShellNoRoam\MUICache	set registry value	C:\WINDOWS\system32\loader_name.exe	loader_name

File loader_name.exe in system folder creates, two more files into the system

C:\WINDOWS\system32\loader_name.exeopen

C:\WINDOWS\system32\loader_name.exeopenopen

File "loader_name.exe" sets the following value "C:\WINDOWS\system32\loader_name.exe" to the registry key "HKCU\Software\Microsoft\Windows\CurrentVersion\Run". This confirms the theory of auto start locations that we mentioned in first chapter. My malware specimen wants to run always when system is on.

Aditionally it attempts to create copies of itself in any folder that contains the characters **"shar"**. The files will have the following file names:

Microsoft Office 2003 Crack, Working!.exe
Microsoft Windows XP, WinXP Crack, working Keygen.exe
Microsoft Office XP working Crack, Keygen.exe
Porno, sex, oral, anal cool, awesome!!.exe
Porno Screensaver.scr
Serials.txt.exe
KAV 5.0
Kaspersky Antivirus 5.0
Porno pics arhive, xxx.exe
Windows Sourcecode update.doc.exe
Ahead Nero 7.exe
Windown Longhorn Beta Leak.exe
Opera 8 New!.exe
XXX hardcore images.exe
WinAmp 6 New!.exe
WinAmp 5 Pro Keygen Crack Update.exe
Adobe Photoshop 9 full.exe
Matrix 3 Revolution English Subtitles.exe
ACDSee 9.exe

Additionally I tested my malware in Client-Server structure by date manipulation of my system and with the help of tool of Wireshark I captured 1880 network packets. I filtered out capture .pcap file using “tcp” filter and I followed TCP stream (client to server) where I realized that my malicious specimen using TCP Port 53 (Port 53 is used by the Domain Name System (DNS), a service that turns human readable names like Google.com into IP addresses that the computer understands. Because port 53 is usually open, malicious programs may attempt to communicate on it) tries to reach some of the following sites:

practicalmalwareanalysis.com
zzz.org
aaa.zzz.org
ddd.com
digicool.com
python.org
anthem.python.org
mail.example.com
zope.com
cravindogs.com
xx.dk
mail.groupcare.dk
cougar.noc.ucla.edu
socal-raves.org
oxy.edu
ucla.edu
babylon.social-raves.org
dom.ain
example.com
zinfandel.lacita.com
wellpartner.com
bb.org
dd.org
ietf.org
bounce2.pbox.com
dot.ca.gov
shore.net
kemtel.ru
bigfoot.com
hello.com
innoncent.com
gmx.net
mirmax.cbs.dk
WINDOWS.GUI.ASM32.ELITE.CODER.COM
GRAPHICS.DESIGNER.COM
oberhumer.com
users.sourceforge.net
BitWagon.com
x-ray.at
whireshark.org
winpcap.org
lists.tcpdum.org
lists.tcpdum.org
sequent.com
alumni.rice.edu
alumni.mit.edu
msdirectservice.com
lebanon-online.com.lb
gto.net.com
bar.baz
bar.foo

3.4 Hybrid-malware analysis

Hybrid-malware analysis is simultaneous analyze of malware specimen static and dynamic. Because above I did behavioral analyze I want to search more about my malicious specimen. First I find with the help of tool **PEiD** that loader_name.exe is **UPX** packed. Using command line tool upx for encoding/decoding binary files I execute the following command: **upx –d loader_name.exe –o Unpacked_loader_name.exe**. Second I continue to analyze my specimen using Bintext tool to get more input strings.

In a difficult world
In a nameless time
I want to survive
So, you will be mine!!
-- Bagle Author, 29.04.04, Germany.

This is the message what malicious specimen has left in binary code. I did search in VirusTotal using the hash value 3495a19bf6275f6e86f7dad561978fbc of “loader_name.exe” suspicious file so from 70 engines I have 55 engines as virus positive categorized as an e-mail worm with name Beagle.32 such as its author left message.

3.5 Automatization of Dynamic Malware Analysis

The citation of the Independent IT-Security Institute Magdeburg, Germany mentioned in second chapter was that they register over 350,000 new malicious programs every day. This huge number of malware developed by actors need to be considered as serious attack over privacy of home and business users. Every single moment a smart device, personal pc or server is under attack with high risk of information to be stealth or business to be interrupted. Dealing such a type of massive invasion of security risks requires to develop a methodologies or automatic responses.

Detection live attacks and performing dynamic analysis requires automatic response. Additionally automatic responses must be enriched using artificial intelligence. Machine learning implementation will be time consuming, effective cost and very strong response for eliminating the rising malware threats. Zero day exploits and validation will be always challenge to this discipline but at least real time detection, classification of malware, automatic generation of reports is existing on market and so many business solutions are provided by many AV vendors.

In this section I tried to find out and develop dynamic malware analysis flow chart through six points, in which all steps could be used and combined to generate automatic response algorithm for rising risks of malicious activates.

The flowchart is listed in **Appendix B** as a part this work and it's generated from previously gathered practice and learnings. All six points mentioned here is very good approach of summarizing of dynamic malware analysis in one single chart.

1. Real-Time traffic capture using WireShark tool which requires extracting content of captured packets and decoding them if the content is encoded.
2. Observation of infection symptoms by executing malware specimen inside malware detection standalone laboratory in which all the:
    - Display **messages or graphics**
    - **Sounds**
    - **Executing other applications** without the user executing it
    - Sudden system **slowdown, shutdowns or reboot**
    - **Crashing** of some programs
    - **Cannot connect** to the Internet

- Presence of **hidden files** in removable media

are written aside with additional observation of common Auto-Start locations, start-up folders, task schedulers and added services of Windows operating systems.

3. Observing process list using InstallRite tool which helps to make discover the differences between clean machine state and infected machine state through taking screenshots of machine between two different states. Additionally using a tool ProcessExplorer or a tool SysTracer there is possibility for looking suspicious process names, registries and DLL's.
4. Checking for suspicious network activity using WireShark tool after malware specimen is executed over the isolated lab. It is important to say that not all malware performs all its malicious activities in one isolated lab. As in our case it is better to create stand-alone isolated virtual network machines to follow all accurate traces of suspicious activities.
5. Retrieve all files related to the suspicious processes, registries and network activates classify them according to their behavioral findings.
6. Finally validating findings by hashing all files using online tools such as VirusTotal or AV vendor tools. Important to mention here that in a case of zero day exploits validation using open sources and AV tools could not be done always because it happens the findings not to have similar features or behaviors to previously known malware.

Conclusion

Overwhelming majority of malware is created to make money illegally, often by collecting confidential data from the victim's computer. To do this malware is designed to install as discreetly as possible running without disturbing the victim and ensuring that the machine is up and ready to be used. A damaged offline machine is no value to cyber criminals, but an infected machine is a powerful asset, able to perform any number of tasks. There has always been lots of speculation about the financial impact of cybercrime. If you search online for 'costs of cybercrime' you will find estimate ranging from millions to hundreds of billions. But the illegal nature of cybercrime activities makes it impossible to give an accurate figure of how much it costs. One thing is for sure; the growing volume of attacks makes it clear that its highly profitable for those involved in the 'dark market' that is cybercrime. The threat landscape has been dominated for almost a decade by random, speculative attack on anyone unfortunate enough to be infected. However the number of targeted attacks is growing. Such attacks are normally aimed specifically at one business. The **motives** can vary. Attackers may want to steal confidential business or customer data, damage a company's reputation, sabotage the normal running of an organization, or even make a political statement. Targeted attacks are highly sophisticated but they often originate by tricking individuals into disclosing information that allows the attacker to access corporate systems.

Malware or Malicious software is a collective term for all kinds of threats including viruses, worms and Trojans, a computer virus, Rootkits, Backdoor Trojans which are classified how they infect machines and travel autonomously from computer to computer. Often malicious codes instead of writing its code to multiple objects on a disk, it installs itself once and then looks for another computer to infect. Additional characteristics of malicious specimen masquerade as something benign, or even useful, but instead it carries out a harmful operation on their computer, without their knowledge or consent. Moreover some dangerous malware hides (itself) objects like (files, process and registry) and results the infected machines to be spied or monitored. Most malware binary codes is often is triggered by a user's action.

First chapter was an in ntroduction to malware analysis pre-requisites before to start any type of malware analysis. Here I gave brief information about operating systems that enables services for software applications to run on a computer. Moreover there was description about the characteristics of Windows OS that implements such as multi-threading, pre-emulative tasking, multiprocessing etc. After I presented the NetMarketShare a web analytics company known for its global market share statistics for Internet Technologies, Desktop operating system market share of Windows operating systems counts about %86.31 of market share. Additionally according to *Spiceworks' Network and Endpoint Security* report, one third of businesses still operate at least one device running Windows XP, which reached the end of its extended support cycle way back in 2014. (Interestingly, the final variant of XP, *Windows Embedded POSReady 2009*, only came out of support in April 2019. This variant of XP was used as a point-of-service operating system by – for example – shops. However many XP Home and Professional users were using a hack to receive security updates intended for this lesser known XP variant. But alas, this has now come to an end.) Of course, Windows XP doesn't get updates anymore. Including security updates. This means as crooks learn about vulnerabilities in XP, there is nothing to stop them using those exploits against XP users. As such, using Windows XP in 2019 is really one of the biggest security faux pas' you can commit. (Charles, 2019).

As a result of high popularity, easy and user friendly GUI, and high hit from malicious codes Microsoft Windows XP was selected as host operating system for researching and analyzing malicious specimen. There was classification of malware on **how it spreads** and how **it payloads** once a target is reached with malware infection routine through installation, propagation and payload. Moreover there was explanation on malware installation by **Processes**, default windows **Auto-start** (a mechanism which malware specimen use to execute in every logon in to the system), and **Registry** as hierarchickal database that contains information about the varios used by operating system in which every malware executes changes. The PE file standard is created by Microsoft and openly available online, which contains additional information for commonly used section names, entry points and import tables.

Suspicious code analysis is extracting information from files through physical file analysis. Microsoft Windows XP and other Microsoft Windows family of operating systems

executable file format is co called **PE (Portable executable)** file. Most malware specimen it's obfuscated from their programmer using so called packers.

Before to go to malware analysis there is need to be distinguished a packed or non-packed file. If malicious file is packed than there is need to be unpacked in order to extract more information while static malware analysis. Malware programmers by using packers (kind of compression the files) are trying to obfuscate malicious activities of malware specimen. Finally there were presented required safety procedures while working with (analyzing) malware specimen, and malware related article 251-a , Criminal code of Republic of North Macedonia in which I am stating the malware samples used here are for demonstration usage.

In second chapter "Static Malware analysis" I build up an environment and tools for malware analysis Lab using Oracle Virtual Box, a virtualization tool of applications in which I installed Windows XP OS, to test, track and analyze interaction of malware specimen with the real working guest system. Additionally I represented some interesting tools which helps dissecting information from dead code, searching strings from binary code, hashing to get digital file signature, tools for real time tracking registry, process and file changes.

Static malware analyze is analysis of malicious specimen without running or executing the code. This is usually done by determining the obfuscation of malware specimen if it is packed or not packed. If it is packed it is decompressed by various tools and techniques.

There are various packers available out there such as .aspack, .upx commonly found as the packers of files. Here I noted that not all packed executable are malicious. Packing legitimate executable is a common practice. Decompression leaves backdoor on malicious code in which by string search there is dissecting information about commands, IP addresses, e-mails, messages etc which happens during execution flow of specimen. Additionally there is digital signature calculation of hash value, which later can be compared with AV data hash sets.

Next very important issue while malware analysis is **Import Table** which contains the list of functions imported by the program from dynamic-link libraries. As I mentioned before packed samples shows few APIs on its **Import Table** than unpacked specimen. To remind

again **Static malware analysis is dissecting information from died executable file.** By looking to the imported functions we can gather more precise and flag positive information about malware itself. Moreover I continued searching strings using BinText tool, where I extracted interesting commands, meaning strings, and additional information to understand what malicious specimen tries to do while execution flow.

Additionally I used to hash my specimen where I got MD5 has digital file signature and searched by hash value on VirusTotal an online search engine to check and verify if it has positive hit over various AV tools.

Because of huge number of new malware variants development as a final topic in the second chapter was generating automated method or a flow chart which is summarizing basic static malware analysis method by implementing a methodology from implementing best practices such as:

- File properties checking because most malware no presents of digital signature or presents incomplete version of information;
- The need of files to be checked if it is packed or unpacked;
- If files are packed there is need to be unpacked using unpacker tools;
- Dead file analysis using Bintex tool to get a hint or initial analysis of what file simple do
- Finally validating findings using md5 or sha-1 hashing over online tools or data hash sets.

The third chapter “Dynamic Malware Analysis on previously captured SMTP packets”, as related with the topic was the core work of this project. Stated above dynamic or as I call live malware analysis is process of executing malicious specimen in sterilized digital lab environment with a goal to track all behaviors of malicious specimen and impacts onto the system. Behavioral analysis has very sensitive nature which explains malware installation on the machines via file system, registry and process. While Static analysis is looking for suspicious strings and commands, Dynamic analysis is process of looking for the suspicious files, looking for the suspicious processes to find suspicious files, and looking for suspicious registries to obtain suspicious file.

The scenario of this topic was analyzing previously captured network packet with the file name **smtp.cap** and hash value **aff1528eaee1ea8d948192fac16d8db1** as a unique digital signature of file.

With the help of network capture tool WireShark I did follow the “TCP Stream” of “SMTP” in order to get conversation between two IP addresses. There was e-mail header investigation in which I found the malicious attachment encoded based base64 (binary to text encoding scheme). I extracted and decoded malicious attachment using various technique with aim to understand and analyze better attachment activities. Then I executed malicious code to monitor its interaction with the system. After execution I stated that some files were created, some file were deleted, and some registry entries added, deleted or updated.

Moreover I listed all the technical details observed about my malicious specimen while I executed it in standalone lab. Additional finding here was that malicious specimen acts different in single isolated environment than the client server environment or virtual network environment. Moreover in digital data investigation preferable way is to use second tool implementation in order to verify results at least from two different sources.

Additionally I tested my malware in Client-Server structure in order to track malware interaction with network where with the help of tool of Wireshark I captured 1880 network packets. I filtered out capture .pcap file using “tcp” filter and I followed TCP stream (client to server) where I realized that my malicious specimen using TCP Port 53 (Port 53 is used by the Domain Name System (DNS), a service that turns human readable names like Google.com into IP addresses that the computer understands. Because port 53 is usually open, malicious programs may attempt to communicate on it) tries to reach some sites

Moreover I did Hybrid-malware analysis (simultaneous analyze of malware specimen static and dynamic) with the aim to verify my facts by third source, I did search in VirusTotal about malicious email attachment where I got result that from 70 engines I have 55 engines as virus positive categorized as an e-mail worm with name Beagle.32 such as its author left message while I was doing searching strings of binary executable code.

Finally the citation of the Independent IT-Security Institute Magdeburg, Germany mentioned in second chapter was that they register over 350,000 new malicious programs every day. This huge number of malware developed by actors need to be considered as serious attack over privacy of home and business users. Every single moment a smart

device, personal pc or server is under attack with high risk of information to be stealth or business to be interrupted. Dealing such a type of massive invasion of security risks requires to develop a methodologies or automatic responses.

Detection live attacks and performing dynamic analysis requires automatic response. Additionally automatic responses must be enriched using artificial intelligence. Machine learning implementation will be time consuming, effective cost and very strong response for eliminating the rising malware threats. Zero day exploits and validation will be always challenge to this discipline but at least real time detection, classification of malware, automatic generation of reports is existing on market and so many business solutions are provided by many AV vendors.

In this section I tried to find out and develop dynamic malware analysis flow chart through six points, in which all steps could be used and combined to generate automatic response algorithm for rising risks of malicious activates.

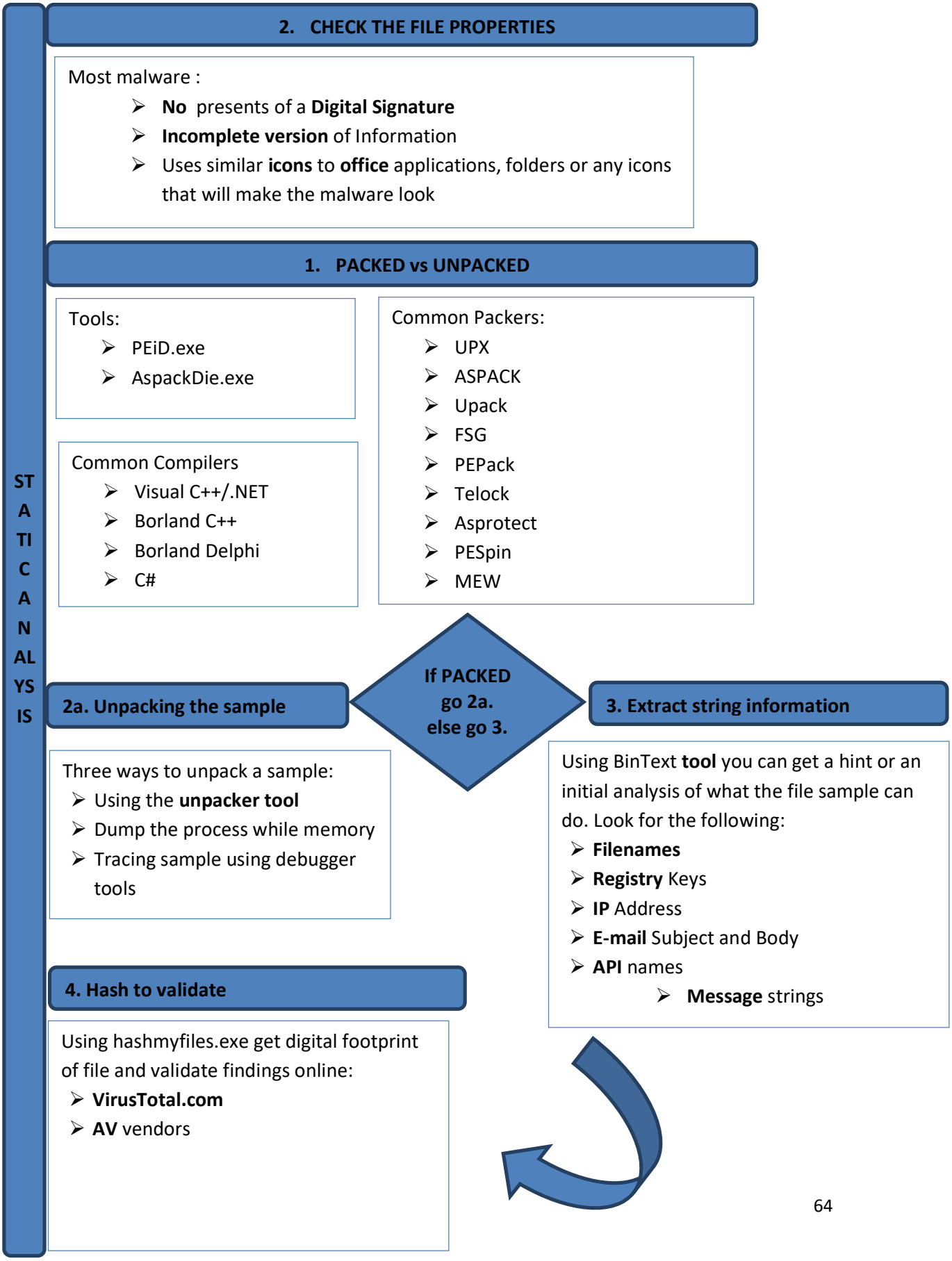
My findings enabled me to generate six point approach on implementing summarized automatic approach while dynamic malware analysis such as:

- Capturing real-time traffic using WireShark tool extracting content of captured packets and decoding them if the content is encoded.
- Gathering infection symptoms by executing malware specimen inside malware detection stand-alone laboratory
- Real picture or better distinction between clean machine state and infected state using screenshots options of InstallRite tool, following the suspicious processes, registries and DLL's using ProcessExplorer tool and Systracer tool.
- Checking for suspicious network activity using WireShark tool after malware specimen is executed over the isolated lab. It is important to say that not all malware performs all its malicious activities in one isolated lab. As in our case it is better to create stand-alone isolated virtual network machines to follow all accurate traces of suspicious activities.
- Retrieve all files related to the suspicious processes, registries and network activates classify them according to their behavioral findings.
- Finally validating findings by hashing all files using online tools such as VirusTotal or AV vendor tools.

Important to mention here that in a case of zero day exploits validation using open sources and AV tools could not be done always because it happens the findings not to have similar features or behaviors to previously known malware.

Finally I want to say that malware and malware related products will grow in future as latest security products will try to take countermeasure using hash based detection, signature based detection, yara rules, regular expression, HIPS/Behavioral Monitoring, vulnerability scanners etc. Malware writer's challenges and security breaches must be considered seriously by the governmental and non-governmental organizations where malware related threats to business or home users should be minimized if we want better and safe digital world. By rising consciousness on usage of digital data handlers and implementing machine learning or artificial intelligence robotic method against malware specimen variations we will have equality of arms.

Appendix A. Static malware analysis flow chart.



Appendix B. Dynamic Malware Analysis Flow chart

DYNAMIC MALWARE ANALYSIS

1. REAL TIME CAPTURE

- **Capture** real time traffic using WireShark
- **Extract** content of captured packed further analysis
- **Decode** using b64.exe tool

2. OBSERVE INFECTION SYMPTOMS

Execute malware specimen:

- Display **messages or graphics**
- **Sounds**
- **Executing other applications** without the user executing it
- Sudden system **slowdown, shutdowns or reboot**
- **Crashing** of some programs
- **Cannot connect** to the Internet
- Presence of **hidden files** in removable media

Common Auto-start locations:

- HKLM\Software\Microsoft\Windows\CurrentVersion\Run
- HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices
- HKCR***file\shell\open\command (Shell Spawning)
- Start-up Folders
- Task Schedulers
- Added Services

3. OBSERVE PROCESS LIST

- InstallRite.exe
- ProcessExplorer.exe
- SysTracer.exe
- Look for suspicious process name. Most malwares will try to use common process names found in the system
- Using Process Explorer, process without description are suspicious enough
- Check the image path of the file related to the process
- If processes under services.exe, check if the owner belongs with SYSTEM, If not it is most likely suspicious
- Check if it should have the correct parent process
- Most processes under services.exe are executed at the same time. If a certain process was exec8ted at a much later time, it is a suspicious process
- Look for suspicious DLLs loaded in each process. (DLL Injection)

5. CHECK FOR SUSPICIOUS NETWORK ACTIVITY

Follow network traffic activities

- WireShark

6. RETRIEVE ALL FILES RELATED TO THE SUSPICIOUS PROCESSES, REGISTRIES AND NETWORK ACTIVITIES

4. Validate findings

Using hashmyfiles get digital footprint of all findings and validate online:

- **VirusTotal.com**
- **AV** vendors

Bibliography

byteptr. (2017, September 8). *stackoverflow*. Retrieved 4 22, 2019, from stackoverflow: https://stackoverflow.com/questions/46108236/whats-the-differences-between-address-of-entry-point-and-original-entry-point/46121656#46121656

Charles, C. (2019, August 1). *Report shows 32% of businesses still use Windows XP*. Retrieved September 9, 2019, from https://www.thatsnonsense.com: https://www.thatsnonsense.com/report-shows-32-of-businesses-still-use-windows-xp/

G.Viscarola, P., & W, A. (1998). *Windows NT Device Driver Development.* New Riders Pub.

Introduction to MS Windows XP. (2009). Center for Educational Technology, University Cape Town.

Krogh, E. (2015). *An Introduction to Windows Operating System* (1st ed.). Bookboon.

Ligh, M. H., Adair, S., Harstein, B., & Richard, M. (2011). *Malware Analyst's Cookbook and DVD tools and techniques for fighting malicious code.* Indianapolis: Willey Publishing Inc.

Microsoft Windows. (2019, 8 26). *PE File*. Retrieved 9 1, 2019, from Microsoft Dev Centar: https://docs.microsoft.com/en-us/windows/win32/debug/pe-format

Popek, G., & Godberg, R. (July 1974). Formal Requirements for Virtualizable Third Generation Architectures. *Communications of the ACM*, 412-421.

S.Tanenbaum, A. (2007). *Modern Operating Systems* (3rd ed.). Pearson.

Singh, R. (September 2018). A comparison of Features of VirtualBox and VMware . *International Journal of Advanced Research Computer Science and Software Engineering, 5*(9), 467-469.

The Independent IT-Security Institute Magdeburg Germany. (2019, Semptember 8). *Malware Statistics*. Retrieved September 9, 2019, from AV-Test: https://www.av-test.org/en/statistics/malware/

Practical Malare Analysiss, The Hands-On Guide to Dissecting Malicious Software, Michael Sikorski and Andrew Honig, Foreword by Richar Bejtlich, published 2012.