Bachelor Degree Project

Antivirus performance in detecting Metasploit payloads: *A Case Study on Anti-Virus Effectiveness*

Authors: Eric Nyberg, Leandro Dinis Ferreira
Student mail: en223de@student.lnu.se
ld222hp@student.lnu.se
Supervisor: Ola Flygt
Semester: VT 2023
Subject: Computer Science

Abstract

This paper will focus solely on the effectiveness of AV (antivirus) in detecting Metasploit payloads which have been encapsulated with different encapsulation modules. There seems to be a significant knowledge gap in the evaluation of commercial antivirus's software and their ability to detect malicious code and stop such code from being executed on IT systems. Therefore we would like to evaluate the capabilities of modern AV software with the use of penetration testing tools such as Metasploit.

The research process is heavily reliant on a case study methodology as it can be argued that each payload generated reflects a case in itself. Firstly the payloads are generated and encapsulated through the self developed software, secondly they are uploaded to VirusTotal to be scanned with the use of their publicly available API, third the results are obtained from VirusTotal and stored locally. Lastly the results are filtered through with the software which in turn generates graphs of the results. These results will provide sufficient data in comparing encapsulation methods, payload detection rates, draw conclusions regarding which operating system may be most vulnerable as well as the overall state of modern AV software's capabilities in detecting malicious payloads. There are plenty of noteworthy conclusions to be drawn from the results, one of them being the most efficient encapsulation method powershell_base64 which had amongst the lowest detection rates in regards to the amounts of payloads it encoded, meaning that its encapsulation hid the malicious code from the AV at a higher degree than most the other encapsulation modules. The most noteworthy conclusion from the results gathered however is the encapsulation methods which obtained the absolute lowest detection rates, these were x86_nonalpha, x86_shikata_ga_nai, x86_xor_dynamic as well as payloads without any encoding at all, which had a few payloads reach among the lowest detection rates across the board (<20%).

Keywords: Antivirus Software, payloads, encapsulation/encoding, efficiency, detection rate.

Preface

We would like to extend our gratitude towards Ola Flygt for his guidance and assistance during the writing of this thesis. His expertise in the field of cyber security and thesis writing made it possible to complete this study for our bachelor degree in Computer Science. We would also like to give our appreciation for VirusTotal and their publicly published API which allowed us to perform this research.

Contents

1 Introduction

As computers become more integrated into society, the attack surface and quantity of attacks also increase. Such that companies invested in developing security products to decrease the success rates of malware and other cyberattacks. The FBI reported a total of potential losses, due to cybercrime, to be at $4.1 billion in 2020 [1] which seems to continuously grow. In 2021 FBI (IC3) reported the total of potential loses in 2021 to be at $6.9 billion [2] which further increased to $10.2 billion in 2022 [3]. According to these reports, cybercrime is not diminishing and since there are multiple important factors influencing the success of cyberattacks it's meaningful to investigate one aspect of defense, which is antivirus (AV) software.

It's often regarded that the antivirus (AV) software is one of the main tools to combat malware and cyberattacks and it has existed within the field of IT for decades [4][5]. Ever since AV software became prevalent for computers to use, AV have been participating in an everlasting arms race against malware. As malware develops to outsmart and outwork the dedicated defense, AV, AV software in turn has been forced to advance its strategies and capabilities to combat malware programs and the attackers tactics.

The development of the security industry, which naturally emerged from the arms race between malware and AV, has helped curb cyberattacks. However, it has also created the need for tools that can test the effectiveness of cyber security software. One such tool is Metasploit [6], an open-source cyber attack framework commonly used in penetration testing attacks [7]. While Metasploit is primarily intended for legitimate security purposes, it could potentially be misused for nefarious acts. Given its significant use in the field of cybersecurity, it is reasonable to recognize its' utilization as well as effectiveness against modern AV software, which is widely utilized across the globe.

1.1 Related Work

There exists substantial amounts of research regarding antivirus software and its effectiveness, both within the educational field as well as outside of it in the form of cyberattacks. If the key selling point of Metasploit is to assist security teams, test vulnerabilities and in general uphold and improve upon cybersecurity, it is compelling to research modern AV effectiveness against such tools. This case was researched back in 2008 in a study called "Effectiveness of Antivirus in Detecting Metasploit Payloads" [8]. As this research was conducted a considerable amount of time ago, the results presented in the study will most likely not be similar to that of this research. However, the utilized payload used were limited and most results did not yield any results higher than a 37.5% detection rate for its' respective payloads. A majority of the payload results in fact attained detection rates below 20%, which is to be believed to have changed for the current study being conducted as both modern AV software as well as the penetration testing framework Metasploit has been severely updated since.

A more recently performed study has been reviewed for this paper, which was conducted in 2019 named "Applied Comparative Evaluation of the Metasploit Evasion Module" [9]. This study chose three payloads, namely windows/meterpreter/reverse_tcp, windows/powershell_reverse_tcp, windows/shell/reverse_tcp. The study in 2019 then chose to encode these payloads with x86/shikata_ga_nai with one iteration to increase the chances of the payload surviving the transport to the destination as well as making it easier to be placed into memory [9]. The payloads were then encrypted using Hyperion

with six bytes of an AES (Advanced Encryption Standard), which does represent a realistic scenario where most payloads will be encrypted to decrease the chances of detection by AV software. To further decrease the chances of exposure, they generated payloads with the Veil-Evasion module supplied by MSFvenom with the intention to add a layer of difficulty for the AV software to detect the payload as malicious. The Veil-Evasion module used however was later deemed to have had a minimal impact, and was even beaten by the encoded trial. The results were found from [9] to be quite limited and did not present the results in its entirety due to the fact that the amount of payloads were small, and could potentially garner different results had the quantity been larger. It is our belief that the amount of payloads used can definitely be expanded upon, and the presentation of results can also be improved. Therefore this paper will in a later section present the results gathered, which will include but not be limited to the number of AV which detected the payload, which type of payload used and with which encoding.

However, since these studies were published the tools and methods that are used around the world have been improved upon or new tools have been created since then. Programs such as MSFVenom [10] and methods such as code polymorphic have been invented. These are tools designed to make it harder for AV software to detect and prohibit malicious payloads from executing on the system. At the time of the latest study the antivirus products did not achieve a 100% detection rate. The results of a study such as this one will not only benefit companies around the globe, which all are in need of antivirus software, but also private individuals's evaluation of which antivirus software they will use for their own computer systems. Mark showed how the payloads were generated via a command line interface [8] and made use of the VirusTotal to calculate the detection rate of each malicious sample [8].

1.2 Background

As shown by the continuous, yearly, reports from FBI's IC3 department [3] the pattern of increasing cyber crimes continue to rise. As stated in the report both AV and malware are advancing their techniques and capabilities in an ever-evolving cycle, which motivates the necessity of evaluating the state of AV software's capabilities before an actual incident occurs. This is one of the main features of Metasploit, to probe the defense of AV and systems, which will be utilized to evaluate the market of AV all at once. Metasploit being one of multiple de facto standards of the industry when it comes to perform offensive testing and payload generation. It is not fully open source but does seem to introduce features and improvements based on community feedback [11]. AV solutions are designed to detect, prevent and remove malicious software before it is allowed to execute on a system, however as mentioned above, the threat landscape of malware keeps advancing and improving their capabilities [12]. The question then becomes: When does society, companies, researchers and developers know the state of AV capabilities? It's theoretically possible that all Metasploit generated payloads achieve significantly low detection rates which shows that the current state of modern AV software is not sophisticated enough to handle the current threats. It is also possible that almost all payloads are detected by the vast majority of AV solutions, which could show that Metasploit is perhaps not as efficient as believed amongst the industry and the authors, or that AV solutions have the upper hand against malware.

Without an evaluation introduced in this study it would not be possible to know the state of such AV software and their capabilities, the most prominent threats and which

areas of AV needs dire improvement, or if any improvement is required whatsoever.

1.3 Problem formulation

A great deal of businesses, people and systems fall prey to malicious code and cyber-attacks yearly [1][2][3], even though malware and AV software have existed in our IT environments for a few decades. To evaluate a certain aspect of why losses to cybercrime are increasing, is to research how efficient one of the lines of defenses is. However, as previously mentioned AV software and malware are in constant change as both sides evolve with new strategies and more advanced features which may render this research less useful in the coming years. It is nonetheless a necessary study to conduct as it will present valuable data which can be utilized for future research. With this in mind, it conceives a few research questions which will be the focus of the research conducted.

- RQ1: How efficient are antivirus software at prohibiting Metasploit payloads from executing?
- RQ2: Are encoded payloads less likely to be undetected by antivirus software compared to payloads lacking encoding?
- RQ3: Are there certain encapsulation methods that are more likely to bypass antivirus scanning than others?

1.4 Motivation

This research is effectively interesting due to the nature of the study and its results for the AV industry. Based upon our results the industry of AV will be able to, at a greater effect than previously, motivate investments into different sections of their AV software. As our results may indicate a lackluster efficiency against certain payloads a significant amount of advancements might need to be made in order for AV to enhance its capabilities against such payloads. It will provide the industry with significant results which provides the industry with knowledge regarding the state of both malicious tools and their capabilities as well as which areas which might need further improvement amongst AV software. Performing a study such as this one also provides the research community with a data bank of results which can be utilized by future researches in the cyber security field to measure improvements amongst AV software, improvements of testing capabilities with penetration testing tools such as Metasploit as well as motivate further research based upon the results.

1.5 Research results

The results gathered throughout this research will later be represented in a section of its own. The results will be presented in a format of graphs which show which payload-type was used, which encapsulation method it was encapsulated with as well as the volume of detections the scanned payload accumulated. With these graphs it will be possible to answer the aforementioned research questions, as well as be able to draw conclusions regarding the state of modern AV software and their capabilities. The graphs will ultimately depict the comparative effectiveness of various encapsulation methods, along with the performance assessment of each respective payload and its capacity to evade detection as malicious code.

1.6 Scope/Limitation

There are substantial limitations regarding this research, for one the study will not be able to draw any conclusions as to how each AV software differentiates between each other and their respective algorithms. Each AV uses similar methods of detecting malware, some are more advanced than others, however their distinct differences between each AV will not be presentable. It's also not possible for this research to optimize any of the binaries generated to decrease detection rates from AV software. A limitation for this paper is that the authors will assume that all payloads generated by the Metasploit framework will be deemed as malicious and should be flagged as such since the payloads generated originate to perform malicious acts against systems.

Another limitation is that due to VirusTotal Terms of Service agreement, any results which may impact the Antivirus industry in any manner is strictly forbidden. The policy which would be in conflict with the intended research is the following policy: "Copy, reproduce, alter, modify, create derivative works, publicly display, republish, upload, post, transmit, resell or distribute in any way material, information or functionalities from the Service – including, without limitation, using the Service in any way for antivirus/URL scanner testing or that could directly or indirectly harm, compete with, or otherwise hinder the antivirus industry/URL scanner industry." [13] which was argued by VirusTotal to us would be exploited if the vendors were revealed in the results. For this reason, the results will be anonymized to solely point out the general market of antivirus effectiveness. The research questions stated above does however still apply to this limitation as the results will still provide sufficient amounts of data to be able to draw educated conclusions from. Another limitation is the speed at which payloads are able to be sent for scanning at VirusTotal's website, which is limited to four requests per minute, but running on two different computers at the same time, increasing the number of requests per minute to 8. However, this comes at the cost of not being able in a timely manner to study payloads for all available encoders. If all 43 available encoders, within the Metasploit framework, were tested with the vast amount of payloads a rough estimate of the duration of the study would take 86 days to solely upload and receive response from VirusTotal. To abide by the time schedule the authors were forced to limit the amount of encoders, which were chosen by the type of encoders which still allowed payloads to execute.

There is a last, noteworthy limitation with the available payloads however. To effectively compare results when using encoders there must be a case for each payload where no encoder is chosen and another case for the same payload where encoders are chosen, to be able to determine the difference of using encoders in the results. This means that there are cases (payloads) where the payload is un-encapsulated as well as encapsulated. Be that as it may, there are payloads from the Metasploit framework which do not have any compatible encoders that allow the payload to execute, and therefore be detected by AV software which means such payloads are excluded from this research. If the payload in question is encapsulated with a certain encapsulation method that does not allow the payload to execute, no AV will detect the payload as malicious and thereby corrupt the results. This effectively limits the research to only include payloads which do have compatible encoders that allow the payload to execute. These payloads will be visible in the results section of the paper. For x64 versions of Windows, the available encoders for x64 systems are limited by Metasploits available encapsulation methods, which is why such payloads are only encapsulated with two encapsulation methods, instead of three.

There are mainly 6 modules of encoders which were chosen, to allow for suc-

cessful execution of as many payloads as possible (considering the limitation mentioned above). The following Metasploit encapsulations were chosen for their respective payloads:

- non_alpha
- powershell_base64
- shikata_ga_nai
- x86_xor_dynamic
- x64_xor_dynamic
- zutto_dekiru

Thus, this paper's scope is clear, to solely focus on the generated, encoded, payload's detection rates as well as other data produced VirusTotal and if different encapsulations (which still allow for payload execution) effect these results. Another limitation is that it will not be possible to explain all tested payloads as this would make the paper too heavy and redundant, which is why a higher level of explanation of each type of payload will instead be explained for the theoretical background. The scope is similar to that of [9] as well as [8] but increases the amount of researched payloads and aims to provide are more recent and broad overview of MSFVenom anti-virus coverage.

1.7 Knowledge Contribution

The knowledge contribution this paper is aiming to provide is the analysis on the available anti-virus software solutions, and if they are up to speed and or need improvement in their detection ability regarding different factors, such as encapsulation methods and payloads. The paper also wish to identify if certain payloads are performing better at evading detection by antivirus software which could imply that a certain aspect of payloads may need further improvements to increase detection rates for that specific payload. This research also wish to deliver a qualitative study which can be used by further research as future payloads arises in the market which will motivate research regarding the capabilities of AV software. This work will provide a data bank of sorts which can be used for multiple purposes in the future, such as evaluation of specific software when attacked with Metasploit payloads, hardening efficiency on systems as well as contribute a source which can be used to compare detection rates in the future amongst AV software. This research can also be used as a reference for future researchers to base their methodology upon, as this study produces a significant amount of cases with varied results, which could then be further expanded upon as development in the AV industry continues. Lastly if the studies that this paper is based upon still reveal the same results today, as they did when said studies were performed.

2 Methodology

This section will delve into the scientific methodology for the research which has been conducted as well as the practical methodology. This research will take form as a case study which is used to research data gathered throughout different scenarios with realistic settings. This case study is set out to fullfill a quantitative aspect as it will produce measurable results in the form of detection rates across software AV's with the help of a developed software and published tools. It was chosen as the most suitable study to conduct as our research will produce a significant number of different scenarios with the use of practical means, where each scenario will produce different outcomes. Therefore, each case's data that is produced will need to be analyzed and collected with the other cases to be able to present meaningful results.

The research began with a small-scale literature study with the intent to gather knowledge regarding previous research and if any had produced useful results for our own research in the field. Another strong motivation for this course of action was also to gather reliable information from previous results which can be used as the foundation for the continued research. The gathered results from the literature research were that our intended subject of research had a notable knowledge gap as as well as lacking research in recent time. The literature study was solely based on published papers and reports. This literature research was the first step in laying the foundation of our own research and will might be seen to be revisited to further the literature research if there is an inadequate amount data. The second part of the research will fulfill the quantitative aspect of the methodology. With the use of a self developed software program which gathers data and logs from the VirusTotal website with the use of their public API-keys. The data which is generated will be analyzed and from there our conclusions will be drawn. Based on each test case there may be different results, positive ones which indicate a successful test and negative ones which indicates a negative results. Once all results have been gathered and analyzed from each case's result, it will be possible to draw truthful conclusions based on said results. There are however limitations to the quantitative research, as the data will not reveal any information regarding how each AV works and therefore the study will not be able to explain in its entirety what separates the performance of each AV. Below, the technical aspect of our quantitative research will be explained progressively.

The goal of our research is to gain a deeper understanding of the effectiveness of modern AV's when faced against a popular attack framework. The main method used to acquire the results required is to create a substantial, quantitative, amount of cases, in the form of malicious payloads, which is to be tested against modern AV's and their abilities to effectively detect the payloads. The study will accomplish the need for a large amount of test cases with the use of an autonomous program which utilizes the popular framework 'Metasploit' with the tool MSFvenom [10]. To further satisfy the need of the large amount of test cases the program will combine each payload available with multiple encapsulation methods, with the aide of MSFvenom. This will create a large amount of payloads, where many are of the same origin but are encapsulated with different encapsulation methods.

Once the payloads have been created and encapsulated, these will then be scanned by multiple commercial AV's with the use of VirusTotal [14] which will provide test results regarding the AV's effectiveness in detecting the malicious payloads. VirusTotal is a handy tool for the research as it allows for quick scans of payloads across a multitude of AV softwares. This increases efficiency by a large magnitude, to be able to test each

payload simultaneously with multiple AVs instead of using a single OS installed with each AV software, is what makes this research reasonable to conduct. Considering the fact that the assortment consists of multiple files that stem from the same payload, but with different encapsulations, the need to use a service such as VirusTotal only grows larger. VirusTotal will also be able to provide all the necessary data from the case results for this work to be able to present reliable results.

After a payload file is uploaded to the VirusTotal website, VirusTotal will scan the payload file with a multitude of AVs, then present a number of which of the AVs detected the payload as malicious, the number of total AV which were used for the scan as well as the total number of AV that detected the payload but stamped it as harmless. All of the provided data from VirusTotal will be requested after the complete collection of payloads has been scanned. This is a preferred methodology to use as the alternative would be to set up a virtual machine/s (VM), install it/them with multiple AV software and then test each software independently from each other with each payload. Already that optional method would harvest a large magnitude of time and effort. There may however surface validity concerns regarding the quantitative research, as each AV may categorize the generated payloads differently, and may state that the payload is harmless or simply state that the payload is a PUP (Probably Unwanted Program) which is a broad generalization and can be seen as an invalid statement. The final step of requesting the data from VirusTotal will be the capstone in answering of all research questions. As the chosen methodology is a case study, each case (payload result) will provide the necessary data required to grant answers to the research questions, RQ1, RQ2 and RQ3. This is a preferred approach as the authors are reliant on the methodology brought up by [8] & [9]. There are nonetheless flaws with the studies [8] & [9] as the methodology used is limited in the amount of payloads chosen in their study, which in turn limits the results presented and adds a coating of uncertainty with the reliability of their results. This is why this paper's methodology will include a vast amount of payloads to increase reliability when presented.

2.1 Validity & Reliability

The chosen methodology can be seen to be used in [9] as well as in [8] where both studies relied on a similar methodology. By creating payloads with the use of Metasploit and then uploading said payloads to VirusTotal for scanning against AV softwares. This also produces a higher level of validity and reliability as the data which is measured comes directly from the source, VirusTotal, and proves that the technical methodology has been tried and tested previously in previous work [9][8]. There is however a concern regarding the reliability of the results, as there may be cases which are received as errors from VirusTotal. Some payloads when being sent to VirusTotal will give an error and is therefore not scanned. However the amount of failed payloads is amongst tens and will therefore not affect conclusions drawn from the results due to the fact that the total amount of successful payloads vastly outnumberes the failed payloads. Therefore the results will still be reliable due to the quantitative aspect of the research. The reliability will grow larger the number of test cases used grows, which motivates the reasoning to use as many payloads as possible to present a reliant & valid result.

2.2 Metasploit motivation

Metasploit's purpose of use is both quantitative as it is qualitative. It is a framework which is regularly updated, used by professionals across the world, easy to use and yields helpful results for many who desire to test their systems defenses. Using Metasploit simplifies the creation of malicious files with its vast library of payloads to simulate realistic attacks. As Metasploit allows users to simulate realistic attacks, it also provides skill development for those that utilize it to gain experience with realistic scenarios. There are also a large community in regards to Metasploit, which enables professionals, security experts and others to share what they have found, exploits, weaknesses or even techniques. By having a community share it's knowledge with each other, it advocates for a mutual defense amongst individuals and businesses whilst simultaneously enhancing the security for all as new information is spread.

2.3 Ethical concerns

When academic research arrive at the topic of cybersecurity, there are often ethical concerns to consider before proceeding with the study in mind. First and foremost, the reader should not be able to gather any practical data, such as code, to be reiterated in their own malicious environment. Due to this reason any code developed and used for this research will remain private and non-published. Second, the results from this research should also not negatively impact the cybersecurity industry, this means that our results should not impact the industry by pointing out specific AV vendors as if they are less qualitative than another AV vendor solely based on the results. This is mainly due to comply with the Terms of Service put forward by VirusTotal [13]. To resolve such an ethical issue, our results regarding specific vendors will be anonymous, and a more general presentation of this paper's findings will be shown.

As mentioned, the framework of Metasploit is heavily relied upon for this paper and with that framework there are also ethical concerns surrounding it. While the framework itself is a legitimate tool and is intended to be used for security improvements, it can also be used for nefarious reasons. The intention behind the use of the tool is important to note, as it is solely used to investigate the aforementioned research questions, raise awareness regarding security as well as improve the field of cybersecurity in terms of AV software. its use will not bring any harm to any parties involved and the software which relies on the framework itself is, as mentioned above, not publicly published.

2.3.1 Data gathering

Our methodology of gathering data from and with VirusTotal is within the confines of ethical, it is a public use of their published API and is within the body of their terms of service, which has been agreed upon before using their service. As have been previously discussed the generated payloads, with the use of Metasploit framework, are sent to VirusTotal with the use of their API. From said API, VirusTotal will store data and information regarding the results from the scan with each payload on their site. Each payload sent, will store a response at VirusTotal's side, which will be gathered all at once after all payloads has been sent and scanned. Once all payloads has been scanned, the response is gathered and the results will be ready to be analyzed. The data which VirusTotal responds with will include information such as the number of AV's which identified the payload as harmless, suspicious, malicious, undetected and type-unsupported. Altough this data is given by VirusTotal, not all of the data will be used in the results as the main focus is the

detection rate for different payloads with different encapsulations and the total detection rate across all payloads.

Once all data has been gathered, it is filtered through another program which filters the JSON data, which VirusTotal responds with, to a CSV file with the necessary data to create graphs from. The data filtered for is the payload type, amount of payload types, encapsulation method and the detection rate.

2.4 Data gathering tool

To be able to produce payloads, encapsulate them, upload the generated payloads to be scanned at VirusTotal (with the use of their API), gather the results and then finally analyze the results a self produced software was needed. The authors developed this software, which excels in generating numerous amounts of payloads. Each payload is encapsulated using a unique method. The software merges all the payloads into a single file, as well as a copy of each respective payload that is treated as the control subject due to it remaining unencapsulated. Finally, the software submits all these payloads for scanning. To limit the risks mentioned in section 3.1 *Ethical Concerns*, namely regarding exploitation of our software, the details regarding the produced software will be kept at a minimum. Hence the idea of the software is what will mostly be brought up and not details regarding its function.

3 Theoretical Background

This section will delve into the theoretical aspect behind numerous of technical aspects which are important to understand for the purpose of this research. To lack knowledge regarding these aspects is to misunderstand the results and the implications that the results presents. Therefore, the general types of payloads will be presented below, as well as some of the encoding methods used in this study.

3.1 How Antivirus software works

Antivirus software all use different as well as combined methods of detecting if there are harmful binaries running on a system. Some are more advanced than others, and some are completely outdated in its effectiveness to detect modern malware (malicious software). Signature Detection is one of the methods used by AV vendors, which utilizes three different analysis methods, static, dynamic or hybrid.

3.1.1 Static analysis

Static analysis functions by dismantling code into manageable parts, where nefarious lines of code can more easily be identified. However, if the code which is being analyzed has used obfuscating methods applied to it, the static analysis may not yield the appropriate results. [15]

Dynamic analysis can be deemed as a larger risk, as it functions by intentionally executing the file and then analyzing how the file affects the operating system, as well as how the operating system reacts to the file's execution [15]. This is not flawless however and there are still possibilities for malware to go through the process undetected.

Lastly, signature detection utilizes the method of hybrid analysis. Hybrid analysis works by combining both static analysis and dynamic analysis. It will, as with dynamic analysis, intentionally execute the code and investigate reactions as well as disassemble the code to detect nefarious code lines within [15]. This will further increase the efficiency of signature detection method amongst AV's but as with all malware, there is always a possibility that the AV software might not properly detect a malicious file. This can be due to multiple reasons, where one of the reasons could be that the malicious file type has not been seen before and is a recent version of a newly found vulnerability that has not been updated amongst AV vendors.

The AV method of signature detection showed in 2019 that most vendors eventually achieved positive results in the later years in their detection study, ranging from the years 2011 to 2018. Once a file has been identified as malicious, the AV may update it's database of known malicious signatures, adding the recently identified file to the data storage. However, for this method to work the system already has to be infected with the malware for it to be analyzed within the machine. The file may not have been executed as of yet but nevertheless the malicious file has entered the system. [15]

3.1.2 Cloud detection

Cloud detection is a slightly different method compared to the more straightforward methods of only having AV software directly installed on the intended system. Cloud detection utilizes a client (the protected, local system) and a remote web service running on a different system. The client has a smaller, less resource costly software running on the local

machine which purpose is to scan the system for nefarious files and malware. The reason why cloud detection consumes less resources than AV directly installed on the local computer is due to the fact that the main powerhouse of the solution is being run on a remote cloud server [16]. This remote service stores data and keeps track of known malware as well as dangerous websites, which is shared with the clients. However, the drawback on such a solution is that if the local computer is attacked or infected with an unknown/lesser known malware, the system may not detect it. [16]

3.1.3 VirusTotal

VirusTotal founded in 2004 in Dublin, Ireland is a company that allows everyone to upload files, executable or otherwise, to the companies website and scans said files/urls with over 70 anti malware solutions and blacklists. VirusTotal has since grown to have more than 500 000 members and provides an easy and reliable way for anyone to check their files for malware. As measured by Liebergen, Caballero, Kotzias and Gates [17], files make their way into the VirusTotal platform on average 4.4 hours after they first appear in users devices. This and other aspects of the VirusTotal platform. Their paper called "A Deep Dive into VirusTotal: Characterizing and Clustering a Massive File Feed" more specifically, made use of VirusTotals, File Feed feature of where entities get access to samples as they are submitted to VirusTotal.

VirusTotal shares samples to multi Anti-Virus vendors and security organizations, thus creating a symbiotic relationship wherein users make use of VirusTotal to scan their files and Anti malware vendors gather pool of possibly interesting samples from which to improve their products from.

Other than the main Web User Interface, VirusTotal also provides an Application Programming Interface (API) [18]. This API is the only interface which will be used to interact with the VirusTotal services through the course of this thesis.

3.2 Payloads

This paper covers a large magnitude of different payload types and most of them are explained in short below. A payload can be thought of as the content which is delivered through packets by network traffic. The payload itself is deemed as having the capacity to execute malicious code or perform malicious acts on a system. It is important to note that there are multiple of other payloads, which are not listed, as these may be very similar to ones explained, and listing them again would take a tremendous amount of effort and time only to display small differences to other similar payloads. There may also be differential version of the mentioned payloads, however, it is not within the scope of this paper to present them all, which is close to an impossible task in itself. Therefore the general idea of each payload which has been used in this study will be presented and explained in short detail.

3.2.1 Meterpreter

Meterpreter is a Interactive Shell that allows attackers to remotely perform functionally the actions on the target system as a users that was present physically would. Meterpreter was developed by the same developers of Metasploit and thereby MSFVenom. Meterpreter

3.2.2 Patchupmeterpreter

The patchupmeterpreter family of payloads make use of the meterpreter interactive shell explained above with the difference being that the payload gets "injected" into another process. This results in the meterpreter instance being "memory-only". A higher level takeaway that results from this payload being delivered va DLL-Inject is that the payload never(as a runable executable) is never placed in the disk. This means that Anti-Virus solutions have a much harder time detecting and removing this payload. Another benefit for attackers is that the meterpreter instanced that is launched get ran with the injected processes permissions.

3.2.3 Powershell

Powershell payloads make use of Microsoft's Powershell. Powershell is as the name suggests, a more powerful Shell [19]. Powershell combines the standard Windows Command Line Interface, conically as "CMD" with Powershell cmdlets and bigger integration with Operating Systems. Cmdlets are collection of functions natively implemented to the Powershell platform that don't require native binaries to run [20]. This environment is commonly used by system administrators to do configure systems through the command line interface. Powershell also allows the creation of scripts that combined with cmdlets make the perfect tool to manage every aspect of an operating system, thus making it the perfect tools for attackers to use [19][21].

3.2.4 Upexec

Upexec is an abbreviation of "Upload and Execute" and is part of a family of malware known as "Droppers" or "Loaders". The main characteristic of this type of malware is it mainly focusing on delivering other types of malware/payloads. This type of payload can be used by malicious actors to bypass security systems, architectural requirements, check if the target system is vulnerable to a certain exploit before deploying the final payload.

3.2.5 Shell

The shell family of payloads, are a group of payloads that achieve their goal by only making use of the built in Command Line Interface of the target systems. This can be very effective at not only avoiding detection, but at avoiding runtime dependencies. The main reason for this sort of payload being so successful at avoiding detection is because it only makes usage of existing resources in the system. This type of payload often make use of a series of Command Line Interface command in a text file, commonly called of "Scripts" to achieve their goal. Command Line Interfaces by themselves serve very little purpose, but combined with a group of programs that are distributed to Operating Systems by default make Command Line Interfaces a useful tools for users and malicious actors alike.

3.2.6 Pingback_*

A pingback payload main objective is to create a covert communication line between a victim's machine and the attacker's machine. By utilizing ICMP (internet-control-message-protocol) the attacker can gain confirmation of an executed action on the victim's

machine. By creating a UUID (universally-unique-identifier) for the pingback payload, the attacker can listen for an ICMP message with the set UUID coming from the victim's machine to verify that the execution has occurred on the remote target. This makes it more difficult for network administrators to catch as the UUID (verification) sent back to the attacker's machine is simply a random string of characters, and they payload may also send at certain time intervals on different ports each iteration. This would increase persistence from the attackers point of view. [22]

3.3 Payload communications

Listed below are some of the techniques used by payloads to achieve communication to or from a target system. There exists plenty of such methods which are not listed here as it is out of scope, but the ones listed are mainly found throughout the payload creation process with Metasploit. To be able to draw proper conclusions later on in this paper these are explained in short manner to provide an understanding of the results and the analysis.

3.3.1 Reverse_http / Reverse_https

A reverse http (hypertext-transfer-protocol) payload's intention is to connect back to a malicious actor's computer to establish an interactive session for the malicious actor to utilize. It is established through the HTTP protocol and may therefor be deemed as valid traffic by some network monitors / firewalls as it is also initiated from the victim's system. However, as this payload only uses http, the traffic can be inspected by monitors and therefore be denied. A payload like reverse https (hypertext-transfer-protocol-secure) would make this action more difficult for a security officer to perform as the traffic would be encrypted and might be allowed to bypass some security measures. [23]

3.3.2 Reverse_tcp

A reverse tcp (transmission-control-protocol) payload often means that the intended event to occur is that the target system will, with some type of terminal, connect back to the malicious actor with an interactive and stable connection through a shell with the use of the tcp. With this connection, the malicious actor can perform a magnitude of desired actions on the system, such as exfiltrate files, gather data, deploy further payloads etc. This payload is possible to bypass some firewalls as the connection is established from the victim behind the firewall and may therefore be regarded as valid traffic meant to be delivered properly. There is also a paylod which uses UDP (user-datagram-protocol) that uses a stateless transmission line instead of a controlled once as with TCP, these payloads are called *reverse_udp*. [24]

3.3.3 Bind_*

A bind module works by opening up a port on the target machine and then exploit the system by the malicious actor through initiating a connection to the opened port. This can be used in a number of different ways, which strategically work in a similar manner but may utilize different ports / protocols combined with different payloads. Once the malicious actor has an established connection it is then possible to execute code, install further payloads or desired software for further nefarious use, control functions on the victim's machine itself (logs, security functions etc) to achieve full control of the target.

This type of method with payloads can be combined for multiple outcomes and techniques as it is more focused on the transfer of data and established connections. [25]

3.3.4 Find_port

A find port payload communication method's main objective is to either find a specific port, or a certain range of ports. Ports are often bound to specified and standardized protocols, such as HTTPS uses the port number 443, and DNS (Domain-name-system) but there are a multitude of ports which are free to use for variable reasons. This payload can be utilized to search for vulnerabilities within a system, e.g if some miss managed ports are open they can potentially be nefariously used as a stepping stone to initiate further attacks on the system.

3.3.5 Reverse_shell

A reverse shell payload / attack can take many shapes, but the idea behind a reverse shell is to let the target system connect back to the malicious actor's system. This is deemed as a safer alternative as it could allow the traffic to bypass security measures such as firewalls, IDS (intrusion-detection-system). The malicious actor will listen on a specified port, and the payload will generate a shell on the victim machine, which then connects to the malicious actor's machine, allowing the malicious actor to perform a plethora of actions, many of which would be deemed as malicious. [26]

3.3.6 Hidden_ipknock

The hidden_ipknock payload explains in it's name that is has an attribute to try and stay hidden and avoid to make noise for systems and its administrators to notice. It also uses the security technique known as "ipknock" which in itself is also particularly a stealthy method which is meant to, without authorization, open ports externally which are intended to be closed. By attempting to send certain packets to defined ports in a specific sequence (knocking), eventually the firewall might open one of the "knocked" ports and therby accepting a connection. Consequently the attacker which imitated the transfer of the payload can initiate a shell and execute further actions. The reason this is regarded as stealthy, and hidden, is due to the fact that the socket is visible as closed on the system, yet it is open logically. [27]

3.3.7 Named_pipe

The named_pipe payload is meant to be used to establish connections with victim machines with the attacker machines to allow further actions from the attacker. The named_pipe payload will either listen for a pipe connection to be made through a port, or attempt to connect back in reversal to the attackers machine. It can also be used for pivoting maneuvers. [28]

3.3.8 nonx

The nonx payload is intended to prevail against DEP (Data Execution Prevention), which is a module implemented in Windows based operating systems. DEP is derived to prevent executable code from launching on the systems from unintended locations that could be vulnerable [29]. The nonx payload is intended to bypass this protective technology and

allow connections from the victim machine to the attacker to grant further access for the malicious actor to the target system. This can be performed with a reverse method where the client initiates the connection from the victim system to the actual attacker [24], or with a bind methodology which connects to a specific port on the attackers machine [25].

3.4 Encoders

Payload encodings in MSFVenom are post-payload generation processes that alter the payload to make it less likely to be detected by antivirus software, but do not alter the behavior or effectiveness of the payload itself. MSFVenom offers a broad range of different encoders that can be applied to each of the various payloads that it can create [11].

Many of the MSFVenom encodings evade antivirus protections by increasing the payload complexity and also randomizing the payload to change the static signatures that can be collected from the payloads [30]. It must be noted, however, that modern antivirus solutions not only perform static signature analysis, but they also perform behavior and heuristic analysis meant to defeat the sort of payload encoding and obfuscation mentioned previously [12]. The caveat for this stipulation is that as encodings and obfuscations adapt and become more complex, so too must antivirus products adapt to the ever-evolving tactics if they want to provide good coverage and defense, as mentioned by [31].

MSFVenom comes with 45 different payload encodings. But they can mainly be split into three different categories of encodings. These being XOR based encodings, Alphanumeric based encodings and Base64 based encodings. Each of these have their benefits and drawbacks. This case study is mean to measure their effectiveness at making payloads evade current anti-virus solutions. As mentioned previously, the research is limited to utilizing a single encoder that belongs to one of the mentioned categories resulting in the use of three different categorized encoders for all available payloads. Below, the encoder's categories are described.

3.4.1 Base64 Encoding

Base64 encryption is a simple and highly effective algorithm that transforms binary data into a base64 encoded string that does not resemble the original data [32]. Base64 although not secure as an encryption algorithm, does provide an highly efficient encryption method that provides data security [32]. Base64 has a history of being used by malware to evade anti-virus solutions as described by Sekar Veerappan et la[30] in their paper "Taxonomy on malware evasion countermeasures techniques" in which the authors describe various methods on how malware attempts to evade Anti-Virus solutions [30]. An interesting feature of the base64 algorithm is the encrypted characters representation is differently depending on the characters position[32]. Which helps payloads to become completly unreadable[30]. The MSFVenom *base64* encoder used for this project in accordance with the limitations stated above is *powershell_base64*

3.4.2 Xor Encoding

XOR encoding makes use of the XOR or Exclusive Or operation present in all computers to encrypt the payload. Xor encryption is a Symmetric key encryption method in which the same key used to encrypt the content is also used to decrypt the content. This means

that the key to decode a le encoded with the technique must be contained in the nal encoded le as well.[33]

The MSFVenom XOR encoders used for this project in accordance with the limitations stated above were x86_shikata_ga_nai, x86_xor_dynamic, x86_xor_dynamic, zutto_dekiru and x64_xor_dynamic

3.4.3 Alphanumeric Encoding

Alphanumeric encoding is similar to base64 encoding in the fact that it is not speci cally made to bypass anti-virus solutions. Similarly to Base64 encoding it also only converts data into the ASCII only format. Although it restricts it to being alphanumeric representation. This is mostly helpful when bypass character set limitations of the different targets of the payloads[34]. But it also is another data representation method that anti-virus must account for, thus making it a valid technique to obfuscate payloads.

The MSFVenom Alphanumeric encoder used for this project in accordance with the limitations stated above is x86_nonalpha

4 Results

In this section the results of the study will be presented. There will be graphs / tables to present the data gathered with the information to answer the research questions presented previously.

Figure 1: Results of payloads encapsulated with all encapsulation methods.

Figure 1 displays a graph that focuses on comparing results between different encapsulation methods and their respective ef ciency in encapsulating their respective payloads. As was mentioned in Scope/Limitations chapter, there are encapsulations which do not suite some payloads which is a factor that has to be considered when viewing the results as there may be differential amounts of payloads per encapsulation method. However the amount of payloads which were not included were not a substantial amount in comparison to the total amount of payloads sent. That factor might however affect conclusions drawn regarding payloads ef ciency for certain operating systems, which will be brought up in the conclusion section below. In Figure 1 there are a total of 981 payloads depicted in the graph above. Nonetheless, Figure 1 shows a clear top performer amongst the payloads, which is the x86_xor_dynamic encapsulation method as well as, alarmingly enough generic_none encapsulation method. These two modules seems to

have achieved the lowest rate of detection rates for their respective payloads, as their amounts of payloads each has a low degree of detection rate. The worst encapsulation methods that provided the highest detection rate amongst its respective payloads were the xor_dynamic, zutto_dekiru and x86_shikata_ga_nai encapsulation methods. These payloads rank amongst the worst encapsulation methods as they had a high figure of detection as well as a significant amount of payloads encapsulated. The actual number of payloads per encapsulation method will be provided below.

Figure 2: Results of payloads encapsulated with non_alpha.

Figure 2 depicts the results achieved with the encapsulation method non_alpha and its respective payload's detection rates. There were a total of 91 successfully generated payloads with the encapsulation method of non_alphaand it showed to have a majority of its detections around 78-89%. This gure does not provide a particularly confident result of it's capability to encapsulate data enough to remain undetected by modern AV software. The payloads with the highest detection rate are the upexec payloads and the payload which achieved the lowest detection rate with this particular encapsulation was meterpreter and shell which achieved around a 9-13% detection rate.

Figure 3: Results of payloads encapsulated with powershell_base64.

Figure 3 depicts the results achieved with the encapsulation method powershell_base64 and its respective payload's detection rates. There were plenty of successfully encapsulated payloads with this encapsulation method, a total of 215 payloads were sent and scanned by VirusTotal. This encapsulated achieved quite low detection rates for its encapsulated payloads, with the majority of payloads resulting in detections rate between 33-36%. The payload type which achieved the highest rate of detection rate were the powershell payloads and the windows_exec payloads which landed at 33-36% detection rate. The payload with at least a noteworthy amount of payloads which achieved the lowest detection rate was still the powershell payload, at 31%.

Figure 4: Results of payloads encapsulated with shikata_ga_nai.

Figure 4 depicts the results achieved with the encapsulation method shikata_ga_nai and its respective payload's detection rates. This encapsulation method was able to encapsulate a total of 89 payloads which were then scanned by VirusTotal by modern AV software. The results display a larger variety in detection rates than the previous encapsulation methods. However, the majority of its encapsulated payloads gained a detection rate of 78-79% but contained payloads that ranged from 5-76%. The payloads which achieved the highest detection rate were a multitude of payloads, which resulted in upexec, shell, meterpreter and patchupmeterpreter having the highest detection rates, around 76-79%. The payload types which resulted in obtaining the lowest detection rates was also a mix of payload types, which were the shell payload type as well as meterpreter which had a large range of detection rates. Albeit the amount of such payloads which realized low detection rates were low, it is noteworthy nonetheless that some of such payloads achieved quite lot detection rates. A total of 25 payloads achieved detection rates below 20%.

Figure 5: Results of payloads encapsulated with x64_xor_dynamic.

Figure 5 depicts the results achieved with the encapsulation method x64_xor_dynamic and its respective payload's detection rates. This particular encapsulation method was able to encapsulate a total of 43 payloads, and resulted in its majority of respective payloads having a 73-79% detection rate. The payload types which achieved the highest rate of detections that were encapsulated with x64_xor_dynamic were quite clearly, vnc_inject, shell, as well as meterpreter which achieved around a 73-79% detection rate. Since the total amount of encapsulated payloads sent with this encapsulation method were not many, and due to the fact that the majority of all payloads sent attained a detection rate of >70% proves it dif cult to determine a payload type which held a strong position of lowest detection rates.

Figure 6: Results of payloads encapsulated with zutto_dekiru.

Figure 6 depicts the results achieved with the encapsulation method zutto_dekiru and its respective payload's detection rates. This encapsulation method did not achieve as many successful encapsulations of payloads, with a total number of 43. The encapsulation method itself did also not impress with the majority of its encapsulated payloads achieving a detection rate of 75%. The payload types with the highest rate of detections were the vnc_inject, windows_exec, meterpreter and vnc_inject which attained a detection rate of a 72-75% detection rate. The payload types with this encapsulation method which attained the lowest detection rate were the powershell. Albeit all payloads with this encapsulation method achieved high detection rates and the powershell payload did not stand out too much considering there were barely 10 payloads of powershell encapsulated and scanned.

Figure 7: Results of payloads encapsulated with x86_xor_dynamic.

Figure 7 depicts the results achieved with the encapsulation method x64_xor_dynamic and its respective payload's detection rates. This particular encapsulation method was able to encapsulate a total of 101 payloads. It is however noteworthy that the encapsulation method successfully sent 25 payloads which had a detection rate or 10% or lower. It is noteworthy for this graph to view the range of detection rates, as these are changing results depending on the type of payload sent. Meterpreter and shell payloads achieve both amongst the highest and lowest detection rates. The payload type which attained the highest detection rate for the amount of payloads, was without a doubt the upexec, shell, pathupmeterpreter and meterpreter payload types with a detection rate of 76-79%. The payload types with the lowest detection rates however were also meterpreter and shell.

Figure 8: Results of payloads without encapsulation.

Figure 8 shows the difference between payload detection rates when no encapsulation method was used. This encapsulation method was used for all respective payloads to be able to measure each respective encapsulations ef ciency compared when there was no encapsulation of the payload. This encoding option for the payload does not alter the payload in any way. The majority of payloads have the detection rate of 33-36%, which is an alarming state of the AV industry as the total amount of unencapsulated payloads scanned were 354. The payload types which attained the highest detection rate whilst lacking encapsulation were upexec, osx_exec, patchupmneterpreter, vnc_inject and meterpreter which landed at a 75-80% detection rate. The payloads which achieved the lowest detection rate whilst having a large volume of payloads from VirusTotal were vnc_inject, upexec, windows_exec and powershell. These payloads have a detection rate around 31-36% detection rate which is a critical result and will be analyzed further.

Figure 9: Results of Powershell payloads encoded versus not encoded.

Figure 9 shows the difference in payload detection rates between base64 encoded Powershell payloads compared to the same payload set without encoding. The majority of the encoded payloads seem to have the same detection rate as non-encoded ones. This seems to indicate that base64 encoding is not a very effective encoding method to evade Anti-virus solutions. Another interesting observation that can be taken from Figure 9 is that the lowest detection rates of 17-18% are from non encoded payloads. Suggesting that the encoding method in itself is being detected as malicious.

Figure 10: Zoomed in view payloads with lower than 20% detection rate

Figure 10 shows the amount of payloads and their respective encoding methods with lower than 20% detection. Of the 981 payloads that were uploaded to VirusTotal only 86 payloads had lower than 20% detection rates. Of those 86 payloads, only 11 have not been encoded. The payloads with the lowest detection rate were non encoded ones as they had a zero percentage detection rate. This two payloads as shown in 8 were payloads of the metsvc type.

5 Analysis of results

This section will delve deeper into what the results can conclude as well as use the results to answer the research questions presented in the beginning of the paper. The analysis will begin with examining each respective encapsulation method and its payload's performance against AV software supplied by VirusTotal.

In Figure 1 there is a lot of data to unpack and analyze. The analysis will begin by looking at each encapsulation method's performance in regards to their payloads to be able to draw conclusions regarding RQ1 and RQ3.

- The encapsulation method zutto_dekiru's performance in its' encapsulation of payload can be seen as quite poor. It achieved a detection rate of 72-75% detection rate for all of its' payloads. This means that the majority of modern AV software are able to detect the majority of all payload types tested which were encapsulated with zutto_dekiru and in this scenario AV software is ef cient at preventing such payloads with this encapsulation type to execute.
- The encapsulation method x86_xor_dynamic resulted in a large variaty of detection rates as can be seen in gure 5. This encoding method however can be deemed as the most ef cient one due to it having 25 payloads sent which resulted in detection rates below 10%, which converts to 24% of the total amount of payloads had detection rates less than 10%. The x86_xor_dynamic encoding method does, like many of the other encoding methods, vary in detection rates depending on certain payloads. Some of the payloads has low detection rates (<10%), such as shell and meterpreter, whilst such payload types simultaneously also resulted in payloads having a 78-79% detection rate (Figure 5. However, the majority of payloads encoded with x86_xor_dynamic resulted in detection rate between 78-79%. This goes to show that in most cases, AV software will prohibit the execution of malicious code if it is encoded with x86_xor_dynamic There are still a minority of cases where the payloads have such low detection rates that, proved through this data, the con dence in the AV softwares abilities to prevent the execution of the payloads fail.
- The encapsulation with a base64 xor_dynamic did not achieve such low detection rates as the other encoding methods. The lowest detection rate for this encoding method were two payloads of the same type, vnc_inject, which resulted in 60% for one and 63% for the other. The remaining payloads, of which there were 43, resulted in detection rate >70%. This, similarly to zutto_dekiru, shows that modern AV software is quite ef cient at prohibiting payloads encapsulated with this encoder from executing on systems.
- The encoding method x86_shikata_ga_nai achieved a very varied results for its' respective payload types. 25 payloads encapsulated with this encoding attained a detection rate of less than 20% which is remarkable, even though it's the minority of the total amount of payloads sent, this means that 28% of payloads encoded with x86_shikata_ga_nai has a detection rate of less than 21%. For this encoding method, the ef ciency of modern AV software does inspire con dently that the AV solutions are fully capable to prohibit execution of malicious payloads.
- The encoding method x86_non_alpha's payloads resulted in similar results to that of the results of x86_xor_dynamic encoder. Referring to the gure 2, there is a

distinct result showing that the encapsulation method has by large failed to hide its malicious payload from AV software's scanning. There were however a total of 24 payloads which amassed a detection rate of <13%, out of a total of 91 payloads as shown in gure 2. This concludes that 26.3% of the payloads encodes with x86_non_alpha's achieved relatively low detection rates. The remaining majority of payloads for this encoder did however achieve reasonably high detection rates, around 78-79%. This also means that in general the encoder will not aide with bypassing AV software, albeit a minority of cases the encoder will increase its' capability to bypass detection from AV softwares. This does also not inspire any con dence in the ef ciency of detection by AV softwares with such a high amount of payloads with low detection rates.

- The encoding method powershell with the base64 did yield interesting results when compared to the results where no encoding was used (gure 9). According to the data gathered, the use of the encoder powershell_base64 did not provide any remarkable results in regards to detection rate when compared to one another. With the exception that the total amount of payloads sent with the encoder powershell_base64 is more vast than the amount of payloads without an encoder. It is nonetheless noteworthy that the detection rate does not change when encoded with the powershell_base64 method compared to when none of the payloads are encoded, as seen in gure 9. This shows that the ef ciency of AV software is not satisfactory as one would expect the detection rate to be higher for generic_none payloads, but also goes to show that the encoding method powershell_base64 would yield payload results with lower detection rates. However, the lowest detection rate amassed with a substantial amount of payloads were 338 payloads which had less than 36%. Considering the total amount of payloads sent with this encoding method was 354, this equates to 95.4% of the total payloads had a detection rate at 36% and lower. Even though the remaining payloads did not attain a signi cantly higher detection rate, it is still meaningful considering the vast amount of payloads.
- The graph which depicts the results from payloads which lacked any encoding whatsoever, seen in gure 8 is also important to review to determine the effectiveness of AV software's capabilities. There were a total of 354 payloads without any encoding, where 337 of these payloads attained a detection rate lower than 36%. Hence, 95.1% of the total amount of payloads which lacked any encoding had a moderately low detection rate across the board. This is absolutely alarming and should be a signal to AV vendors to review, test and update their own products to increase their respective ef ciency. Such a low detection rate for the majority of payloads is one of the key ndings from this research, even though a minority of payloads reside amongst detection rates >70% it is not the focal point of the gure 8. One would hope that payloads without encoding would have its' majority of payload results attaining detection rates well above 70%, not the minority which the actual results.

Encoding Method	Lowest Substantial Detection Rate	Amount of payloads
Zutto_dekiru	<62%	1
x86_xor_dynamic	<10%	25
x64_xor_dyanmic	<63%	2
x86_shikata_ga_nai	<20%	25
x86_non_alpha	<13%	24
Powershell_base64	<36%	338
Generic_none	<36%	337

Table 1: Each encoding method's payload's attained detection rate & the amount

In light of these aspects mentioned above, there are three main outliers which resulted in the lowest detection rates comprehensively. These encapsulation methods are, x86_shikata_ga_nai, powershell_base64, x86_xor_dynamic and x86_non_alpha, which all had compelling amounts of payloads in total as well as a signi cant amount of payloads which yielded low detection rates for their respective payloads. The worse performer amongst these four encoders were the powershell_base64 which had its' majority of payloads as well as detection rates amongst 31-36% and very few payloads below that percentage. The other three, x86_xor_dynamic, x86_shikata_ga_nai and x86_non_alpha had 24%, 26% and 28% of their payloads yielding detection rates below 30%. This can also be seen in gure 10, although powershell_base64 is not included as the gure is limited to the encodings which attained a detection rate below 20%. There is however a difference between powershell_base64 against x86_xor_dynamic, x86_shikata_ga_nai and x86_non_alpha, as powershell_base64 encoding method uses different payload types. This factor might be one of the causes as to why there is a difference, but there may also be a lot of other different factors in play. This was mentioned previously in the limitations chapter, as some payloads are not compatible with certain encoding methods.

As shown in Figure 10 the majority of payloads that achieve a lower than 20% detection rate were encoded payloads. The payloads with the lowest detection rate of 0% were non encoded ones, but the majority, 87%, were encoded. This leads to the conclusion that encoded payloads do achieve a lower detection rate. As also evidenced by Figure 10, the encoders with the lowest detection rate x86_shikata_ga_nai, x86_xor_dynamic and x86_nonalpha The observation that the majority of the lowest detected payloads being encoded answers RQ2.

For the payload type meterpreter, modern AV provided varying results. Meterpreter payloads can be seen in the gures 2, 4, 5, 6 and 8 which displays a large range of different detection rates for the payload type. As it simultaneously results among the lowest detections rates for multiple encoders, it also results in having amongst the highest detection rates. Notable, meterpreter payloads encapsulated with x86_shikata_ga_nai, x86_xor_dynamic and x86_non_alpha does fail to detect a vast amount of these payload types, where meterpreter has amongst the lowest detection rates for those encoding methods.

6 Discussion

With these results provided previously, as well as the analysis being completed, we believe that the stated research questions has been answered dutifully. The efficiency of the antivirus softwares are analyzed and presented with graphs and comparisons between different results to be able to determine the most proper results. Thus answering the first research question, RQ1, to find the effectiveness of modern AV softwares and if they are sufficient in prohibiting malicious payloads from executing. Even though the results varied, we believe that the analysis of the presented results yielded answers to this matter.

By viewing the graphs which showcases the different encoding methods used, we were able to single out a few encoding methods which achieved the lowest detection rates. These were seen in figure 10 which displays payload results attaining less than a 20% detection rate. Compared to payloads which lack any sort of encoding, this figure proves that there are some encapsulation methods which does lower detection rate. Thus the second research question, RQ2, was answered.

Encoding has been a major part of the conducted research and as such the last research question, RQ3, was also answered. By studying the graphs of different encoding methods and their respective payload results and detection rate, we were able to determine the encoding methods which had the lowest detection rate amongst the different encodings. There are however factors which might affect the results to make it less reliable, as mentioned previously. An example of such a factor would be the fact that different payload types are not supported by all encoding methods, resulting in certain encoding methods utilizing different sets of payloads. Explained in the limitations chapter, this is due to certain payloads not being able to be executed if they are forcefully encoded with an incompatible encoding method. This is something which future research could expand upon, as well as focus solely on certain systems instead of a broader research such as this one.

Our findings found that two out of 981 payloads, which lacked any encoding whatsoever, had achieved a 0% detection rate. This is also alarming, even though the percentage is low, one would hope that no malicious software would be allowed to execute on virtually all systems without being detected. Compared to that of [8] the detection rates has significantly been improved upon. Even though some of the selected payloads by [8] are not a 100% match against ours, almost no results from their work achieved a highest detection rate than 37.5%. Their selection of payloads were also significantly smaller than ours, but it is nonetheless possible to see a direct improvement amongst AV software in their detecting capabilities since 2008. The study performed in 2019 however does bring in somewhat similar detection rates [9] but as mentioned previously, the amount of payloads used are momentously smaller than ours, making this research more comprehensive. Their study solely focused on windows platforms with the x86 architecture, which we also expanded upon for this paper as we include Linux systems as well as x64 systems.

Is it possible to deem the current state of AV software to be sufficient enough to combat the modern payloads of Metasploit? We do not believe the general vendors of AV software are sufficient enough in their performance as of yet. But as stated in the introduction, this is the arms race between malware and AV software. As it is not possible for the results to single out ace performers amongst AV software, the general statistics

depict that currently not enough of the AV armada reach satisfactory levels of detection. To remedy this, AV vendors in the industry need to continue to invest in their products, enhance their software's capabilities to detect malware in code which is not encoded to be able to determine it's characteristics as malicious or not. Being able to determine the behaviour of code is something which is still being developed and improved upon, and perhaps with the recent advancements in A.I as well as ML (Machine learning), plenty of AV software may get necessary updates. The ability to detect code which is encoded with some sort of method had decently high detection rates across the board, which is an acceptable result as the code is indeed trying even further to remain undetected. However, as the numerous amounts of payloads which lacked encoding still remained undetected, this aspect desperately needs improvements for the sake of cyber security.

This is partially one of the most frustrating aspects of our research, namely that we were not allowed to demonstrate which AV software vendors are on top of their products, and which vendors needs improvements. VirusTotal is its' own company however and they have deemed that a study like this could damage their business which is also why they have stated clear guidelines in their Terms of Service [13]. Nonetheless we believed such data would bene t the industry, it could showcase quite clearly which aspects of each AV vendor's software was failing and which aspects of their software were ourishing. Such data would also provide useful information for private individuals who may not be very insightful in the eld of Computer Science or cyber security. Surely, the theoretical data could have also done damages to some vendors reputation in case their AV software performed worse than others, as well as elevate other AV software who had good results.

7 Conclusions & Future Work

When referencing the graph shown in fig10 compared to fig8 it is quite clear to note that to gain a detection rate below 20%, an encoder is definitely required. However, as can be seen in fig8 a lot of payloads without any encoding whatsoever still has a decently low detection rate. This, as previously mentioned, is alarming as to why so many payloads generate such low detection rates. The reasons for this results are unknown, and only provides the state of AV's capabilities when handling pure, malicious, payloads, and those capabilities from the author's point of view is failing.

Another noteworthy conclusions to be drawn seems to stem from the encoders themselves. When comparing fig1 with fig8 the detection rates have plummeted drastically. It seems, to the author's surprise, that many of the payloads sent with different encoders are deemed malicious by AV software based on the single fact that they are encoded, not based on the content of the payload itself, as shown in fig10. This could be a potential problem as it could generate false positives. AV software can alarm systems, managers and such of intrusion of malware when in fact it could have been benign files who are simply encoded with untrustworthy methods. This is an area which could truly benefit from future research, more vividly research how payloads without any encoding differ from payloads, both malicious and benign, attain detection rates.

As VirusTotal keeps track of all uploaded files, one could keep track of how the detection rates for all of the payloads created for this theses change over time.

This research mostly focused on detection rates of payloads and how they are affected by encoders. But VirusTotal provides a lot more information on the payloads then presented. This includes but is not limited to the signatures given to detect malware, sandbox information, metadata information. This information could be leveraged to check how encoders might affect shell integrity and operationality. This information could also be correlated to the payload creation arguments and to the different signatures reported by AV solutions to check a sorts of signature-to-matter-of-fact-threat fidelity, or in other words, are AV solutions detecting the payload or only the malicious characteristics. Another interesting field of study could be the uniqueness of the MSFVenom tools. Most if not all of the payloads created and uploaded to VirusTotal have never been seen before, as in the hash is unknown to VirusTotal. This might point to the conclusion that MSFVenom might take into account other runtime varibles when creating payloads. How could this affect detection rate?

Another aspect that this thesis didn't delve into is the how detections rates differ between platforms. Or in other words, are certain operating systems more vulnerable than others? We believe that our data set could be used to draw some conclusions about that question or be used to further this research in that area alas we did not have the time to analyze the data set further.

References

[1] "Internet crime complaint center(ic3)," Mar 2021. [Online]. Available: https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf

[2] "Aarp® official site - join amp; explore the benefits," Mar 2022. [Online]. Available: https://www.aarp.org/content/dam/aarp/money/scams_fraud/2022/03/2021-ic3-annual-report.pdf

[3] "Internet crime report 2022," Mar 2023. [Online]. Available: https://www.ic3.gov/Media/PDF/AnnualReport/2022_IC3Report.pdf

[4] G. Post and A. Kagan, "The use and effectiveness of anti-virus software," *Computers Security*, vol. 17, no. 7, pp. 589–599, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404899800595

[5] C. Nachenberg, "Computer virus-antivirus coevolution," *Commun. ACM*, vol. 40, no. 1, p. 46–51, jan 1997. [Online]. Available: https://doi.org/10.1145/242857.242869

[6] "Penetration testing software, pen testing security." [Online]. Available: https://www.metasploit.com/

[7] F. Holik, J. Horalek, O. Marik, S. Neradova, and S. Zitta, "Effective penetration testing with metasploit framework and methodologies," in *2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*, 2014, pp. 237–242.

[8] M. Baggett, "Effectiveness of antivirus in detecting metasploit payloads," *SANS Institute*, vol. 61, 2008.

[9] P. Casey, M. Topor, E. Hennessy, S. Alrabaee, M. Aloqaily, and A. Boukerche, "Applied comparative evaluation of the metasploit evasion module," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, 2019, pp. 1–6.

[10] "Msfvenom." [Online]. Available: https://www.offsec.com/metasploit-unleashed/msfvenom/

[11] S. Rahalkar and N. Jaswal, "Chapter 3: Metasploit components and environment configuration anatomy and structure of metasploit metasploit components," 2019.

[12] J. Cloonan, "Advanced malware detection - signatures vs. behavior analysis," Apr 2017. [Online]. Available: https://www.infosecurity-magazine.com/opinions/malware-detection-signatures/

[13] "Virus total - terms of service." [Online]. Available: https://support.virustotal.com/hc/en-us/articles/115002145529-Terms-of-Service

[14] "Virus total intelligence." [Online]. Available: https://www.virustotal.com/gui/intelligence-overview

[15] M. Al-Asli and T. A. Ghaleb, "Review of signature-based techniques in antivirus products," in *2019 International Conference on Computer and Information Sciences (ICCIS)*, 2019, pp. 1–6.

[16] S. S. Hatem, M. M. El-Khouly *et al.*, "Malware detection in cloud computing," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 4, 2014.

[17] K. van Liebergen, J. Caballero, P. Kotzias, and C. Gates, "A deep dive into virustotal: Characterizing and clustering a massive file feed," 2022.

[18] VirusTotal, "Virustotal api v3 overview." [Online]. Available: https://developers.virustotal.com/reference/overview

[19] S. Wheeler and A. Buck, "What is powershell? - powershell." [Online]. Available: https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.3

[20] S. Wheeler, "Cmdlet overview." [Online]. Available: https://learn.microsoft.com/en-us/powershell/scripting/developer/cmdlet/cmdlet-overview?view=powershell-7.3

[21] Mar 2023. [Online]. Available: https://www.sans.org/blog/offensive-powershell-metasploit-meterpreter/

[22] B. Watters, "Introducing pingback payloads: Rapid7 blog," Aug 2019. [Online]. Available: https://www.rapid7.com/blog/post/2019/08/01/introducing-pingback-payloads/

[23] Packtsecurity, "Reverse shells from payloadsnbsp; - packt - secpro," Apr 2022. [Online]. Available: https://security.packt.com/reverse-shells-from-payloads/

[24] C. Atwell, T. Blasi, and T. Hayajneh, "Reverse tcp and social engineering attacks in the era of big data," in *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, 2016, pp. 90–95.

[25] M. Miller, "Command and control: Bind vs reverse payloads " triaxiom security," Mar 2021. [Online]. Available: https://www.triaxiomsecurity.com/command-and-control-bind-vs-reverse-payloads/

[26] K. Kaushik, S. Aggarwal, S. Mudgal, S. Saravgi, and V. Mathur, "A novel approach to generate a reverse shell: Exploitation and prevention," *International Journal of Intelligent Communication, Computing and Networks (IJICCN), Open Access Journal*, vol. 2, 2021.

[27] "Port knocking," 2023. [Online]. Available: https://wiki.archlinux.org/title/Port_knocking

[28] "Penetration-testing-software." [Online]. Available: https://docs.metasploit.com/docs/using-metasploit/intermediate/pivoting-in-metasploit.html#smb-named-pipe-pivoting-in-meterpreter

[29] "Microsoft-support." [Online]. Available: https://support.microsoft.com/en-us/topic/what-is-data-execution-prevention-dep-60dabc2b-90db-45fc-9b18-512419135817

[30] C. S. Veerappan, P. L. K. Keong, Z. Tang, and F. Tan, "Taxonomy on malware evasion countermeasures techniques," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, 2018, pp. 558–563.

[31] Y. P. Yeap, "Council post: Why antivirus signatures and next-gen malware detection aren't enough," Sep 2021. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2021/09/03/why-antivirus-signatures-and-next-gen-malware-detection-arent-enough/?sh=677ad4006a9d

[32] S. Wen and W. Dang, "Research on base64 encoding algorithm and php implementation," in *2018 26th International Conference on Geoinformatics*, 2018, pp. 1–5.

[33] F. Huo and G. Gong, "Xor encryption versus phase encryption, an in-depth analysis," *IEEE Transactions on Electromagnetic Compatibility*, vol. 57, no. 4, pp. 903–911, 2015.

[34] InfosecMatter, "Non-alpha encoder - metasploit." [Online]. Available: https://www.infosecmatter.com/metasploit-module-library/?mm=encoder%2Fx86%2Fnonalpha