

# Chilli Awesome Dissertation

**Charles Graham**

12345

**Supervised By**

Dan Goldsmith

12/12/2024

School of Computing, Engineering and Data Science



## **Abstract**

This is the abstract

If you keep it in the indented block shizzle will work I hope that new lines also work correctly.

A new paragraph.

Not sure about it being centered, I will have a think about that, possibly move so it's got a bit more space at the top than the usual stuff.

## **Dedication**

We also like to add a dedicaaion Like the abstract anything in the block wll be kept

Would lke to thank my parents, god, the family cat / dog. Obviously Dan because he is awesome

Again, not sure about the layout, but we can fix that.

# Contents

|          |   |           |
|----------|---|-----------|
| 0.1      | Introduction . . . . .                      | 1         |
| 0.1.1    | Aims and objectives: . . . . .              | 2         |
| 0.2      | Legal and Ethical considerations: . . . . . | 2         |
| 0.3      | Literature review: . . . . .                | 3         |
| 0.3.1    | Computer viruses . . . . .                  | 3         |
| 0.3.2    | Computer virus history . . . . .            | 3         |
| 0.3.3    | Lifecycle of a computer virus . . . . .     | 4         |
| 0.3.4    | Papers . . . . .                            | 7         |
| 0.4      | Key Findings: . . . . .                     | 10        |
| <b>1</b> | <b>Methodology</b>                          | <b>12</b> |
| 1.1      | Environment . . . . .                       | 12        |
| 1.1.1    | Vail 3.1 . . . . .                          | 13        |
| 1.1.2    | Pe.Cloak.py . . . . .                       | 13        |
| 1.1.3    | Msfvenom . . . . .                          | 13        |
|          | <b>References</b>                           | <b>14</b> |

## List of Figures

## List of Tables

## 0.1 Introduction

Within the cyber world, there is a constant battle between viruses and antivirus software. Due to computers evolving at an exponential rate, so are the techniques of virus development and mitigation. In this paper I aim to understand the extent to which the stock antivirus system Windows Defender is able to prevent malicious executables from being run on a stock Windows machine. With this data, I will conclude how effective Windows Defender is when different forms of evasion are used.

This research will focus on evaluating the effectiveness of different virus creation tools for Windows 10 I have chosen to focus on the Windows 10 operating system due to the market share in operating systems, with Windows holding a substantial percentage. “As of December 2020, over 76% of all computers worldwide were running some version of Windows” (Zaharia, 2024). Having such a large share in the market means that an attacker’s priority when trying to infect a machine is mainly aimed at a Windows machine; by evaluating the effectiveness of Windows Defender, we will be able to understand the possible future mitigations to be put in place so that all devices are protected from a multitude of potential viruses.

There are many types of viruses, but the one I will be focussing on is mainly a Remote access Trojan (RAT). The reason I have chosen to focus on a RAT is due to the severity and impact it can cause on a user. Remote access aims to gain full access to the victim’s machine and can open up doors to more attacks. By understanding how these viruses are created and the techniques used to obfuscate code avoiding AV will allow us to put in the necessary protection mechanisms to defeat malicious users.

As discussed in Section 1 above, as Windows holds a majority share in the market when it comes to computers, there are a range of tools that can be used to create effective viruses. The impact of these tools regarding AV detection is continuously altering, and therefore, the environment is rapidly changing. With plenty of innovations from both attackers and defenders, it is not easy to produce long-lasting solutions.

Up-to-date sources indicate that “560,000 new pieces of malware are detected every day” (Jovanovic, 2023). With this vast number of potentially 0-day viruses being created every day, there is no way that antivirus software can keep up and block all these threats before they infect multiple machines.

(Jovanovic, 2023) Also says that “Every minute, four companies fall victim to ransomware attacks” This relates nicely to (Samarati, 2017) report on how much cybercrime cost the UK in 2016, “Although ransomware ranked last in terms of the number of organisations affected (388,858), it ranked first in terms of financial losses (£7,356,060,699)”. These statistics show us that more resources need to be dedicated to researching the techniques that attackers use to produce malware. After understanding why Antivirus is evaded, we will then be able to mitigate the impact of readily available tools and protect our computers. The reason so many organisations were affected is due to evasion techniques such as obfuscation and cryptography. Encoding and changing the structure of the payload will bypass specific

detection techniques, such as signature-based scanning and potentially heuristic-based scanning, depending on the severity of the obfuscation. Also, techniques such as polymorphic code will allow code to be changed every time the program is run, further evading Antivirus by constantly changing its signature.

Antivirus software is arguably one of the most critical processes on our computers, blocking internal and external threats and protecting our computers from malicious users. My projects will be important to the blue and red teams within cybersecurity. With more excellent knowledge of how antivirus systems cope with different forms of evasion, will ensure that new protections are put in place to secure devices further. Also, I'm positive that my research will also benefit individual users who have little knowledge of cyber threats. Reasoning for this is that demonstrating the different ways in which a virus can come will encourage users to be more cautious when opening a file from an unsigned source. Furthermore, antivirus developers will be able to use my research to locate the flaws in any existing products. If a particular technique used guarantees antivirus evasion, then developers will be able to rectify any mistakes.

### **0.1.1 Aims and objectives:**

- How well do off-the-shelf tools evade antivirus?
- What are the key characteristics of a virus?
- What behaviours cause Antivirus to flag a file as malicious?
- What techniques, if any, can be used to evade Antivirus?

## **0.2 Legal and Ethical considerations:**

### **0.2.0.1 Legal**

Throughout this project I will be researching and developing computer viruses, and therefore bare any responsibility over the programs I create and how they are used. Because legal considerations around Computer Misuse Act (1990) and Civil Liability Laws, all versions of computer virus that are created will be tested locally. This means that we wont address how viruses spread over the internet, but we will dive into how well AV detects them once on a "victims" computer. During this research no data will be lost or destroyed and no person(s) will be affected by my research. ##### Ethical As my research explores the techniques and mindset that a malware developer may use there are some ethical implications. Firstly, any working code that bypasses AV will be concealed as to not encourage readers to use for malicious reasons. Secondly, I will practice responsible disclosure. In the case that



my research leads to a security hole being uncovered, this will be reported to the relevant parties. Lastly, I will ensure that the intentions of my research are purely educational.

### **0.3 Literature review:**

#### **0.3.1 Computer viruses**

There are many different types of computer virus but the one that I intend to research is the Trojan horse virus. ("Trojan," 2022) Describes this as a type of malicious computer software, usually disguised within a legitimate or beneficial program. Once this software is installed on a users computer, the trojan will allow the attacker remote access to the victims computer. Remote access gives the attacker a very threatening position, with the ability to access all computer files, applications and data, the attacker has the opportunity to steal and sell your confidential data (Bhardwaj, 2022). In this paper by (Robinson, 2017) The capturing of personal data was valued at US\$60 for each internet user at the time (2012). Although outdated this paper sets an example of how personal data can be sold for profit. This breach in personal data can have a snowball affect of implications on the user and even cause problems such as identity theft and financial loss. Moreover, when data is lost from a company due to virus related incidents, this can cause reputation damage and consequently loss of revenue.

#### **0.3.2 Computer virus history**

##### **0.3.2.1 ILOVEYOU Virus**

Computer viruses have been around since early computer development. One of the most significant computer viruses was the "ILOVEYOU" virus. The web article by (Root, 2022) explains the computer virus would be received via email and appear to be a text document described as a "Love letter coming from me". However, most users didnt realise the file received was a VBS script which ran a program to corrupt documents on the hard drive and then replicate itself through the use of your address book, and access to the email account. This virus certainly wasn't the first to abuse this exploit, but it was a pivotal moment for malware development and the damages incurred are estimated at about \$10 billion.

##### **0.3.2.2 Zeus Virus**

<https://www.cynet.com/malware/zeus-malware-variants-methods-and-history/>

Zeus/Zbot was a malware package that used a client/server model (Trojan). The main function of Zeus was to gain unauthorised access to financial systems by stealing credentials, banking information and

financial data. This virus over the years infected over 3 million computers in the US, including companies such as NASA and the Bank of America. Zeus disguised itself as legitimate software and enables the malware when it spots the use of a banking site or financial transaction. Zeus was so successful in infecting machines, there were two common ways this was done. Firstly, Drive-by-downloads. This involved compromising legitimate websites and exploiting known browser and operating system vulnerabilities to download the Zeus malware when a user accessed the site. Secondly, As Zeus grew it gained the ability to infiltrate user accounts on social media and over email. With the compromised accounts Zeus would be able to create phishing messages tricking users into downloading this malware.

### 0.3.3 Lifecycle of a computer virus

The author (Alrawi et al., 2021) looks to investigate the life cycle of a virus. Upon reading the paper, a framework is created where the lifecycle can be broken down into five simple components. 1) The infection vector; This is how the malware attacks a system 2) The payload; This is the dropped malware code after exploitation 3) Persistence; This is how the malware installs on a system 4) Capabilities; The functions in the malware code 5) C&C infrastructure; How the malware communicates with the operator. Using these components we are able to break down how a virus is able to infect, execute, and replicate on machines. This framework allows us to assess how impactful a virus could potentially be. ### Protection mechanisms ##### Signature based detection To stand a chance at bypassing AV we need to understand the complexities around its detection mechanisms. This author breaks down the different detection types (Rohith & Kaur, 2021) Firstly, there is a signature analysis. This involves scanning the file for a specific signature that is found within a known virus, if the signature matches then the file will be reported as suspicious. One major disadvantage of using a detection system like this is that it can only scan for signatures that are known viruses; a new 0-day virus will not be detected by signature based detection. Because of this we have a heuristic detection system. With the help of probabilistic algorithms it will “scan the file structure with the compliance with the virus patterns are checked”. As this technique uses a probability algorithm it will cause false positives from time to time, an example of this may be a file that injects code into another application's memory, such as a mod for a game. As these methods aren't fool proof there are multiple methods that antivirus can use to detect malware. ##### Behavioural analysis Another popular type of detection mechanism is the behaviour based method. Unlike signature based detection, behavioural analysis is dynamic, therefore does not share the same shortcomings as signature based detection. The author (Bazrafshan et al., 2013) breaks down how this detection method works. “Behavioural based detection techniques observe the behaviour of a program to conclude whether it is malicious or not” This means that the AV will break down the core functions of the program and assess whether the functions and how the program executes to determine whether any suspicious activities such as editing registry keys or starting outgoing connections to an unrecognised address. Different strains of malware can all be classified under a single behavioural signature as they may utilise the same type of behaviour. It's important to note that because behavioural

analysis is dynamic it does not have the same restraints against 0-day viruses like signature based systems. This strength allows behavioural detection techniques to quarantine suspicious programs where coding techniques like polymorphic coding are employed. Overall, this enables behavioural analysis to detect newer malware as often viruses share the same properties as one another. One disadvantage to using this detection method is for the possibility of a false positive. As some programs may share behaviours that can be found in viruses there is the possibility that the program is flagged as unwanted. An example of this could be a mods for a game downloaded off the internet. As the mods may interfere with the integrity of the games memory, that might be flagged as suspicious and therefore a false positive. ##### Heuristic Detection In the paper by (Rohith & Kaur, 2021) Heuristics is defined as a set of rules applied to a program to determine if it contains a virus or not. Heuristic based detection is described as utilizing machine learning and data mining techniques to learn a program's intentions. Heuristic based systems are both static and dynamic. With the static analysis comparing the possible behaviour or the decompiled code with a database. The dynamic analysis will run the program in an isolated environment to observe its operations. These two types of analysis allow the AV to assess the risks of the threat without exposing the users device to a potentially harmful program. Like behavioural analysis there is the disadvantage that there could be false positives. As (Rohith & Kaur, 2021) suggests, it is impossible to create an algorithm that can definitively distinguish between a virus and an uninfected program with 100% accuracy. ##### Sandboxing The author (Andryani & Sutabri, 2023) explains that a sandbox provides an isolated environment used to execute suspicious binary interactions. A sandbox is a security component for ongoing projects, usually with the ultimate goal of mitigating flaws. It is often used to execute untested or untrusted projects or code. Within Antivirus sandboxing is employed as a virtual machine where potentially harmful files are executed to monitor their actions. (Rohith & Kaur, 2021) Sandboxes allow the programs to execute freely without risk of damaging or impeding the security of the users device.

Overall these protection mechanisms all have their strengths and weaknesses. For an antivirus system to be reliable and effective, all 4 of these techniques should be employed to provide the most accurate and effective detection. In modern day antivirus systems these techniques will be highly developed and accurate to ensure the best possible security, however that doesn't mean that they're un-evadable. ### Bypass methods As I mentioned in section 1, computer malware infects thousands of companies per day (Jovanovic, 2023) "Every minute, four companies fall victim to ransomware attacks". With more research being poured into discovering the techniques that black hat hackers may use to create a backdoor into a machine, the better chance we have at developing prevention techniques that quarantine these unwanted programs. If we could stop the effectiveness of computer viruses then this would enable a higher level of confidentiality and data integrity, saving companies compensation claims and loss of reputation.

As virus detection techniques have evolved over the years so has the need to create tools that bypass these techniques. The author (Garba et al., 2019) says "Malwares generated using tools such as

Metasploit are easily detected nowadays due to the techniques used by antiviruses” As a direct reaction tools have had to evolve to keep up with antivirus detection. In this section I will discuss some popular techniques that malicious users may use to bypass AV. ##### Encryption The author (Rad et al., 2012) defines encryption as the earliest and simplest method employed by the malware programmers to achieve their goal. The first known encrypted virus appeared in 1987 and was named, Cascade. Encryption can be achieved in many ways, a simple way a payload can be encrypted is through the reversible XOR function. Since this paper more tools have developed such as msfvenom which will take a payload and encrypt using a specified algorithm. ##### Obfuscation “General code obfuscation techniques aim to confuse the understanding of the way in which a program functions. These can range from simple layout transformations to complicated changes in control and data flow” (Balakrishnan & Schulze, 2005). When employed into bypassing antivirus, Obfuscation can be used to hide the intentions of an executable. As obfuscation transforms or changes the code, this in turn changes the signature of a file, making antivirus software unable to detect the executable through signature based detection. Like encryption, due to more detection mechanisms, obfuscation would have to be used in compliment to another evasion technique to have a chance of being successful. Although bypassing one detection is a step in the right direction, it is not enough to evade AV on its own. ##### Morphic coding Morphic coding techniques such as polymorphism and metamorphism aim to change their own code each time they run. With polymorphism adapting its appearance with simple techniques such as encryption, data appending or prepending. (Ali et al., n.d.) states that polymorphism allows objects or methods to operate in multiple forms and adapt to different conditions within a programs code. One problem identified by (Konstantinou & Wolthusen, 2008) is that polymorphic viruses eventually have to decrypt their constant body in memory in order to function, advanced detection techniques can wait for the virus to decrypt itself and then detect it reliably. On the other hand, Metamorphic viruses have the ability to transform their code after each run. Through the use of obfuscation techniques metamorphic viruses spawn hundreds of iterations of a virus with the same core functions. Some more advanced metamorphic viruses may also have the ability to alter their execution depending on whether they are in a sandboxed environment. This is later confirmed in (Andryani & Sutabri, 2023) paper, where they inform us that Malware dodge sandbox is another type of malware that can tell if it is in a sandbox or virtual machine. ##### Off-The-Shelf Tools There are tools out there such as Veil, Avet, TheFatRat and PeCloak.py that aim to quickly create and undetectable executable file that can infect a victims device. Although these tools are quick and easy to use, the results generated from them are usually sub-par. This is due AV developers being able to watch the development of the tools and patch any exploits that may have been found by the tool developers. Although as some of these tools allow you to highly customise your payload, this may open a window of opportunity in that AV may take a while to detect the malicious file.

### 0.3.4 Papers

#### 0.3.4.1 Paper 1

The author (Kaushik et al., 2022) of 'A systematic approach for evading antiviruses using malware obfuscation'. The paper is from 2022 and describes the process they go through to obfuscate a payload with Graffiti and Veil-Evasion. The purpose of this research is to identify methods that can be used to bypass the antivirus, while mainly focussing on tools such as Veil-Evasion and Graffiti. Similarly to my proposed project the author starts trying to evade antivirus by using readily available tools such as Graffiti and Veil-Evasion. According to the results, Graffiti failed most of the time to avoid the antivirus, and was not effective. However, through the use of Veil-Evasion and a .bat to .exe converter they were able to produce a payload that gave remote root access to a windows machine (bypassing Windows 10 Defender). This is a significant result as it demonstrates that even off-the-shelf tools can be used to bypass Windows 10 Defender. Reviewing this paper gives an insight to the techniques I will need to follow to generate the same results and the methodology I should follow.

The paper also goes into a good depth on how the different tools behave and the outputs they produce. Such as Veil can output the obfuscated payload as a .py, .bat or .exe file. This is important to understand as these tools may be relied upon to create payloads for my future experiments. In conjunction with using Veil and Graffiti the author also uses "If-else Deletion" to understand which portion of the code is required for the payload to run as intended. They also utilise "Dead code insertion" This allows the payload to be filled with code that won't affect the execution of the payload but also differs the signature from the genuine payload.

One limitation on this paper is the sample size. The author concludes that "We have shown effectively that we can easily make a malware undetectable utilizing code obfuscation strategies". Although these results have proved that creating an obfuscated payload can evade antivirus, I do not believe that there is enough evidence to back up this argument, and that the scope is too limited to creating a singular working payload. The paper shows the results of only one payload where Veil-Evasion bypassed 13 out of 59 Antivirus systems. Due to viruses constantly evolving, Antivirus systems have had to keep up in a never ending battle. There is no evidence to suggest that this method for creating a virus is replicable on a large scale. Replicability is the most important factor when it comes to creating a virus, the ability to create a virus that can infect hundreds to thousands of machines without waking up the antivirus present is the goal of virus developers. As discussed earlier, antivirus systems work using databases and probabilistic algorithms to determine whether a program is suspicious. My aims for this study is to try demonstrate the ways that viruses can be produced, with techniques that evade Windows Defender.

One strength of this paper is that the conclusion provides an optimal approach on how the research could've been finer, "The best system to dodge protector is to make your own obfuscate tools whether

that be with a custom obfuscator or transforming them physically by hand.” What this highlights to us is that for virus creation to be effective in bypassing antivirus, tools need to be created where we can obfuscate a payload using polymorphic or metamorphic systems. This would entail creating a payload that can transform its code to ensure that signature based systems won't detect our payloads. The author (Ali et al., n.d.) also exhibits the uses of polymorphic code, and displays some compelling strengths as to how it is useful for programmers when coding in an OOP style approach. The benefits of coding in polymorphic ways will allow the developer to develop code quickly, reuse of code and more flexibility in object-oriented programming paradigms. Under normal circumstances within virus development, reuse of code is generally a poor approach. However, the circumstance changes when techniques like polymorphic and metamorphic systems are employed. This will allow the developer to create a basic virus template to which the program will follow, after each run the programs appearance or structure may change slightly but the core processes and functions will stay the same, ensuring a working payload.

In conclusion this paper has a strong connection with my proposal. The paper has some crucial methodology that is relevant to the approach that I intend to take. The overall results demonstrate that evasion of modern day antivirus systems is possible through the use of readily available tools. The paper is also very clear on the steps that need to be taken to reach the same results, and through this I will do my best to firstly replicate the results and then build upon them.

#### **0.3.4.2 Paper 2**

In contrast to the results of this work (Aminu et al., 2020) found concluded that the best evasion tools were Avet and PeCloak.py. “Bypassed most of the selected antivirus by 83% and 67% respectively”. These findings are super interesting to us as a later study by (Kaushik et al., 2022) was able to bypass Windows 10 defender with Veil. This change in results could be caused by the development in the tools used. (Aminu et al., 2020) aimed to evaluate the effectiveness of selected antivirus systems through the use of evasion tools such as, Veil 3.0, PeCloak.py, Shellter, and a Fat Rat. These tests were conducted against a windows platform. Although these results are older and occurred while testing different antivirus systems to (Kaushik et al., 2022) the results are still relevant and need to be understood. Since, Windows Defender and the tools used in this experiment have arguably advanced. This also relates back to section 1, where I emphasise the need for more research to be done over antivirus systems and their counterpart. With technology evolving at exponential rates, so are the tools and techniques that create unwanted programs and leave backdoors into our machines.

Interestingly, the author talks about how Avet was able to evade so many of the antivirus systems. It seems the main reason for this is the ability for Avet to avoid sandboxing and emulation. “Avet includes two tools, avet.exe and AV evasion technique to avoid sandboxing and emulation”. This result opens up many opportunities to explore, if the technique still functions this would allow more simplistic but

effective payloads and Windows Defender would have to rely on more dated detection techniques such as signature based detections.

Some strengths of this paper are that the author used a virtual environment VM to stage both the attacker and the victim machines and connected them together through a NAT setup. This allowed the author to freely create payloads and infect the Windows 10 machines without the consequence of infecting a real machine. The methodology and recording of results were also really clear in this experiment. The results clearly demonstrate the effectiveness of both the antivirus and the evasion tool used. With this data strong conclusions could be made as to what AV systems provide the most protection and what tools rain supreme in evading AV.

In conjunction with (Kalogranis, 2018) these authors also conclude that results could be more substantial if the use of hand crafted payloads were employed. This would stop the signature based detection system from flagging files that have come from readily available tools and slow down the speed to which antivirus flags the file as suspicious. Although this is a farfetched idea to hand create every virus, it also demonstrates how the coding techniques such as polymorphic and metamorphic systems could be employed to create more devastating strains of computer virus.

#### **0.3.4.3 Paper 3**

<https://ieeexplore.ieee.org/abstract/document/9519480>

In the paper by.... discusses a sophisticated evasion technique called Living-Off-The-Land (LotL). This evasion technique is one of the major evasion techniques used in many attacks. The premise behind this attack is to leverage binaries that are already present in the system to conduct the attack. This is an interesting technique as it potentially bypasses the need for the attacker to inject a suspicious program onto the system. The conclusions of this paper highlight that almost every popular AV product tested had difficulties detecting malicious usage of LotL binaries. Later development included working with AV vendors and working to implement detection methods for this style of attack.

One reason why this paper is so significant is due to LotL creating unforeseeable challenges for the AV, as the technique uses present binaries on the windows system this may have the effect of tricking AV into trusting the process. By favouring LotL, malicious processes are hidden in plain sight and function behind the legitimacy of a Windows binary. This is significant as the use of LotL will make static analysis methods such as signature based detection obsolete. Relying purely on dynamic based detection will leave space to where viruses evade the unreliable detection algorithms. Although LotL based viruses can be seen as difficult to detect they are also very specific to the machine. For example, most systems are configured in slightly different from person to person. The issue is being able to create a reliable, comprehensive binary list that encompass all the possible configurations is unforeseeable. With this issue, the developer would have to handpick the binaries for a chance of infecting a system.

The authors first analyse several databases of up to roughly 31,000,000 different malware samples. They find that they average about 9.41% prevalence in LotL techniques. Also, when conducting the experiment they use some of the top market leading antivirus products, including Windows defender. The experiment also makes use of Windows 10 machines that have been up-to-date to the latest patches. All these variables give an advantage to the antivirus software's. However this is not reflected in the initial results, almost every popular AV product that was tested had difficulties detecting malicious usage of LotL binaries. The authors went one step further and worked with AV developers disclosing information on the attacks and the severity of LotL, despite this only a fraction of AV vendors implemented a successful detection mechanism. Also some vendors implemented detections for the specific payload that was disclosed but not for detecting the technique employed. One reason for vendors unsuccessfully implementing a countermeasure is due to the risk of generating false positives.

Like all research on virus/antivirus developers, both sides benefit from the findings. Like the authors concluded, the findings of the paper helped AV developers in creating new detection mechanism for LotL. On the other hand, weaknesses have been exposed in current versions of windows and the antivirus companies that don't have any protections in place. For There will be need for ongoing research on this topic so that AV can keep up with the evolving LotL techniques. Since this research (INSERT [https://www.sciencedirect.com/science/article/pii/S016740482300562X?casa\\_token=2WuA2-8cOwUAAAAA:g4QnD\\_-cp6RHFAjZfQLUGEkL61zEisyKpNZO6ho6RpFE84zM\\_rLs59wM5nIXUQ-bkJi5\\_8Hg1JY](https://www.sciencedirect.com/science/article/pii/S016740482300562X?casa_token=2WuA2-8cOwUAAAAA:g4QnD_-cp6RHFAjZfQLUGEkL61zEisyKpNZO6ho6RpFE84zM_rLs59wM5nIXUQ-bkJi5_8Hg1JY)) analysed the evolution of fileless attacks. Results found that the prevalence of fileless attacks has been increasing year-by-year, and with this so has LotL. LotL is not solely a windows based problem, and are a significant concern on Linux too. They are challenging to attack due to the stealthy ways they abuse binaries and require special attention.

## 0.4 Key Findings:

After reading and discussing these papers I want to emphasize some key findings and patterns. Firstly, across the reviewed papers, a consistent pattern emerges; Malware developers employ evasion techniques such as obfuscation, encryption to hide their payload from traditional detection methods such as signature based systems. Also, as my papers span over a a few years there are contradictions in the findings of different authors. The contradictions exemplify the battle between virus and AV developers. Some techniques or tools that may have worked 6 months ago may now be obsolete, whereas tools that may previously of not worked could be the key to creating a working payload. Lastly, we have discussed the strengths and limitations of each detection mechanisms such as, signature, behavioural and heuristic. Lastly the tools and current evasion techniques have been discussed.

Some negative patterns emerged too, authors understood the importance of handcrafted payloads and the implications it could have for AV detection, however this assumption was not reflected in their



papers. Most experimentation consisted of payload generation through the usage of tools. Although this is a very convenient for experimentation, it wont allow for the results to be fully reflective of real life scenarios where it is missing a key aspect in ways that payloads are crafted. Through my experiment I will attempt to fill the gaps in research by developing my own payload through the use of python. Understanding this research enables me to start answering my research questions. Firstly, key characteristics of a virus was defined by (Alrawi et al., 2021) as having 5 simple components (The infection vector, the payload, persistence, capabilities, C&C infrastructure) Lastly, (Bazrafshan et al., 2013) defines how behavioural based mechanism use probabilistic algorithms to determine whether a programs behaviour is malicious.

# 1 Methodology

I will adopting an experimental methodology to answer, to what extent can Windows Defender detect malicious code where evasion techniques are used. To answer this question fully I have broken broken down the research questions into sub questions - 1) How well do off-the-shelf tools evade antivirus? - 2) What evasion techniques, if any can successfully evade antivirus? - 3) Is it feasible to handcraft payloads, as opposed to generation through tools.

## 1.1 Environment

In order to test the hypothesis, firstly an environment will need to be setup. The experimentation will be carried out within a labatory and using virtual machines hosted on (VMware Workstation 16) with the target running Windows 10 to the latest patch, and the attacker running Kali Linux. The two machines will be connected through the same private NAT network device. Reasoning for this is to enable the two machines to freely communicate with one another using local IP addresses, ensuring that no other devices accidentally get infected with any generated payloads and that there are no external variables unaccounted for.

![[NetworkConfig.png]]

Kali Linux comes with a wide range of pre-installed pen-testing applications and tools, due to the freedom that kali Linux enables, this will also allow creation of malicious payloads without the risk of the operating system deleting any crucial work. As discussed above, the purpose of this research is focussing on the effectiveness of windows defender to when a malicious file is already present on the machine. Furthermore, the target will be infected by through the aid of pythons http server command. `python -m http.server`. Through the usage of this command I will be able to setup a temporary local webserver from the attackers current directory, that enables the target machine to download any payloads that are created. This will be further highlighted in the figure below ![[WebserverUpload 1.png]]

Using this configuration the attacker machine will be able to upload new iterations of any viruses created, in a secure fashion. This also replicates how a virus may be downloaded onto a machine in real world scenarios. ## Tools In this section i will discuss the tools and methods that will be employed

to evaluate how well off-the-shelf tools evade antivirus. I have chosen a selection of well known and frequently updated tools for this section, this will hopefully give the best chance at evading antivirus.

### **1.1.1 Vail 3.1**

The reason Veil has been selected is due to the contradictory results that were generated over the span of a few years in the section above. Veil-Evasion is a famous framework written in python. We can utilize this framework to produce payloads that can sidestep most of AVs (Kaushik et al., 2022).

### **1.1.2 Pe.Cloak.py**

### **1.1.3 Msfvenom**

## References

- Ali, H. M., Hamza, M. Y., & Rashid, T. A. (n.d.). *Exploring polymorphism: Flexibility and code reusability in object-oriented programming*.
- Alrawi, O., Lever, C., Valakuzhy, K., Snow, K., Monroe, F., Antonakakis, M., et al. (2021). The circle of life: A {large-scale} study of the {IoT} malware lifecycle. *30th USENIX Security Symposium (USENIX Security 21)*, 3505–3522.
- Aminu, S. A., Sufyanu, Z., Sani, T., & Idris, A. (2020). Evaluating the effectiveness of antivirus evasion tools against windows platform. *Fudma Journal of Sciences*, 4(1), 112–119.
- Andryani, A., & Sutabri, T. (2023). DETECT AND MITIGATE MALWARE THREATS USING SANDBOXING TECHNOLOGY. *Jurnal Scientia*, 12(03), 3263–3269.
- Balakrishnan, A., & Schulze, C. (2005). Code obfuscation literature survey. *CS701 Construction of Compilers*, 19, 31.
- Bazrafshan, Z., Hashemi, H., Fard, S. M. H., & Hamzeh, A. (2013). A survey on heuristic malware detection techniques. *The 5th Conference on Information and Knowledge Technology*, 113–120.
- Bhardwaj, P. (2022). What are the risks of remote computer access? In *MUO*. <https://www.makeuseof.com/remote-access-risks/#:~:text=Risks%20Associated%20With%20Remote%20Computer%20Access%201%20Security,associated%20with%20remote%20access%20is%20performance%20issues.%20>
- Garba, F. A., Kunya, K. I., Ibrahim, S. A., Isa, A. B., Muhammad, K. M., & Wali, N. N. (2019). Evaluating the state of the art antivirus evasion tools on windows and android platform. *2019 2nd International Conference of the IEEE Nigeria Computer Chapter (NigeriaComputConf)*, 1–4.
- Jovanovic, B. (2023). *Malware statistics in 2023*. dataprot.
- Kalogranis, C. (2018). *AntiVirus software evasion: An evaluation of the AV evasion tools* [Master's thesis]. Πανεπιστήμιο Πειραιώς.
- Kaushik, K., Sandhu, H. S., Gupta, N. K., Sharma, N., & Tanwar, R. (2022). A systematic approach for evading antiviruses using malware obfuscation. In *Emergent converging technologies and biomedical systems: Select proceedings of ETBS 2021* (pp. 29–37). Springer.
- Konstantinou, E., & Wolthusen, S. (2008). Metamorphic virus: Analysis and detection. *Royal Holloway University of London*, 15, 15.
- Rad, B. B., Masrom, M., & Ibrahim, S. (2012). Camouflage in malware: From encryption to metamorphism. *International Journal of Computer Science and Network Security*, 12(8), 74–83.

- Robinson, S. C. (2017). What's your anonymity worth? Establishing a marketplace for the valuation and control of individuals' anonymity and personal data. *Digital Policy, Regulation and Governance*, 19(5), 353–366.
- Rohith, C., & Kaur, G. (2021). # antivirus evasion methods in modern operating systems. 2021 2nd *International Conference on Intelligent Engineering and Management (ICIEM)*, 429–434.
- Root, E. (2022). Iloveyou: The virus that loved everyone. In *Daily English Global blogkasperskycom*. <https://www.kaspersky.com/blog/cybersecurity-history-iloveyou/45001/>
- Samarati, M. (2017). *Cyber crime cost UK businesses £29 billion in 2016*. It governance.
- Trojan. (2022). In *Encyclopædia Britannica*. Encyclopædia Britannica, inc. <https://www.britannica.com/technology/trojan-computing>
- Zaharia, A. (2024). *300+ terrifying cybercrime & cybersecurity statistics*. comparitech.