

Proposal for research into a Remote Access Trojan.

Matthew Cragg
Cyber Security and Networking (of MSc Year 1)
Research Methods and Professional Development
University of Wales Trinity St.Davids
Swansea, Wales
Student ID 2207432

CONTENTS

I	Introduction	1
I-A	Objective	2
I-B	Potential Issues	2
I-C	Methodology	2
II	Literature Review	3
II-A	Encryption	3
II-B	Networking	3
II-C	RAT Detection	4
II-D	Discussion	4
III	Research	4
III-A	Programming Language	4
III-A1	Python	4
III-A2	C	4
III-A3	Java	5
III-A4	Visual Basic	5
III-B	Programming Language - Conclusion .	5
III-C	Survey on Ethicality	5
IV	Further Expansion	6
IV-A	Encryption	6
IV-B	Cloud Computing	6
IV-C	Delivery	6
V	Results	6
VI	Terminology	6
VII	Reference List	7

LIST OF FIGURES

1	Design diagram of a Crypter, from ‘Cloud Crypter for bypassing Antivirus [1]’.	3
2	‘Example of network traffic monitoring during the early stage of a RAT virus attack [2]’.	4

Abstract—The proliferation of affordable technology and computer systems within the average persons life has brought both concern and risk, alongside the usefulness of technology and the assistance it may provide. This paper aims to explore the various exploits that attackers have developed throughout the computers lifetime, specifically a Remote Access Trojan. This exploration will be expanded into research how such a trojan may be hidden from anti-viruses, and methods that attackers exploit to hide their program from view.

I. INTRODUCTION

The term ‘Computer Virus’ was first defined by Fred Cohen in his paper ‘Fred Cohen: Theory and Experiments’. In his paper, he states ‘We define a computer ‘virus’ as a program that can ‘infect’ other programs by modifying them to include a possibly evolved copy of itself. With the infection property, a virus can spread throughout a computer system or network using the authorizations of every user using it to infect their programs. Every program that gets infected may also act as a virus and thus the infection grows [3]’.

By this definition alone, a virus is already a severe security risk to any and all computer systems that are vulnerable to being infected.

There are multiple types and definitions of a computer virus, with each one hosting a number of subsections themselves. The paper will focus on one type of virus that is often associated with ‘Spyware’. This virus is called a Remote Access Trojan, or a RAT. This virus, when installed, will infect a computer with a program that can record any keystrokes, mouse movements or inputs made by the victim. This information is then sent to the attacker for them to look for anything useful. The 2020 paper ‘Growth and Commoditization of Remote Access Trojans’ claims ‘That the amount of RATs increased drastically in the last ten years and that nowadays they have become standardized commodity products that are no very different from each other [4]’. This growth only increases interest and risk on both sides of the fight. More people with malicious intent will learn of the RAT virus and begin to research and develop extra functionality and methods of avoiding detection, and the inverse side will work heavily on developing their anti-viruses to find any and all possible ways of detecting and preventing viruses such as the RAT. The same paper found that the number of

This type of virus is dangerous because the attacker will receive any log-in information that the victim may input while

the virus is live, such as banking information, application usernames and passwords. As most people have a habit of re-using passwords between different applications and websites, an attacker can make an educated guess on a victim's account, even if they have not revealed their password by studying other account passwords and patterns.

This virus can also be used to bypass two-factor authentication, if the attacker is quick enough to use the code given. An example of this would be Google Authenticator. As the application generates a new code for the user to use every thirty seconds, the attacker may have enough time to input the same code generated by Google Authenticator to sign in at same time.

The information gathered from this virus may be used in a variety of ways. The most common aim of infecting another with a RAT virus is financial gain, by viewing and stealing banking information, or other finance services such as PayPal. Another purpose for the RAT virus could be blackmail, where an attacker would contact the victim with partial information that is private, such as a photo, with the threat of leaking or revealing the information publicly unless money is paid.

This project may become quite complex in finding various methods to hide a RAT. Due to this and the fact that development could continue on indefinitely to try and create it as undetectable as possible, this project would be considered a success if an anti-virus is able to scan the vulnerable machine and not find the virus at least once. The following anti-viruses will be used in this experiment to try and detect the RAT.

- Windows Defender.
- Malwarebytes.
- McAfee.
- Avast.

These four anti-viruses will allow for a broad range of experimentation as each anti-virus has a different database and methods of searching for an anti-virus.

The prevailing method of anti-virus approach is named a Signature-based anti-virus [5]. This type of anti-virus uses already known strings of data or code that exists in viruses that have been detected before. If a scanned file contains code or data that matches a virus previously found, it will flag and quarantine the application.

Another approach to detecting a virus is the heuristic approach. In this approach, an anti-virus uses a similar technique as signature-based anti-viruses but differentiates by scanning files via various rules or algorithms, instead of parts of signatures or data.

The third and final technique that is commonly found in anti-viruses is the sandbox approach. In this design, an anti-virus isolates files and programs and runs them in a virtual environment, through virtualisation. Depending on the results and behaviour of the files within the virtual environment, the anti-virus will decide if it should allow or deny the files to persist, or be removed.

A. Objective

While most research papers focus upon the exploration or exploitation of various viruses and RATs, this paper will attempt the inverse. Rather than focusing upon the detection of a virus or the expansion of an anti-virus, this paper will concentrate on hiding and attempting to find ways of being hidden from sight, either through a low-level computing cycles or by exploiting anti-virus weaknesses.

By providing a contrasting viewpoint from most papers on a similar topic to computer viruses, the intention will be to focus upon developing a RAT tool that will infect a local computer through various payloads, and then be further developed to try and bypass multiple anti-virus programs. The overall purpose of this will be to try and find various ways of hiding a virus from an anti-virus program, so that further exploration can be in using the same tactics to detect viruses, a method similar to reverse engineering.

B. Potential Issues

There are a number of critical issues that should be considered before development and research can begin.

- The foremost complication that arises with this project is the possibility that no new methods of hiding a RAT from an anti-virus may be discovered.

If this issue was to occur, then development would focus on ensuring that the RAT would be hidden from at least one anti-virus of choice, rather than multiple. By shrinking the scope of the project, more time can be devoted to ensuring some results can be found, and therefore the project will be considered to be a success.

- Windows detects the virus as it is being developed and proceeds to delete the application during development. Windows Defender and similar anti-viruses will be disabled for the majority of this project until it is time to scan the machine.
- The application may have vulnerabilities and offer external attackers a chance to exploit the test machine. The RAT virus will be installed on a self-contained machine called a Virtual Machine through a method called Virtualisation, which would ensure that no computers are ever at risk, and the virtual computer can be deleted easily when it is no longer in need to reduce any risk.

C. Methodology

For this research-focused proposal, the best methodology to adopt and use throughout would be the quantitative research design. This is due the fact that a survey will be used to conduct research on how large of a threat this type of virus can be in a modern, technological world, and experiments will be ran across all anti-viruses to see which ones are able to detect the virus. The results produced from this testing will be numerical, based on the accuracy of the anti-virus and how quick it was able to find and remove the threat. This design will measure success based on length of time of detection, accuracy of the anti-virus (To remove any false positives, for example.),

and if a delivery method is able to safely and securely deliver the payload to the target.

II. LITERATURE REVIEW

To gain a better understanding of the current state of research within the constant evolution of viruses and their counterparts, scholarly papers were found and reviewed through IEEE that were relevant to this proposal and eventual project. Despite the age of some of the papers reaching almost five years old, they remain relevant due to the lack of modern day articles. This would mean that the area of developing and defending against certain viruses is potentially underdeveloped, especially in a modern setting where technology is evolving at dangerous speeds.

Using IEEE as mentioned, specific keywords such as ‘Rat’, ‘Remote Access Trojan’, ‘Virus Ethicality’ and ‘Cloud Encryption’ were used to gather results, with 2015 being the earliest year that a paper may be shown. Roughly twenty papers were found and examined for topics relevant to this proposal, with some proving invaluable to the research of this topic.

A. Encryption

Encryption is one of the main fundamentals in the space of security, regardless of if it is in the digital world or in the real world. It’s the art of scrambling data or words into a code that can only be understood or read by those who hold the knowledge to unscramble it’s true meaning. The first form of ‘modern’ encryption stems from the year 1553, where a keyword had been agreed upon that was required to decrypt the coded message. This form of encryption would eventually find it’s way into the digital world, first appearing in the early 1970s, developed by IBM. This encryption method would eventually be named DES (Data Encryption Standard), and would be used to keep their customers information safe. Even if the DES encryption method has been cracked and deciphered many years ago, any form of encryption is better than none, and a basic layer of encryption could be applied to the delivery of the virus to ensure safer delivery, and delay the time it may take for an anti-virus to identify the virus. The longer it takes for the anti-virus to act, the more information the RAT is able to gather and root itself within the computers lower layers (Depending on it’s functions, of course.)

Yousif Ali and Abdul Hameed in their paper ‘Cloud Crypter for bypassing Antivirus’ claim ‘Results show that our cloud based crypter can be accessed remotely and avoids antivirus detections up to 100% [1]’. They propose a solution to bypassing anti-viruses by delivering their payload through Microsoft Azure, a cloud computing service, which runs the virus through an application called ‘Crypter’. This application is designed to encrypt, manipulate and ultimately hide the virus from the victim and any installed anti-viruses.

Figure 1, page 3, shows how the Crypter creates the encrypted executable. By encrypting the executable code, the Crypter attaches a ‘Stub’, or a decrypter, with the intention of creating a new executable file. This file becomes difficult

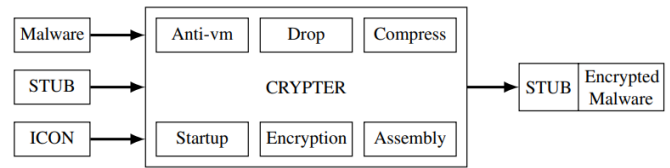


Fig. 1. Design diagram of a Crypter, from ‘Cloud Crypter for bypassing Antivirus [1]’.

for an anti-virus to identify due to the mixing of code and encryption. The decrypter is used to decrypt the virus when it is ready to deploy and the attack to be launched.

The first issue with this method of delivery is the connection to Azure that is required. It relies on the victim to have access to Azure for long enough for the files to be downloaded. Azure is also not an application many people use in their daily lives. As the encryption and decryption of the virus runs both locally on the victim’s computer, there is no logical reason to use Microsoft Azure as a method of delivery as there was no visible reason or benefit proposed or mention in the paper, and was seemingly only used to mention Microsoft Azure and the Cloud within the title. The second issue with this method comes in the form is the decrypter. Decrypting the virus would take time and some computing cycles, as it would require the decrypter to read the whole application. This applies the risk of the victim noticing the process being run, or the anti-virus may notice the files as they become decrypted.

B. Networking

Almost all of the papers found on IEEE focus upon the detection or prevention of IP Spoofing, with few papers discussing the advantages of IP spoofing for preventing DDoS (Direct Denial of Service) attacks. In the paper ‘IP Spoofing Detection for Preventing DDoS Attack in Fog Computing’, Amakan Elisha Agoni and Mqhele Dlodlo propose a method to detect a trick called IP Spoofing. IP Spoofing is a trick used by attackers to hide their tracks by faking what the source IP address of network packets. IP Spoofing will be discussed in greater detail in the Further Expansion section later in this paper. While the paper written by Amakan and Mqhele focuses on detecting IP Spoofing, it still explains how IP Spoofing is used by attackers, claiming ‘Source IP address spoofing is carried out by attackers to conceal the identity of the attacker, thereby thwarting any trace back or falsifying activities in the network in order to access certain services reserved for a legitimate host [6]’.

The paper titled ‘A Study on the IP Spoofing Attack through Proxy Server and Defence Thereof’ written by Young-Hyun Chang, Kyung-Bae Yoon and Dea-Woo Park explains in detail how IP spoofing works and how to set up a proxy server for use in IP spoofing. While most of the papers explain how IP spoofing works on a basic level, this paper explains in detail how the system and functions work, which will be invaluable if IP spoofing is explored in the final project.

C. RAT Detection

As mentioned earlier in this proposal, the vast majority of papers focus upon the detection of RAT viruses, and there were no papers on IEEE that explored RAT viruses further, specifically in the development section. This may be due to the ethicality of publishing a paper that focuses on developing a virus. However, these papers are still useful to the research of this project, as the detection methods discussed by the papers may be used in the reverse. If a paper, for example, focuses on the detection of a RAT virus through machine learning by monitoring network usage, then the virus may be developed to either always keep a constant connection, or to work in smaller bursts as to not arouse suspicion, and then detection. This is the case with the article 'An Approach to Detect Remote Access Trojan in the Early Stage of Communication', written by Dan Jiang and Kazumasa Omote. In this article, they propose a machine learning algorithm that monitors the traffic to detect a constant pattern as displayed in figure 2, page 4.

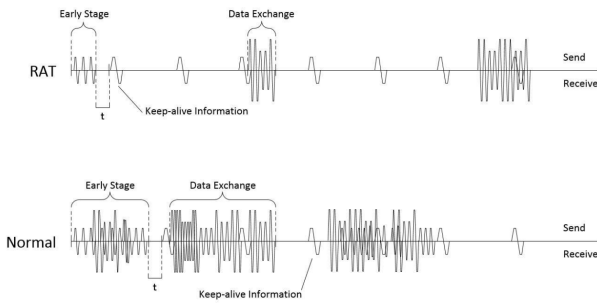


Fig. 2. 'Example of network traffic monitoring during the early stage of a RAT virus attack [2]'.

This figure clearly shows the difference in a RAT virus attack compared to normal internet traffic. Once finding this information, an attacker can then use it to change their application to work around or deploy differently, effectively rendering the machine learning algorithm useless. This type of research is crucial for both the attackers and defenders in the development cycle of virus and anti-viruses. While it is extremely unlikely that this particular algorithm explained in the article is used in a real life anti-virus, it does help educate attackers and virus developers what to expect and what to look out for when they are about to launch an attack.

D. Discussion

The vast majority of already conducted research on this topic is either out-dated by modern technology standards or focusing on the detection of a 'RAT'. Therefore, more research will need to be conducted in this area with hands-on experience, especially in the area of transferring a virus from the attackers machine to a victims, as this area is severely underdeveloped and can provide useful information in learning what areas are most vulnerable for attack.

None of the articles found on IEEE or Google Scholar discuss the ethicality of using a virus for the purpose of

research, nor do they discuss the methods of how a 'RAT' may be hidden, giving the impression that this area of Cyber Security is not very developed in an academic setting.

III. RESEARCH

A. Programming Language

While an application may technically be created using any programming language, there are often better languages suited to application needs. Each language has their own advantages, and the general semantics remain the same between each language. A RAT can be developed in any of the main, well-known languages such as Python, C++ or Java, but picking which language to use for the projects needs are important.

1) *Python*: Usually, small scale applications tend to be developed using the python programming language due it's ability to create powerful and complex applications easily. However, this language will not be suitable for the Remote Access Trojan, as it would require Python to be installed on the victim's computer. This is not reliable to rely on, as it is likely that the majority of computers may not have Python installed.

A method to alleviate this problem could be to include a self-contained executable, which will include all of the required libraries and code within itself, but this would increase the viruses file size considerably. This would then cause issues by the slower transfer rate when the initial infection begins, and it increases the risk of the virus being discovered due to the larger than average file size.

The advantage of using Python for the programming language is that Python is cross-platform, meaning the virus would be compatible with a Windows computer, MacOS, Linux, and even Android phones. While Python is not natively supported on the iPhone platform, there are resources to support Python applications to run on devices such as iPhones and iPads.

Python has another advantage in that there are multiple libraries for use already available which will be useful in speeding up the development speed of the application, and reducing the overall size of the application as the code length will be shorter.

Python is a high-level programming language, which means that it would rely on external scripts to be installed to run correctly. It also would not allow the application to pull, communicate or read any data or information from API calls or from the memory, reducing it's effectiveness as a virus.

2) *C*: C is popular programming language used by developers to create low level applications that are designed to be hidden. As the RAT virus has no visual component, C++ would not be viable for this project.

C works on a low-level programming language, meaning that the application can access the underlying system, enabling the application to work and retrieve from the memory directly.

When an application written in C is compiled, everything is packaged into it's own isolated executable, so that it would not need to rely on any external scripts or files. This enables the surreptitious application to be delivered as a singular

executable file. This single file will be of smaller size with fast execution as it does not require additional scripts or libraries to search and communicate with, resulting in a fast, small and effective executable.

3) *Java*: Java suffers the same main issue as Python where the victim computer would require Java to be pre-installed. While Java is a popular requirement for some games and applications, there is the risk that the victim's computer has yet to download the required software, rendering the virus useless.

Java is known for its large cost in terms of memory, and this would potentially flag the program as a virus due to the amount of RAM it would be using.

While Java has been used in the past for malware development, Java tends to "sandbox" Java applications. To run native code or access arbitrary files, a Java application must ask for permission, which creates digital signatures and certificates which may be tracked back to the perpetrator, or lead to the discovery of the malware.

4) *Visual Basic*: Visual Basic boasts one main advantage in the development of malware. This advantage stems from the ability to use Microsoft Office macro support to deliver a payload. This method of payload combined with social engineering creates a very powerful combination.

This is a common method of delivering a payload that is usually used in workplace environments. For example, if an attacker wished to attack a specific company, the easiest targets would be the employees on the lower levels. Using a website such as LinkedIn, the attacker can easily find information on who an employee's boss is.

According to Ruth Bearden and Dan Chai-Tien Lo in the paper 'Automated Microsoft Office Macro Malware Detection Using Machine Learning', 'Microsoft Word documents containing macro viruses typically try to persuade viewers to enable the macros. However, informing MS Office users of the dangers of enabling macros is no longer a sure defence. Attackers are applying social engineering techniques when designing malicious documents in an effort to make them appear legitimate. To increase the techniques effectiveness, attackers are using sophisticated information harvesting and data mining from creative places, such as a corporation documents metadata, to launch targeted attacks which are more successful than previous spam campaigns [7]'.

An example of this tactic mentioned in the paper would be an email that is sent to the employee with a fake email that is very close to the boss's email, tricking the employee into believing it to be them. The email could be something as 'harmless' as 'Look at next week's schedule', or pretending the document was sent to them in error, and it was really a spreadsheet was a list of who is going to be fired. This type of fear will help the attacker in opening the document, as the employee would be more concerned about if they are going to be fired instead of checking the email address.

When the Microsoft Office document has been downloaded, the attacker can prompt the employee for a password. However, the attached macro could have already been deployed, and the RAT would already be at work. The employee may

just discard the document as they don't know the password, and not realise that their computer has just been injected.

B. Programming Language - Conclusion

Each programming language has its own advantage and disadvantage for this specific project. After discussing and reviewing each of the languages, the Remote Access Trojan in this project will be developed using the C language. The reasoning for this decision is that the language will be vital in being small in size, and able to access the vital low-level CPU functions and memory. This will allow the virus to be harder to detect, easier to deploy, and allow for further expansion if the attacker wished to modify critical functions within the victim's computer.

Visual Basic would be a viable language to use if the payload is dependant on Microsoft Office, though there are many other payloads that exist already. The time spent on re-creating the malware in another language for a singular payload could be considered a waste of time, though the method of delivery is effective.

C. Survey on Ethicality

A survey will be created and sent to people of various age groups and experience with technology with the intention of gathering a wide range of data. The survey will consist of the following questions.

- Do you understand what a Remote Access Trojan is?
- What operating system do you use?
- Have you ever been effected by a RAT or similar virus before?
- Which age group do you belong to? (Note, this is used only to determine if there is a pattern between certain ages).
- Do you think it would be ethical to create and develop a RAT for research purposes only?
- Do you have Python, the programming language software, installed on your computer?
- Do you have Java, the programming language software, installed on your computer?
- Do you use a VPN?
- Which desktop anti-virus do you currently use?
- Has your anti-virus ever picked up a false positive? (This means that your anti-virus flags a file/application as a virus, but it's actually safe.)

These questions will be sent to friends, relatives and posted anonymously in various forums related to technology to assist in further gathering data that will be non-biased. These questions are tailored to be non-intrusive and to assist in developing the eventual virus.

As operating systems have advanced, newer operating systems have stronger security that will be relevant to how the application will be made. The questions regarding Python and Java will be interesting to gather information about as they will assist in knowing if people do have these languages installed (and therefore be vulnerable to viruses developed in these languages). Knowing what the vast majority of anti-viruses

people use will be useful to know what anti-viruses should be used in testing when the virus has been developed, and which to focus on bypassing.

The ethics of developing and then deploying a virus is questionable, and one that has been debated multiple times outside of academic studies. Though the paper's purpose is intended to be educational and for research, it may be considered dangerous as it would be exploring various methods of hiding, masking and concealing a potentially dangerous virus.

The purpose of the aforementioned survey will be put together containing various questions relevant to the virus and the ethicality of its development, or finding who has been effected by a RAT virus before. This question will be followed by another question asking for the users age, if they are willing to share it. These responses will be useful in determining if RAT viruses are still used and relevant to the modern age, and if so, how many people are aware of them. Knowing the age of the victim will be useful to determine if this type of virus is easier to exploit on the older generations, or if younger people are effected in the same capacity.

IV. FURTHER EXPANSION

Beyond the development of the RAT virus and attempts to hide the virus from an anti-viruses detection, time will be spent on developing the tools in other ways to avoid detection. The next few sections will outline the possibilities of exploration that the project may evolve into if possible and there is enough time.

A. Encryption

One method that, as mentioned earlier, would be an effective tool to assist in the obscuration of the virus would be encryption. This would come in the form of a Crypter, which would be researched and developed to encrypt, decrypt and build a stub with the C programming language, similar to the virus itself.

B. Cloud Computing

Another aspect that may be explored within this project would be the benefits of incorporating Cloud Computing. As the data gathered by the virus needs to be sent back to the attacker, it may be easily traced through the network in the forms of a packet sniffer.

One way of mitigating this risk would be to have the data sent to a server on the cloud, such as Microsoft Azure. Azure supports a concept called Load Balancing, which splits the packets evenly across multiple virtual machines, or servers, to lessen the burden of network traffic. Each packet contains a destination and source IP address, and are usually sent through TCP ports. TCP ports don't allow for modification of the destination IP address (Otherwise it would be sent to the fake IP, and therefore become lost), but it does allow for modification of the source IP address. This is known as 'IP Spoofing', and could be used in this project as a technique to hide the path that the data has travelled upon, effectively concealing where the data has come from, and who the victim was.

C. Delivery

The delivery is a crucial part of every virus and must be carefully considered before any development may even begin. If a virus is to be encrypted, then the decryption key has to be delivered alongside the virus, increasing the overall size, and therefore the chance of detection. Some attackers may bypass this by sending the decryption key separately, though this requires a constant connection to be maintained between the first and second payload. This further increases risk, as the connection is not easily concealed.

Because of these risks, one potential expansion for this topic may be to compare various methods of delivery. These potential delivery methods would include the delivery through the cloud with Azure, where the virus is delivered remotely through a website. Another would be a phishing scam attempt where an email is faked with an attachment, where the attachment is the virus. An additional delivery method could be a Raspberry Pi that, when connected to a Windows machine, begins transferring files to the root of the computer, though this requires physical access and is very risky. The final potential delivery method is similar in that it could be transferred over a USB stick, though this method contains the same risk as the previously mentioned Raspberry Pi method.

V. RESULTS

The results gathered from this project will come in the form of detection. If the virus has been detected by an anti-virus, then it would be noted as a failure. If the virus manages to go undetected by the anti-viruses installed and can safely connect and transmit data, then it would be considered a success. As the virus would need to be delivered safely, multiple methods of payloads will be tested, and the results of each method will be noted. If the virus is detected by one anti-virus at the beginning, and then is able to be developed further to the point it is no longer detected, then this research will be considered a success. This may be done through a different method of delivery. For example, an anti-virus may struggle to detect a virus delivered through cloud software, or the sections of the virus that are detected may be removed or tweaked to avoid detection. As anti-viruses discover viruses by matching what is in their database, they may struggle to detect a custom made anti-virus. If this is the case, the experiment will be considered a success, and the methods of virus delivery will be explored.

VI. TERMINOLOGY

List of terminologies used in this document:-

- RAT - Remote Access Trojan.
- RAM - Random Access Memory.
- IP Address - Internet Protocol Address.
- DDoS - Direct Denial of Service.
- DES - Data Encryption Standard.

VII. REFERENCE LIST

- [1] Y. Ali and A. Hameed, "Cloud crypter for bypassing antivirus," in *2019 15th International Conference on Emerging Technologies (ICET)*, 2019, pp. 1–6.
- [2] D. Jiang and K. Omote, "An approach to detect remote access trojan in the early stage of communication," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, 2015, pp. 706–713.
- [3] F. Cohen, "Computer viruses: Theory and experiments," pp. 22–35, 1987. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0167404887901222>
- [4] V. Valeros and S. Garcia, "Growth and commoditization of remote access trojans," pp. 454–462, 2020.
- [5] V. G. Tasiopoulos and S. K. Katsikas, "Bypassing antivirus detection with encryption," New York, NY, USA, p. 1–2, 2014. [Online]. Available: <https://doi.org/10.1145/2645791.2645857>
- [6] A. E. Agoni and M. Dlodlo, "Ip spoofing detection for preventing ddos attack in fog computing," in *2018 Global Wireless Summit (GWS)*, 2018, pp. 43–46.
- [7] "Protect yourself from macro viruses - Microsoft Support." [Online]. Available: <https://support.microsoft.com/en-us/office/protect-yourself-from-macro-viruses-a3f3576a-bfef-4d25-84dc-70d18bde5903>