UNIVERSITY OF PIRAEUS

DEPARTMENT OF DIGITAL SYSTEMS

Postgraduate Program

«Techno-economic Management and Security of Digital Systems»

Master's Thesis

AntiVirus Software Evasion: An Evaluation Of The AV Evasion Tools.

Christos Kalogranis

MTE1512, c.kalogranis@ssl-unipi.gr

Under the supervision of:

Dr. Christoforos Dadoyan, dadoyan@unipi.gr

Piraeus, February 2018

This thesis is dedicated to my wife, who supports me and encourages me in all my decisions; and to our two beloved children that are the source of my inspiration and the reason that I am struggling every day to become a better man.

ABSTRACT

This thesis focuses in the efficiency of the free given to the internet AV Evasion tools which have been developed for penetration testing. A selection of these tools has been made for the purpose of testing on how they can generate undetectable malwares against the most popular AV software products of the market.

The selected AV Evasion tools for evaluation are used for patching malicious Windows Portable Executable files.

A brief review is made for the most popular malware detection and evasion techniques and the selection criteria for the AV Evasion tools and for the AV Software products are presented. Additionally, it is described the lab test and the evaluation results are presented.

Keywords: Portable Executable file, AV Evasion, Encoding, Decoding, Code Cave, Encryptor, Decryptor.

CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

[This page intentionally left blank]

1 Introduction

In the era of Internet of Things, the use of a personal computer device or a smartphone device is an every task for the most of us. Services like e-Shopping, e-Banking and e-Government have great acceptance from the public and are preferred than the physical presence way of transactions.

The criminals of the today know that, your wallet or any identification paper of yours is useless and the only thing that they have to do is to have get access to your smartphone or your personal computer. The internet gives a lot of information on how to learn and do this. Moreover, the tools that have been developed for penetration testing with the purpose to raise the level of security to our devices and e-services, have been used also from the adversaries in order to gain access to our devices and then get everything they need for their final purposes.

One way to get access to a device is via the use of a malware, which easy for someone to create one by using one of the tools that are freely given to the internet. There are tools like Metasploit that generates malicious payloads for creating malware, but todays AntiVirus software products, use various techniques in order to discover malicious software and this kind of malwares are been detected easily . In addition, there have been developed tools that employ some evasion techniques in order to make the known malwares undiscoverable from the AV software products.

In this thesis, we briefly review malware detection and evasion techniques. Then, we focus on the evasion tools by making an evaluation test for selected AV Evasion tools and AV software products and present the results of this lab test.

1.1 Malware Detection Techniques

An antivirus is a program that has the ability to scan several types of files on the disk, by comparing it with a known database. It is able to collect original file size and compare it within the time to validate that it has not grown. It also can scan over the Master Boot Record (MBR), boot sectors, bad sectors, among others to determine if it has been infected. In regards of malware detection methods, three main categories have been recognized: Signature based, Behavior based, and Heuristic based methods. [1]

1.1.1 Signature Based Method

This technique searches sequences of bytes in order to identify a particular piece of malicious software. A signature is composed of a particular sequence of code or data. This signature is stored on a database which is used to compare to the scanned files.

This is the most common method used to detect malware, since it produces a small error rate. However, there are some limitations. For example, it cannot detect new malwares since their signatures have not been generated yet. Additionally, internet connectivity is a must to download new signatures from the server and keep the host protected. Furthermore, since new malware appears every day, it is necessary to store large amounts of signatures, demanding considerable storage, making slow to search a particular signature, and affecting system performance. In addition, it is estimated that at least 24 hours are required for a new signature to be added to the database, giving new variants the ability to compromise a system without being detected. [1]

1.1.2 Behavior based Method

The approach is to identify a malware by inspecting its behavior while it is being executed. A behavior-based detector can determine if a program is malicious or not by examining what it does. The architecture of a behavior-based detector, it consists of:

- Data collector which acquires dynamic and static information from an executable
- Interpreter which transforms collected data to intermediate representations.
- Matcher which compares representations with behavior signatures.

In spite of this approach focuses on behavior, it might produce a considerable false positive ratio as well as high amount of scanning to achieve its purpose. Behavior based approach relies on identifying patterns that are not common, such data being sent between two nodes, attempts to change boot sector or the flash memory. [1]

1.1.3 Heuristic based Method

It was born as an alternative to signature-based detection. It works by examining system behaviors and keystrokes to determine if there is an abnormal behavior. Also, it does not require constant updates. But, it can produce several false positives as well as consuming more computing resources. Besides, it requires the inclusion of a third party component, such as tools to analyze protocols which might open a new breach in terms of vulnerabilities. Finally, heuristic detection has to know the vulnerability rather than the malware.

It is important to clarify that heuristic approach reviews the code to find possible variants of a malware, while behavior-based method examines if a malware misbehave. The combination of heuristic and behavior methods could help to reduce false positives. On the other hand, excluding the inclusion of a third party component would prevent vulnerability exploitation from other sources. This approach would reduce system performance effect.

Heuristic method is considered as part of a technology that uses Artificial Intelligence. This method has the ability of self-discovering and analyzing the code in an intelligent way performing a deep inspection of code instruction sequences as well as discovering unusual or unopened system calls. This technique might cause a high rate

of false positives and negatives but combining with other traditional techniques, its efficiency could improve considerably.

Some heuristic methods are listed below:

API/System calls: Based on the calls made to the Operating System (OS) by a particular program through the use of application programming interface (API). The aim is to analyze what piece of code uses a request to the OS.

OpCode: It is based on Operational Codes (subdivision of machine language) in order to identify code sequences that might be associated to malware programs. Some of the approaches shown previously, count them and compare with valid programs.

N-Grams: This approach is based on reading binary code either from reading PE section, plain-text strings encoded in executables and sequences of bytes.

Control flow graph: A program is represented through a series of steps; every section is represented by a node and helps to understand the way a program functions.

Hybrid features: This method considers the inclusion of Machine Learning classifiers which depends on two elements: features and algorithms. In order to improve accuracy of this method, it is required to combine features. [1]

1.1.4 Sandboxing

This mechanism consists on running programs in a separate/isolated environment by controlling all the resources allocated in case of damage. This is considered an appropriate contention mechanism against malware's obfuscation techniques. There are several tools that use this technique. Such tools are able to imitate malware interaction as well as catching and documenting changes made to an infected system. Once a sample of malware is in execution, it is important to collect as much information as possible in order to determine changes performed in the system, identify patterns and understand its behavior. [1]

1.2 Malware Evasion Techniques

As there are ways to detect malware, there are also ways for being undetected; just like a game of thieves and cops. There are several techniques used by attackers that are leaving behind anti-malware vendors. [1]

1.2.1 Encryption

Encrypted malware is composed of two main sections: a decryption loop and a main body. Decryption loop is capable of encrypting and decrypting the main body. Main body contains the code of the malware itself which is encrypted with simple algorithms like XOR or using complex and robust ones such as AES. Anti-malware solutions must decrypt the main body to get the valid signature and detect the malicious piece of software. [1]

1.2.2 Oligomorphism

The purpose of this technique is to produce a different decryptor on every new infection. An additional improvement is that there are several decryptors which are randomly chosen making a new type of code on every instance. This technique can be detected but requires more time. [1]

1.2.3 Polymorphism

Polymorphic malware is harder to detect as there is an unlimited number of new decryptors. The main feature of this technique is that the code constantly changes on every new variant. Code obfuscation is used to mutate the decryptor to produce a new version for another victim. [1]

1.2.4 Metamorphism

It does not contain an encrypted part. However, it uses mutation engines to change the body on every compilation, rather than using cryptography for protecting the code. A metamorphic engine should consist of the following components as shown [1]:

- Disassembler
- Code analyzer
- Code transformer
- Assembler

1.2.5 Obfuscation

Hiding information to avoid being caught is a common practice among attackers, in order to defeat certain security devices like IDS or any other based on signature detection. Using encoding and manipulating strings is a practice that can easily bypass signatures based scanners. For example, either by replacing a / with a \ or by using Hex, Unicode or UTF-8 can avoid detections. Moreover, using encryption to encode a whole session is a classic way to avoid being detected. The problem lies in the lack of the appropriate key to decrypt information. Polymorphic code aims to mutate its code while maintaining its original algorithm. It is usually combined with a cipher/decipher module that is embedded in the code. Encrypting malware is the first stage to bypass signature-based solutions as they do not have readable information to compare.

Obfuscation has helped polymorphic and metamorphic malware to bypass anti-malware solutions. Indeed, they have used several coding techniques, to achieve its goal, some of them are as follows:

Dead-Code insertion: It simply adds not effective instructions to a program to change its appearance but its behavior remains intact.

Register Reassignment: It consists on switching registers from one version to another.

Subroutine Reordering: A subroutine is obfuscated and reordered in a random way giving n chances of variants.

Instruction Substitution: Its objective is to replace original code with others that are equivalent to the original.

Code Transposition: It reorders the sequence of a set of instructions from the original code, either by using unconditional branches or based on independent instructions.

Code Integration: A malware joins its code with a valid program by decompiling the original one and rebuilding it with infected instructions. [1]

1.2.6 Code reuse attacks

This type of attacks does not inject code in order to work. First of all, it uses return-into-libc which aims to reuse executable code from a valid running process. A malicious user might try to change the pointer to a different function rather than following the normal process. Libc is a standard library that is part of C-language.

Return-Oriented Programming (ROP) can create malicious computation orders by linking small pieces of code in the existing space address of a running program. These types of attacks are organized in sets of instructions and do not require to inject code or call a function which is a way to bypass current anti-malware solutions. [1] [2]

2 Selection of the AV Evasion Tools

Most of the personal computers today are using Microsoft's Windows Operating System. Specifically the 82.68% of the active personal computers are using a version of Windows based on the statistics shown in *Figure 1*. [3]

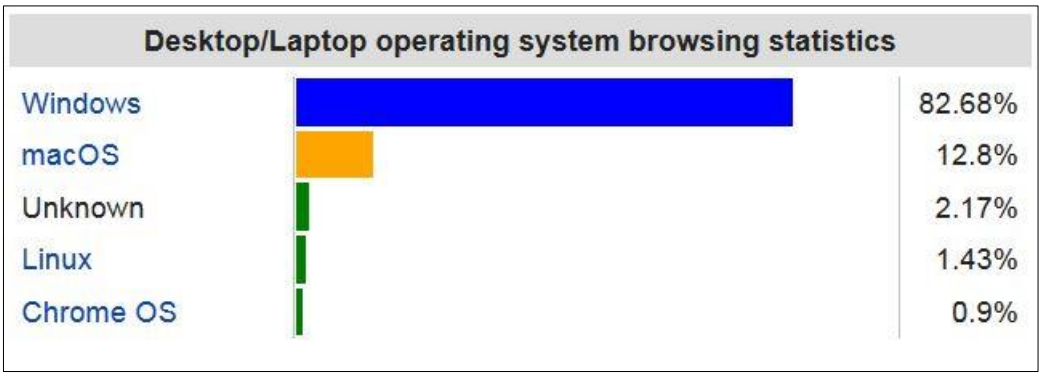

Figure 1: Desktop OS market share as of January 2018 (Wikipedia)

Furthermore, the Windows version as shown in *Figure 2* is either Windows 10 or Windows 7. [3]

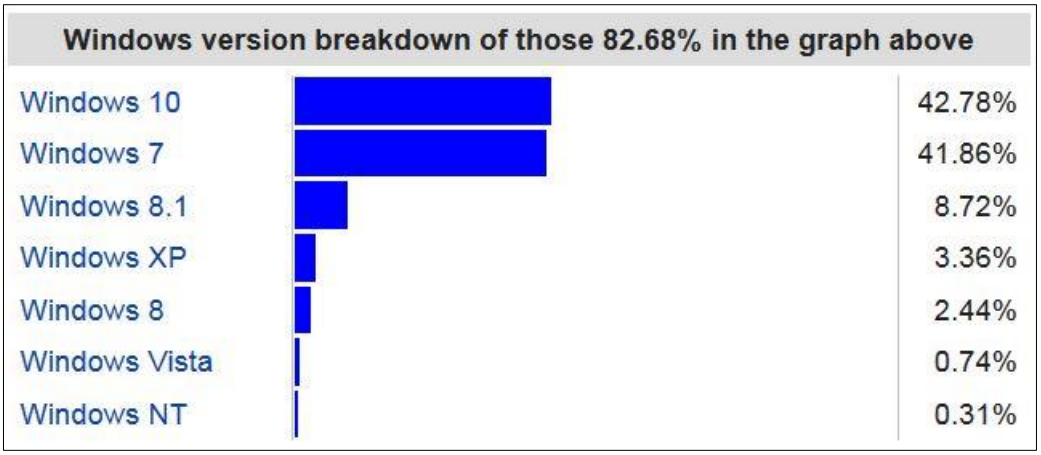

Figure 2: Windows versions breakdown as of January 2018 (Wikipedia)

According to the above statistics information, the targeted operating system is Windows and specifically the version 7 instead of version 10, because is the older version among them and it's possible to be more vulnerable than the latest version; furthermore the patching scheme for this product will finish sooner.

So, it's logical to use malicious infected Windows executable files (Portable Executable files) generated by Metasploit framework in order to attack in a desktop PC. But, this is not so easy, since there is AntiVirus software installed on the

Windows OS. Additionally, there are some AV Evasion tools to use, in order to overcome the possibility that the malicious PE file been detected by AV software. The most interesting of AV Evasion tools are selected, based on popularity and the effectiveness of these. The four AV Evasion tools that are been selected, are presented in the following chapters.

2.1 AVET

AVET is the AntiVirus Evasion Tool, which was developed to support the pentesters job and for experimenting with antivirus evasion techniques.

For evading antivirus software AVET employs the following techniques/methods.

A ***shellcode binder*** is necessary to alter the encoded or obfuscated payload before execution. It is quite simple and does not contain enough information for creating a pattern for signature based recognition.

The payload itself has also to be encoded to make it invisible for the antivirus software. To accomplish this AVET implements an ***ASCII encryptor***.

For evading sandboxing/heuristics different techniques are possible. Emulators are breaking up their analysis at a point. So, when a run cycle limit is reached the emulation stops and the file is passed as not malicious. This can be accomplished by using lots of rounds of an encoder or to perform an action that the emulator is not capable of. This includes opening files, reading parameters from the command line and more. [4]

2.2 peCloak.py

The peCloak.py, is a python script, which automates the process of hiding a malicious windows executable from AV detection.

In order to avoid signature-based detections, it is been employed encoding, with the constraint of the use only basic add, sub, xor instructions, and a dynamic construction of the encoding order. The number of instructions, the order, and the modifiers are all chosen pseudo-randomly to increase variation.

By default, the script encodes the entirety of the PE section that contains the executable code (typically the .text or .code section) but that is a configurable option, since there is the pefile library. Which is been employed for mapping the contents of the PE file and retrieving various sections of it.

As for the heuristic bypass, It is been employed a routine which uses a set of instructions like NOPS, INC/DEC, ADD/SUB, PUSH/POP chosen randomly for a finite number of iterations, that wastes cycles in order to trick the AV scanner that the executable is benign.

Eventually, this script encodes the designated sections of a PE file and then inserts a code cave containing the heuristic bypass and the corresponding decoder function. [5]

2.3 Shellter

Shellter is a dynamic shellcode injection tool. It can be used in order to inject shellcode into native Windows applications (currently 32-bit apps only). The shellcode can be something yours or something generated through a framework, such as Metasploit.

Shellter takes advantage of the original structure of the PE file and doesn't apply any modification such as changing memory access permissions in sections (unless the user wants to), adding an extra section with RWE access, and whatever would look dodgy under an AV scan.

Shellter is not just an Entry-Point Obscuring (EPO) infector that tries to find a location to insert an instruction to redirect execution to the payload. Unlike any other infector, Shellter's advanced infection engine never transfers the execution flow to a code cave or to an added section in the infected PE file.

Shellter uses a unique dynamic approach which is based on the execution flow of the target application. This means that no static/predefined locations are used for shellcode injection. Shellter will launch and trace the target, while at the same time will log the execution flow of the application.

Shellter traces the entire execution flow that occurs in userland. That means, code inside the target application itself (PE image), and code outside of it that might be in a system dll or on a heap. This happens in order to ensure that functions actually belonging to the target executable, but are only used as callback functions for Windows APIs will not be missed. [6]

During tracing, Shellter will not log or count any instructions that are not in the memory range of the PE image of the target application, since these cannot be used as a reference to permanently inject the shellcode.

2.4 Veil-Evasion (Veil 3.0)

Veil is a penetration-testing framework that was originally designed to evade antivirus protection on the target system.

To bypass antivirus protection, Veil generates random and unique payloads for exploits. This ability to make random changes to the payload is similar to polymorphic malware that changes as it moves from host to host, making it much more difficult to discover than traditional malware, which has a distinct signature.

Veil's exploits are compatible with popular penetration testing tool frameworks like Metasploit, which makes it very easy to incorporate Metasploit into your existing penetration testing routine.

Veil aggregates various shellcode injection techniques across multiple languages, putting the focus on automation and usability. [7]

3 Selection of the AntiVirus Products

The anti-malware industry is evolving every day, the reason for that is the technological evolution in conjunction with the user needs. So, there are so many anti-malware products that are impossible to test it all. For that, I chose only five AntiVirus products, based on two criteria. Firstly, the product's market share and secondly, the ranking for the protection that provides.

Furthermore, the portal Statista.com gives statistics about the anti-malware industry market share for 2017, specifically for anti-malware vendors that offer applications designed for Windows as of August 2017 and as shown in *Figure 3*, Avast, Eset, and McAfee held over the 43% of the anti-malware application market. [8]

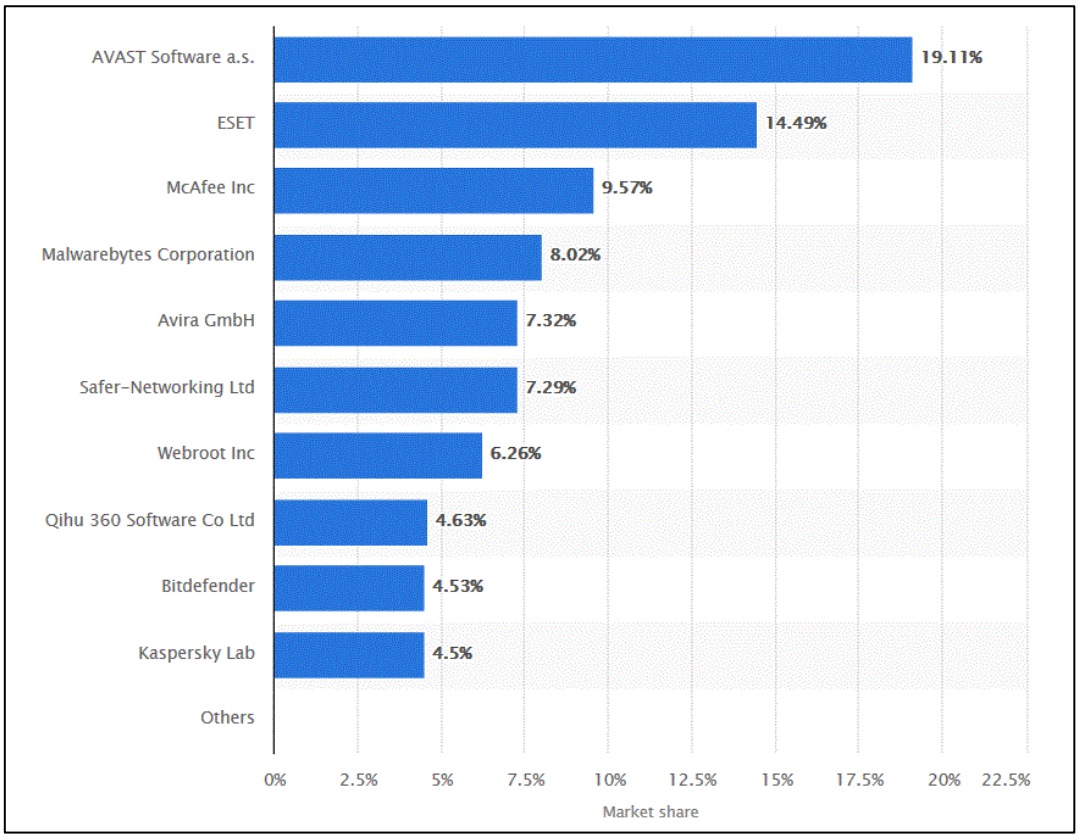

Figure 3: The anti-malware industry market share for 2017, for Windows OS (Statista.com)

Additionally, the AV-Comparatives report on December 2017, as about the "Real-World" Protection Test, resulted that Avast, Bitdefender, Eset, McAfee and Avira are an "ADVANCED" level malware protection solutions, as shown in *Figure 4*. [9]

AWARD LEVELS	PRODUCTS
AV ADVANCED+ ★★★ -REAL WORLD- PROTECTION TEST comparatives DEC 2017	Panda Bitdefender Tencent Trend Micro Kaspersky Lab AVIRA Avast AVG VIPRE Emsisoft ESET McAfee
AV ADVANCED ★★ -REAL WORLD- PROTECTION TEST comparatives DEC 2017	F-Secure* Symantec* BullGuard* Microsoft* Fortinet* CrowdStrike
AV STANDARD ★ -REAL WORLD- PROTECTION TEST comparatives DEC 2017	Seqrite* eScan
AV TESTED -REAL WORLD- PROTECTION TEST comparatives DEC 2017	Adaware

Figure 4: "Real-World" Protection Test Level (AV Comparatives)

Moreover, the AV-TEST ranking results on December 2017 for the category AntiVirus software for home users as shown in *Figure 5*, awarded 6/6 points for the overall protection provided in Avast Free AntiVirus, Bitdefender Internet Security, McAfee Internet Security and are following the Eset Internet Security and Avira Antivirus Pro. [10]

Figure 5: The best antivirus software for Windows Home User (AV-TEST)

Based on the above information, the five AntiVirus software products that were selected for the lab test are shown in the list below.

1. *Avast Free Antivirus* [11]
2. *Bitdefender Internet Security 2018* [12]
3. *Eset Internet Security* [13]
4. *McAfee Total Protection* [14]
5. *Avira Antivirus Pro* [15]

These five Antivirus software products have installed in the 55% of the Windows PCs that have AntiVirus software and are the top ranked for the overall protection that provide.

4 Evaluation of the effectiveness of these AV Evasion Tools

4.1 The Lab Test description

For the evaluation of the effectiveness of these automated tools I did a lab test. The test lab was consisted of the malware generation machine which was a Kali-Linux 2017.2-i386 virtual machine and a number of the target Windows OS machines which were instances of the same Windows 7 Professional N virtual machines but with different AntiVirus software installed in it.

Each of the four selected AV evasion tools ran on the malware generation machine and generated two different windows OS Portable Executable file (PE file). Then these PE files are shared via python's *SimpleHTTPServer*.

Every instance of the target Windows OS machine is updated and additionally the installed AV software is also updated with the latest malware signatures, before the execution of the lab test. So, after this phase, there is no internet access for the test lab virtual machines, in order to avoid any connectivity with the AV's vendor malware intelligence systems.

If a PE file was not detected by the AV software as malware during the download or execution, it is awarded one point to the AV Evasion tool generated it, else zero points is awarded it.

4.2 The Lab Test results

4.2.1 AVET

In order to evaluate the AVET, we generated 4 PE file samples as listed in Table 1, with the most popular Metasploit payloads, like Reverse TCP Meterpreter and Reverse TCP Shell, as source shellcode.

Table 1: List of generated PE file samples by AVET

Sample Name	Description
pwn1	Windows reverse tcp meterpreter with Shikata encoding.
pwn2	Windows reverse tcp meterpreter with generic/none encoding for 3 iterations
pwn3	Windows reverse tcp shell with Shikata encoding
pwn4	Windows reverse tcp shell with generic/none encoding for 3 iterations

All of the PE file samples bypass at least the 2 out of 5 AVs. Bitdefender and Avira bypassed by all of the samples. Eset and McAfee detected all of the samples as malicious. But the malicious samples pwn1 and pwn2 bypass the Avast AV, because of the Shikata encoding.

Table 2: AVET AV evasion scoring

AV Product	PE File			
	pwn1	***pwn2***	***pwn3***	***pwn4***
Avast Free Antivirus	1	0	1	0
Bitdefender Internet Security	1	1	1	1
Eset Internet Security	0	0	0	0
McAfee Total Protection	0	0	0	0
Avira Antivirus Pro	1	1	1	1
Total Score	**3**	**2**	**3**	**2**

As shown in the Table 2 above the best effort for the AVET was to bypass the 3 out of the 5 selected AVs.

4.2.2 peCloak.py

For the evaluation of the peCloak.py, we generated 2 PE file samples, which are explained in Table 3.

For both of the samples, they used the default options for cloaking. Which are 3 iterations for emulation bypass and no encoding.

Table 3: List of generated PE file samples by peCloak.py

Sample Name	Description
cloaked1	Cloaked payload of Windows reverse tcp meterpreter with no encoding.
cloaked2	Cloaked payload of Windows reverse tcp shell with no encoding

The 2 Cloaked PE file samples bypass the Avast and Avira AVs. The other 3 AVs detected these samples as malicious.

As shown in Table 4, the peCloak.py bypasses the 2 out of 5 AVs.

Table 4: peCloak.py AV evasion scoring

AV Product	PE File	
	cloaked1	***cloaked2***
Avast Free Antivirus	1	1
Bitdefender Internet Security	0	0
Eset Internet Security	0	0
McAfee Total Protection	0	0
Avira Antivirus Pro	1	1
Total Score	**2**	**2**

4.2.3 Shellter

As for the evaluation of Shellter, we selected the *putty.exe* as target PE file to inject different payloads and finally generate 7 different malicious samples. In the first four samples we inject custom payloads and in the last three we inject predefined payloads. All samples generated by Shellter in Auto Mode and only for the predefined payloads additionally enabled the Stealth Mode. The above mentioned samples are listed in detail in Table 5

Table 5: List of generated PE file samples by Shellter

Sample Name	Description
shell_cp1	Windows reverse tcp meterpreter with Shikata encoding.
shell_cp2	Windows reverse tcp shell with Shikata encoding
shell_cp3	Windows reverse tcp meterpreter with generic/none encoding for 3 iterations
shell_cp4	Windows reverse tcp shell with generic/none encoding for 3 iterations
shell_p1	Windows reverse tcp meterpreter (option preset 1).
shell_p5	Windows reverse tcp shell (option preset 5)
shell_p7	Command Shell Test, WinEXE (option preset 7)

Three of the AVs detected all the samples as malicious. Furthermore, one of the two remaining AVs, detected only the samples with the predefined payloads as malicious.

Table 6: Shellter AV evasion scoring

AV Product	PE File					
	shell_cp1	***shell_cp2***	***shell_cp3***	***shell_cp4***	***shell_p5***	***shell_p7***
Avast Free Antivirus	0	0	0	0	0	0
Bitdefender Internet Security	0	0	0	0	0	0
Eset Internet Security	0	0	0	0	0	0
McAfee Total Protection	1	1	1	1	0	0
Avira Antivirus Pro	1	1	1	1	1	1
Total Score	**2**	**2**	**2**	**2**	**1**	**1**

As shown in Table 6, the malicious samples which are injected with custom payloads bypass 2 out of 5 AV in conjunction with those which are injected with predefined payloads. So, the best AV evasion effort was 2 out of 5.

4.2.4 Veil-Evasion (Veil 3.0)

For the evaluation test of Veil-Evasion, we selected two custom payloads and two predefined payloads. The two custom payloads were a reverse tcp meterpreter and a reverse tcp shell, both encoded with Shikata.

An explanation of the generated malicious samples is shown in Table 7.

Table 7: List of generated PE file samples by Veil-Evasion

Sample Name	Description
veil_cp1	Windows reverse tcp meterpreter with Shikata encoding.
veil_cp2	Windows reverse tcp shell with Shikata encoding
veil_p7	Windows reverse tcp (predefined payload 7)
veil_p8	Windows reverse tcp service (predefined payload 8)

The two samples, which were generated by predefined payloads detected as malware by all the selected AVs. In adverse, the other two samples which were generated by custom Metasploit payloads, was detected as malicious, only of the two out of five AVs.

Table 8: Veil AV evasion scoring

AV Product	PE File			
	veil_cp1	***veil_cp2***	***veil_p7***	***veil_p8***
Avast Free Antivirus	1	1	0	0
Bitdefender Internet Security	0	0	0	0
Eset Internet Security	0	0	0	0
McAfee Total Protection	1	1	0	0
Avira Antivirus Pro	1	1	0	0
Total Score	**3**	**3**	**0**	**0**

As shown in Table 8, the best effort for the Veil-Evasion is with the custom payloads which bypassed the 3 out of 5 AVs.

4.2.5 Comparison of Evasion Ratio between the AV Evasion Tools

The selected AV Evasion tools, bypassed at least two out of the five selected AVs. Most of them achieved that with custom payloads. The best effort was to evade 3 out of 5 AVs.

Furthermore, the evasion ratio (AVs evaded/Total Number of Selected AVs) was 40% to 60%. AVET and Veil-Evasion were those with the 60% evasion ratio, the other were those with the lowest 40% ratio, as shown in Figure 6.

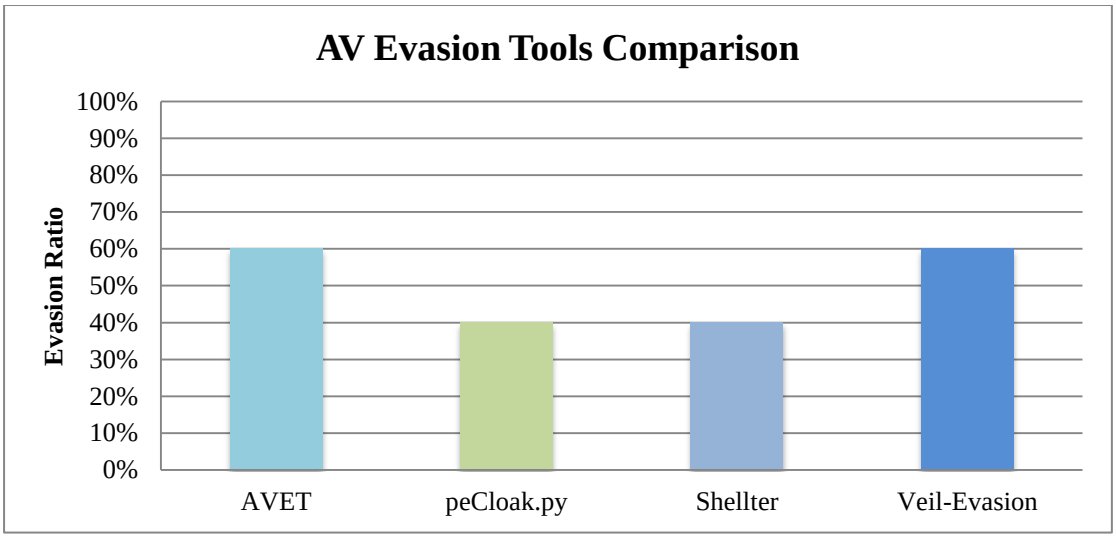

Figure 6: Comparison of evasion ratio between the selected AV Evasion Tools

The AV Evasion tools that tested in this evaluation got an average evasion score over 40%.

AVET and Veil-Evasion, managed to bypass the most of the AV software products, by using Metasploit encoded payloads.

In addition, peCloack.py and Shellter, managed to bypass only two of the five AV software products, by using encoded and un-encoded payloads

5 Conclusions

This evaluation revealed that the AV evasion task is not an impossible mission. Actually, is easy enough to bypass enough of the most known AV software products of the market, with no deep technical knowledge for the malware and antimalware software.

There are enough AV evasion tools that efficiently are completing their task. Most of them are managing to successfully alter the malicious payloads from Metasploit in a way that AV software is not able to discover it.

As concern the technical aspect of this, most of the tools employing code caving and encrypting techniques, in order to bypass signature base detection. In addition, the use of iterations is the easiest way to evade sandboxing and heuristic detection.

REFERENCES

[1] S. G. Yoo and J. J. Barriga, "Malware Detection and Evasion with Machine Learning Techniques: A Survey," *International Journal of Applied Engineering Research,* Vols. ISSN 0973-4562 Volume 12, pp. 7207-7214, 2017.

[2] G. Poulios, "Advanced Antivirus Evasion Techniques," University of Piraeus, Piraeus, 2015.

[3] Wikipedia, "Usage share of operating systems," Wikimedia Foundation, Inc., February 2018. [Online]. Available: https://en.wikipedia.org/wiki/Usage_share_of_operating_systems#Desktop_and_laptop_computers. [Accessed February 2018].

[4] D. Sauder, "AntiVirus Evasion Tool (AVET)," in *Blackhat USA 2017 Tools Arsenal*, 2017.

[5] M. Czumak, "peCloak.py – An Experiment in AV Evasion," 9 March 2015. [Online]. Available: https://www.securitysift.com/pecloak-py-an-experiment-in-av-evasion/. [Accessed February 2018].

[6] K. Ekonomou, "Shellter," [Online]. Available: https://www.shellterproject.com/Downloads/Shellter/Readme.txt. [Accessed February 2018].

[7] D. J. Dodd, "ADMIN Magazine - Network & Security," 2016. [Online]. Available: http://www.admin-magazine.com/Archive/2016/32/Slipping-your-pen-test-past-antivirus-protection-with-Veil-Evasion. [Accessed February 2018].

[8] Statista, "Statista.com," Statista Inc., August 2017. [Online]. Available: https://www.statista.com/statistics/271048/market-share-held-by-antivirus-vendors-for-windows-systems/. [Accessed February 2018].

[9] AV-Comparatives, "AV-Comparatives.org," December 2017. [Online]. Available: https://www.av-comparatives.org/wp-content/uploads/2017/12/avc_prot_2017b_en.pdf. [Accessed February 2018].

[10] AV-TEST, "AV-TEST.org," AV-TEST GmbH, January 2018. [Online]. Available: https://www.av-test.org/en/antivirus/home-windows/windows-10/. [Accessed February 2018].

[11] AVAST, "AVAST.com," AVAST Software, Inc. (US), February 2018. [Online]. Available: https://www.avast.com/el-gr/index. [Accessed February 2018].

[12] Bitdefender, "Bitdefender.com," Bitdefender SRL, February 2018. [Online]. Available: https://www.bitdefender.com/solutions/internet-security.html. [Accessed February 2018].

[13] ESET, "ESET.com," ESET, spol. s r.o., February 2018. [Online]. Available: https://www.eset.com/gr/home/internet-security/. [Accessed February 2018].

[14] McAfee, "McAfee.com," McAfee, Inc., February 2018. [Online]. Available: https://www.mcafee.com/consumer/en-us/store/m0/catalog/mtp_521/mcafee-total-protection-trial.html/. [Accessed February 2018].

[15] Avira, "Avira.com," Avira Operations GmbH & Co. KG, February 2018. [Online]. Available: https://www.avira.com/en/download/product/avira-antivirus-pro. [Accessed February 2018].

[16] J. Koret and E. Bachaalany, The Antivirus Hacker's Handbook, Wiley, 2015.

[17] D. Keragala, "Detecting Malware and Sandbox Evasion Techniques," The SANS Institute, 2016.