VSB–TECHNICAL UNIVERSITY OF OSTRAVA
VŠB–TECHNICKÁ UNIVERZITA OSTRAVA

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DEPARTMENT OF COMPUTER SCIENCE
KATEDRA INFORMATIKY

ARTIFICIAL INTELLIGENCE IN CYBERSECURITY

DOCTORAL THESIS
DIZERTAČNÍ PRÁCE

AUTHOR
AUTOR PRÁCE
Msc. THANH CONG TRUONG

SUPERVISOR
VEDOUCÍ PRÁCE
prof. Ing. IVAN ZELINKA, Ph.D.

OSTRAVA 2020

VSB – TECHNICAL UNIVERSITY OF OSTRAVA
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

Artificial intelligence in cybersecurity

Ph.D. Thesis; Delivered in October 2020
Doctoral Study Programme:
P1807 Computer Science, Communication Technology and Applied Mathematics
Doctoral Study Branch:
1801V001 Informatics

Student: M.Sc. Thanh Cong Truong
Department of Computer Science
Faculty of Electronic Engineering and Computer Science
VSB - Technical University of Ostrava
17. listopadu 15/2172, 708 00 Ostrava - Poruba
cong.thanh.truong.st@vsb.cz

Supervisor: Prof. Ing. Ivan Zelinka, Ph.D.
Department of Computer Science
Faculty of Electronic Engineering and Computer Science
VSB - Technical University of Ostrava
17. listopadu 15/2172, 708 00 Ostrava - Poruba
ivan.zelinka@vsb.cz

OSTRAVA, 2020

ABSTRACT

Artifcial intelligence (AI) and machine learning (ML) have grown rapidly in recent years, and their applications in practice can be seen in many felds, ranging from facial recognition to image analysis. Recent developments in Artificial intelligence have a vast transformative potential for both cybersecurity defenders and cybercriminals. Anti-malware solutions adopt intelligent techniques to detect and prevent threats to the digital space. In contrast, cybercriminals are aware of the new prospects too and likely to adapt AI techniques to their operations.
This thesis presents advances made so far in the field of applying AI techniques in cybersecurity for combating against cyber threats, to demonstrate how this promising technology can be a useful tool for detection and prevention of cyberattacks. Furthermore, the research examines how transnational criminal organizations and cybercriminals may leverage developing AI technology to conduct more sophisticated criminal activities. Next, the research outlines the possible dynamic new kind of malware, called X-Ware and X-sWarm, which simulates the swarm system behaviour and integrates the neural network to operate more efficiently as a background for the forthcoming anti-malware solution. This research proposes how to record and visualize the behaviour of these type of malware when it propagates through the file system, computer network (virus process is known) or by observed data analysis (virus process is not known and we observe only the data from the system). Finally, a paradigm of an anti-malware solution, named Multi agent antivirus system has been proposed in the thesis that gives the insight to develop a more robust, adaptive and flexible defence system.

KEYWORDS

Cybersecurity, artificial intelligence, machine learning, deep learning, swarm intelligence, neural network

ABSTRAKT

Význam umělé inteligence (AI) a strojového učení (ML) v posledních letech rychle rostl a na jejich aplikacích lze vidět, že v mnoha oblastech, od rozpoznávání obličeje až po analýzu obrazu, byl učiněn velký pokrok. Poslední vývoj v oblasti umělé inteligence má obrovský potenciál jak pro obránce v oblasti kybernetické bezpečnosti, tak pro útočníky. AI se stává řešením v otázce obrany proti modernímu malware a hraje tak důležitou roli v detekci a prevenci hrozeb v digitálním prostoru. Naproti tomu kyberzločinci jsou si vědomi nových vyhlídek ve spojení s AI a pravděpodobně přizpůsobí tyto techniky novým generacím malware, vektorům útoku a celkově jejich operacím.
Tato práce představuje dosavadní pokroky aplikace technik AI v oblasti kybernetické bezpečnosti. V této oblasti tzn. v boji proti kybernetickým hrozbám se ukázuje jako slibná technologie a užitečný nástroj pro detekci a prevenci kybernetických útoků. V práci si rovněž pokládme otázku, jak mohou nadnárodní zločinecké organizace a počítačoví zločinci využít vyvíjející se technologii umělé inteligence k provádění sofistikovanějších trestných činností. Konečně, výzkum nastíní možný nový druh malware, nazvaný X-Ware, který simuluje chování hejnového systému a integruje neuronovou síť tak, aby fungovala efektivněji a tak se celý X-Ware a X-sWarm dal použít nejen jako kybernetická zbraň na útok, ale i jako antivirové obranné řešení. Tento výzkum navrhuje, jak zaznamenat a vizualizovat chování X-Ware, když se šíří prostřednictvím systému souborů, sítí a to jak analýzou jeho dynamiky (proces je znám), tak analýzou dat (proces není znám, pozorujeme jen data). Nakonec bylo v disertační práci navrženo paradigma řešení proti malwaru, jež bylo nazváno „Multi agent antivirus system“. Tato práce tedy poskytuje pohled na vývoj robustnějšího, adaptivnějšího a flexibilnějšího obranného systému.

KLÍČOVÁ SLOVA

kybernetická bezpečnost, umělá inteligence, strojové učení, hluboké učení, rojová inteligence, neuronová síť

DECLARATION

ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest thankfulness to my supervisor Prof. Ing. Ivan Zelinka, Ph.D. for his encouragement and great support. I appreciate all his contributions of knowledge, time, ideas, and constant support through the process of completion of my thesis. Without his motivation and instructions, the thesis would have been impossible to be done effectively. Next, I want thank to all my friends during my study period in VSB - Technical University of Ostrava. Their kindly help, care and motivation gave me strength and helped me up all the trouble. Finally, I would like to thank my family who always encouraged me with love and devoutly supported me. Especially, I am very much thankful to my wife for all her love, support, understanding and constant motivation to complete this research work.

Ostrava . . . .

author's signature

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACO	Ant colony optimization
AI	Artificial Intelligence
AIS	Artificial immune systems
ANN	Artificial neural networks
AR	Association rule algorithms
C&C	Command and Control
CIFS	Common Internet File System
CNNs	convolutional neural networks
DBNs	Deep belief networks
DL	Deep learning
DT	Decision trees
EDLNs	Ensemble of Deep Learning networks
EL	Ensemble learning
ES	Evolution strategies
FNN	Feedforward neural networks
GA	Genetic algorithms
GANs	Generative adversarial networks
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPC	Interprocess Communication
KNN	K-nearest neighbor
MAAS	Multi agent anti-virus system
ML	Machine learning
OR	Onion Router
P2P	Peer to Peer
PCA	Principal component analysis
PSO	Particle swarm optimization
RBMs	Restricted Boltzmann machines
RF	Random forest
RL	Reinforcement Learning
RNN	Recurrent neural networks
SAE	Stacked autoencoders
SI	Swarm intelligence
SMB	Server Message Block
SSL	Secure Sockets Layer
SVM	Support vector machines
TCP	Transmission Control Protocol
TOR	The Onion Router.

1 INTRODUCTION

1.1 Overview

Today, cybersecurity become an essential part for computer systems and infrastructures with a concentrate on the defence of valuable resource stored on those systems from malicious actors who want to steal, damage, destroy or forbid access to it. The increasing use of Internet services and computer systems has led to a sharp increase in the number of attacks on cyberspace. The Internet is not only the chief source of information, but it is also a viable medium for cybercriminals to use it as a tool to carry out cyber-attacks. Contemporary with the development of technology, cyber-threats are becoming more sophisticated and automation. Cybercriminals are always changing their methods, making attacks challenging to predict and prevent.

Over the years, scientists have investigated a considerable number of methods to protect information systems from unauthorized access and unauthorized use. Such techniques may involve proper implementation of security mechanisms such as passwords, encryption, access control lists as well as complicated security protocols. Nevertheless, conventional security systems, which use rules and signatures, have become ineffective against flexible, continually evolving cyber-attacks. Furthermore, current cyber defence systems involve humans at multiple levels, which may cause the information flow slow and asynchronous. As a consequence, such a system is unavailable to adapt to the speeds of cyber threats. Thus, we need advanced approaches such as applying Artificial intelligence (AI) techniques that provide flexibility and learning capability to assist humans in combating cybercrimes.

On the contrary, malicious actors are aware of the new prospects too and will probably attempt to use for nefarious purposes. This technology will also be used as a way to improve threats. For example, malicious actors can leverage Machine learning (ML) technique to generate a hard-to-detect malware variant with machine speed. What is more, AI might be able to personalize the phishing scheme better and raising the scale of the attack, making the attack more likely to succeed. Hence, it is logical to expect that, shortly evolutionary techniques, artificial intelligence as well as swarm intelligence will be used in computer viruses and antivirus solution.

1.2 The impact of AI on Cybersecurity

The rapid development of computing technology and the internet has a significant impact on people's daily life and work. Unfortunately, it also caused many new cybersecurity challenging issues: First, the explosion of data makes manual analysis impractical. Second, threats are growing at a high rate, which also means that new short-lived species and highly adaptive threats become commonplace. Third, at present, the threats compromise various techniques for propagation, infection, and evasion, therefore, they are hard to detect and predict. Moreover, the expense to prevent threats also should be considered. It takes much time, money, and effort to generate and implement the algorithm. Also, employing or training specialists in the field is hard and expensive. What is more, many threat variations emerge and spread continuously. Hence, AI-based methods are expected to cope with these cybersecurity issues.

1.2.1 The positive uses of AI

In the field of cybersecurity, AI is already being used to advance defensive capabilities. Based on its powerful automation and data analysis capabilities, AI can be used to analyse large amounts of data with efficiency, accuracy, and speed. An AI system can take advantage of what it knows and understand the past threats to identify similar attacks in the future, even if their patterns change. Undoubtedly, Artificial Intelligence has several advantages when it comes to cybersecurity in the following aspects:

- *AI can discover new and sophisticated changes in attack flexibility*: Conventional technology is focused on the past and relies heavily on known attackers and attacks, leaving room for blind spots when detecting unusual events in new attacks. The limitations of old defence technology are now being addressed through intelligent technology. For example, privileged activity in an intranet can be monitored, and any significant mutation in privileged access operations can denote a potential internal threat. If the detection is successful, the machine will reinforce the validity of the actions and become more sensitive to detect similar patterns in the future. With a larger amount of data and examples, the machine can learn and adapt better to detect anomalous, faster and more accurate operations. This is especially useful when cyber-attacks are becoming more sophisticated, and hackers are making new and innovative approaches.
- *AI can handle the volume of data*: AI can enhance network security by developing autonomous security systems to detect attacks and respond to breaches. The volume of security alerts that appear daily can be very overwhelming for security groups. Automatically detecting and responding to threats has helped to reduce the work of network security experts and can assist in detecting threats more effectively than other methods. When a large amount of security data is created and transmitted over the network every day, network security experts will gradually have difficulty tracking and identifying attack factors quickly and reliably. This is where AI can help by expanding the monitoring and detection of suspicious activities. This can help network security personnel react to situations that they have not encountered before, replacing the time-consuming analysis of people.
- *AI security system can learn over time to respond better to threats*: AI helps detect threats based on application behaviour and the entire network activity. Over time, AI security system learns about a regular network of traffic and behaviour and making a baseline of what is normal. From there, any deviations from the norm can be spotted to detect attacks

1.2.2 Drawbacks and limitations of using AI

The advantages highlighted above are just a fraction of the potential of AI assist cybersecurity, but the application of this technology had some limitations as described below.

- *Data sets*: Creating an AI system demand a considerable number of input samples, which however, obtaining and processing the samples can take a long time and resources.
- *Resources requirement*: Building and maintain the fundamental system need an immense amount of resources, including memory, data, and computing power. What's more, skilled resources necessary to implement this technology requires a significant cost.
- *False alarm*: Frequent false alarms are an issue for end-users, disrupt business, potentially delaying any necessary response and generally affecting efficiency. The process of fine-tuning is a trade-off between reducing false alarms and maintaining the security level.
- *Attack the AI-based system*: Attackers can use various attacks techniques that target AI systems such as adversarial inputs, data poisoning and model stealing.

1.3 Contributions of the thesis

This thesis exposes issues related to the application of AI techniques in network security. The principal contributions of this thesis are:

- We conduct comprehensive research about how AI-based techniques could be applied to enhance the cybersecurity solution as well as the potential maliciously used of AI methods.
- We propose a new type of malware that combines SI, ANN as a background for an anti-malware solution.
- Based on the architectural model of the X-Ware, we have designed an anti-malware paradigm for the upcoming swarm malware.

1.4 Thesis organization

The rest of this thesis is organized as follows: Chapter 2 is focused on the thesis motivation and main goals of the dissertation. Chapter 3 presents the knowledge background employed in the thesis. Chapter 4 describes the state of the art. Chapter 5 presents the application of AI in the cyber domain. Chapter 6 describes all essential information about the methodology of X-Ware creation as well as summarises and discussed author's results in developing the X-Ware. In chapter 7, a design of a novel worm with swarm characteristics, whose communication relies on the Tor privacy infrastructure is proposed. Chapter 8 presents a paradigm design to intelligent anti-malware solution. Chapter 9 is focused on summary and discussion about achieved results and proposal of the future work.

2 MOTIVATION AND GOALS

2.1 Motivation

Recent days, the exponential growth of the Internet and its technologies and the rising dependency of users upon the applications for most essential demands has raised the importance of cyberspace security. As a consequence, cybersecurity becomes a vital part of computer systems and infrastructures with a concentrate on the defence of valuable resource stored on those systems from adversary action.

Over the years, numerous solutions have been proposed for protecting the information systems from unauthorized access and unauthorized use. These methods may involve proper implementation of security mechanisms such as password protection, firewalls, access control and encryption. Nevertheless, these techniques have become ineffective against the increasingly sophisticated, continually evolving cyber-attacks.

Furthermore, current cyber defence solutions involve humans at multiple levels, which may cause the information flow slow and asynchronous. As a consequence, such a system is unavailable to adapt to the speeds of cyber threats. Thus, more advance techniques need to be research to assist humans in combating cybercrimes.

In recent days, a variety of AI architecture designs have blossomed in the context of cybersecurity. On the offence side, cyber threat actors can employ AI to improve the sophistication and scope of their attacks. On the defence side, AI is utilized to enhance the defence strategies, so that the defence systems become more robust, flexible, and efficient, which can adaptively with the environment and decrease the impacts occurred. On the other side, attackers are using the same tools to increase the sophistication of their attacks. However, there is a lack of literature dealing with the malicious uses of AI technologies. The research and industry communities need to understand how AI can be applied to cyber-attacks and where the weak points are to find the best vaccine for them.

The motivation of this dissertation work is to seek the answer for the question: “what is the impact of Artificial intelligence in cybersecurity“. This question is formed in part by the perception that AI plays a very significant role in the of cybersecurity in recent time.

2.2 Goals of the thesis

This research is aimed to illustrate how artificial intelligence techniques can be utilised in cybersecurity. The following highlights are stated as the main goals of the dissertation.

2.2.1 Goal 1

Study the applications of AI on cybersecurity from two respects, how AI improve the cyber defence solutions and the misuse of AI and its impact, with already existing examples:

- First, I will introduce a brief explanation of AI methodology for cybersecurity.

- Second, AI-based approaches for defending against cyberspace attacks will be listed.
- Next, a range of possible uses toward which AI could be put for the malicious end will be illustrated.
- Finally, I will summarise the main points and discuss the challenges and open research directions.

2.2.2 Goal 2

Propose a prototype of X-Ware that use swarm intelligence principle and AI technique as a backbone of its functionality and generation:

- First, the conception of a new type of malware that is a combination of swarm intelligent, ANN, and a traditional virus will be proposed.
- Second, the conception will be applied to design a prototype.
- Finally, the prototype will be evaluated via empirical experiments and analysis.

2.2.3 Goal 3

Monitor and analyse the behaviour of the X-Ware on the simulation of the real network system:

- First, a collection of analysis techniques and monitoring tools for observing the X-Ware will be studied.
- Second, I will apply these techniques and tools to monitor the prototype.
- Finally, the results from empirical experiments will be analysed and discussed.

2.2.4 Goal 4

Design structure and the principles of the possible swarm anti-malware solution:

- First, I will propose the conception of an anti-malware solution based on the conception of the swarm virus.
- Second, the conception will be utilised to design a Multi-agent antivirus system.
- Finally, the main points of the proposed framework will be discussed. Additionally, suggestions for future work will also be presented.

3 BACKGROUND

This chapter provides the necessary theoretical background, that is, the methods and procedures for the establishment of a key material which is in the constant focus of how AI would leverage on the cyber applications.

3.1 Artificial intelligence

Defining AI can take two approaches. First, it is a science that strives to discover the nature of intelligence and develop smart machines in which scientists apply information, logic, self-learning, and determination to make machines becoming intelligently. To put it simply, humans create machines with intelligence. This intelligence can think, learn, decide, and work while trying to solve a problem as human intellect. On the other hand, scientists define AI as a science that researches and develops methods for resolving complexity problems that impossible to be resolved without adopting intelligence. For example, scientists build an AI system for real-time analysis and decision making based on enormous amounts of data. In recent years, AI has resulted in advances in many scientific and technological fields such as computerized robots, image and recognition, natural language processes, expert systems and other majors. The next few chapters will focus on some of the areas listed previously. These chapters will discuss the utilization of some of these artificial intelligent areas in cybersecurity.

In fact, AI has represented a broad category of methods for teaching computers how to perform the tasks as a human. This includes Machine Learning (ML), a technique that trains a machine how to learn; and Deep Learning (DL) that helps machines to analyze data so that they can draw the same conclusions as a human. As a core building block for AI, Machine Learning uses algorithms to parse data, learn from that data and make the decision based on what it taught. Deep Learning is a strict subset of machine learning that uses a structure as close as possible to the human brain as a model for learning and makes intelligent decisions on its own.

3.2 Cybersecurity

The main aim of cybersecurity is to protect computers, networks, and software programs from such cyber attacks. While a variety of definitions of the term cybersecurity have been suggested, this paper will use the definition suggested by Craigen et al. (2014) [32] following as: "*Cybersecurity is the organization and collection of resources, processes, and structures used to protect cyberspace and cyberspace-enabled systems from occurrences that misalign de jure from de facto property rights.*" Furthermore, cybersecurity concerns with the practice of ensuring integrity, confidentiality, and availability of information of the organization and user's assets [128], which is further described with the following definitions

- Integrity refers to preventing unauthorized modification or deletion of information.

- Confidentiality refers to protecting sensitive information from unauthorized individuals or systems.
- Availability refers to keeping information accessible by authorized users.

The term “cybersecurity“ includes other concepts, which are defined by the ISO [1] as follows:

- Information security: *“concerned with the protection of confidentiality, integrity, and availability of information in general, to serve the needs of the applicable information user“.*
- Network security: *“concerned with the design, implementation, and operation of networks for achieving the purposes of information security on networks within organizations, between organizations, and between organizations and users“.*
- Internet security: *“concerned with protecting internet related services and related ICT systems and networks as an extension of network security in organizations and at home, to achieve the purpose of security. Internet Security also ensures the availability and reliability of Internet services“*
- Critical information infrastructure protection: *“ensures that those systems and networks are protected and resilient against information security risks, network security risks, internet security risks, as well as Cybersecurity risks“.*
- Cybercrime: *“criminal activity where services or applications in the Cyberspace are used for or are the target of a crime, or where the Cyberspace is the source, tool, target, or place of a crime“.*
- Cybersafety: *“condition of being protected against physical, social, spiritual, financial, political, emotional, occupational, psychological, educational or other types or consequences of failure, damage, error, accidents, harm or any other event in the Cyberspace which could be considered non-desirable“.*

Along with the development of technology, malicious software is becoming more sophisticated and automated. Cybercriminals are always changing their methods, making attacks difficult to predict and prevent. Conventional security systems, which use rules and signatures, have become ineffective against flexible, continually evolving cyber-attacks. Thus, we need advanced approaches such as applying AI-based techniques that provide flexibility and learning capability to assist humans in combating cybercrimes.

3.3 Swarm intelligence

Nature’s inspiration always helps humanity to solve real problems. Over the years, the applied of bio-inspired techniques have been seen in various areas ranging from computer science, engineering, economics, medicine, and social sciences. Among those biologically-inspired techniques, SI is a research direction that has attracted scientists in recent times. This domain focus on designing robust systems (or principles) that can operate efficiently and intelligently under threat and catastrophic conditions without centralized control [20]. SI-based algorithms have emerged as nature-inspired algorithms that are capable of providing quick, robust, and inexpensive solutions to various complex problems [29][94].

The expression "Swarm Intelligence" was first introduced by Beni and Wang in the context of cellular robotics system [15]. This term refers to the collective behavior of agents in nature such as ant colonies, honey bees, fireflies, and bird flocks. These agents in the swarm collaborate with one another to achieve tasks necessary for their survival. Generally speaking, SI is a population-based system that comprises many individuals in it that communicate locally with each other and with their environment. The individuals interact with each other via direct or indirect to solve the problem [91]. Direct interaction may through direct contact, like ants touch each other with their antenna or through audio and visual. On the other hand, indirect interaction, which is referred to as stigmergy, means the individual communicate through the environment, such as ant product pheromone use for exchanging information [34]. SI can therefore be defined as a branch of AI that is used to model the collective activities of social agents in nature.

To date, many SI algorithms have been proposed in the literature and successfully applied in practice, including function optimization problems, finding optimal routes, scheduling, structural optimization, and image analysis [37]. Examples of SI algorithms are: Ant Colony Optimization [34], Particle Swarm Optimization [36], Artificial Fish Swarm [78], Artificial Bee Colony [66], Self-Organizing Migrating Algorithm [143], Bacterial Foraging [96], Cat Swarm Optimization [30], Glowworm Swarm Optimization [70], Firefly Algorithm [139], Bat Algorithm [140], Grey Wolf Optimizer [86], and Whale Optimization Algorithm [85]. The most well known and very basic one is Ant Colony Optimization (ACO), followed by Particle Swarm Optimization (PSO) and Self-Organizing Migrating Algorithm (SOMA). As a consequence, these methods were utilized in a variety of problem domains as well as cybersecurity.

Recently, scientists proposed numerous techniques that have utilized SI for ensuring integrity, confidentiality, and availability of information. In this part, we category applicability of SI in the field of cybersecurity into three main groups: feature selection, parameter and weight optimization, and other approaches

3.3.1 Feature selection

In term of applying SI in cybersecurity, one of the most research direction that attracts the research community is adopting these techniques for optimizing the feature selection. It is derived from the fact that Machine learning (ML) approaches have been widely used to improve the efficiency of cyber defence systems. In fact, feature selection is one of the core concepts in ML, which hugely impacts the performance of the model. Generally speaking, feature selection can be described as the process of choosing the optimal subset of features based on specific criteria. This works by selecting the features with minimum redundancy and maximum relevance score. In this way, it examines for a projection of the data onto fewer features which conserves the information as much as feasible. Feature selection techniques often category into wrapper-based, filter-based or embedded.

3.3.2 Optimum the parameter and weight

ML is often considered as a crucial part of the evolution of cybersecurity solutions. In fact, ML models are parameterized so that their performance can be tuned for a given problem. In general, a model parameter is a configuration variable that is inherent to the model and whose value can be estimated from the provided data. Parameters, which are also learned from historical training data, are crucial to ML methods. In brief, the ML models can have many parameters and discovering the best configuration of parameters can be treated as a search problem. Hence, the application of metaheuristic algorithms to optimize the fine-tuning of parameters is also an attention topic to the research community.

3.3.3 Other applicabilities

Besides the application to optimize the selection of attributes as well as the optimization of parameters, SI algorithms are also carried out in various fields of cybersecurity. For instance, Si-based algorithm had been utilized to generate a traffic profile in [45], for investigating the probable origin of attack in [129], to initiate the parameter [53].

3.4 Malware

3.4.1 Overview

Malware is a generic term generally used for viruses, trojan horses, worms, rootkits, spyware, ransomware, and others. They are intrusive and can perform harmful, unauthorized activities on computer systems without sending any prior consent. Malware is easily propagated into any computer systems causing long-lasting destructions as well as system corruptions. Fig. 3.1 shows a variety of malware in computer systems.

Among other malware, computer virus is the most pervasive one. Similar to biological virus, [114], computer virus, as a program, can automatically replicate itself and do harm to a variety of host files. The list of host files include device drivers, boot code, executable files, and files that can be executed only via some specific applications (such as Visual Basic scripts, Microsoft Word,...). As the hosts are run, the virus code is activated and propagates itself further by autonomous replicating and attaching to other hosts.

Besides virus, different terminologies are used to address different malware based on their purposes, and behaviours, in other words, spreading strategies. To name a few, a worm exploits system vulnerabilities to spread over a network. A spyware hijacks personal and confidential data. An adware automatically and generically delivers unwelcome advertisements into a system. A ransomware locks or encrypts data from a victim for money extortion. A rootkit sneaks in to take the highest possible administration right to control a machine. A pack of botnets is used for an entire network control from a remote server.

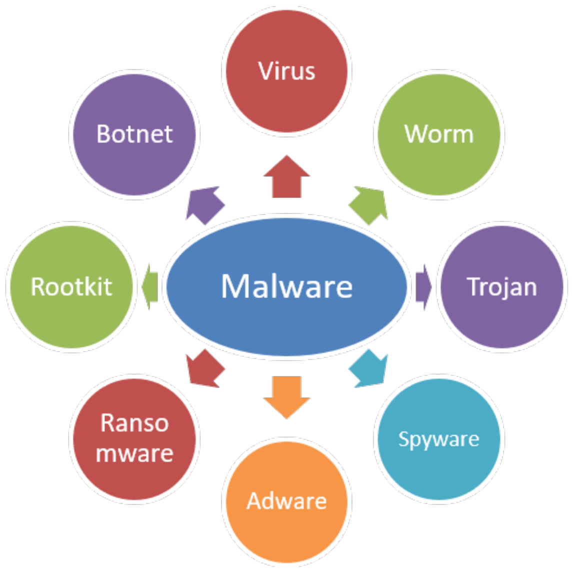


Fig. 3.1: Type of malware

3.4.2 Propagation mechanisms

The propagation of malware can be categorized into several groups as follows:

- Spreading through e-mail: The user open or execute the attachment of the email cause the virus executes. In some cases, opening the e-mail with might lead to execute the malicious code. In addition, Links in e-mail can lead to a website comprised of viruses, which often exploits browser and operating system vulnerabilities. In other words, the link leads to executing a piece of code, and the computer may be infected with a virus.
- Spreading through a network: In this case, a malware spreads in a network by exploiting the vulnerabilities that allows it to compromise itself into a target host.
- Transferring via physical media: A human actor might cause the malicious software to spread, by physically transferring the infected media to new locations.
- Local spreading: The malicious program can self-replicate on a host locally within the limits of access permissions. What's more, the shared network drives (NFS for instance) might allow the malware to spread across systems.

3.5 Virus

3.5.1 Overview

The first idea of computer virus was brought up by John von Neumann in 1950s regarding his study of cellular automata and self-replicating software [47]. Nevertheless, not until 1971 that the Creeper, first self-replicating program developed by Bob Thomas was presented [24]. In 1983, Fred Cohen introduced a program which is capable of infecting a computer, replicating itself and propagating to other computers [31]. The first “computer virus“ was born and used ever since. Later on, in 1986, a software named Brain was considered to be the first malware in real world that have ever been observed [119].

There are four phases that a computer virus goes through when infecting a system: Dormant phase, Trigger phase, Propagation phase and Execution phase as describe below:

- Dormant phase: The dormant phase is in which the virus infiltrates into a system but does not function yet.
- Trigger phase: The virus needs a trigger for its initializing code in trigger phase. This trigger can be a pre-set timer or a condition changing events such as internal clock changing, or a user opens a file containing virus.
- Propagation phase: The propagation phase starts then as the virus replicates itself, and autonomously triggers new viruses, if there are.
- Execution phase: The virus performs its intended function. The virus can delete, beside itself, a number of local files, or attempt to slow down the system performance by consuming disk space and power with self propagation.

3.5.2 File infection techniques

Overwriting

Overwriting is a simple infection technique, in which the virus copies its code over the host computer system’s file data, thus destroying the original program [119].

Appending

In this case, the virus appends itself at the end of the original file. Technically, the virus inserts the malicious code at the end of the last section of the file. This technique allows the virus modifies the host file so that it is able to execute without cause damage to the host file [119]. Figure 3.2 illustrates a typical appending virus.

Prepending

In this technique, the virus inserts its malicious code right before the entry point of the host program. When the user executes the infected host program, the malicious code will run first before the original host code takes over. To do this, the virus executes the host file by copying the host file as a temporary file to disk and using a function call to run the host file [119]. Figure 3.3 illustrates the prepending technique.

Cavity

Cavity infectors typically overwrite part of files that contain zeros in binary files so-called caves of the file. Nevertheless, the other parts also can be leverage, such as $0xCC$ - filled blocks that C compilers often use for instruction alignment, or areas that contain spaces $0x20$. Cavity

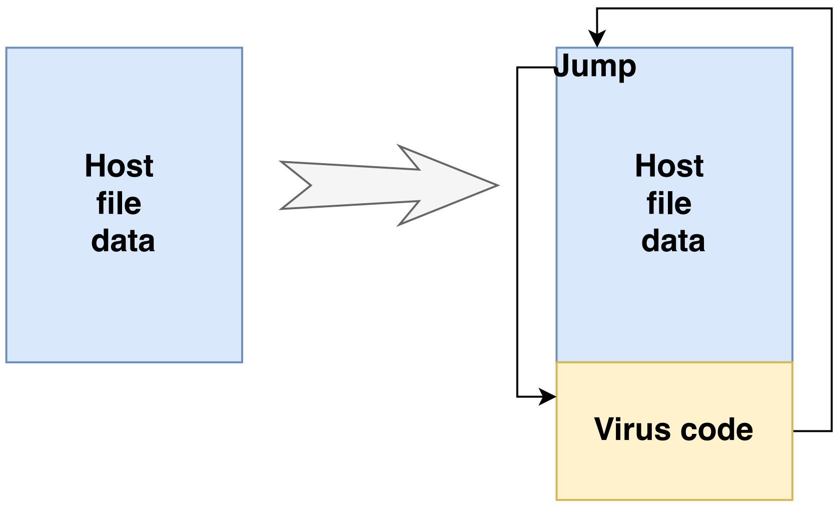


Fig. 3.2: A typical appending virus

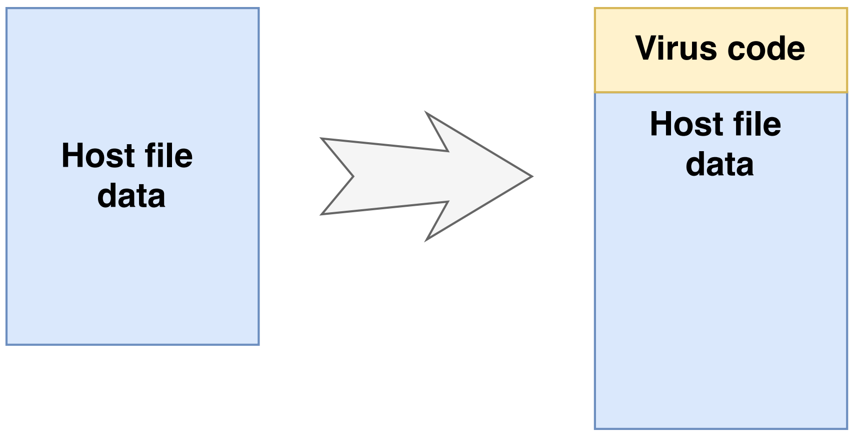


Fig. 3.3: A typical prepending virus

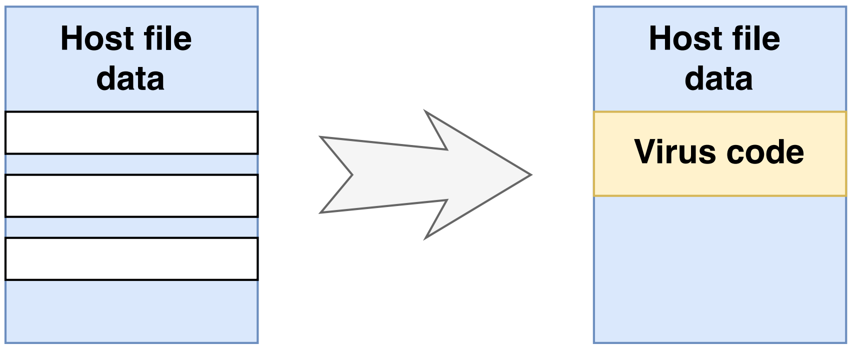


Fig. 3.4: Cavity technique

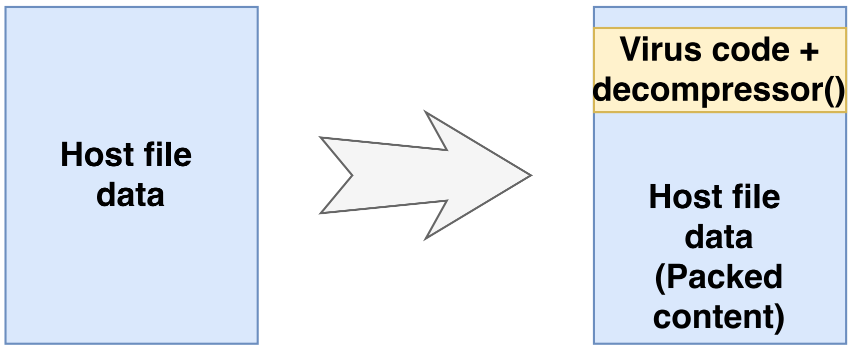


Fig. 3.5: Compressing technique

technique typically does not increase the size of the host file. [119]. Figure 3.4 shows the cavity technique.

Compressing

A particular virus infection technique that utilizes uses run-time packers to compress the host files code. Sometimes this technique is used to hide the increase in host size after infection by packing the host sufficiently with the binary packing algorithm. Figure 3.5 illustrates the compressing technique.

Ameoba infection technique

This technique splits the virus into two parts. The head part of the virus is prepended to the front of the file, and the tail part is appended to end of the file [119]. When executing, the head accesses to the tail and is loaded later. The original program is re-built as a new file for execution afterwards. Figure 3.6 shows the host program before and after infection by a virus that uses the Amoeba infection technique.

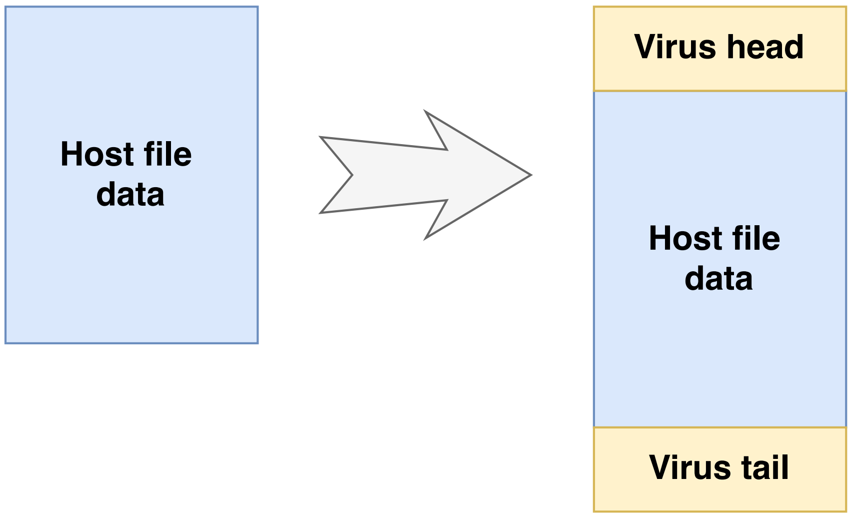


Fig. 3.6: Ameoba infection technique

Companion

A companion virus is a kind computer virus that does not modify any files but names itself similar to another program file that is frequently executed.

Entry Point Obscuring (EPO) technique

In this technique, the virus hide its entry point in order to avoid detection. Instead of taking control directly the host file, the virus patches the host file with a jump or call routine, and acquire the control that way.

Dll injection technique

This kind of infection happens by deceiving an application to call a malicious DLL file which then gets executed as part of the target process. The original content of the host file is not modified.

Embedded decryptor technique

This kind of virus embeds the decryptor to the host file, in which the entry point of the host is modified to point to the decryptor code. When the infected host executes, the virus code is decrypted and take control [119].

3.6 Artificial neural network

3.6.1 Overview

Artificial Neural Network (ANN), in short, Neural Network (NN) is a brain-inspired computational system that replicates the human brain in the learning process. An ANN connects neurons, technically termed, nodes, together to form a computational network, which is able to perform many sophisticated tasks. Information passed from nodes to nodes is evaluated with “weight“ multipliers assigned to the links between the nodes. The higher the weight factor, the more important the input or, the stronger the signal is. Weighted input is then fed to an activation function to generate a final value for the next corresponding node. Figure 3.7 shows a basic structure of a neural network.

ANN is well-known for its ability to handle with ease complex natural problems with large inputs, and deliver results with relatively high accuracy compared to many other methods utilised in ML. ANN works as an information managing model that imitates the biological nervous system function of a human brain. A human brain consists of a number of connected neurons forming a network that receives, processes and sends signals to perform human actions. There is a large amount of ANN applications in many specific computational areas ranging from human-like cognitive tasks such as three-dimensional object recognition, facial recognition, handwriting recognition, speech recognition, to more outreaching fields such as texture analysis, diagnosis of hepatitis, undersea mine detection, data recovery from faulty software in telecommunications, and many other applications. ANNs, like human brains, learn from examples or by trial and error mechanism. A human brain learning ability depends on adjusting the synaptic relationship between neurons, so does an ANN. For ANN learning paradigms, there are three training methods: supervised learning, unsupervised learning and reinforcement learning. Supervised learning proceeds a set of well-labelled input to create a mapping function for predicting a specified output. Unsupervised learning proceeds unlabeled input and produces unspecified output. Reinforcement learning utilises reward-penalty, trial and error mechanism. The system learns from feedbacks obtained from its interactions with the surrounding environment.

3.6.2 Multilayer perceptron

The multilayer perceptron (MLP) is a feed forward network with interconnected neurons, in which an input vector map with its specific output vector. The neurons, or nodes, are connected by weight and each node includes a nonlinear activation function. An MLP consists of one input layer, one or several hidden layers, and eventually an output layer. In a MLP, each node connects to every node in the next and previous layer, which leads to a fully connected network [49].

Through training, MLP can learn to solve specific problems. There are many algorithms that can be utilized to train an MLP. A common algorithms for training is the back-propagation algorithm [102], which proceeds in two phases :

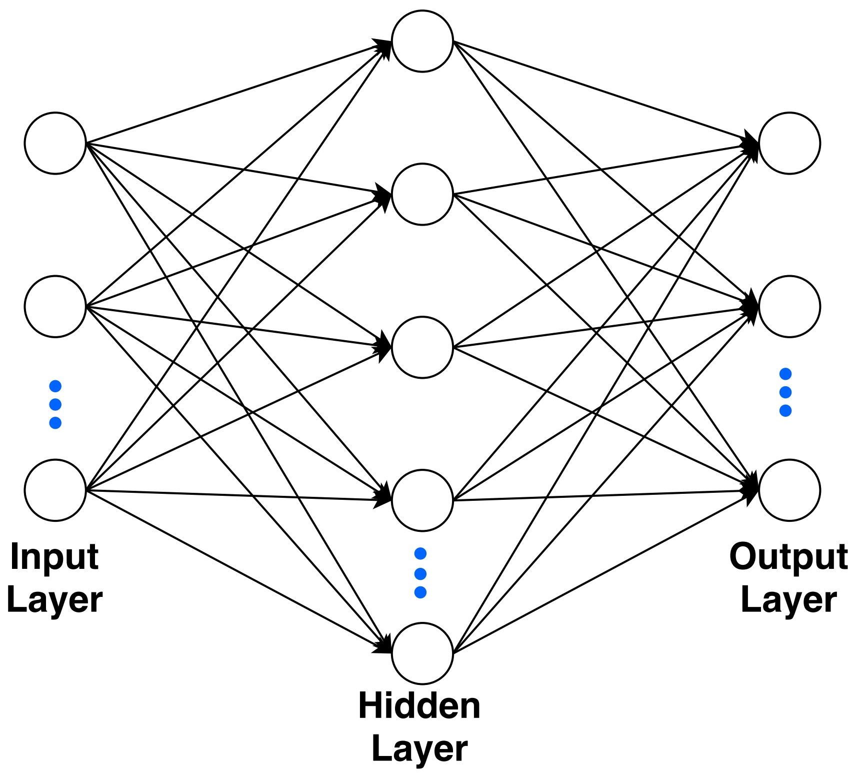


Fig. 3.7: Basic structure of a neural network

- Forward phase: the weights of the network are fixed and the input signal is propagated through the network until it reaches the output.
- Backward phase: The output of the network is compared with a target value to produce an error signal. The resulting error signal is propagated through the network in the backward direction. In the backward phase, continuous adjustments are made to the weights of the network.

Summarize of the back-propagation algorithm describes as following [49]:

1. Initialise network weights
2. Present input to the network
3. Calculate the output based on input
4. Comparing actual output to the target output to calculate error signal
5. Propagate error signal back to the network
6. Adjust weights to minimise overall error
7. Repeat steps 2–7 with next input until the error reaches a satisfactory level.

3.7 Network analysis and centrality measures

Network analysis provides a set of robust quantitative techniques to depict relations among actors and to examine the structure of the system. In the cybersecurity domain, network analysis used to study the processes that rule the propagation of the malware specimen. These analysis allows us not only make predictions about the behavior of advanced malware spread but it is also possible to test the effectiveness of control procedures or security countermeasures

These analysis techniques include the graph density and number of connected components, which describe the network as a whole. They also consist of vertex metrics such as degree centrality, betweenness centrality, eigenvector centrality, closeness centrality, and clustering coefficient that can be utilised to classify unique or essential factor within a network.

3.7.1 Degree centrality

The *degree centrality* of a vertex x is the number of its edges. The degree centrality values are normalized by the maximum possible degree in graph G. Hence, it is the fraction of edges that x is connected to. This metric indicates the probability that a message propagating in the network flows through vertex x, which actually reflects the intensiveness of network traffic. The *degree centrality* of vertex i in an undirected network is given by equation:

$$C_D(i) = \sum\nolimits_{j=1}^{n} A_{ij} \tag{3.1}$$

where n is the number of vertices, A_{ij} is adjacency matrix. If there is a link between i and j node, then $A_{ij} = 1$, and if opposite, $A_{ij} = 0$. Normalized version of degree centrality is:

$$C'_D(i) = \frac{C_D(i)}{n-1} \tag{3.2}$$

3.7.2 Closeness centrality

The *closeness centrality* of a vertex i is the inverse of the sum of the shortest paths between i to every other vertex in the network [48]. This metric reflects the speed at which the message spreads from the i to all other vertices in the network. Considering that the sum of distances depends on the number of vertices, the calculation needs to be normalised as denoted in equation 3.3.

$$C_C(i) = \frac{n-1}{\sum_{j=1}^{n-1} d(i,j)} \tag{3.3}$$

where n is the number of vertices, $d(y,v)$ is the shortest path between vertex x and y.

3.7.3 Betweenness centrality

The *betweenness centrality* quantifies the fraction of shortest paths between any couple of vertices passing through a vertex u [48]. The higher the betweenness, the higher the likelihood that the vertex will become a mediator in the data flow between the other nodes. Simply speaking, this centrality points out the vertex who influence the communication flow of the system, because more information will pass through that vertex. The *betweenness centrality* of an undirected network denotes by the following equation:

$$C_B(v) = \frac{2\sum\limits_{j<k} \frac{g_{jk}(i)}{g_{jk}}}{(N-1)(N-2)} \tag{3.4}$$

where N is the number of vertices, g_{jk} is the number of shortest paths between vertex j and k, and $g_{jk}(i)$ is the number of paths via vertex i among shortest paths between vertex j and k.

3.7.4 Eigenvector centrality

The *eigenvector centrality* metric takes into consideration not only the number of connection that a vertex has but also the significance of its neighbours. If a vertex has many connections to other vertices which are themselves well connected, then, it possesses high eigenvector centrality value. In the network, the vertices with high eigenvectors are considered to be more influential than other nodes. This metric can be utilised in measures of malware diffusion power. The equation of this metric is given in Eq. 3.5:

$$C_E(i) = \frac{1}{\lambda}\sum_j A_{ij} C_{E_j} \tag{3.5}$$

where A_{ij} is adjacency matrix of vertex i and j at the edge k, λ is the largest eigenvalue of the adjacency matrix, and C_{E_j} is the eigenvector centrality of vertex j.

3.7.5 Clustering coefficient

The *clustering coefficient* measures of the number of triangles in a graph. There are two versions of this metric: global and local. The global clustering coefficient measures the total number of closed triangles in a network. On the other hand, the local clustering coefficient indicates how close a member of a group is to all other nodes in the same group. The local clustering coefficient of a vertex i is defined as [133]:

$$C_i = \frac{2L_i}{k_i(k_i-1)} \tag{3.6}$$

where k_i is the degree of vertex i, L_i is the number of connections of the k_i neighbors of vertex i.

The clustering coefficient of entire graph is presented by the average clustering coefficient, $\langle C\rangle$. This metric calculates the probability that two neighbors of a randomly selected vertex connect to each other. The average clustering coefficient is defined as [133]:

$$\langle C\rangle = \frac{1}{N}\sum_{i=1}^{N} C_i \tag{3.7}$$

where N is the number of vertices.

4 STATE OF ART

4.1 Studies relevant to AI techniques in cybersecurity

While the concept AI was proposed in the 1950s, in recent years, it has grown at a significant pace and now influencing all aspects of community and occupations. Many areas benefit from AI, such as gaming, natural language processing, health care, manufacturing, education and others. This trend is also affecting the cybersecurity field where AI has been utilized to both attacking and defending in the cyberspace. On the offence side, cyber threats can employ AI to improve the sophistication and scope of their attacks. On the defence side, AI is utilized to enhance the defence strategies, so that the defence systems become more robust, flexible, and efficient, which can adaptively with the environment and decrease the impacts occurred.

Recently, researchers presented several literature surveys in the domain of AI and cybersecurity. However, some of them just focused on the adopting machine learning methods to cyber problems such as those in [22, 122, 56, 135]. Other research [16, 134] just focused on Deep learning methods. Additionally, there is a lack of literature dealing with the nefarious use of AI.

Apruzzese et al. [10] performed a survey on ML and DL methods for cyber security. Nevertheless, their research was just covering attacks related particularly to network intrusion detection, malware investigation, and spam identification.

The author in [75] discuss the intersection of AI and cybersecurity. More particular, the paper reviewed some ML and DL approaches to counter against cyber attacks. What is more, the author introduced the possibility of attacking the AI model. Nevertheless, the paper just discussed adversarial attacks and ignored other kinds of attack the AI model, such as poisoning data, extraction the model.

Another approach by the authors in [135] pointed out the differences between traditional ML and DL methods for cybersecurity. However, their survey just concentrated on intrusion detection.

4.2 Malware and advance techniques

The malware spreading has been a topic for numerous researches. Studies of its dynamic and behaviours when placed in real-world networks are among the main trends. For example, virus infection on computers was investigated by authors in [93] implementing epidemiologically compartmental models to identify potential contagious sources. Concurrently, another group, [111], introduced a moderate epidemiological model inspired by the fractional epidemiological model for descriptions of computer viruses with an arbitrary order derivative with a non-singular kernel.

Behaviours of malware are studied also in [74], [99], [95]. As in [74], the Susceptible Infected (SI) and the Susceptible Infected Recovered (SIR) model were combined and implemented to Barabasi-Albert network to study the influence of infection rates over virus propagation. In another research carried out by Parsaei et al. [95], the Lyapunov function and the Volterra-Lyapunov matrix properties are combined for building a computer virus propagation model. Researchers in [99] also built a virus propagation model utilizing Lyapunov function but in combination with Outh-Hurwitz criterion with a kill signal, namely SEIR-KS.

On the other hand, frameworks for malware evolution based on evolutionary computation for computer malware, in [83], [116], and for Android malware, by Noreen et al. [90], were proposed. Moreover, the researchers in [24] exploited the Evolutionary Algorithm (EA) to generate malware automatically. Kudo et al. in other research [71], proposed botnets with adopted ML techniques for prediction of system vulnerabilities and malware self-governed evolution.

Lately, the combination of malware and computational intelligence has been raised as a new research trend among scientists. Many researches such as [9], [136], [8], [63], and [7] focused on utilizing ML engines to evade anti-malware.

In another approach, Zelinka et al. [145] came up with a swarm virus prototype mimicking a swarm system behaviour. This paper is to outline a possible dynamics, structure, and behavior of a hypothetical (up to now) swarm malware as a background for a future anti-malware system. The research shows, how to capture and visualize the behavior of such malware when it walks through the operating system. The swarm virus prototype, designed here, mimics a swarm system behaviour and thus follow the main idea of a swarm algorithms. The information of its behavior is stored and visualized in the form of a complex network, reacting virus communication and swarm behaviour.

5 APPLICATIONS OF AI IN CYBER DOMAIN

This chapter presents a detailed survey on the application of AI in the cyber domain, which have already been presented and published at scientific conferences and journals [rel1, rel2, rel3, rel4, rel6].

5.1 AI methodology for cybersecurity

In this section, the author group give an overview of the learning algorithms, an essential concept of AI. Furthermore, we present some brief introduction about ML, DL and bio-inspired computation methods that frequently utilize in the area of cybersecurity.

5.1.1 Learning algorithms

AI is a branch of computer science that seeks to produce a new type of intelligent automaton that responds like human intelligence. To achieve this goal, machines need to learn. To be more precise, we need to train the computer by using the learning algorithms. Generally, learning algorithms help to enhance performance in accomplishing a task through learning and training from experience. There are currently three major types of learning algorithms which we use to train machines:

- *Supervised learning*: This type requires a training process with a large and representative set of data that has been previously labelled. These learning algorithms are frequently used as a classification mechanism or a regression mechanism.
- *Unsupervised learning*: In contrast to supervised learning, unsupervised learning algorithms use unlabeled training dataset. These approaches are often used to cluster data, reduce dimensionality, or estimate density.
- *Reinforcement learning*: Reinforcement learning is a type of learning algorithm that learns the best actions based on rewards or punishment. Reinforcement learning is useful for situations where data is limited or not given.

5.1.2 Machine learning methods

Machine learning (ML) is a branch of AI that aims to empower systems by utilizing data to learn and improve without being explicitly programmed. ML has strong ties to mathematical techniques that enable a process of extracting information, discovering patterns, and drawing conclusions from data. There are different types of the ML algorithm, but they can generally be classified into three main categories: supervised learning, unsupervised learning, and reinforcement learning. In the computer security domain, the standard ML algorithms are decision trees (DT), support vector machines (SVM), Bayesian algorithms, k-nearest neighbour (KNN), random forest (RF), association rule (AR) algorithms, ensemble learning (EL), k-means clustering, and principal component analysis (PCA).

5.1.3 Deep learning methods

Deep learning (DL) is a sub-field of ML, and it uses data to teach computers how to do things only humans were capable of before. Its motivation lies in the working mechanisms of the human brain and neurons for processing signals. The core of deep learning is that if we construct more extensive neural networks and train them with as much data as possible, their performance continues to increase.The most important advantage of DL over the conventional ML is its superior performance in large datasets. Similarly to ML methods, DL methods also have supervised learning, unsupervised learning, and reinforcement learning. The benefit of DL is the leverage of unsupervised learning to select feature automatically. The typical DL algorithms are frequently utilized in cybersecurity domain are: feed forward neural networks (FNN), convolutional neural networks (CNNs), recurrent neural networks (RNN), deep belief networks (DBNs), stacked autoencoders (SAE), generative adversarial networks (GANs), restricted Boltzmann machines (RBMs), and ensemble of DL networks (EDLNs).

5.1.4 Bio-inspired computation methods

Bio-inspired computation is a branch of AI, which emerged as one of the most studied during recent years. It is a collection of intelligent algorithms and methods that adopt bio-inspired behaviours and characteristics to solve a wide range of complex academic and real domain problems. Among many biological-inspired methods, the following techniques are most commonly used in cybersecurity domain: genetic algorithms (GA), evolution strategies (ES), ant colony optimization (ACO), particle swarm optimization (PSO), and artificial immune systems (AIS).

5.2 AI-based approaches for defending against cyberspace attacks

Recently, scientists proposed numerous techniques that have utilized AI methods to detect or categorize malware, detect network intrusions, phishing, spam attacks, counter Advanced Persistent Threat (APT) and identify Domain Generation Algorithms (DGAs). In this section, we category these literature into four main groups: malware identification; network intrusion detection; phishing and SPAM identification, and other, which compromise countering APT and identify DGAs. Fig. 5.1 illustrates the primary areas of utilising AI to cybersecurity.

5.2.1 Malware identification

Malware is a general term for many types of malicious software such as virus, worm, trojan horse, exploits, botnet, retrovirus and today is a popular method of cyber-attack. The malware's impact on digital society is enormous, so a considerable amount of research about adopting AI techniques has been done to prevent and mitigate malware. The most recent and noteworthy

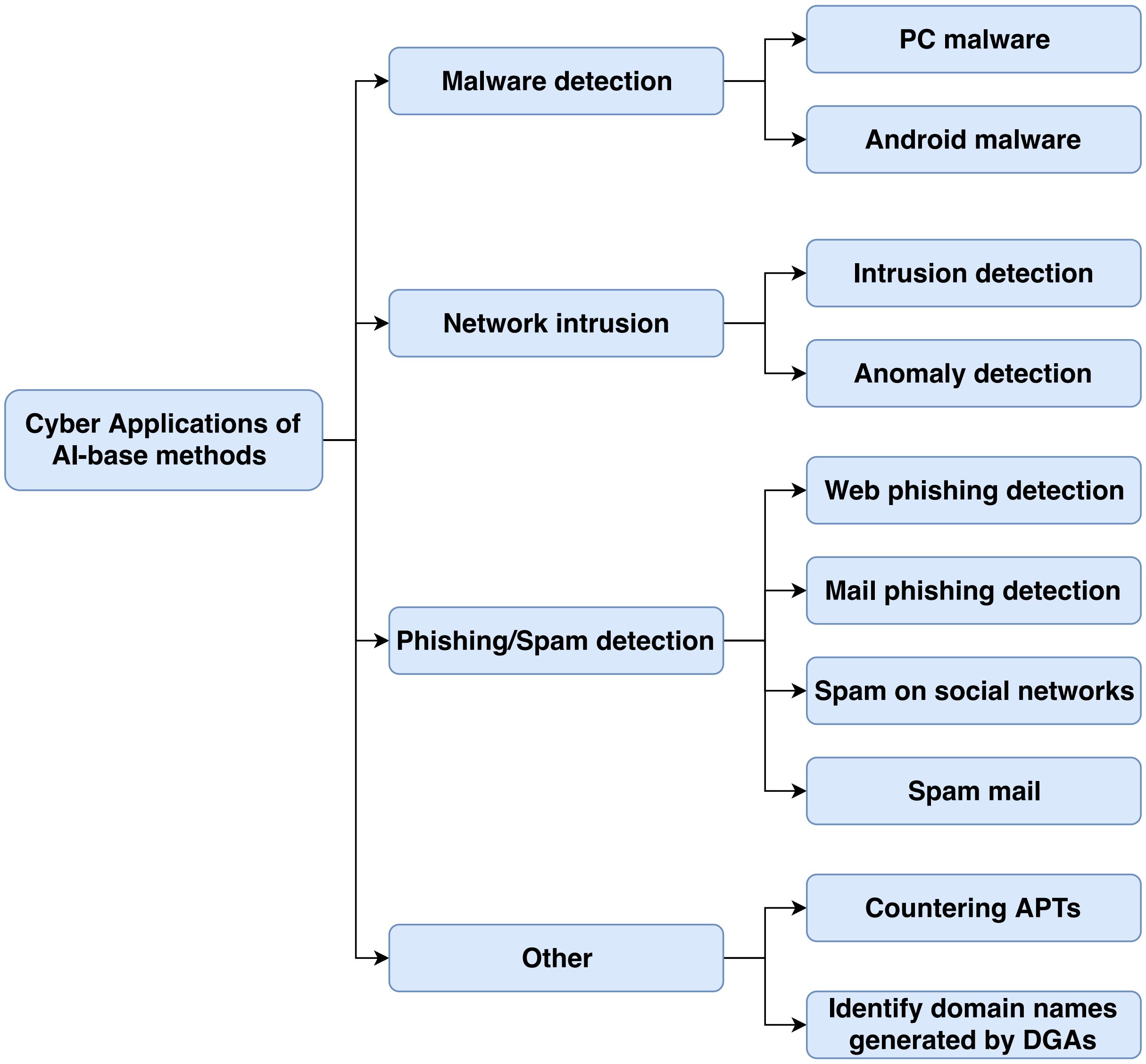


Fig. 5.1: Main branches of cybersecurity application adopting AI techniques

contribution works utilizing intelligent for malware detection, and prevention are described as follow.

In [137], the authors adopted ML to create an online framework for hardware-assisted malware detection based on virtual memory access patterns. The proposed method used logistic regression, support vector machine and random forest classifier and performed on the RIPE benchmark suite for the experiments. The authors reported that the framework has a true positive rate of 99% with less than 5% false positive rate. Meanwhile, the scholars in [28] presented a framework for classifying and detecting malicious software using data mining and ML classification. In this works, both signature-based and anomaly-based features were analyzed for detection. Experimental results showed that the proposed method outperformed other similar methods.

The authors in [59] utilized operational codes (OpCode) and k-nearest neighbors (KNN), support vector machine (SVM) as a ML classifiers to classify malware. The OpCode was represented as a graph and embedded into eigenspace, then particular or ensemble of classifiers were utilized to classify each vector as malware or benign. The empirical result showed that the proposed model is efficient with a low false alarm rate and high detection rate.

Later, Ye et al. [141] built a deep learning architecture for intelligent malware detection. In this work, they utilized an AutoEncoder stacked up with multilayer restricted Boltzmann machines (RBMs) to detect unknown malware. The author claimed that heterogeneous deep learning framework could improve the overall performance in malware detection compared with traditional shallow learning methods, deep learning methods.

A recent trend of research in malware detection focused on mobile malware in general and Android malware in particular. Machine learning, along with Deep learning, was a significant breakthrough in this area. In [82], a deep convolutional neural network (CNN) was adopted to identify malware. The raw opcode sequence from a disassembled program was used to classify malware. The authors in [76] utilized the Support vector machine (SVM) and the most significant permissions from all of the permission data to distinguish between benign and malicious apps. In [146], the authors presented novel ML algorithms, namely Rotation Forest for malware identity. Artificial neural network (ANN) and the raw sequences of API method calls were utilized in [67] to detect Android malware. A recent study by Wang et al. [131] introduced a hybrid model based on deep autoencoder (DAE) and convolutional neural network (CNN) to raise the accuracy and efficiency of large-scale Android malware detection.

Another research direction that attracted the attention of scientists was the used of bio-inspired methods for malware classify. These techniques were mainly used for features optimization and optimizing the parameter for the classifiers. For example, Particle Swarm Optimization (PSO) was adopted in [6], [2], [17]; the Genetic Algorithm (GA) was utilized in [4], [42] to enhance the effective of malware detection system. Table 5.1 abstracts some characteristics of the discussed malware identification approaches, concerning to the focus area, techniques, features, the dataset and validation metrics used to evaluate the model's performance. For the validation metrics, we present the best performing method in the paper.

Tab. 5.1: Selected literature of AI-based approaches in malware investigation

Ref.	Year	Focus	Tech.	Features	Dataset	Validation metrics
[137]	2017	PC malware	SVM,RF Logistic regression	MAP's feature sets	RIPE	DR: 99% FPR: 5%
[28]	2017	PC malware	BAM, MLP	N-gram, Windows API calls	Self collection: 52,185 samples	ACC: 98.6% FPR: 2%
[59]	2017	PC malware	KNN, SVM	OpCode graph	Self collection: 22,200 samples	ACC, FPR
[82]	2017	Android malware	CNN	Opcode sequence	GNOME, McAfee Labs	ACC: 98%/80%/87%, F-score: 97%/78%/86%
[6]	2017	Android malware	ANF, PSO	Permissions, API Calls	Self collection: 500 samples	ACC: 89%
[4]	2017	Botnet	C4.5, GA	Multi features	ISOT , ISCX	DR: 99.46%/95.58% FPR: 0.57%/ 2.24%
[141]	2018	PC malware	AutoEncoder, RBM	Windows API calls	Self collection: 20,000 samples	ACC: 98.2%
[76]	2018	Android malware	SVM, DT	Significant permissions	Self collection: 54,694 samples	ACC: 93.67% FPR: 4.85%
[146]	2018	Android malware	Rotation Forest	Permissions, APIs, system events	Self collection: 2,030 smaples	ACC: 88.26%
[67]	2018	Android malware	ANN	API call	Malgenome, Drebin, Maldozer	F1-Score: 96.33% FPR: 3.19%
[2]	2018	Android malware	PSO, RF, J48, KNN, MLP, AdaBoost	Permissions	Self collection: 8500 samples	TPR: 95.6% FPR: 0.32%
[131]	2019	Android malware	DAE, CNN	Permissions, filtered intents, API calls, hardware features, code related patterns	Self collection: 23000 samples	ACC: 98.5%/98.6% FPR: 1.67%/1.82%
[17]	2019	Android malware	PSO, Bayesnet, Naïve Bayes, SMO, DT, RT, RF J48, MLP	Permissions	UCI, KEEL, Contagiodump, Wang's repository	ACC: 79.4%/47.6%/ 82.9%/94.1%/ 100%/77.9%
[42]	2019	Android malware	SVM, ANN	App Components, Permissions	Self collection: 44,000 samples	ACC: 95.2%/96.6%

5.2.2 Intrusion Detection

An intrusion detection system (IDS) is a system that is supposed to protect the system from possible incidents, violations, or imminent threats. AI-based techniques are appropriate for developing IDS and outperform other techniques because of their flexibility, adaptability, fast calculation and learning ability. Hence, many researchers studied intelligent methods to improve the performance of IDS. The focus is on developing features optimization and improving the classifiers to reduce the false alarm. Some recent notable studies are listed as follows: Al-Yaseen et al. [3] combined support vector machine (SVM) and extreme learning machine with modified K-means as a model for IDS. Using KDD'99 Cup dataset [61], their model archived the result up to 95.75% accuracy and 1.87% of false alarms. Meanwhile, Kabir et al. [65] introduced a method for intrusion detection system based on sampling with Least Square Support Vector Machine (LS-SVM). The proposed methodology is validated through KDD'99 Cup dataset and obtained a realistic performance in terms of accuracy and efficiency.

The authors in [11] introduced a fuzziness based semi-supervised learning approach for IDS. In their work, they utilized unlabeled samples assisted with a supervised learning algorithm to enhance the performance of the classifier. The algorithm was tested on the KDD'99 Cup dataset and outperformed other comparative algorithms.

Later, Shone et al. [109] proposed a novel deep learning-based intrusion detection method called nonsymmetric deep autoencoder (NDAE). The authors used TensorFlow and evaluated their method by using KDD Cup '99 [61] and NSL-KDD [120] datasets. They have claimed that their model has achieved an accuracy of 97.85%.

Another approach using Genetic Algorithms (GA) and Fuzzy Logic for network intrusion detection is presented by Hamamoto et al. [58]. The GA is used to create Digital Signature of Network Segment using Flow Analysis (DSNSF), a prediction of the networks traffic behaviour for a given time interval. Additional, the Fuzzy Logic approach is adopted to assess whether an instance represents an anomaly or not. The evaluation was conducted by using real network traffic from a university and obtained an accuracy of 96.53% and false alarm of 0.56%.

One point to be taken into account is that the use of Swarm Intelligence (SI) for IDS. Botes et al. [21] presented a new method namely Ant Tree Miner (ATM) classified, which is is a decision tree using ACO instead of conventional techniques such as C4.5 and CART [92], for intrusion detection. Using NSL-KDD datasets, their approach archived the accuracy of 65% and false alarm rate 0%.

In a later study [118], the authors presented an IDS using binary PSO and kNN. The proposed method consists of feature selection and classification step. Based on the results obtained, the algorithm showed excellent performance, and the proposed hybrid algorithm raised the accuracy generated by KNN by up to 2%. Meanwhile, Ali et al. [5] introduced a learning model for fast learning network (FLN) based on PSO named PSO-FLN, and then the model had been utilized to the problem of IDS. The PSO-FLN model was tested on the KDD'99 Cup datasets and achieved the highest testing accuracy compared to other meta-heuristic algorithms. In the

Tab. 5.2: Selected literature focusing on network intrusion detection

Ref.	Year	Focus	Tech.	Anomaly types	Dataset	Validation metrics
[3]	2017	intrusion detection	SVM, K-means	DoS, Probe, U2R, R2L	KDD'99	ACC: 95.75%, FPR: 1.87%
[11]	2017	intrusion detection	NN with random weights	DoS, Probe, R2L, U2R	NSL-KDD	ACC: 84.12%
[21]	2017	intrusion detection	ACO, DT	DoS, Probe, R2L, U2R	NSL-KDD	ACC: 65%, FPR: 0%
[118]	2017	intrusion detection	PSO, KNN	DoS, Probe, R2L, U2R	KDD'99	ACC: - Dos: 99.91% - Probe: 94.41% - U2L: 99.77% - R2L: 99.73%
[65]	2018	intrusion detection	LS-SVM	DoS, Probe, U2R, and R2L	KDD'99	ACC: Over 99.6%
[109]	2018	intrusion detection	DAE, RF	DoS, Probe, R2L, U2R	KDD'99, NSL-KDD	Average ACC: 85.42% - 97.85%
[58]	2018	Anomaly detection	Fuzzy logic, GA	DoS, DDoS, Flash crowd	University dataset	ACC: 96.53%, FPR: 0.56%
[5]	2018	Intrusion detection	PSO, FLN	DoS, Probe, R2L, U2R	KDD'99	ACC: - Dos: 98.37% - Probe: 90.77% - U2L: 93.63% - R2L: 63.64%
[50]	2018	Anomaly detection	CSO, K-means	DoS, Probe, R2L, U2R	UCI-ML, NSL-KDD	ACC: 97.77%, FPR: 1.297%
[57]	2018	Intrusion detection & classification	ABC, AFS	DoS, Probe, R2L, U2R, Fuzzers, Analysis, Exploits, Generic, Worms, RA, Shellcode, Backdoors	NSL-KDD, UNSW-NB15	ACC: 97.5%, FPR: 0.01%
[27]	2019	anomaly detection	PSO, SVM, K-means, AFS	DoS,Probe, R2L, U2R, RA, RI, CI	KDD'99, Gas Pipeline	ACC: 95%
[51]	2019	Anomaly detection	GWO, CNN	DoS, Probe, U2R, R2L	DARPA'98, KDD'99	ACC: 97.92%/98.42% FPR: 3.6%/2.22%
[68]	2019	anomaly &misuse detection	Spark ML, LSTM	DSoS, DoS, Botnet, Brute Force SSH	ISCX-UNB	ACC: 97.29% FPR: 0.71%
[105]	2019	Anomaly detection	FA, C4.5, Bayesian	DoS, Probe, U2R, R2L	KDD'99	DoS(ACC: 99.98%, FPR: 0.01%) Probe(ACC: 93.92%, FPR: 0.01%), R2L(ACC: 98.73%, FPR: 0%), U2R(ACC: 68.97%, FPR: 0%)
[55]	2019	Intrusion dectection	Tabu search, ABC, SVM	DoS, Probe, U2R, R2L	KDD'99	ACC: 94.53%, FPR: 7.028%

recent study by Chen et al. [27], a multi-level adaptive coupled intrusion detection method combining white list technology and machine learning was presented. The white list was used to filter the communication, and the machine learning model was used to identify abnormal communication. In this article, the adaptive PSO algorithm and Artificial Fish Swarm (AFS) algorithm were used to optimize the parameters for the machine learning model. The method was tested on KDD'99 Cup [61], Gas Pipeline, and industrial field datasets. The empirical result showed that the proposed model is efficient with various attack types.

In [50], the authors introduced the Fuzzified Cuckoo based Clustering Technique for anomaly detection. The technique consists of two phases: the training phase and the detection phase. In the training phase, Cuckoo Search Optimization (CSO), K-means clustering, and Decision Tree Criterion (DTC) were combined to evaluate the distance functions. In the detection phase, a fuzzy decisive approach was utilized to identify the anomalies based on input data and previously computed distance functions. Experimental results showed that the model was effective with an accuracy rate of 97.77% and a false alarm rate of 1.297%.

Meanwhile, the authors in [57] incorporated Artificial Bee Colony and Artificial Fish Swarm algorithms to cope with the complex IDS problems. In this work, a hybrid classification method based on the ABC and AFS algorithms was proposed to improve the detection accuracy of IDS. The NSL-KDD and UNSW-NB15 datasets were used to evaluate the performance of the method. Based on the results obtained, the proposed model was efficient with a low false alarm rate and high accuracy rate.

In later research, Garg et al. [51] proposed a hybrid model for network anomaly detection in cloud environments. The model utilized Grey Wolf Optimization (GWO) and Convolutional Neural Network (CNN) for feature extraction and identifying the anomalies on real-time network traffic streams. The empirical result showed that the proposed model was efficient with a low false alarm rate and high detection rate.

Another approach [68] presented a hybrid IDS utilizing Spark ML and the convolutional-LSTM network. The ISCX-UNB dataset was used to evaluate the performance of the method. Based on the results obtained, the proposed model obtained a significant result and outperformed the compared method.

In Ref. [105], the authors adopted the firefly algorithm for feature selection and C4.5, Bayesian Networks classifier for detection network intrusion. The proposed approach was tested on the KDD'99 Cup dataset, and obtained a promising result and outperformed the compared method for feature selection.

Recently, research conducted by Gu et al. [55] introduced an IDS based on SVM with the Tabu-Artificial Bee Colony for feature selection and parameter optimization simultaneously. The main contributions of their work included the adopting of Tabu Search algorithm to improve the neighbourhood search of ABC so that it could speed up the convergence and prevented stuck in the local optimum. According to their experiments, although the accuracy rate was high 94.53%, the false alarm rate was 7.028%.

Table 5.2 abstracts some characteristics of the discussed network intrusion detection approaches, concerning to the focus area, techniques, features, the dataset and validation metrics used to evaluate the model's performance. For the validation metrics, we present the best performing method in the paper.

5.2.3 Phishing and SPAM detection

Phishing attack is a cyber-attack that attempts to steal user's identity or financial credentials. Today, phishing attacks are one of the most menacing threats on the Internet. Various novel intelligent approaches were used to cope with these problems.

The authors in [112] presented a phishing detection scheme called Phishing Email Detection System (PEDS), which joined the evolving neural network and reinforcement learning. Their model obtained 98.6% of accuracy rate and 1.8% false positive rate.

The authors in [64] introduced an anti-phishing method, which utilized several different ML algorithms and nineteen features to distinguish phishing websites from legitimate ones. The authors claimed that their model achieved 99.39% true positive rate.

Another approach by Feng et al. [43], applied neural network for identification the phishing web sites by adopting the Monte Carlo algorithm and risk minimization principle. Empirical results showed that their model reached to 97.71% precise detection rate and 1.7% false alarm rate.

A recent study conducted by [104] introduced a real-time anti-phishing system, which utilized seven different classification algorithms and natural language processing (NLP) based features. According to the authors, their approach obtained a promising result with a 97.98% accuracy rate.

Another study [79] built a stacking model by combining GBDT, XGBoost and LightGBM using URL and HTML features for classifying the phishing web pages. The authors reported that their approach reached to 98.60% accuracy rate.

The terminology "SPAM" refers to unsolicited bulk email (junk email). Spam email may lead to security issues and inappropriate contents. To overcome the drawbacks of this cyber-threats, recently scientists applied various novel intelligent techniques to build spam filter systems.

Feng et al. [44] combined support machine vector and Naive Bayes to develop a spam filtering system. The proposed system was evaluated by DATAMALL dataset and obtained a great spam-detection accuracy.

The authors in [72], designed a spam categorization technique using a modified cuckoo search to enhance the spam classification. In their work, the step size-cuckoo search was utilized for feature extraction, and the SVM was used for classification. The proposed approach was tested on two spam datasets: Bare-ling, Lemm-ling, and obtained a competitive result.

Later, research conducted by [113] proposed a system to filter the spam message of Facebook using SI-based and machine learning technique. The PSO algorithm was adopted for feature selection and the SVM, decision tree for classification. The authors claimed that the proposed system was efficient. Unfortunately, the details of the results were not provided.

Tab. 5.3: Selected literature focusing on phishing and spam identification

Ref.	Year	Focus	Tech.	Features	Dataset	Validation metrics
[44]	2016	Spam detection	Naive Bayes, SVM	99 features	DATAMALL	Not provide
[72]	2017	Spam classification	CSO, SVM	101 features	Ling-spam corpus	ACC: 87%/88%
[112]	2018	Mail phishing detection	NN, RL	50 features	Self collection: 9900 samples	ACC: 98.6%, FPR: 1.8%
[64]	2018	Website phishing detection	RF, SVM, NN, logistic regression, naïve Bayes	19 features	Phishtank, Openphish, Alexa, Payment gateway, Top banking website	ACC: 99.09%
[43]	2018	Website phishing detection	NN	30 features	UCI repository phishing dataset	ACC: 97.71%, FPR: 1.7%.
[113]	2018	Spam message detection	PSO, DE, DT DB index, SVM,	13 features	Self collection: 200,000 samples	Not provide
[12]	2018	Spammer detection	LFA, FCM	21 features	Self collections: 14,235 samples	ACC: 97.98%
[104]	2019	Website phishing detection	Naive Bayes, KNN, Adaboost, K-star, SMO, RF, DT	104 features	Self collection: 73,575 samples	ACC: 97.98%
[79]	2019	Website phishing detection	GBDT, XGBoost, LightGBM	20 features	Self collection: - 1st: 49,947 samples - 2nd: 53,103 samples	ACC: 97.30%/98.60% FPR: 1.61%/1.24%
[41]	2019	spam detection	GA, RWN	140 features	Spam Assassin, LingSpam, CSDMC2010	ACC: 96.7%/93%/90.8%

Recently, Aswani et al. [12] provided a hybrid approach for detecting the spam profiles on Twitter using social media analytics and bio-inspired computing. Specifically, they utilized a modified K-Means integrated Levy flight Firefly Algorithm (LFA) with chaotic maps to identify spammers. A total of 14,235 profiles was used to evaluate the performance of the method. The empirical result showed that the proposed model was efficient with an accuracy of 97.98%.

A recent study conducted by Faris et al. [41] presented an email spam detection and identification system based on Genetic Algorithm (GA) and Random Weight Network (RWN). According to the experiments, the proposed system obtained remarkable results in terms of accuracy, precision, and recall.

5.2.4 Other: counter APT and identify DGAs

Advanced Persistent Threat

Advanced Persistent Threat (APT) is a sophistication cyber-attack that uses advanced techniques to exploit sensitive data and remain undetected. The attackers often focus on valuable targets, such as large corporation's security agencies and government organizations, with the ultimate goal of long - term information stealing. To defend against APT attacks, scholars proposed a variety of AI techniques to deal with these cyber-threats.

In [87], the authors applied a decision tree to build IDS to detect APT attacks. It can detect intrusion from the beginning and quickly react to APT to minimize damage. Empirical results showed that the proposed system achieved a high rate of APT detection. Meanwhile, Sharma et al. [108] presented a framework architecture for the detection of APTs, which based on multiple parallel classifiers. According to the authors, the proposed framework achieved great effectiveness and accuracy.

The authors in [101] investigated how deep neural networks (DNN), which used raw features of dynamic analysis could be employed for nation-state APT attribution. Evaluated with the training set contained 3200 samples, the proposed approach reached an accuracy of 94.6%.

Burnap et al. [23] used machine activity metrics and self-organizing feature map approach to distinguish legitimate and malicious software. The authors reported that their method showed promising for APT detection.

Another approach [52] introduced a ML-based approach named MLAPT to identify and predict APTs. According to the authors, their system had the ability of early prediction of APT attacks. The experiments showed that MLAPT had a true positive rate and the false positive rate with 81.8% and 4.5% respectively.

Table 5.3 exhibits some characteristics of the discussed phishing and spam detection approaches, concerning to the focus area, techniques, features, the dataset and validation metrics used to evaluate the model's performance. For the validation metrics, we present the best performing method in the paper.

Identify domain names generated by DGAs

Domain Generation Algorithms (DGAs) are algorithms that use to create an immense number of pseudo-random domain names to hide the operator's command and control (C&C) server and evade detection. Lison et al. [80] adopted recurrent neural networks (RNN) to identify domain names generated by DGAs with high precision. According to the authors, the model could detect 97.3% of malware-generated domain names with a low false positive rate. Curtin et al. [33] also took a similar approach using the generalized likelihood ratio test (GLRT) and achieved promising results.

Yu et al. [142] performed a comparative analysis on convolutional neural network (CNN) and recurrent neural network (RNN) based architectures and tested on the dataset with 2 million domain names. The authors reported that all comparative models performed well with high accuracy rate and low false positive rate.

The authors in [124] introduced a novel Long Short-Term Memory network (LSTM) based algorithm to handle the multiclass imbalance problem in DGA malware detection. Based on the results obtained, the proposed algorithm provided an improvement as compared to the original LSTM.

In a recent study [132], the authors utilized IF-TF for a DGA and DNS covert channel detection system based machine learning. According to the authors, the proposed approach achieved outstanding accuracy at 99.92%. Another approach in [138], proposed a framework for identification word-based DGAs by utilizing the frequency distribution of the words and an ensemble classifier constructed from Naive Bayes, Extra-Trees, and Logistic Regression. The authors reported that their method outperformed the comparative researches. Table 5.4 describes the main details of the selected researches focusing on APTs detection and identifying domains generated by DFGAs, concerning to the focus area, the algorithms, the dataset, and the evaluation measures.

5.3 The malicious use of AI

Regarding the fact that AI tools already developed in open source, it is logical to expect that, AI technologies may be leveraged for creating new types of advanced and sophisticated threats. In this section, we illustrate a range of feasible uses toward which AI could be put for nefarious ends. Some of them are already occurring in the limited form in practice but could be scaled up or strengthen with further technological advances in the future [121]. Fig. 5.2 highlights some branches of leveraging of AI for malicious activities.

Tab. 5.4: Selected studies focusing on APTs and DGAs domains detection

Ref.	Year	Focus area	Tech.	Features	Dataset	Validation metrics
[87]	2017	APTs detection	DT	API calls	Self collection: 130 samples	ACC: 84.7%
[108]	2017	APTs detection	GT, DP, CART, SVM	Log events	Self collection	ACC: 98.5%, FPR: 2.4%
[101]	2017	nation-states APTs detection	DNN	Raw text	Self collection: 3200 samples	ACC: 94.6%
[80]	2017	DGA domains detection	RNN	Letter combinations	Self collection: over 2.9 million samples	ACC: 97.3%
[23]	2018	APTs detection	SOFM, DT, Bayesian, SVM, NN	Machine activity metrics	Self collection: 1188 samples	ACC: 93.76%
[52]	2018	APTs detection and prediction	DT, KNN, SVM, EL	Network traffic	Self collection, university live traffic	ACC: 84.8%, FPR: 4.5%
[33]	2018	DGA domains detection	RNN	Characters	Self collection: 2.3 million samples	FPR: <=1%
[142]	2018	DGA domains detection	RNN, CNN	Strings	Self collection: 2 million samples	ACC: 97–98%
[124]	2018	DGA botnet detection	LSTM	Characters	Alexa, OSINT	F1:98.45%
[138]	2019	DGA detection	Ensemble	words	Self collection: 1 million samples	ACC: 67.98%/ 89.91%/91.48%
[132]	2019	DGA, DNS covert chanel detection	TF-IDF	Strings	Self collection: 1 million samples	ACC: 99.92%

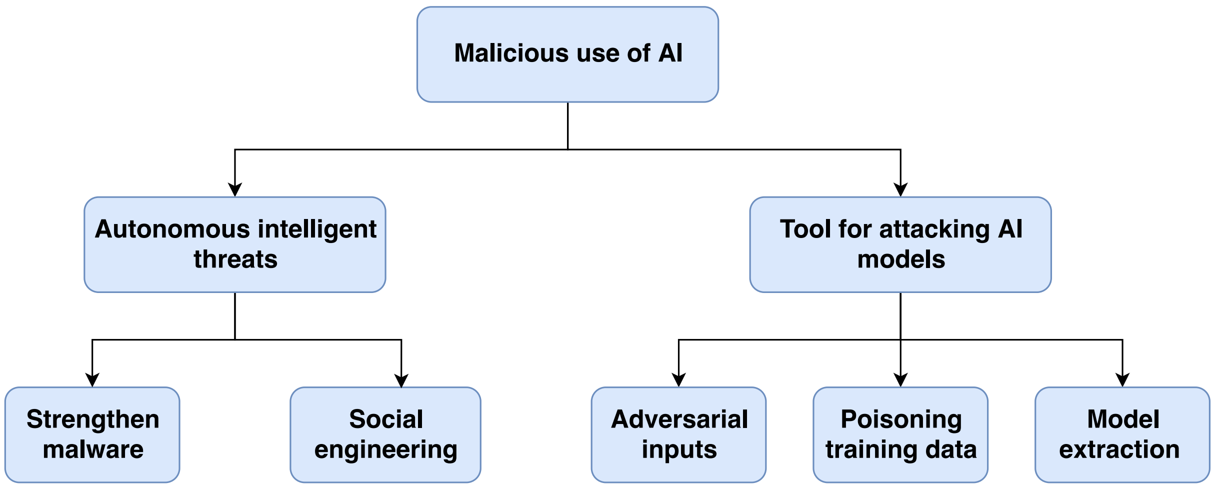


Fig. 5.2: The use of AI for malicious activities in cybersecurity

5.3.1 AI and autonomy intelligent threats

AI-powered malware

AI technologies can further be weaponized to increase the effectiveness of the malware, making it more autonomous, sophistication, increase in speed, and hard to detect. With the support of AI, the new generation of malware become smarter and capable to operate autonomously. The intelligent malicious programs can self-propagate in a network or computer system base on a sequence of autonomous decisions, intelligently custom-made to the parameters of the host system and. Autonomous malware capable of choosing the lateral movement techniques, thus increasing the likelihood of fully compromising the targeted networks.

What is more, malware authors could adopt the ability to adapt to a new environment or to use the knowledge acquired from past occurrences of AI in creating intelligent viruses and malware or modelling adaptable attacks. Consequently, malware becomes independent, integrating into its environment, taking countermeasures against security tools and could leverage data acquired from the past to attack the system.

One of the ultimate goals of malware is to hide their presence and malicious intent to avoid being detected by anti-malware solutions. Cybercriminals will certainly discover ways to implement the most advanced technology into evasive techniques.

The researchers from IBM [115] presented malware enhanced by the Deep learning (DL) technique that was capable of leveraging facial recognition, voice recognition, and geolocation to identify its target before for attacking.

In [100] Rigaki and Garcia adopted DL techniques to generate malicious malware samples that avoid detection by simulating the behaviours of legitimate applications.

Concurrently to the development of malware, there are attempts to apply bio-inspired techniques into malware. For instance, Ney ea al. [89] presented how to compromise a computer by encoding malware in a DNA sequence. Later, the authors in [145] outlined a hypothetical swarm malware as a background for a future anti-malware system. More precise, the swarm virus prototype simulated a swarm system behaviour, and its information was stored and visualized in the form of a complex network. As a further improvement, the authors in [126] fused swarm base intelligence, neural network, and a classical computer virus to form a neural swarm virus.

AI used in social engineering attacks

AI can be leveraged to mine large amounts of big datasets containing social network data to extract personally identifiable information, which can be used for compromising user accounts. What is more, based on user information, malicious actors could adopt AI to generate custom malicious links or creating personalized phishing emails automatically.

There have been researches on adopting AI to carry out complex social engineering attacks. In [106, 107], the authors introduced a long short-term memory (LSTM) neural network that was trained on social media posts to manipulate users into clicking on deceptive URLs.

5.3.2 AI as a tool for attacking AI models

As AI is being integrated into security solutions, cybercriminals attempt to exploit vulnerabilities in this domain. Attacks on AI systems are typically discussed in the context of adversarial machine learning. The offences on AI systems often appeared in three areas:

- *Adversarial inputs*: This is a technique where malicious actors design the inputs to make models predict erroneously in order to evade detection. Recent studies demonstrated how to generate adversarial malware samples to avoid detection. [54] crafted adversarial examples to attack the Android malware detection model. Meanwhile, scholars in [63] presented a generative adversarial network (GAN) based algorithm called MalGAN to craft adversarial samples, which capable of bypassing black-box machine learning-based detection models. Another approach by Anderson et al. [9] adopted GAN to create adversarial domain names to avoid the detection of domain generation algorithms. The authors in [69] investigated adversarial generated methods to avoid detection by DL models. Meanwhile, in [7], the authors presented a framework based on reinforcement learning for attacking static portable executable (PE) anti-malware engines.
- *Poisoning training data*: In this kind of attack, the malicious actors could pollute the training data from which the algorithm is learning in such a way that reduces the detection capabilities of the system. Different domains are vulnerable to poisoning attacks, for example, network intrusion, spam filtering or malware analysis [77],[26].
- *Model extraction attacks*: These techniques are used to reconstruct the detection models or recover training data via black-box examining [123]. On this occasion, the attacker learns

how ML algorithms work by reversing techniques. From this knowledge, the malicious actors know what the detector engines are looking for and how to avoid it.

Table 5.5 describes the main details of the selected studies focusing on malicious use of AI, with regard to the focus area, the techniques, the innovation point and the main idea.

5.4 Challenges and open research directions

In this section, we discuss the challenges when adopting AI-based approaches in practice. Additionally, we also offer a vision about some areas that need to be further researched.

5.4.1 Challenges

AI methods have played a crucial role in cybersecurity applications and will continue a promising direction that attracts investigations. However, some issues must be considered when applying AI-based techniques in cybersecurity. First, the accuracy of AI models is a significant barrier. Specifically, the false alarms may cause waste of time triaging, or an AI system might miss a cyberattack entirely. Another barrier to adoption is that many of the approaches proposed today are model-free methods. These models require a large quantity of training data, which are hard to obtain in real cybersecurity practice. Next, in designing AI-based solutions for cybersecurity, approaches need to consider the adversary. Adversary attacks are hard to detect, prevent and counter against as they are part of a battle between AI systems.

AI can help protect the system against cyber-threats but can also facilitate dangerous attacks, i.e., AI-based attack. Malicious actors can leverages AI to make attacks flexible and more sophisticated to bypass detection methods to penetrate computer systems or networks.

5.4.2 Open research directions

There are diverse promising and open topics for incorporating AI techniques and cybersecurity. Some research area is as follows.

First, the combination of several AI-based techniques in a defence solution may still an interesting research direction. For example, the incorporate between bio-inspired computation and ML/DL approaches shown the promising results in malware detection [6], [2], [17], [4], [42] or [68], [105], [55] in detect the network intrusion. Hence, the combination of these two techniques is a very potential research direction due to the number of bio-inspired algorithms exploited in cybersecurity is still limited.

Second, the corporation between a human intellect with machines for cyber defences also needs further investigation. In this human-machine model, the agents will autonomously execute the task whilst humans can supervise and intervene only when necessary.

Third, outstanding literature has proved that the threat actors could utilize the AI-based method to bypass or attack the AI models such as in [9], [123], [54], [63],[69],[26], [25]. Hence, the defence strategy against these types of attacks would be an inevitable trend in the future.

Tab. 5.5: Selected references in term of the malicious use of AI

Ref.	Year	Focus	Tech.	Innovation point	Main idea
[9]	2016	Adversarial attacks	GAN	New attack model	create adversarial domain names to avoid the detection of domain generation algorithms
[123]	2016	Stealing model	AE	model extraction attacks	extract target ML models by the machine learning prediction APIs
[106]	2016	Social engineering attacks	RNN	New attack model	Automated spear phishing campaign generator for social network
[89]	2017	Compromise computer	Encoding DNAs	Encoding malware to DNAs	compromise the computer by encoding malware in a DNA sequence
[54]	2017	Adversarial attacks	AE	New attack algorithm	adversarial attacks against deep learning based Android malware classification
[63]	2017	Adversarial attacks	GAN	New attack model	present a GAN based algorithm to craft malware that capable to bypass black-box machine learning-based detection models
[115]	2018	Malware creation	DNN	AI-powered malware	Leverage deep neural network enhance malware, make it more evasive and high targeting
[100]	2018	Malware creation	GAN	AI-powered malware	avoid detection by simulating the behaviours of legitimate applications
[145]	2018	Malware creation	ACO	SI-based malware	use ACO algorithms to create a prototype malware that have a decentralize behaviour
[7]	2018	Adversarial attacks	AL	New attack method	a generic black-box for attacking static portable executable machine learning malware models
[69]	2018	Adversarial attacks	AM	New attack algorithm	adversarial generated methods to attack neural network-based malware detection
[77]	2018	Poisoning attack	EPD	New poisoning data method	present a novel poisoning approach that attack against machine learning algorithms used in IDSs
[26]	2018	Poisoning attack	AM	Analysis poisoning data method	present three kind of poisoning attacks on machine learning-based mobile malware detection
[107]	2018	Social engineering attacks	LSTM	New attack model	introduced a machine learning method to manipulate users into clicking on deceptive URLs
[126]	2019	Malware creation	ANN	next generation malware	fuse swarm base intelligence, neural network to form a new kind of malware

Another aspect that captures further investigation is the use of AI in malware, such as in [115], [145], [126]. Specifically, the combination of swarm communication and other AI-based techniques need to be noted. Such malware will exhibit extremely high robustness of information preservation against swarm network damage. Swarm communication also exposes the research direction to apply this idea to other malware like worms, trojans, or ransomware so that their activities can be more distributed and stealth.

5.5 Discussion

The utilize of AI in cybersecurity creates new frontiers for security investigate. Scientists views AI as an essential response to the continuous growth in the number and increase the complexity of cyber-threats and the need for quick reaction and substantially automatic responses to security attacks. On the other hand, AI technology also leads to some security issues that need to be resolved. In this section, we summarize the essential points in this study. Other methods to enhance cybersecurity are also mentioned. To conclude, the authors compared this study with several existing surveys.

It is clear from the literature that AI-based approaches could be adopted in the cyber domain, encompassing a variety of methods that have developed over many decades, have demonstrated effectiveness, and are currently in use.

At present, the prime targets for AI applications are malware classification and analysis, intrusion detection (focusing on anomaly network-based attacks), phishing and spam, advanced persistent threat detection and characterization. Furthermore, a rapidly emerging topic for application is automated vulnerability testing and intrusion resistance.

Intrusion detection systems typically rely on hybridization techniques that combine several methods: signature-based methods for rapid detection of known threats with low false alarm rates and anomaly-based methods to flag deviations. What is more, another trend is combining with other computational intelligent models such as ACO, PSO.

The absence of datasets for research and development in network intrusion is a problem. Precisely, publicly available datasets are extremely dated, such as DARPA (1998), KDD (1999), and NSL-KDD (2009), and the characteristics and volume of attacks have significantly changed since that time. What is more, the majority use of these datasets may offer a one-sided vision about collected data and not reflect real-world situations.

There are indications that AI-based models can be bypassed. Several published examples in the cybersecurity field indicate that the AI system can be challenged with the adversarial inputs or poisoning the training data. Furthermore, the potential threats of malicious use of AI need to be taken into account. For example, AI technology can be utilized to power malware, establish a spear-phishing campaign, or perform a social engineering attack.

5.6 Summary

This chapter focuses on the application of the AI-based technique in cybersecurity issues. Specifically, we present the application of AI in malware detection, intrusion detection, APT, and other domains such as spam detection, phishing detection. Furthermore, our study offers a vision of how AI could be adopted for malicious use.

In contemporary research, the primary targets for AI application in cybersecurity are network intrusion detection, malware analysis and classification, phishing, and spam emails. In those areas, the adoption of DL gradually become the primary trend. Furthermore, the combination of other intelligent techniques, such as bio-inspired methods, together with ML / DL, also attracted the attention of researchers. Such combinations yield very promising results and continue a trend for further research.

Although the role of AI in resolving cybersecurity matters continues being researched, some of the problems that exist around the deployment of AI-based defences are also striking. For instance, the adversarial attack against the AI models or the emergence of autonomous intelligent malware. Hence, research on discovering solutions to these threats should be further explored.

6 X-WARE: THE SWARM MALWARE

In chapter 5 we focus on the application of the AI-based technique in cybersecurity issues. Furthermore, we predict that in upcoming time, the cyber-threat actors will leverage the AI technology in malicious intention, for instance in malware.

This chapter described the methodology of developing a prototype of the hypothesis X-Ware, which is an improvement of the swarm virus introduced in [145]. In this study, the authors outlined a possible dynamics, structure, and behavior of a hypothetical (up to now) swarm malware as a background for a future anti-malware system. The research shows, how to capture and visualize the behavior of such malware when it walks through the operating system. The swarm virus prototype, designed here, mimics a swarm system behaviour and thus follow the main idea of a swarm algorithms.

The content in this chapter have already been presented and published at scientific journals [145] and conferences [110], [126].

6.1 Methodology

6.1.1 X-Ware design

Malware has evolved drastically since its early days being an unharmful annoyer. Various techniques have been adopted to virus development such as oligomorphism, metamorphism, polymorphism, encryption, armouring (armoured viruses) [46] and obfuscation [98] to defeat anti-malware. One of the latest modern viruses, Stuxnet, [73], is controlled with command and control (C&C) infrastructure method. C&C has a weak spot of being immobilized when its control centre is demolished.

Acting as a malware developer, in order to eliminate the weak spot of the botnet structure being its C&C centre, the author group adopted swarm-based intelligence and NN to a traditional virus to produce a new malware named X-Ware.

Technically, this virus consists of instances (many individuals) forming a swarm (population) that propagate in a computer file system. The individuals in the swarm communicate via command lines (when shifting from file to file) and among themselves. The global data is stored inside each virus so that the swarm can be expected to perform decentralized behaviour. Whenever there is a change in the swarm (e.g., the virus moves to another host), the information will be updated to every individual in the pack.

Furthermore, when a member of the swarm is eliminated (removed by a user or deleted by antivirus software), another one will be regenerated by swarm to ensure that the number of individuals in the population is constant.

In addition, the ANN embedded on the prototype, which presented in [rel4], can be used as a "trigger conditions" to execute the payload on the aimed target, or control behavior of the swarm. On the other hand, the virus can simulate the working mechanism of an ANN to enhance the robustness. More details about this are discussed in section 6.1.4. The workflow of

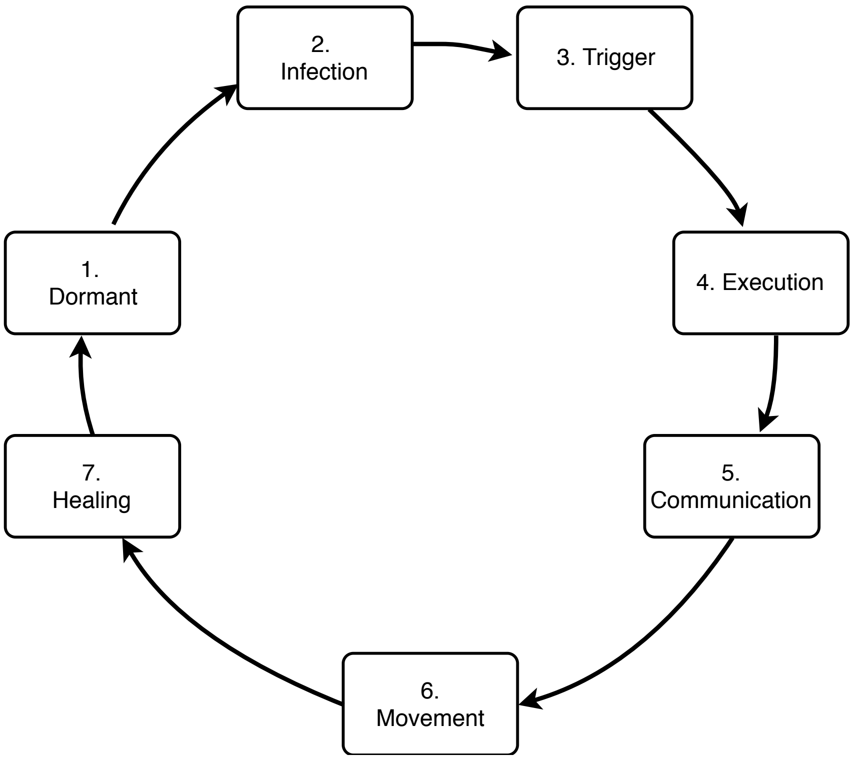


Fig. 6.1: X-Ware work flow

the X-Ware, which illustrated in Fig. 6.1, is as follows: The workflow of the X-Ware consist of :

- *Dormant*: The virus is idle; none of the other virus instances is operating.
- *Infection*: This phase is responsible for generating the swarm population and maintains the virus population size. Moreover, the infection phase is also a kind of passive defence mechanism. In case an anti-malware recognizes some of the viruses and eliminates them, the same amount of viruses will be regenerated to ensure the same swarm size.
- *Trigger*: The virus is activated to perform the tasks for which it was intended to. System events can cause this triggering.
- *Execution*: The virus delivers a payload onto the infected system.
- *Communication*: The communication phase decides which virus should be activated as well as updates the global memory of the swarm.
- *Movement*: This phase determines the next target to infect.
- *Healing*: The virus wipes its old copy when the movement to another host is completed. This phase helps to maintain the cardinality of the swarm.

6.1.2 X-Ware structure and functionality

The X-Ware is a self-replicating structure consisting of components to perform its task. The functionality of each component is as follows:

- Infector: This component is responsible for some tasks, respectively, as follows. The first task is copying the virus and attaching it to a suitable host. The second one is healing the host file after moving to a new location. The third one is checking whether the file is infected or not; if the file is infected, then there is no need to infect it again.
- Propagator: This component is the most important because it decides how the virus is propagated. It will indicate where to move (i.e., which file to infect) by evaluating the files.
- Communication: Communication is responsible for exchanging information in the swarm. The virus instances interact with each other to decide which one should be activated through the command line arguments. Furthermore, it allows the virus to update the stored information such as locations of all virus instances, and the network topology when moving to a new file. That means the entire communication traffic goes from one virus instance to another without any central communication point.
- Payload: In this prototype, the payload is to test the swarm functionality. No destructive payload was implemented.
- Trigger: The trigger launches the payload at a given condition. The triggering can be set by a variety of system events such as setting off at a given time, a given number of times a program runs, the physical condition of the disk, a specific date or time.

6.1.3 Spreading mechanisms

The spreading of malware is quite complicated and mainly depends on the kind of malware (e.g., viruses, trojans, worms) and the environment it is in (inside the computer or network). The distribution of malicious programs has expanded beyond traditional ways, such as from removable media, download files from the Internet, or email attachments to more sophisticated approaches like drive-by-downloads from a compromised website or using social engineering.

In fact, the malware could use various infection techniques to move inside a PC environment such as prepending, appending, or inserting it into an executable file. For the herein virus prototype, the prepending technology is adopted. With this technology, the virus attaches itself to the starting code of the host so that it will execute first when the program starts.

Furthermore, the spreading strategy resembles the classical worm called “Rabbit“ [13]. With this strategy, the virus will erase its copy on previously visited files when moving to a new one. Consequently, the host will be recovered to the original state. In this view, the virus behaviour is similar to evolutionary or swarm algorithms, whose individuals jump over the search space. This strategy controls the spread of the virus to avoid excessive population growth, which may cause system slowdown and lead to possible detection. Hence, the population of the swarm virus will remain unchanged during the contagion.

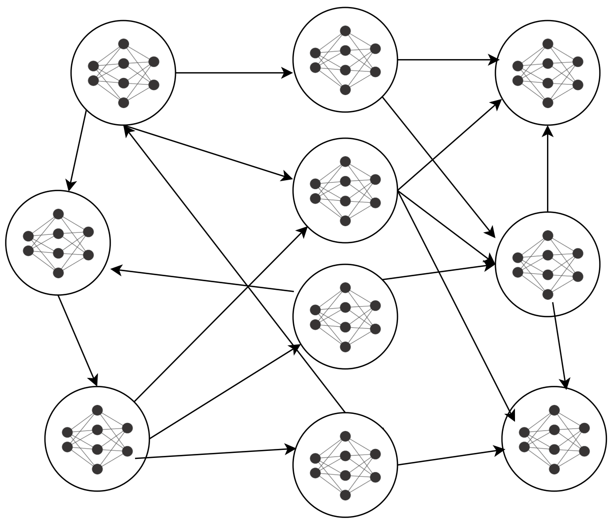


Fig. 6.2: Each virus in the swarm consist of a MLP

6.1.4 Incorporate ANN and the virus

This part discusses how to incorporate ANN and the proposed virus. In this study, the concept of MLP neural network has been use to enhance the malware. More specifically, we adopt MLP in two cases: (1). each of the virus in the swarm will contain an MLP, (2). a number of individuals in the population will simulate the behaviour of the MLP to perform actions.

Each virus comprise an ANN

In this implementation, each individual in the swarm had its own MLP as shown in Fig. 6.2. The MLP implementation, in this case, is composed of three layers: input layer, hidden layer and output layer. The file's size was used as input to the network's input layer, which contains a total of 2 neurons. The hidden layer consists of 2 neurons for network training, and an output layer consists of 1 neuron as it produces the result of whether to perform the tasks of the malware or not. In terms of activation function, there are a variety of methods which could be used for training. In this study, for the purpose of binary classification, the logistic sigmoid function is utilised. Fig. 6.3 shows the architecture of the ANN integrated into this prototype.

The back-propagation algorithm [102] is used to train the MLP in the proposed approach. Back-propagation is short for "backward propagation of errors". It is a supervised training algorithm in the multilayer feed-forward networks using gradient descent [60]. The dataset used for training is collected from system files. After training the model, the optimized network weights are integrated into the virus. When the virus executes, these weights are used on the embedded MLP to make the computation. Then, the network makes the "trigger conditions" to perform the execution or can be responsible for another activities.

Viruses are trained to perform system searches for finding a suitable target. The ANN then generates signals for conducting a task. For the herein experiments, the task is to display a

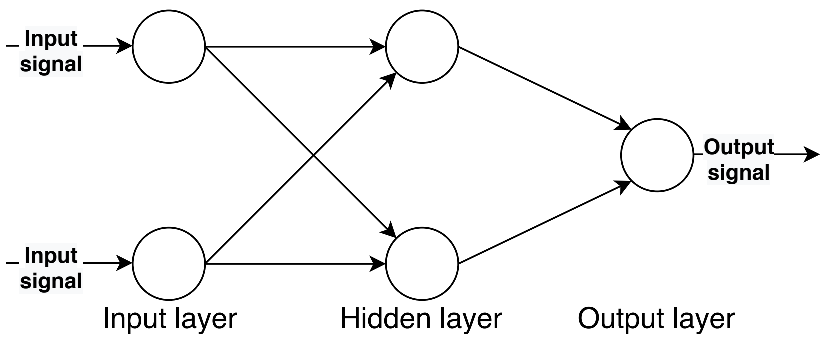


Fig. 6.3: Architecture of the MLP embedded in virus

message. Additionally, the file size is utilised for target identification. Other attributes, such as system-level features, can be utilised. Subsequently, it is impossible to reverse the ANN to work out the target specifications.

Malware can use an ANN model, which is a black box, instead of a traditional “if-then“ command line to camouflage its trigger condition. This seems to be much more effective camouflage technique that obfuscation. Technique wise, this complicates the deciphering jobs of anti-malware by hiding the target categories or triggering conditions.

Virus act as a node of an ANN

This kind of implementation uses several viruses in the swarm to act as input layers, hidden layers and output layers of. Fig. 6.4 illustrates this idea. As shown in this figure, only some individuals in the swarm (black coloured) act as nodes in the network. More precisely, two individuals are utilised as input nodes to receive signals, three for hidden nodes and one for the output node.

The MLP is also trained to obtain the optimized weights in the same manner as mentioned above. The difference is the weight allocated on individuals that act as nodes in the network. In other words, each node comprises its own weights.

When the virus executes, the swarm simulates the working mechanism of an ANN network. More specifically, the signal values propagated from the “inputs virus“, through the connection to the “hidden virus“, and then onward through more connection to the “output virus“. Following this strategy, the payload of the virus could be distributed, and it is extremely hard to reverse engineer the virus (reverse engineers must capture whole swarm).

6.1.5 Visualizing X-Ware behaviour

Malware activities can be logged, recorded, in two ways: using network dumps method, or tracking that malware’s exploitation and operation on a system to collect relevant data for security purposes. Malware samples can be dynamically analysed in a controlled environment. Analysts can then monitor the malware’s execution, record its reactions to the operating system

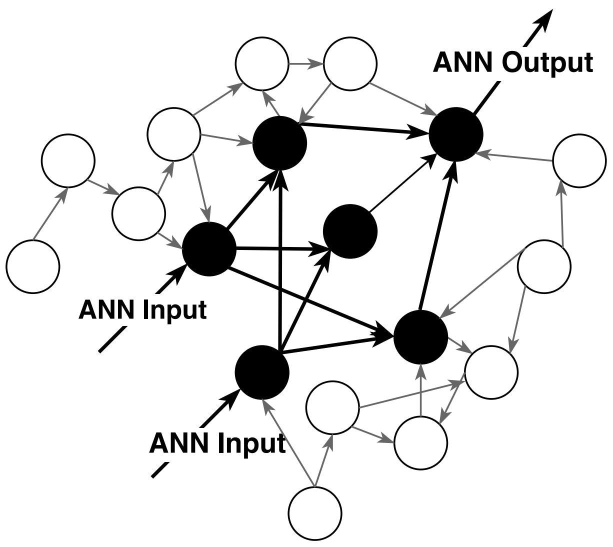


Fig. 6.4: Virus simulate the ANN working mechanism

and network. Subsequently, base on the collected data, its specific behaviour profile can be established.

Because of the continually rising number of malware, there is a massive amount of data generated from analysing them. There are inevitable needs for data storages and adequate, structured analysing tools to gain useful insights for counteracts, or for at least menace identification. Nevertheless, there are some difficulties in working with data. For instance, raw data is extremely hard for analysts to proceed. Furthermore, inappropriate tools can generate unusable results.

One of the most effective ways to observe a tremendous multi-dimensional amount of data for human beings is to visualize it. Application of data visualization in the security field is still in its early stage. There are still a wide range of visualizing tools as well as applications to explore, and many more yet to come. Additionally, there are needs for ad-hoc solutions for specific systems and for most of the cases, these do not satisfy all needs.

As mentioned above, so as to capture X-Ware's behaviour for later visualization, the author group performed a dynamic analysis on a sample of it. An X-Ware sample was put in a virtual, isolated environment. X-Ware's activities, for instance, file writing and deleting, registry writing and reading, process creating and terminating, as well as system calls invoked by the malware while performing these actions, were recorded. Table. 6.1 presents the actions currently monitored by the author group's analysis platform.

To protect the host system from any hostile takeovers, the dynamic analysis was performed in a separate virtual machine. This can be done by separating the host OS and the guest's kernel. Before executing any malware on a VM, a snapshot, or record, of the clear state of the VM was taken. This enabled the analyst to revert the machine back to that snapshot after each analysis. This isolated environment was installed with appropriate tools to capture the X-Ware

Tab. 6.1: OS-related action

Type	**Operation**
File	Create, Read, Write, Rename
Process	Create, Terminate
Registry	Create, Read, Write, Delete
NET	Connect, Send, Disconnect

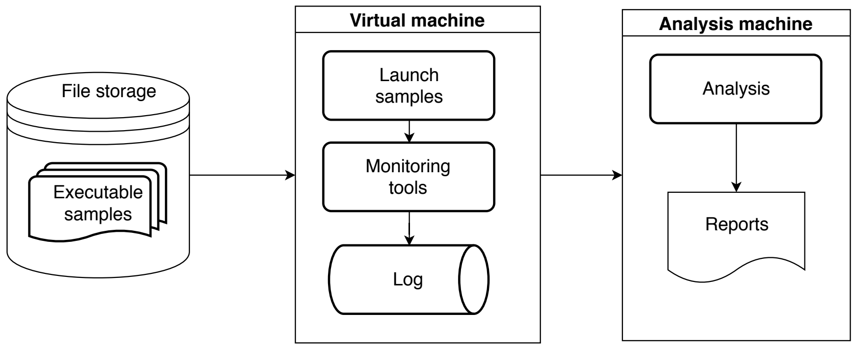


Fig. 6.5: X-Ware behavior analysis architecture

activities. Fig. 6.5 shows the platform for monitoring the behaviour of the X-Ware sample.

For observation and analysis of a swarm virus, its behavioural pattern is the essential factor. Behavioural pattern is usually a portion of significant malware behaviour records in a system, which is easily accessible for the analysts. As a virus moves from one file to another across a PC system, its behaviour follows and can be mapped down like a tree structure. Nonetheless, the tree structure contains dead-ends and no-cycles, which are not effective to capture the complex coordinate-to-coordinate movement of the viruses. Thus, the tree structure is converted to another structure which is inspired by swarming dynamic nature. This complex structure, or network, has proven itself to be an effective format for visualising the swarm dynamics [144] and performing analysis upon it. In this study, the Bianconi-Barabási model [18], for the mentioned advantages, is utilised

Generally speaking, Bianconi and Barabási instructed how to build a complex network by nodes and links between nodes. In Bianconi-Barabási model, a new node i is added to a network by establishing a link to another node which is selected from a set of existing neighbouring nodes. If in neighbour with node i, there are two nodes: j with probability of p, and node k with probability of $1 - p.$, the link will be established between i and the node with higher probability. There can be more neighbouring nodes, and each possesses its own probability. i will form a link at a time to a neighbouring node with the highest probability. This node-to-node connectivity is built up to form a complex network in which a node can be connected, to the time being, for several times.

The previous literature [145] has pointed out that the movement of the virus in the swarm and the communication inside the swarm likely follow the complex network topology so that any swarm intelligence based techniques, which permit search over the graph, could be adopted to build the complex network. The principal idea is such that a vertex represents each individual, and edges between vertices should reflect dynamics in population, i.e., interactions between individuals. For instance, SOMA [143] could be used to form a complex structure. The SOMA algorithm consists of a Leader drawing the entire population in each migration loop. Hence, the position in the population of activated Leaders shall be recorded like vertex, and the interacts between Leaders and individuals shall be recorded like edges.

6.2 Experiment setups

The proposed hypothesis can be validated with the help of a virus prototype in a high programming language - C#. This prototype is very convenient for laboratory experiments and researches.

This virus has no payload, and the contagion can be controlled. To train the ANNs for the virus, a labelled dataset was prepared. It consisted of 1000 files from a clean computer. A validation set that consisted of 100 samples were also prepared, which were then used to tune the parameters of the model.

In addition, an environment for virus evaluation had been established to test this virus's performance. The main goal of this environment was to focus on analysing the behaviour of X-Ware as well as showing the possibility of visualising X-Ware. For hardware specification, a desktop machine with 16 GB of RAM and an Intel Core i7-8750 processor were used.

To analyse the behaviour of the virus, the "out-of-the-box" (OOB) method was utilised. This method helped to keep physical machines from being infected. The OOB method was brought up using VMware software [127]. VMware was used to establish a virtualised environment, in which Windows 10 played the role of a host operating system and Windows 10 Pro as a guest operating system. The guest operating system included dynamic monitoring and behavior analysis which involved runtime execution of the X-Ware samples inside an isolated environment. Table 6.2 lists the tools for dynamic monitoring and capturing the fingerprints or file path traces. Table 6.3 indicates the tools utilised for analysing and visualisation. Table 6.4 shows the specifications of the sandbox configuration information used for experimentation. Noriben sandbox [103] was utilised to perform dynamic analysis of X-Ware samples whereas they were executed - forced to start, and their behaviours were monitored.

The behavioural pattern is a significant feature when studying malicious software. To record and visualize the behaviour of X-Ware, a slightly adjusted Bianconi-Barabási model for creating scale-free networks was used. In Bianconi-Barabási model, there are two parameters to be set up being the probability p and the number of links to attach for each new vertex m. In these experiments, the parameter p was set to at least 0.5. The probability 0.5 would mean that the ratio of connecting to neighbours or random vertex is equal. Another parameter to set is the

Tab. 6.2: Tools for monitoring the behaviour of X-Ware

Tools	**Description**
Process Monitor	used to track I/O operations, registry, file operation and processes
Process Hacker	used for monitoring system resources
Regshot	used to compare registry snapshots
RAMMap	used to analyze memory usage of any application

Tab. 6.3: Tools used to analysis and visualise X-Ware

Tools	**Description**
VM-snapshot	Snapshot manager to copied and stored the current working state of an virtual machine at a specified time
Noriben	A sandbox for malware analysis
Procdot	Software used for visualising malware
Gephi	Tools for network analysis and visualization

Tab. 6.4: Specification of environment used for experimentation

System	**Execution environment**	**Operating System**	**Network interface**	**Configuration**
Host	Physical	Windows 10 Pro	Ethernet	Intel i7-8750, 16 GB RAM
Sandbox	Virtual Machine	Windows 10 Pro	Virtual Ethernet	Processor: 2 RAM: 8 GB

Tab. 6.5: Recommended parameters of X-Ware.

Parameter	**Value**
number of individual	5–up to user
Visited host length	20–up to user
p	0.4–0.9
m	1–10

m parameter, which represents the number of connections it will initially have. Experiments showed that any value higher than four would give better performance. In this network, each edge weights $1/d$, where d is the distance of the file, measured by the number of folders between files (vertices) + 1. The complete set of experimental verified value are shown in Table 6.5.

6.3 Experimental results and analysis

In this section, the experimental results are presented and analysed. The behavioural analysis provides detailed information about the malware that is suitable for understanding the X-Ware samples. The samples were executed in a Windows virtual machine environment, and their behaviour was identified during program execution. Malware analysing tools, as stated in Table 6.2 and Table 6.3 were used to track the X-Ware activities and produce the log files.

6.3.1 Behavioural analysis

The result of executing the X-Ware in the isolated environment is illustrated in Fig. 6.6, Fig. 6.13, and Fig 6.12. In the presented case study, we obtained the behaviour data (named X-Ware.exe) composed of the malware actions on the system until the termination of the primary process. It was noticeable that these data were all typical file operations, including creating, reading, writing and renaming files. Besides, there were registry access and modification, process creation and termination, and network access.

Based on the logs, the behaviour of the virus prototype was visualized for concept demonstration.

As shown in Fig. 6.6, 6.7, 6.8, the initial virus population was created by the main process of X-Ware. This initializing process was started by utilizing a *Process Create* to run the binary. Then, a *CreateFile* function with *Generic Read* in *Desired Access* was performed to open the file. Lastly, there was a copying process for reprinting a *CreateFile* with *Generic Write* of the binaries, from *Hello1* to *Hello5.* The following step of the generating population was a read/write operations from the X-Ware file to the host. All of the above operations were performed by a single thread.

[1]Visualization using ProcDot[97]

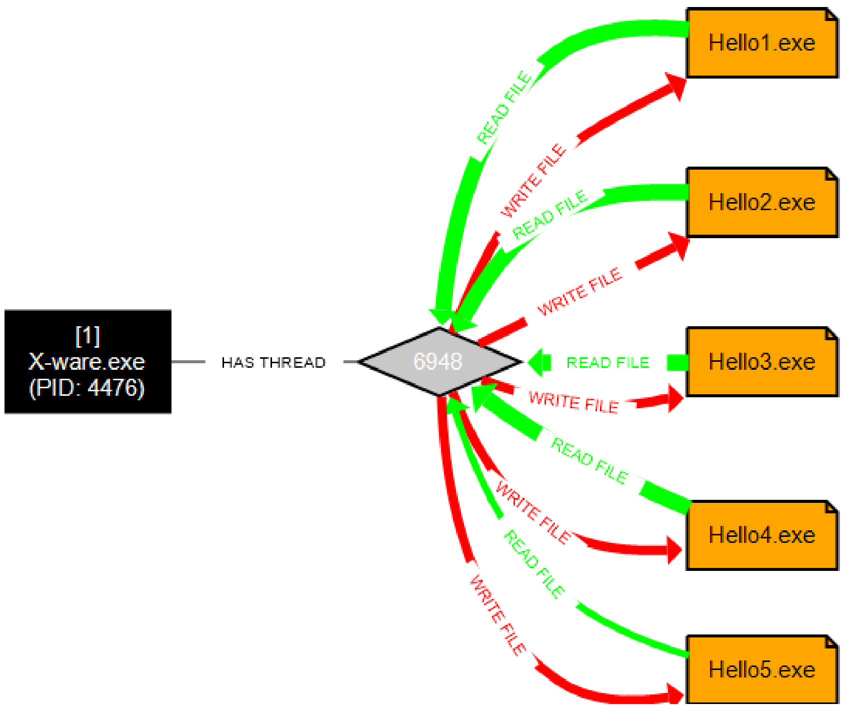
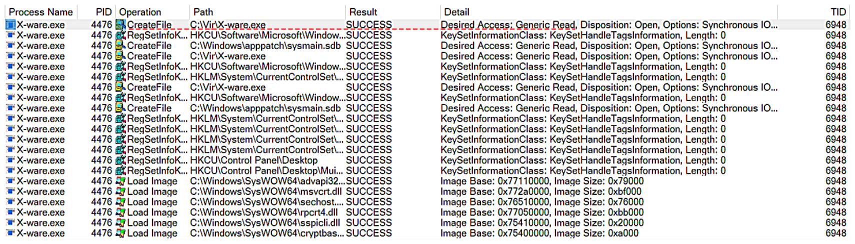
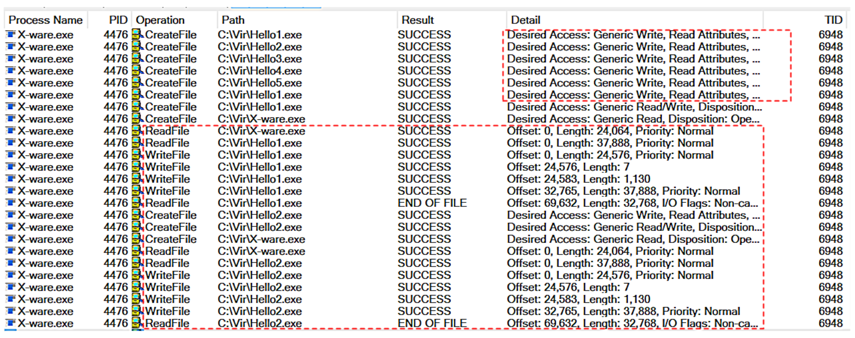


Fig. 6.6: Generate virus population[1]



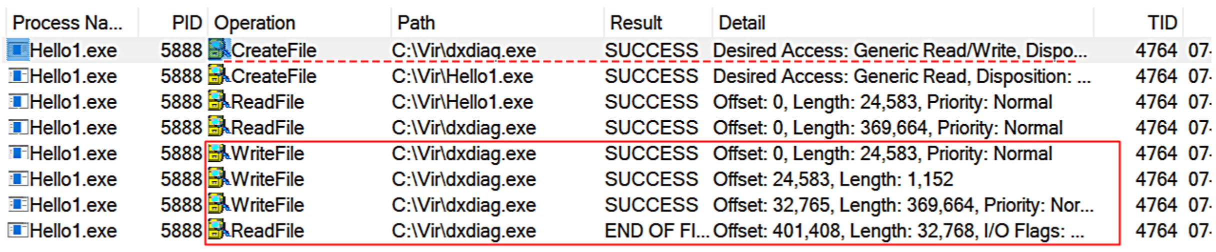
Process Name	PID	Operation	Path	Result	Detail	TID
X-ware.exe	4476	CreateFile	C:\Vir\X-ware.exe	SUCCESS	Desired Access: Generic Read, Disposition: Open, Options: Synchronous IO...	6948
X-ware.exe	4476	RegSetInfoK...	HKCU\Software\Microsoft\Window...	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0	6948
X-ware.exe	4476	CreateFile	C:\Windows\apppatch\sysmain.sdb	SUCCESS	Desired Access: Generic Read, Disposition: Open, Options: Synchronous IO...	6948
X-ware.exe	4476	CreateFile	C:\Vir\X-ware.exe	SUCCESS	Desired Access: Generic Read, Disposition: Open, Options: Synchronous IO...	6948
X-ware.exe	4476	RegSetInfoK...	HKCU\Software\Microsoft\Window...	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0	6948
X-ware.exe	4476	RegSetInfoK...	HKLM\System\CurrentControlSet\...	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0	6948
X-ware.exe	4476	CreateFile	C:\Vir\X-ware.exe	SUCCESS	Desired Access: Generic Read, Disposition: Open, Options: Synchronous IO...	6948
X-ware.exe	4476	RegSetInfoK...	HKCU\Software\Microsoft\Window...	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0	6948
X-ware.exe	4476	CreateFile	C:\Windows\apppatch\sysmain.sdb	SUCCESS	Desired Access: Generic Read, Disposition: Open, Options: Synchronous IO...	6948
X-ware.exe	4476	RegSetInfoK...	HKLM\System\CurrentControlSet\...	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0	6948
X-ware.exe	4476	RegSetInfoK...	HKLM\System\CurrentControlSet\...	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0	6948
X-ware.exe	4476	RegSetInfoK...	HKLM\System\CurrentControlSet\...	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0	6948
X-ware.exe	4476	RegSetInfoK...	HKLM\System\CurrentControlSet\...	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0	6948
X-ware.exe	4476	RegSetInfoK...	HKCU\Control Panel\Desktop	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0	6948
X-ware.exe	4476	RegSetInfoK...	HKCU\Control Panel\Desktop\Mui...	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0	6948
X-ware.exe	4476	Load Image	C:\Windows\SysWOW64\advapi32...	SUCCESS	Image Base: 0x77110000, Image Size: 0x79000	6948
X-ware.exe	4476	Load Image	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS	Image Base: 0x772a0000, Image Size: 0xbf000	6948
X-ware.exe	4476	Load Image	C:\Windows\SysWOW64\sechost....	SUCCESS	Image Base: 0x76510000, Image Size: 0x76000	6948
X-ware.exe	4476	Load Image	C:\Windows\SysWOW64\rpcrt4.dll	SUCCESS	Image Base: 0x77050000, Image Size: 0xbb000	6948
X-ware.exe	4476	Load Image	C:\Windows\SysWOW64\sspicli.dll	SUCCESS	Image Base: 0x75410000, Image Size: 0x20000	6948
X-ware.exe	4476	Load Image	C:\Windows\SysWOW64\cryptbas...	SUCCESS	Image Base: 0x75400000, Image Size: 0xa000	6948

Fig. 6.7: Beginning of population creation[2]



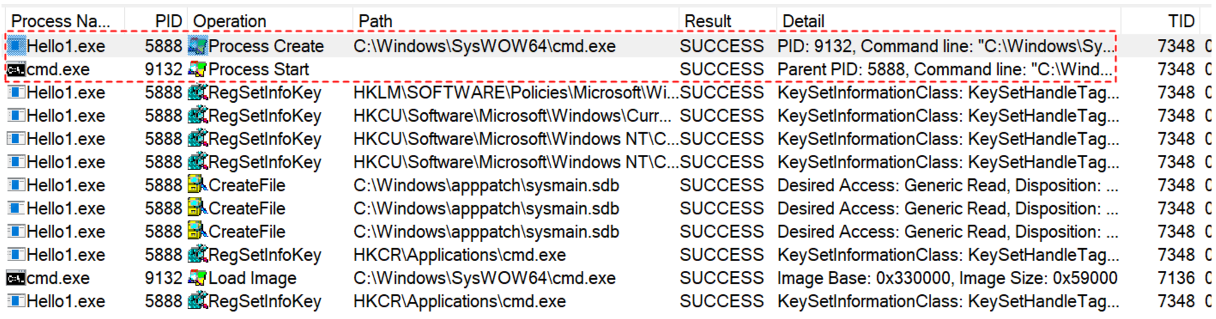
Process Name	PID	Operation	Path	Result	Detail	TID
X-ware.exe	4476	CreateFile	C:\Vir\Hello1.exe	SUCCESS	Desired Access: Generic Write, Read Attributes, ...	6948
X-ware.exe	4476	CreateFile	C:\Vir\Hello2.exe	SUCCESS	Desired Access: Generic Write, Read Attributes, ...	6948
X-ware.exe	4476	CreateFile	C:\Vir\Hello3.exe	SUCCESS	Desired Access: Generic Write, Read Attributes, ...	6948
X-ware.exe	4476	CreateFile	C:\Vir\Hello4.exe	SUCCESS	Desired Access: Generic Write, Read Attributes, ...	6948
X-ware.exe	4476	CreateFile	C:\Vir\Hello5.exe	SUCCESS	Desired Access: Generic Write, Read Attributes, ...	6948
X-ware.exe	4476	CreateFile	C:\Vir\Hello1.exe	SUCCESS	Desired Access: Generic Write, Read Attributes, ...	6948
X-ware.exe	4476	CreateFile	C:\Vir\Hello1.exe	SUCCESS	Desired Access: Generic Read/Write, Disposition...	6948
X-ware.exe	4476	CreateFile	C:\Vir\X-ware.exe	SUCCESS	Desired Access: Generic Read, Disposition: Ope...	6948
X-ware.exe	4476	ReadFile	C:\Vir\X-ware.exe	SUCCESS	Offset: 0, Length: 24,064, Priority: Normal	6948
X-ware.exe	4476	ReadFile	C:\Vir\Hello1.exe	SUCCESS	Offset: 0, Length: 37,888, Priority: Normal	6948
X-ware.exe	4476	WriteFile	C:\Vir\Hello1.exe	SUCCESS	Offset: 0, Length: 24,576, Priority: Normal	6948
X-ware.exe	4476	WriteFile	C:\Vir\Hello1.exe	SUCCESS	Offset: 24,576, Length: 7	6948
X-ware.exe	4476	WriteFile	C:\Vir\Hello1.exe	SUCCESS	Offset: 24,583, Length: 1,130	6948
X-ware.exe	4476	WriteFile	C:\Vir\Hello1.exe	SUCCESS	Offset: 32,765, Length: 37,888, Priority: Normal	6948
X-ware.exe	4476	ReadFile	C:\Vir\Hello1.exe	END OF FILE	Offset: 69,632, Length: 32,768, I/O Flags: Non-ca...	6948
X-ware.exe	4476	CreateFile	C:\Vir\Hello2.exe	SUCCESS	Desired Access: Generic Write, Read Attributes, ...	6948
X-ware.exe	4476	CreateFile	C:\Vir\Hello2.exe	SUCCESS	Desired Access: Generic Read/Write, Disposition...	6948
X-ware.exe	4476	CreateFile	C:\Vir\X-ware.exe	SUCCESS	Desired Access: Generic Read, Disposition: Ope...	6948
X-ware.exe	4476	ReadFile	C:\Vir\X-ware.exe	SUCCESS	Offset: 0, Length: 24,064, Priority: Normal	6948
X-ware.exe	4476	ReadFile	C:\Vir\Hello2.exe	SUCCESS	Offset: 0, Length: 37,888, Priority: Normal	6948
X-ware.exe	4476	WriteFile	C:\Vir\Hello2.exe	SUCCESS	Offset: 0, Length: 24,576, Priority: Normal	6948
X-ware.exe	4476	WriteFile	C:\Vir\Hello2.exe	SUCCESS	Offset: 24,576, Length: 7	6948
X-ware.exe	4476	WriteFile	C:\Vir\Hello2.exe	SUCCESS	Offset: 24,583, Length: 1,130	6948
X-ware.exe	4476	WriteFile	C:\Vir\Hello2.exe	SUCCESS	Offset: 32,765, Length: 37,888, Priority: Normal	6948
X-ware.exe	4476	ReadFile	C:\Vir\Hello2.exe	END OF FILE	Offset: 69,632, Length: 32,768, I/O Flags: Non-ca...	6948

Fig. 6.8: CreateFile Function for the Copy[2]



Process Na...	PID	Operation	Path	Result	Detail	TID	
Hello1.exe	5888	CreateFile	C:\Vir\dxdiag.exe	SUCCESS	Desired Access: Generic Read/Write, Dispo...	4764	07
Hello1.exe	5888	CreateFile	C:\Vir\Hello1.exe	SUCCESS	Desired Access: Generic Read, Disposition: ...	4764	07
Hello1.exe	5888	ReadFile	C:\Vir\Hello1.exe	SUCCESS	Offset: 0, Length: 24,583, Priority: Normal	4764	07
Hello1.exe	5888	ReadFile	C:\Vir\dxdiag.exe	SUCCESS	Offset: 0, Length: 369,664, Priority: Normal	4764	07
Hello1.exe	5888	WriteFile	C:\Vir\dxdiag.exe	SUCCESS	Offset: 0, Length: 24,583, Priority: Normal	4764	07
Hello1.exe	5888	WriteFile	C:\Vir\dxdiag.exe	SUCCESS	Offset: 24,583, Length: 1,152	4764	07
Hello1.exe	5888	WriteFile	C:\Vir\dxdiag.exe	SUCCESS	Offset: 32,765, Length: 369,664, Priority: Nor...	4764	07
Hello1.exe	5888	ReadFile	C:\Vir\dxdiag.exe	END OF FI...	Offset: 401,408, Length: 32,768, I/O Flags: ...	4764	07

Fig. 6.9: Beginning of the movement [2]



Process Na...	PID	Operation	Path	Result	Detail	TID	
Hello1.exe	5888	Process Create	C:\Windows\SysWOW64\cmd.exe	SUCCESS	PID: 9132, Command line: "C:\Windows\Sy...	7348	C
cmd.exe	9132	Process Start		SUCCESS	Parent PID: 5888, Command line: "C:\Wind...	7348	C
Hello1.exe	5888	RegSetInfoKey	HKLM\SOFTWARE\Policies\Microsoft\Wi...	SUCCESS	KeySetInformationClass: KeySetHandleTag...	7348	C
Hello1.exe	5888	RegSetInfoKey	HKCU\Software\Microsoft\Windows\Curr...	SUCCESS	KeySetInformationClass: KeySetHandleTag...	7348	C
Hello1.exe	5888	RegSetInfoKey	HKCU\Software\Microsoft\Windows NT\C...	SUCCESS	KeySetInformationClass: KeySetHandleTag...	7348	C
Hello1.exe	5888	RegSetInfoKey	HKCU\Software\Microsoft\Windows NT\C...	SUCCESS	KeySetInformationClass: KeySetHandleTag...	7348	C
Hello1.exe	5888	CreateFile	C:\Windows\apppatch\sysmain.sdb	SUCCESS	Desired Access: Generic Read, Disposition: ...	7348	C
Hello1.exe	5888	CreateFile	C:\Windows\apppatch\sysmain.sdb	SUCCESS	Desired Access: Generic Read, Disposition: ...	7348	C
Hello1.exe	5888	CreateFile	C:\Windows\apppatch\sysmain.sdb	SUCCESS	Desired Access: Generic Read, Disposition: ...	7348	C
Hello1.exe	5888	RegSetInfoKey	HKCR\Applications\cmd.exe	SUCCESS	KeySetInformationClass: KeySetHandleTag...	7348	C
cmd.exe	9132	Load Image	C:\Windows\SysWOW64\cmd.exe	SUCCESS	Image Base: 0x330000, Image Size: 0x59000	7136	C
Hello1.exe	5888	RegSetInfoKey	HKCR\Applications\cmd.exe	SUCCESS	KeySetInformationClass: KeySetHandleTag...	7348	C

Fig. 6.10: Execution of cmd.exe[2]

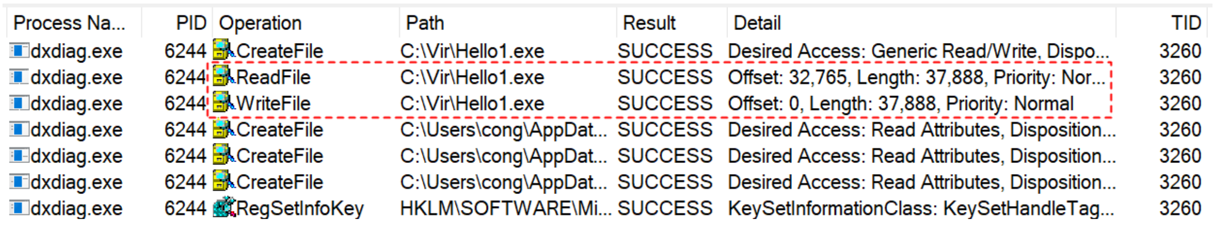
Figures 6.9, 6.10, 6.11, and 6.12 demonstrate the movement of a swarm's individuals. The movement began with *Process Create* to execute the file named *Hello1*; this file compromised the virus code. Followingly, the virus searched for victim files by performing *CreateFile* function with *Read Data/List Directory* in *Desired Access*. Once the target was found, there was the kick-start of the infecting process by a *CreateFile* with *Generic Read/Write* of the binary *dixdiag.exe*. The next step was the read/write operation from the virus file to the host. Once the migration was successful, the virus updated its new position information to other members of the flock. Then, the executed binary brought the execution to *cmd.exe* to clean its old copy when the move to another host was made. At this point, the jumping was conducted successfully. Figures 6.13 illustrated the regenerating process when the user or anti-virus software eliminated members of the swarm. The regenerating process aimed to keep the cardinality of the swarm constant.

6.3.2 Network analysis

Complex network analysis can be accomplished using many components. In this study, we utilised the adjacency weight matrix to indicate the linkage between different individuals in the population.

In the herein experiments, a swarm consisting of 5 individuals was created, which means there

[2]Visualization using Process Monitor[84]



Process Na...	PID	Operation	Path	Result	Detail	TID
dxdiag.exe	6244	CreateFile	C:\Vir\Hello1.exe	SUCCESS	Desired Access: Generic Read/Write, Dispo...	3260
dxdiag.exe	6244	ReadFile	C:\Vir\Hello1.exe	SUCCESS	Offset: 32,765, Length: 37,888, Priority: Nor...	3260
dxdiag.exe	6244	WriteFile	C:\Vir\Hello1.exe	SUCCESS	Offset: 0, Length: 37,888, Priority: Normal	3260
dxdiag.exe	6244	CreateFile	C:\Users\cong\AppDat...	SUCCESS	Desired Access: Read Attributes, Disposition...	3260
dxdiag.exe	6244	CreateFile	C:\Users\cong\AppDat...	SUCCESS	Desired Access: Read Attributes, Disposition...	3260
dxdiag.exe	6244	CreateFile	C:\Users\cong\AppDat...	SUCCESS	Desired Access: Read Attributes, Disposition...	3260
dxdiag.exe	6244	RegSetInfoKey	HKLM\SOFTWARE\Mi...	SUCCESS	KeySetInformationClass: KeySetHandleTag...	3260

Fig. 6.11: Finishing the movement[2]

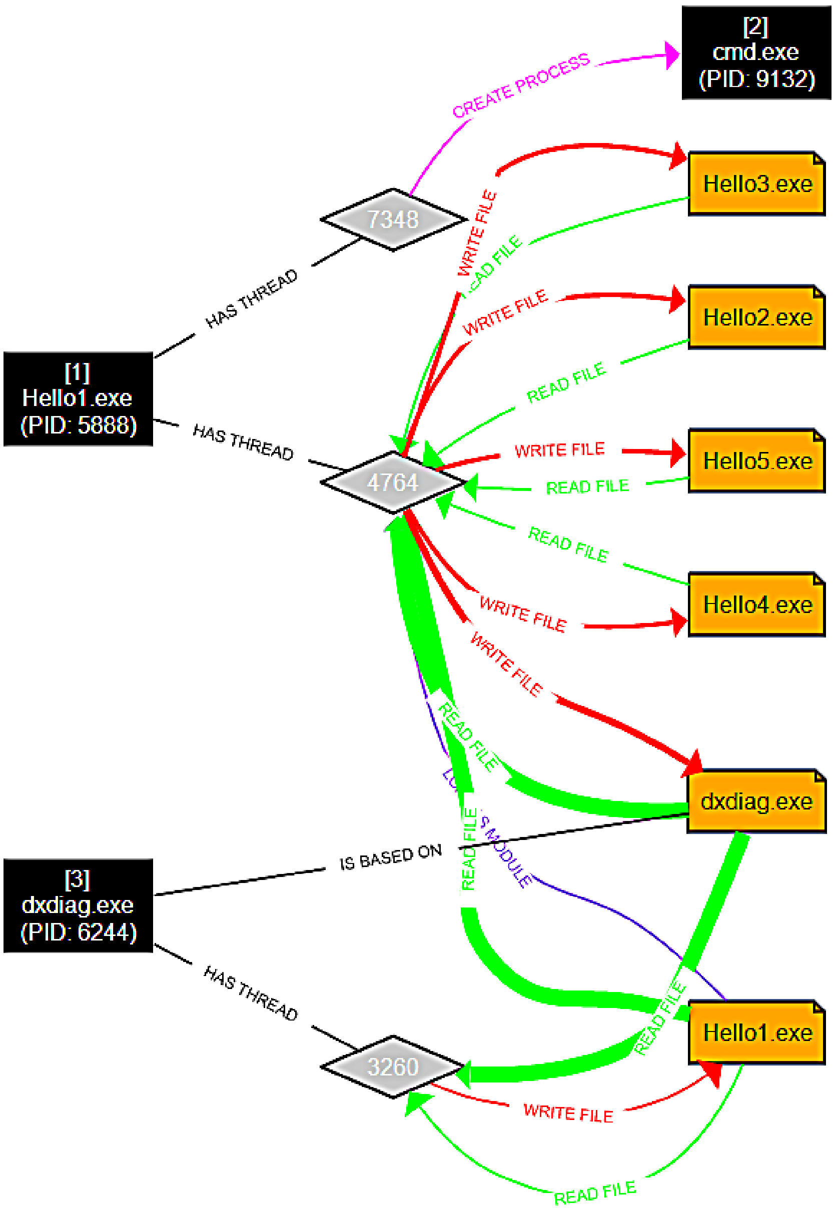


Fig. 6.12: Virus movement[1]

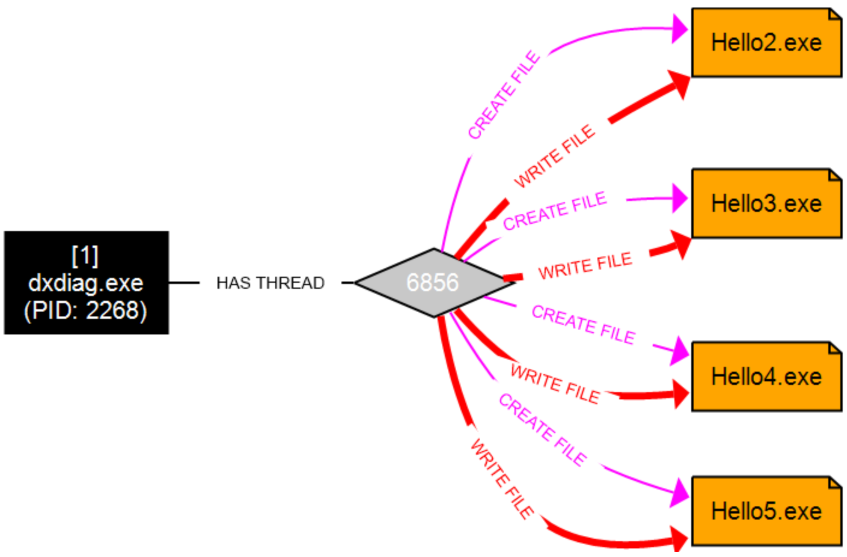


Fig. 6.13: Regenerate virus files[1]

Tab. 6.6: Swarm virus network centralities

	Min	**Median**	**Max**
Degree centrality	0.095	0.057	0.485
Closeness centrality	0	0.178	1
Betweenness centrality	0	0.02	0.162
Eigenvector centrality	0	0.004	1

are five virus instances. In each experiment, an individual jumped from file to file (infected a file) 20 times in total. The fascinating behaviour of the swarm was recorded and visualised. The figures shall be read as follow: the node size is related to file importance. The node colour indicate the communication between the subgroups. Furthermore, the virus behaviour in the system had been analysed for different network attributes such as the vertex centrality including degree , closeness, betweenness, eigenvector. Alongside that, the clustering coefficient was also utilised to examined the network structure. The obtained statistical data is given in Table 6.6.

Degree Centrality

Generally referred to as the basic centrality; this centrality is defined as the number of direct connections a node has with other nodes, which means the number of relation (edge) that the node has. Technically, a node with a higher degree has more neighbour than the others. Degree centrality is an important distribution hub in the network as it connects and thereby distributes the most of information flowing through the network. Degree centrality is one of the most significant features considering a complex network. Figure 6.14 presents the visualisation of

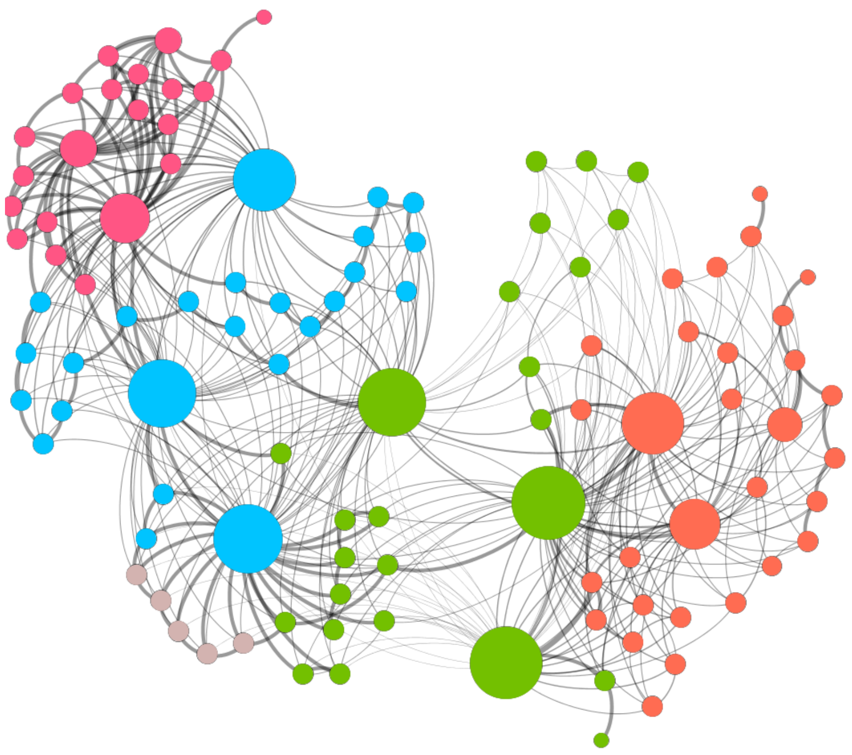


Fig. 6.14: Degree centrality

the networks in term of degree centrality. As can be seen from the graphs, there are multiple nodes which are increasing (distinguished by their size), emphasising their prominence in the population.

Closeness centrality

The closeness centrality is defined as the average of the shortest path from one node to other nodes in the network. This centrality represents the utility and efficiency of connections between the nodes. In other words, this measure points out the individuals who are the most influent in the entire network.

Closeness is a crucial measure as to the rate of distribution of information in the graph. The visualization of this centrality is given in Figure 6.15. As shown in the figure, the closeness centrality is mainly distributed over the entire network. This actually conveys that the majority of the nodes are contributed evenly in the network.

Betweenness centrality

The betweenness centrality is defined as a ratio of the shortest paths between other nodes passing through the node, to the total number of shortest paths between nodes. The higher the betweenness, the higher the likelihood that the node will become a mediator in the data

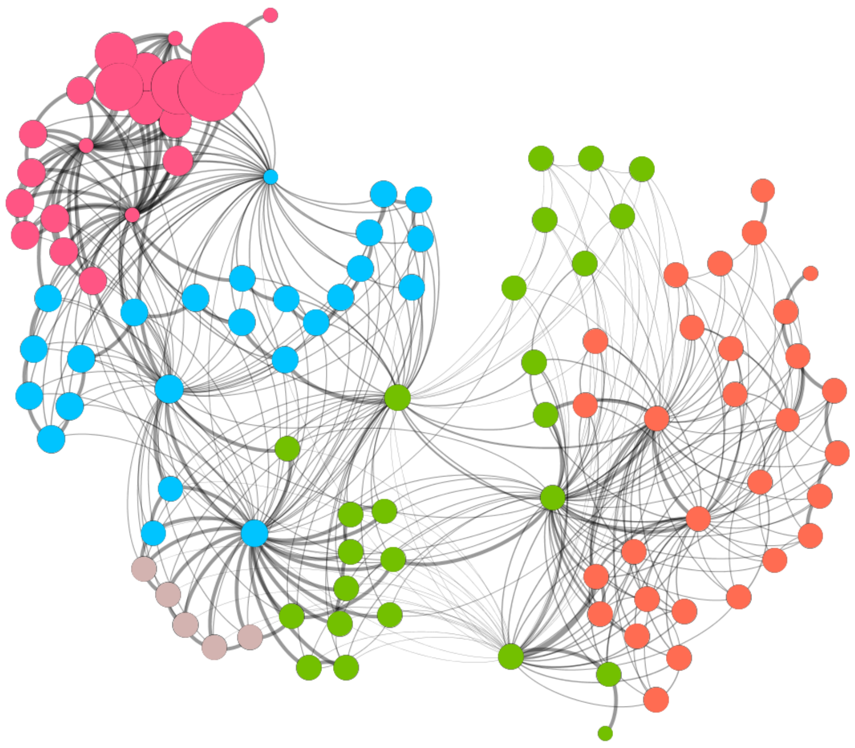


Fig. 6.15: Closeness centrality

flow between the other nodes. Simply speaking, this centrality points out the individuals who influence the communication flow of the system.

Betweenness is a fundamental measure in terms of the control and management of information within the graph. This centrality is illustrates in figure 6.16. In this graph, the bigger nodes have higher betweenness centrality. Incidentally, the node which has the highest betweenness centrality also has the best fitness. One crucial aspect to be taken into account is that once the node for the best fitness changes, the betweenness centrality of the system alter as well.

Eigenvector centrality

The eigenvector centrality indicates that the importance of a node depends not only on the number (its degree) but also the significance of its neighbouring nodes. If a node is attached to many other nodes which are themselves well connected, then, it possesses high eigenvector centrality value. In the network, the nodes with high eigenvectors are considered to be more influential than other nodes.

Eigenvector centrality is used as a measure of influence and power of a node in the network. Figure 6.17 demonstrates the distribution of eigenvector centrality. The colour coding is utilised to distinguish the centrality. Darker nodes are more central and lighter nodes are less central. As shown in the figure, some individuals are more influent than the others in the swarm.

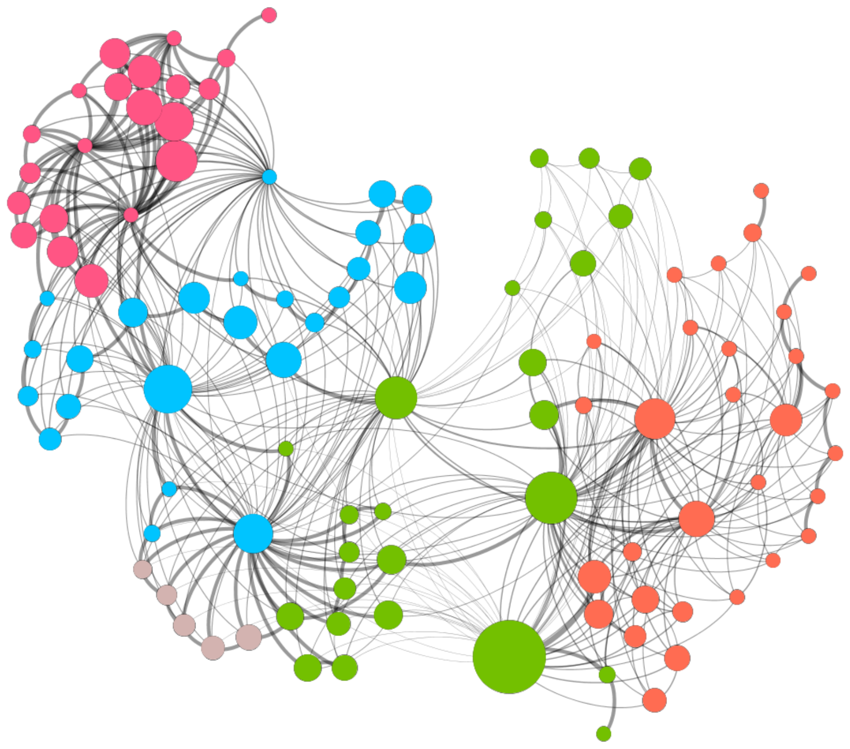


Fig. 6.16: Betweenness centrality

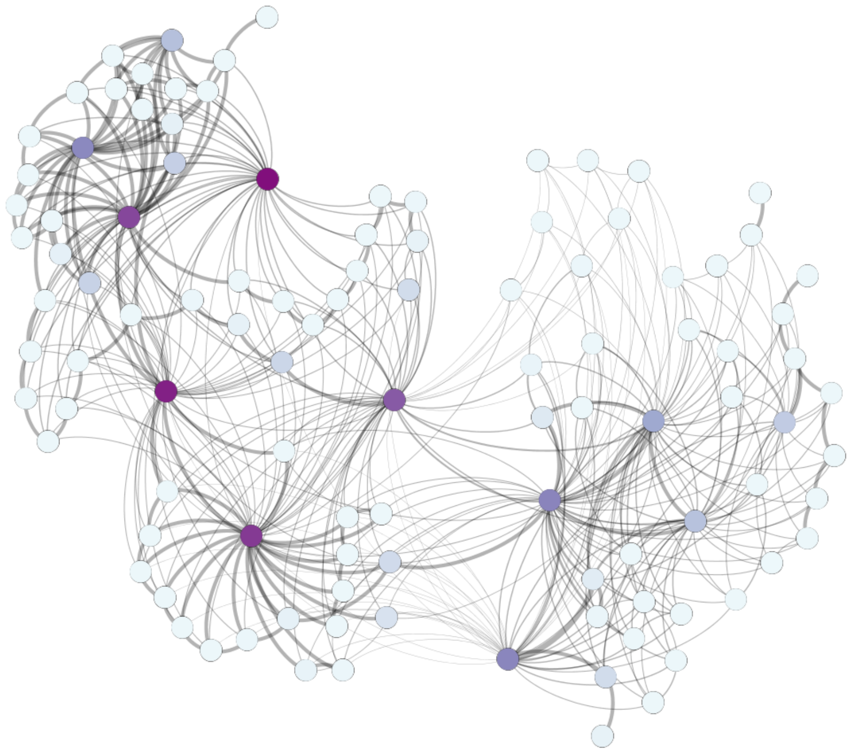


Fig. 6.17: Eigenvector centrality

Clustering coefficient

Clustering coefficient is a measure of the density of triangles in a network. In many networks it is found that if vertex i link to j and k, then vertex j and k are likely to connect each other [88]. A lower clustering coefficient's vertex indicates that it probably connect to many unconnected vertices which belong to different communities.

In the previous studies, the authors in [39, 40] argued that higher clustering coefficients lead to lower speeds of propagation. In addition, a high clustering coefficient indicates that persistence of the transmission in the network. Furthermore, the correlations between the two factors of the X-Ware properties has been plotted in Fig. 6.18. The figure indicates that as vertex degree increases, the range of clustering coefficient shrinks from $[0, 1]$ exponentially towards 0. This means that a vertex with a higher degree has a lower value of the clustering coefficient. There are some possible explanations for the decrease of clustering coefficient in as vertices degree increases.

- First, the X-Ware members tend to group in communities, sharing mostly neighbours within the same community.
- Second, the smaller communities are denser. Hence, the value of clustering coefficients is larger.
- Finally, communities may be connected by large degree vertices, and being a connector will decrease its value of clustering coefficient of these large degree ones.

Histograms of the four measures

Fig. 6.19, 6.20, 6.21, and 6.22 depict the histograms of observed network attributes. The empirical result shows that the more important a node is, the higher centrality it has. Subsequently, if a node has a higher centrality, then it has more probability of being visited. In other words, the important files have a higher centrality, which means they have a higher chance to be infected by the malware prototype. In contrast, less important files have lower centrality values and have a lower chance of being visited by the virus.

Discussion

Network analysis is a set of methods that originated from network theory to exhibit the power of social network influences. It provides several statistics that help characterize a network. Network science has been applied widely in many areas, from studies of the social networks to an assortment of biological systems and even to complex networks.

In recent years, the research community has witnessed efforts to modelling several complex systems as the complex network [19]. This omnipresent scale-free structure has an essential meaning for system dynamics. The complex network could be witnessed in numerous systems, from world wide web to citation networks, and as well as the epidemics of malware. Therefore, it is of essential to understand the effect of evolving complex networks on virus spreading to help make policies to prevent and mitigate the damage of malicious code.

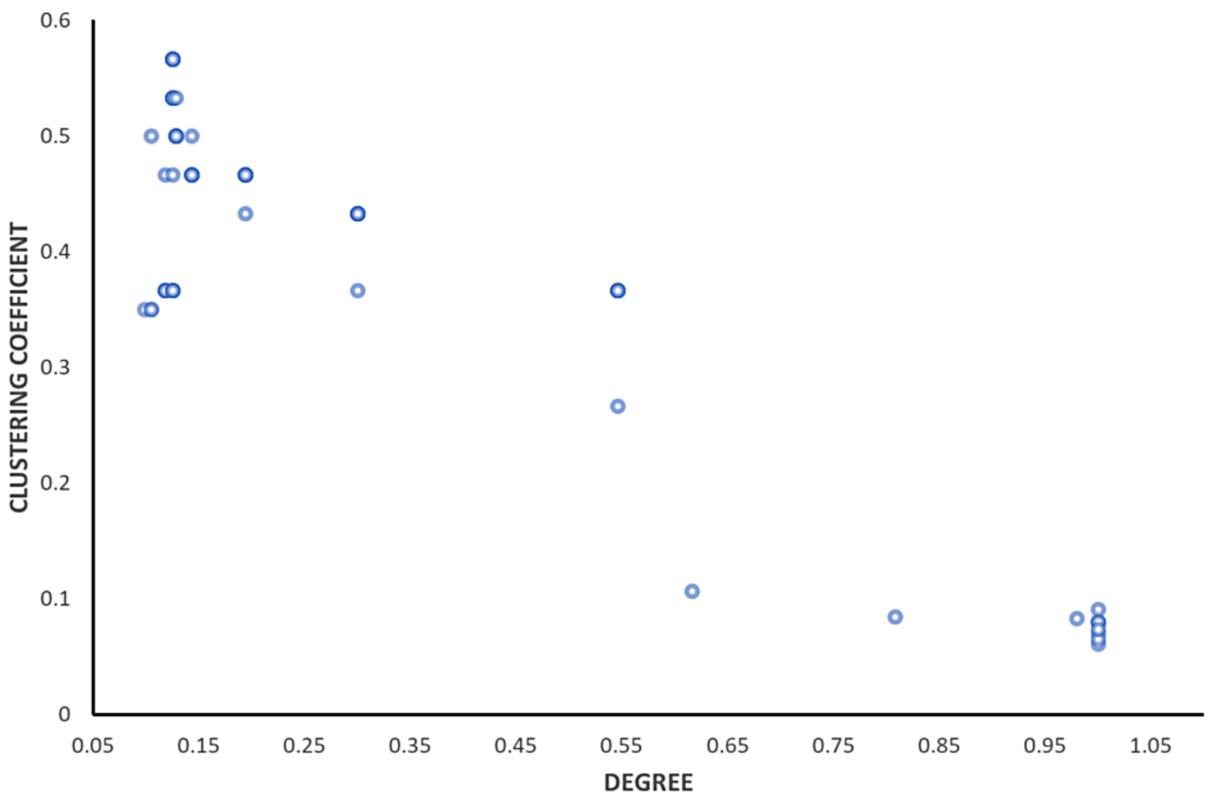


Fig. 6.18: Correlation of clustering coefficients and vertex degree

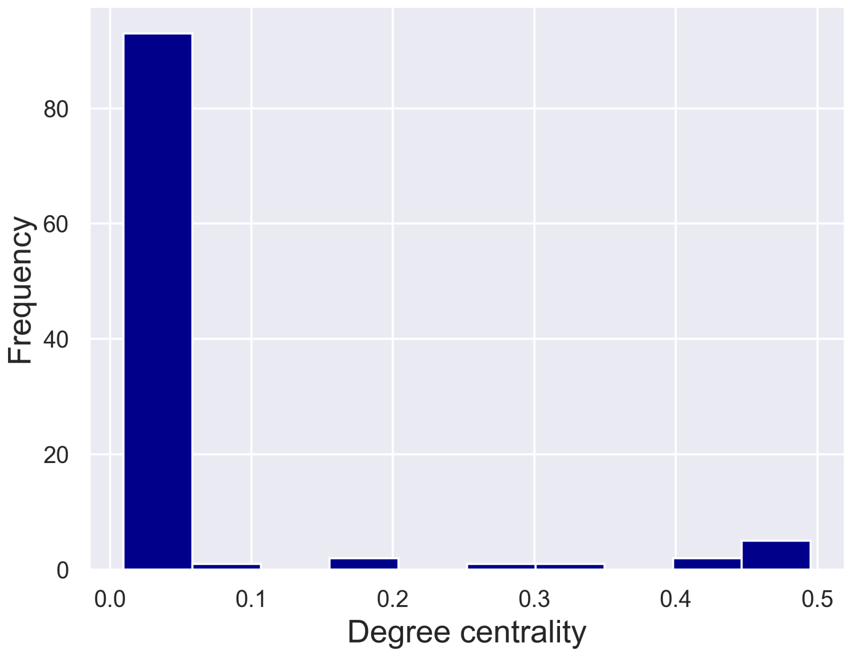


Fig. 6.19: Histogram of the network degree centrality

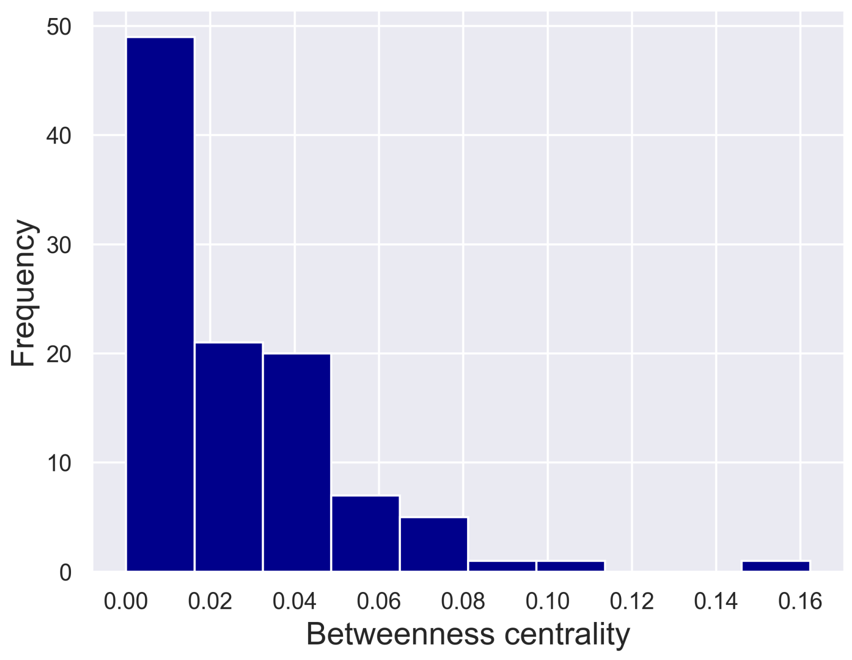


Fig. 6.20: Histogram of network betweenness centrality

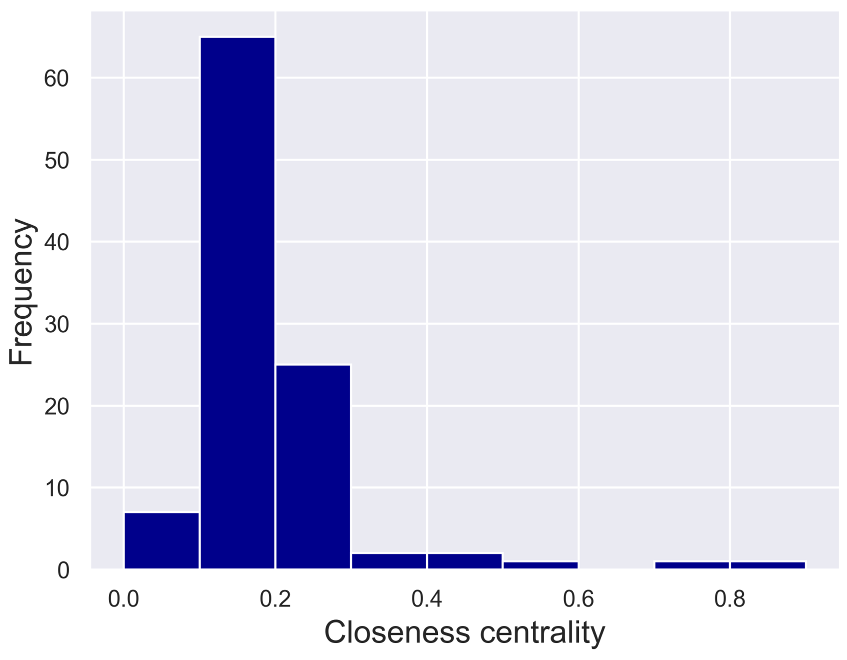


Fig. 6.21: Histogram of network closeness centrality

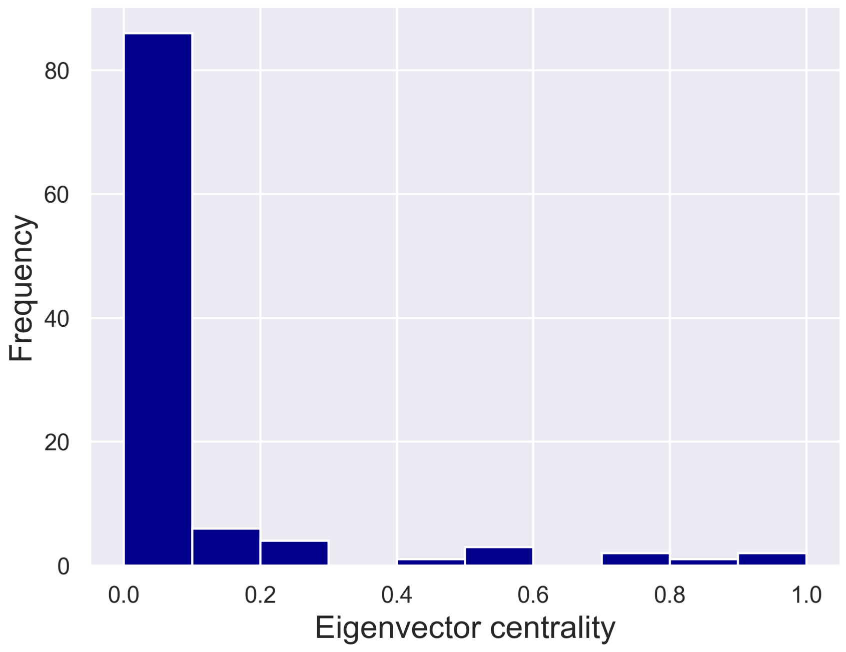


Fig. 6.22: Histogram of network eigenvector centrality

At present, numerous metrics have been used to define and classify complex networks. In which, centrality is a crucial property of complex networks that affects the operation of dynamical processes, like synchronization and epidemic spreading, and can bring valuable information about the organization of complicated systems. The centrality measures aim for identifying the more influence nodes in a network. They are used for understanding the power and the social influence in a network.

In the scope of this thesis, the author utilized some basic centrality measure metric to evaluate the spreading of the hypothesis swarm malware: degree, closeness, betweenness, eigenvector and clustering coefficient.

Our empirical experiments show that the files with higher fitness also have more top centrality metrics, and this means these files are more important. They are distinguishable from others. In contrast, less critical files have lower centrality estimates. This means a random X-Ware instance, which is moving through the system files or network, has a higher probability of infecting a file with the higher fitness files.

The essential factor is that files with higher fitness should be distinct. In our experiments show that most important files also have higher centrality. In figures, the nodes that represent important files are bigger than the other. In contrast, less important should be smaller. Graphs 6.14 6.15,6.16 are showing the dependency between importance and centralities. In these graphs, the most important file is the biggest node in the figure. So it is the most influence node in the network also. Furthermore, in figure 6.17, the darker nodes indicate that they are more influence than the other. These results suggest that if we can identify the most centrality node in the network then the virus' spreading rate could be decrease and potentially eradicate the virus.

6.4 Countermeasure

This section aims to suggest an approach to counter the X-Ware prototype. The prototype version of X-Ware is developed and observed in a controlled environment so that its behaviour and data can be easily obtained. Nevertheless, in reality, identifying the self-replicating swarm structures is a difficult task. As our experiments verify, the X-Ware has two significant features 1) it is moving over the hosts while keeping the constant of the population and 2) the communication among them. These features shall act as the critical criteria for identifying the to-be-created likewise prototypes. Hence, we suggest that the protection systems should not destroy them instantly but observe them and analyze their activities data as a whole in order to discover the activities of some subset of such malware that can be expected from the X-Ware (i.e. movement, communication, trigger).

Additionally, complex network visualization and analysis can help a lot. By applying the network analysis, we can discover the critical nodes, which play an essential role in the swarm network and thus can take corresponding actions based on the analysis.

6.5 Summary

In this chapter, we presented X-Ware, which is a new virus based on botnet evolution using AI and SI for the purpose of studying its features to form the anti-malware solution in future. This chapter discusses the methodology to develop a prototype of X-Ware in which the ANN acts as an intelligent centre that keeps payload, triggers conditions with no payload and controllable contagion. Furthermore, there are practical experiments to visualise, measure, and analyse the behaviour of the X-Ware under the form of a complex network.

7 X-SWARM: THE UPCOMING SWARM WORM

Inheritance from Chapter 6, in this chapter we study the modification the X-Ware as a worm, named X-sWarm, its simulations and analysis. The content of this chapter have already been presented and published at scientific journals [125].

7.1 Introduction

Recent years have witnessed a dramatic growth in utilizing computational intelligence techniques for various domains. Based on developments of evolving trends of cyber-threat, it is reasonable to predict that cybercriminals will begin to integrate malware with artificial intelligence in general, swarm intelligence technology in particular, to create more effective attacks. Generally, artificial swarm malware can share the collected information, speed up the process of trial and error, and leverage the specialized members of the swarms in the specifics environment.

Alongside that, an emergency trend needs to be concerned is that the abuse of anonymity networks like Tor to evade detection and anonymize the location of the command and control (C&C) servers. With the deployment of Tor, a device is able to build a web-based hidden service (HS) for accepting connections without revealing their physical location.

A natural question which arises is what happens if the two mention technique are combined. To seek the answer to this question, we design a prototype called X-sWarm, which combine the above techniques, so that we can conduct analysis, understanding of their potential and limitations. From this result, we suggest the idea for developing the mitigation techniques for this kind of upcoming threats as shown in figure 7.1

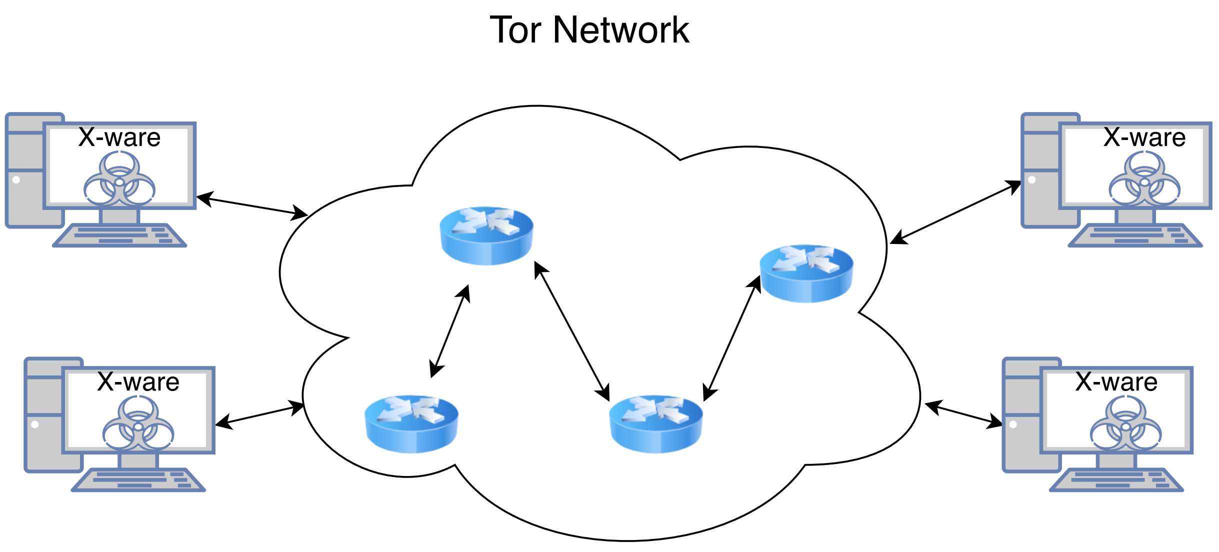


Fig. 7.1: X-sWarm architecture, the individual communicate through Tor network

In this chapter, we assess the threat of malware with swarm behaviour that relies on Tor infrastructure. Accordingly, we present a design of the first generation of an X-sWarm, in which the communication channel is established through hidden services. We also propose a graph maintenance algorithm with high resiliency and repair in the event of a take-down. Furthermore, we also suggest the countermeasure technique based on the same stealthy features of the X-sWarm. Our main contributions are summarized as follows:

- We propose a novel reference design of a new type of malware with swarm characteristics, whose command, communication, and management are fully anonymized by leveraging the Tor privacy infrastructure.
- We define a communication topology with self-repair mechanisms that enhance the resiliency and performance of the network.
- We discuss the possible countermeasures to mitigate similar threats in the future.

7.2 Background

In this section, we present some of the basic concepts that will be useful to better explain the proposed method by presenting the significant objects and interactions among them.

7.2.1 Worm

Peter Szor, in his research [119] described the worm was a subclass of computer viruses but primarily propagated on networks. The main difference between a computer virus and worm is the propagation mechanism. While the virus spreads by infecting files on computer or network, worms usually propagate as independent programs. A copy of a worm will be called a worm instance to avoid ambiguity. Furthermore, worms can exploit the vulnerabilities of the remote system and compromise these systems without the assisting of the user. Today, many worms act a carrier for other malware, such as trojan horses and bots.

A typical worm contains the following components: target locator, infection propagator, payload routines self-tracker, and life-cycle manager. In this structure, two key components are the target locator and infection propagator; the other components are non-essential and vary according to the worm.

7.2.2 Tor and hidden services

Tor, which name derived from the acronym of the project name “The Onion Router“, is a distributed low-latency anonymity-network. It aims to help user protecting their privacy, circumvent censorship, as well as keep the user’s confidential communication un-monitor. What is more, the users capable of concealing their activities and location by using Tor.

Users establish anonymous communications by forwarding their traffic through other Onion Router (OR). The client negotiates with each relay in the circuit a separate set of encryption keys in order to enhance the privacy on the circuit. The client negotiates with each relay in the symmetric circuit locks to enhance the privacy on the circuit. Next, clients transmit data using an encrypted channel, using previously negotiated keys. The data are delivered from the relay to relay until it reaches the destination. In addition to providing anonymous communications, Tor also allows publishing services inside the network anonymously, which called hidden services.

The architecture of Tor's hidden services consists of the following components:

- The service (e.g. web-server)
- client: the specific user who want to access the service
- Introduction Points: a set of relays, which is chosen by the hidden service that transmits the initial messages between the hidden service and client at the Rendezvous Point
- Hidden Service Directories: store the service descriptor.
- Rendezvous Point: The client randomly chooses the Tor relay, which is used to transmit the data between the client and the hidden service.

Generally, it is essential for a service to be published in the network in order to be reachable. The process begins with the hidden services selects a randomly set of relays to be its introduction points. Next, it creates a descriptor containing the public key and the address of its introduction points, then inserts the descriptor in a dynamic hash table using an address like *abcxyz.onion* as key. Tor implements a dynamic hash table for storing the onion address by utilizing the hidden service directory. By using the .onion address, the client obtains the descriptor and forms a new virtual circuit to a random relay as a rendezvous point. The client then utilizes the introduction points to notify the hidden service about the rendezvous point. Finally, the hidden service establishes a virtual circuit to the rendezvous point and starts the connection with the client.

7.3 X-sWarm design

In this section, the author describes the methodology to design the prototype. The X-sWarm consists of the following components: target selection, infection propagator, communicator, payload. These components integrated to compromise a machine.

7.3.1 Target selection

This component is responsible for discovering new targets in order to spread the worm through the network. This is crucial to the success of the worm. There are several methods to identify the targets, such as the following:

- E-mail addresses: There are many ways worm can collect e-mail addresses for attacks. For example, the address books on the victim's machine, collecting from the web, or through social networks.
- Host Lists: the host list on the compromised machine may contain information about the potential targets.

- Network neighbourhood: The worms explore the network neighbourhood to find new potential targets, for instance, by sending queries using Server Message Block (SMB) protocols.
- Random generation of target IP addresses: This is the most common and easily implemented technique. A worm could randomly select a target address to infect.
- Combine method: some worm's author utilized some of the mentioned methods to discover the victim.

For demonstration purpose, the target engine is implemented with a simple method. First, the worm discovers the IP address of the host. Next, it uses the Class-C boundary of that IP address and commences an Internet Control Message Protocol (ICMP) scan from A.B.C.0 through A.B.C.255. If a host responds to the ICMP echo request, the worm adds the host to the target list. Contemporary, the worm attempt to establish an SMB NULL Session when traversing through the IP address range.

After all systems on the local network have been vanquished, the targeting mechanism turns its attention to spreading across a wider area.

7.3.2 Infection propagator

This part describes the strategy which is used by the worm to propagate itself to a new bud. Generally, there are four typical approaches to propagate the worm

1. Through security vulnerabilities: every computer operating system has its vulnerabilities, and some worms are specifically coded to take advantage of these weak points.
2. Through email: this is a common way for computer worms to spread themselves. Worm's author can deceive the user into executing the malicious code in the email or click the malicious link. What is more, the attachment also could be utilized to spread the worms.
3. Through shared folder: worm replicates themselves to a shared P2P folder on the disk or even produce a shared folder and deceive the user into running the malicious code.
4. Through instant messaging: use social engineering and send messages that trick recipients into executing a link or an attachment.

In the scope of this research, we examine the null session technique to spread the worm. A null session is the unauthenticated sessions of the Server Message Block (SMB) protocol enables anonymous access to hidden administrative shares on a system. Consequently, the user can enumerate information about the system and environment when connecting to the share through a null session.

The propagation process

The propagation process starts by examining the local network address space, attempting to spread to as many machines as it can locally. After attempting to exploit all computers on the local network, it tries to intrude random external IP addresses.

The first thing when the worm tries to compromise a machine is by establishing a null session. This step provides the information on whether the machine it is attacking supports the Common Internet File System (CIFS) protocol and is likely to be a Windows machine. Next, the worm uses the SMB protocol to enumerate the list of account names on the remote machine. It also establishes some basic properties about the user for guessing the password process. After that, it makes an SMB connection with the target computers, attempting to access the IPC$ connection. If the worm successfully connects to the IPC$ share, it copies itself over to the remote machine. After the copy process, the worm uses remotely schedule a job to run itself on the target machine.

7.3.3 Communicator

This component is responsible for communication in the swarm. In fact, we design a virtual network between infected computers to establish communication and control operations. Each worm has a list of other known, running copies of the worm and capable of creating encrypted communication channels to spread information. This virtual network is built based on swarm intelligence principle. Such a network could be utilized to pass information rapidly to all running executables, lead to decentralize the C&C communication, and preventing the communication channel from being disrupted by others, make the worm hard to track. The two techniques are utilized for transmitting information are TCP and through Tor network.

In our design, the prototype would form a peer-to-peer (P2P), self-healing network that maintains a low degree and a low diameter with other instances to relay messages. In the following, we present an abstract graph representation of a worm communication topology, which is capable of self-repairing and dynamic distributed. This communication topology is simple, stealthy and resilient, which formed over a privacy infrastructure such as Tor.

Graph structure

We propose to use the concept of Neighbour-of-Neighbour (NoN) [81] to construct the abstract graph. In the literature [81], the authors examined the neighbour of neighbours for making better routing decisions. In the scope of this study, we investigate the concept of NoN to create a self-healing network. Note that in this study, we use vertex and node with equivalent meaning. We define our graph as below:

Definition 1. *Consider a graph G having n vertices (V) and m edges (E), where each vertex $v_i \in V, 0 \leq v < n$, is linked which a number of vertices. The set of neighbour verices of v_i is denoted as $N(v_i)$. Furthermore, the vertex v_i has the information of vertices that are linked to $N(v_i)$. In other words, each vertex knows the information of its neighbour of neighbour.*

Figure 7.2 give an example of a network connection diagram with 16 vertices. In this diagram, the neighbours of vertex 1 are: $N(v_1) = \{7, 11, 13, 15\}$. The neighbours of vertex 7 are $N(v_7) = \{1, 8, 12, 15\}$, similarly $N(v_{11}) = \{1, 4, 6, 12\}$, $N(v_{13}) = \{1, 5, 10, 2\}$, $N(v_{15}) =$

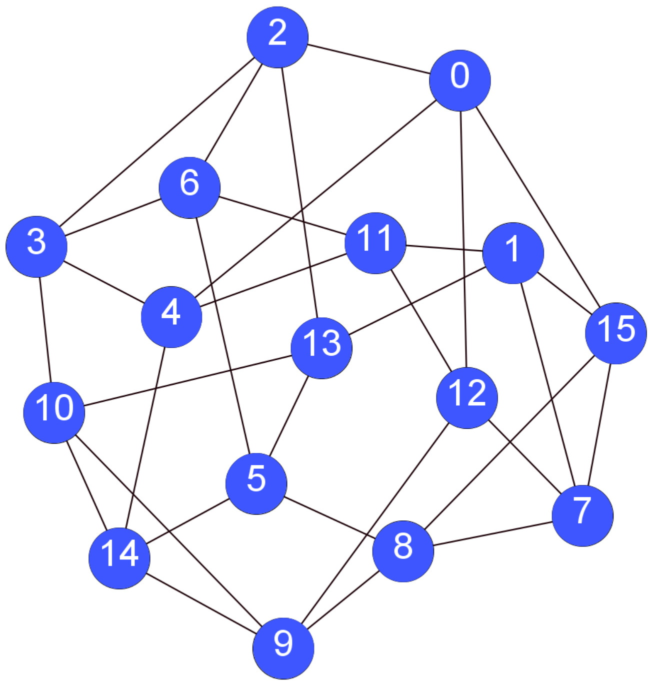


Fig. 7.2: Network diagram

$\{0, 1, 7, 8\}$. Thus, the NoN of vertex 1 are $\{8, 12, 15, 5, 2, 10, 0, 6, 4\}$. Vertex 1 stores a list that contains the information about its neighbours vertices $N(v_1)$ and the vertices that connected to $N(v_1)$. In the content of this work, the information stored is the *.onion* addresses themselves.

Network self-healing mechanism

Today, many networks have a valuable property that they can change their topology by reconfiguring. More precise, the nodes in the network can establish new connections or disconnect the existing one. These networks should be able to tolerant the failures and capable of recovering. The process where the network can recover in response to failure is what we call self-healing.

Based on the neighbour of the neighbour graph, we propose a network self-healing mechanism to form a new connection and substitute the relay function of the removed node. Thus, the connectivity of the network is maintained. The specific process of the repairing is: Assuming a vertex v_i is deleted, the neighbours of v_i react to this deletion by adding some set of edges amongst themselves. These edges can only be between nodes which were previously neighbours of v_i. This is to ensure the locality information in the underlying network which is maintained after inserted the edges.

One aspect to take into account is that the insertion of the new edge may lead to the growth in the connectivity degree of each vertex, denoted by $\delta(v)$. Indeed, the degree of some vertices may rise significantly after repeated deletion. Nevertheless, increasing the degree of such vertices is undesirable for the worm's resilience and clandestine operation. Hence, we propose to keep the degree of the vertices in the range $[\delta(Min), \delta(Max)]$ when add a new edge. The flow of the self-healing operation is presented in algorithm 1.

Algorithm 1 The self-healing algorithm

```
Input: Graph G
Output: Graph G' after self-healing
for each deleted vertex $v_i$ do
    for each pair neighbours of $v_i$: $v_j, v_k \in N(v_i)$ do
        if $\delta(v_j), \delta(v_k) < \delta(Max)$ and $e_{jk} = \{v_j, v_k\}$ not in $E$ then
            Add edge between $v_j, v_k$
            Update state of vertices
```

Figure 7.3 depicts the self-healing process in a 4-regular graph with 16 nodes. The red lines indicate the newly established connection between the vertices. For example, when vertex 1 is deleted, its neighbors $N(1) = \{7, 11, 13, 15\}$ react to this deletion and traverse their neighbor list to check whether they are linked to each other, and then establish a new one. In this case, the following edges are created: (7, 13), and (11, 15). Similarly, when vertex 12 is removed together with its connected edge, the new edges are appeared: (7,9) and (0, 11).

Command and control communication

In our prototype, communication is entirely encrypted by using Tor and Secure Sockets Layer (SSL). Furthermore, the encryption keys are unique to each link. Additionally, there is no central server; instead, all requests are handled by peers within the network. Each worm member acts as a command server and a client. Consequently, this structure helps the worm more resilient against the defences method than the traditional centralized structure.

The communication inside the swarm relies on the peer list that contained in each worm. This list is fixed and has a limited size for each worm. Thus, when a worm is revealed, only a few numbers of worms in its peer list are exposed. To forward a command, a worm could use its neighbours as targets and rely on these neighbours to continue passing on the command in the swarm worm.

Furthermore, the peer list based architecture can be utilized to implement strong encryption as suggested in [130]. Technically, each worm i generates its symmetric encryption key K_i. Assume the worm x has its peer list, which is denoted by PL_x. This peer list would consist of not only the N .onion address but also the symmetric keys of its neighbours. Hence, the peer list on worm x is:

$$PL_x = \{(O_{i_1}, K_{i_1}), (O_{i_2}, K_{i_2}), ..., (O_{i_n}, K_{i_n})\} \tag{7.1}$$

where $\left(O_{i_j}, K_{i_j}\right)$ are the *.onion* address and symmetric key used by the worm i_j. This encryption ensures that if a worm is captured, then just the keys in the captured worm's peer list are revealed. Hence, the encryption among the remaining worms will not be endangered.

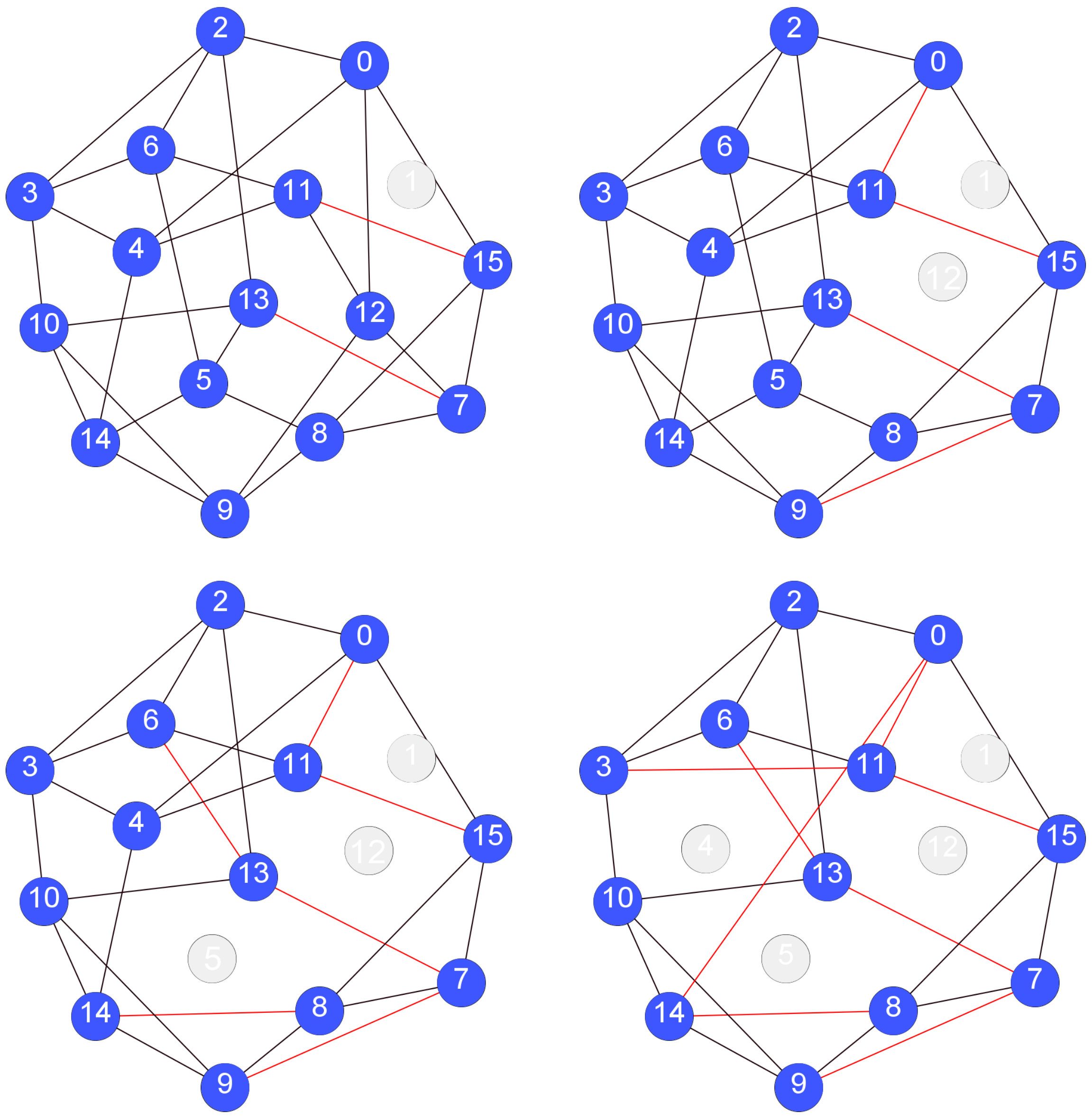


Fig. 7.3: Node deletion and the self-healing process

7.3.4 Payload

A worm's payload is designed to perform specific actions on behalf of the worm's author on the victim system. In this prototype, the payload is to test the swarm worm functionality. Hence, no destructive payload was implemented, except spread to other machines.

7.3.5 X-sWarm swarm behaviour

In this subsection, we describe how to build a malware network with swarm-based technology to create more effective attacks. Additionally, we discuss the swarm characteristics of our envision prototype.

Over the past few years, we have seen that traditional worm and botnet has a critical weakness that is the centralise C&C communication. Thus, cyber-threat actors attempt to discover a different method to overcome this disadvantage. One potential approach is to leverage the swarm intelligence (SI) to overcome the centralise weakness. With the latest advances of swarm technology, it is logical to expect that in the upcoming time swarm intelligence (SI) will be utilised to obtain this goal. Hence, to deal with this future threat, we need to have knowledge about this emergence trend so that we can design an efficient solution for countering the SI-based malware threats. For this reason, in this work, we propose a new swarm-based C&C behaviour in malware network. We aim to decentralise the infrastructure as well as autonomy the role of each member in such a network.

Accordingly, we suggest to design a P2P malware network that is able to share information – between malware nodes – and act on their own without a malware author issuing any commands. In this network, the nodes capable of communicating with each other and share local intelligence. For example, the malware attempts to learn information about a potential victim, and when it discovers the victims, it will share this information for the rest of the swarm. Furthermore, each node can make autonomous decisions with minimal supervision, use the collective intelligence to solve problems. This network allows node executes commands without the central instruction, and recruit and train new members of the swarm. Consequently, as a swarm compromises the more devices, it will be able to grow exponentially and thereby enhance its ability to attack multiple targets simultaneously.

In other respects, the communication-feedback mechanisms are required in order to help the swarm operate autonomously. We suggest that each peer in the P2P network supports bidirectional commands, enabling a peer within a network request and receive a response. Accordingly, each peer contains a set of commands that allow it to interact with other peers using custom-built P2P communication for performing multi-tasks routines. In the context of our work, the Tor protocol is utilised as a communication channel. Table 7.1 depicts the necessary commands for the communication process.

Tab. 7.1: List of command establish the autonomous of swarm.

Type of command	Description
Peer list	These type of commands are utilised to maintain the peer list up to date
Update config	The update commands are leveraged to ensure that every member have the latest config
Report	These commands are responsible for reporting the potential target.
Data transfer	These commands allow to transmit data between peers

7.4 X-sWarm life cycle

The life cycle of X-sWarm consists of the following stages, which cycle through until it is eliminated:

1. **Choosing target**: In this phase, the worm select the uninfected host to attack. In other words, the worm performs reconnaissance to determine other potential victims.
2. **Infection**: In the infection stage, X-sWarm exploits the file-sharing vulnerability to compromise the target system. By using Windows file share, users can read or write files across the network transparently. Our prototype takes advantage of these file-sharing services by using them to copy the worm code to a target's file system. After that, the worm might be manually executed by a user or scheduled to execute on the victim machine automatically. This is a simple but yet effective propagate technique.
3. **Communication and update**: When infecting a new host, the worm transmits the attacker's address to the victim machine. Whenever a new node address is introduced, it will join the virtual network. Then it will exchange the peer list with its neighbours. The data transfer executes via the virtual network, which uses Tor protocol to ensure that the information encrypted.
4. **Payload execution**: In this phase, the X-sWarm executes the payload. Nevertheless, for academic purpose, the worm is not implemented with a destructive payload but just propagate the worm.

7.5 Evaluation and Results

In this section, we conduct several experiments to prove the concept that is proposed in the previous section. First, we simulate to evaluate the resiliency and performance of the self-healing algorithm. Next, we execute the proposed X-sWarm to monitor its behaviour.

7.5.1 Simulation and analysis

In this part, we consider the performance measurements for our prototype along two dimensions: efficiency and robustness. The experiment involves the simulation process of the self-healing network resiliency and performance. For studying the robustness and the attack tolerance of networks, we conduct the procedure of removing a node from a network, where the node is chosen randomly, which mean the removal of a set of nodes happens with a certain probability.

Efficiency evaluation

The efficiency is a measured metric that needs to be concerned when studying communication in the network. A worm may be evaluated by its communication efficiency, such as how long it would take to transmit messages, update binary code, or collect the host's information.

One metric to express the efficiency of the network is the average shortest path length, l. This metric measures the average shortest path length that link any of two vertices in the network. The dynamics of the network (i.e. communications, information) is slow if l is high and vice versa. The average shortest path length defined by equation 7.2

$$l = \frac{1}{n*(n-1)} \sum_{i \neq j} d(v_i, v_j), \tag{7.2}$$

where $d(v_i, v_j)$ is the minimum number of edges between v_i and v_i, and n is the number of vertices.

Nevertheless, as the number of deleted vertices rises, the network will eventually turn into many disconnected components. Consequently, the average shortest path length becomes infinite. Hence, the author in [62] proposed to utilise the mean inverse shortest path length defined by the equation 7.3

$$l^{-1} = \frac{1}{n*(n-1)} \sum_{i \neq j} \frac{1}{d(v_i, v_j)}, \tag{7.3}$$

This way, if no edge connects v_i and v_j, the distance of $d(v_i, v_j) = 0$. Furthermore, the inverse length ranges from 0 (no edges) to 1 (fully connected). As such, the speed of information transformation is measured by l^{-1}: the larger l^{-1} is the better. In the context of our work, the l^{-1} refers to the overlay network of peer to peer connections created by the malware, instead of the physical topology of the network.

To investigate the effect on the network connectedness, we simulate and construct some generic models to analysis. Among various existing models for generating networks Erdos Renyi model [38] of the random networks and the Barabási Albert model [14] of the scale-free network are widely used. Hence, in the simulation process, we simulate the following models: two regular models ($N = 1000$, $k = 4$) in which one have the self-repairing mechanism, an Erdos Renyi model ($N = 1000$, $p = 0.05$) and an Barabási Albert model ($N = 1000$, $m = 5$).

Figure 7.4 illustrates the results for the vertex attack vulnerability measured by the average inverse shortest path length l^{-1}, which is defined in equation 7.3 when a fraction number of

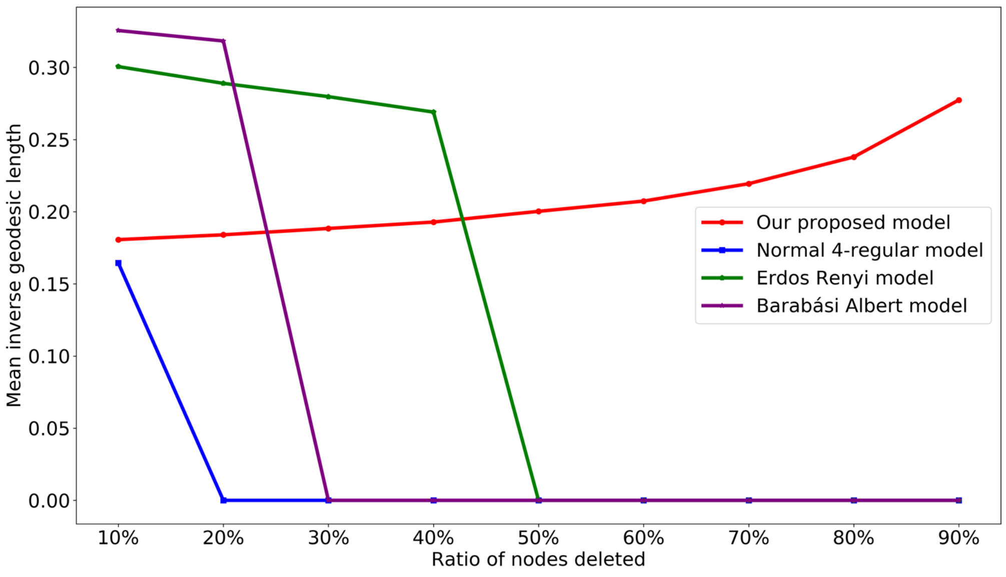


Fig. 7.4: Mean inverse shortest path when removing nodes

vertices are removed. As shown in Figure 7.4, the regular model decay exponentially after 20% of nodes is removed, while with the Erdos Renyi model, the rate is 30%. This can be explained from the finding that each node has approximately the same degree and thus contributes to the network by relatively the same amount. For the Barabási Albert model, the l^{-1} slightly decreases until 50% of node removal. This is of course, due to the large variation in the importance of the nodes, i.e., there exist significant vertices, hubs. These hubs act as a crucial role in network functionality. As long as the hubs are not eliminated, the connectivity of the network remains. On the contrary, in the regular model applying our proposed self-healing, as the nodes are deleted and the number of nodes decreases, the l^{-1} of the graph also slightly rises accordingly, even when 90% of the node is deleted. This should be interpreted as the network functionality remains after removing a large of nodes. Furthermore, as the number of nodes decreases, the average of inverse shortest path length rises, which mean the dynamics of the network increases. Taken together, these results show that our algorithm helps maintain connectivity and even increases the efficiency of the network when deleting vertices.

Robustness evaluation

To evaluate the robustness, we examine how resilient our network is to failure in the network, such as members being eliminated. We utilise some metrics that are used in graph theory, such as the closeness centrality, degree centrality and a number of the connected component after nodes removal.

In the experiment, the closeness centrality of a single vertex can't reflect the whole network. Hence, the mean centrality of the whole network is obtained during the test, which is calculated by equation 7.4.

$$\mu_{C_C} = \frac{\sum_{i=1}^{n} C(x_i)}{n} \tag{7.4}$$

where n is the number of vertices, $C(x_i)$ is the *closeness centrality* of vertex x_i.

In our simulations, the average degree of all vertices can be utilised as a test indicator and calculated by equation 7.5

$$\mu_{C_D} = \frac{\sum_{i=1}^{n} C'_D(i)}{n} \tag{7.5}$$

To examine our proposed algorithm's ability to repairing the network, some experiments are deployed. We simulate the node removal process in the network of 1000 nodes, with up to 90% (900) node deletions. Four models are utilised including two k-regular (k=4), an Erdos Renyi model, and an Barabási Albert model. In these models, one k-regular model applies our proposed self-repairing algorithm while the others use a naive self-repairing algorithm (each node may add edges joining it to any other neighbour nodes as desired). The figure 7.5 illustrates the mean degree centrality and the mean closeness centrality when deleting nodes. From the figure, it can be seen that the model apply our algorithm to keep the mean centralities stable. This result may be explained by the fact that our method keeps the degrees of the nodes in the bound constraint during the repairing process. On the other hand, in figure 7.5, there is a clear trend of increasing the degree and closeness centrality in the models that utilise the naive self-healing algorithm. From the result, we can see that the naive self-healing approach cause high degree increase (which may lead to overload and eventual network breakdown) or increase in distances between nodes (which may lead to poor communication). Whereas, our proposed method keep the metrics stable, and even after 90% of node removal, the degree centrality and closeness slightly increase. Low degree centrality is desirable because it decreases the chances of detection and takes down.

In order to understand how node removal affects the network, we simulate the nodes deletion process of four types of model: a normal 4-regular, an Erdos Renyi, an Barabási Albert, and finally a 4-regular model with self-healing mechanism. Figure 7.6 depicts the simulation result when removing nodes of two network of size 1000 (a) and 10000 (b), respectively. From the data in Figure 7.6, it is apparent that the self-repairing model remains connected even when a large portion (80%) of the nodes are deleted, compared to other types of the model (with no self-repairing mechanism). Note that, in a normal model after 30% node deletion, the number of partitions rise sharply. The similar phenomenon happens in Erdos Renyi and Barabási Albert model when removing 50% of nodes. From this result, we notice that the normal 4-regular model is the most vulnerable to random node removal whereas Erdos Renyi and Barabási Albert models have better tolerance with node deletion. On the contrary, the model with the self-healing mechanism is the most resilient network. It can, therefore, be assumed that our self-healing algorithm makes the k-regular network more resilient and robustness.

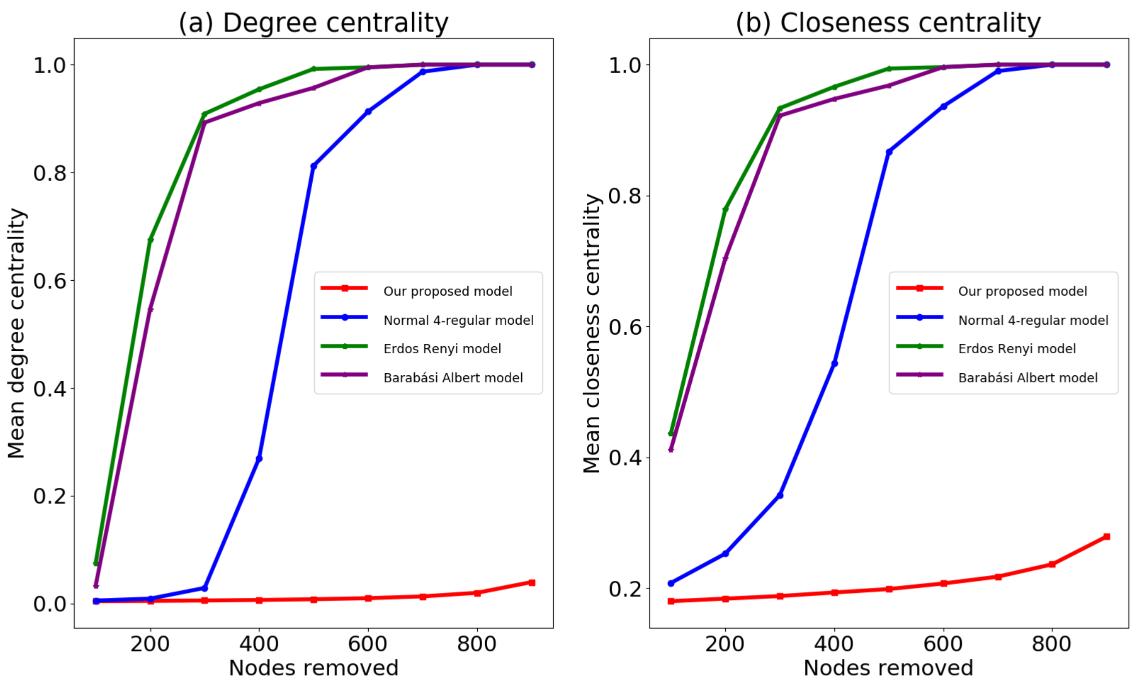


Fig. 7.5: Mean centrality of the network when removing node

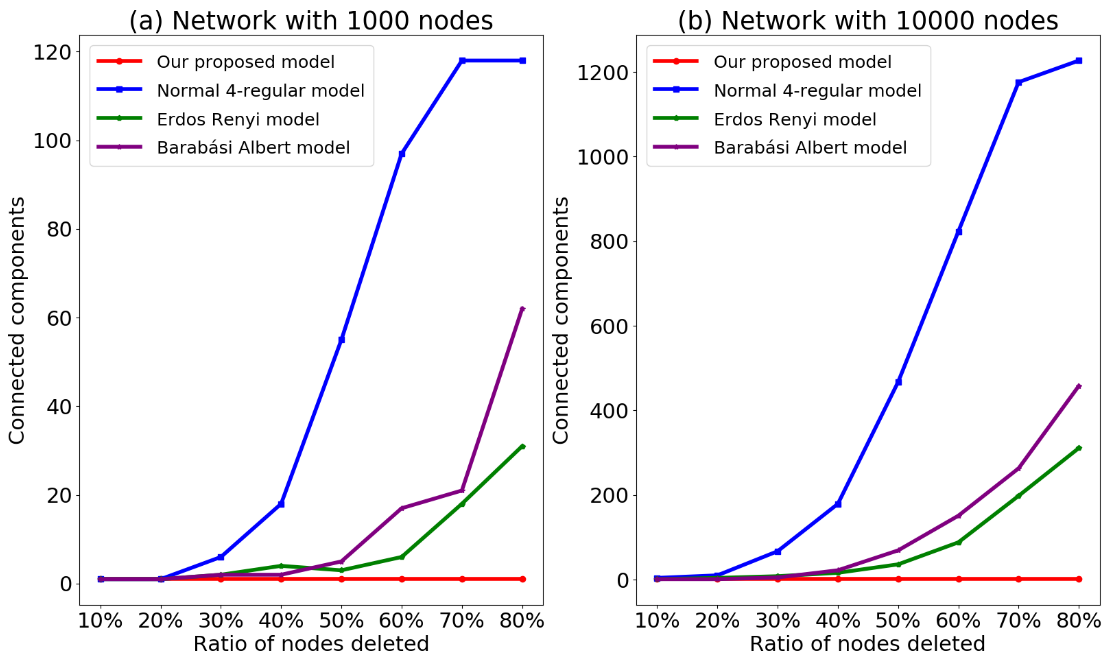
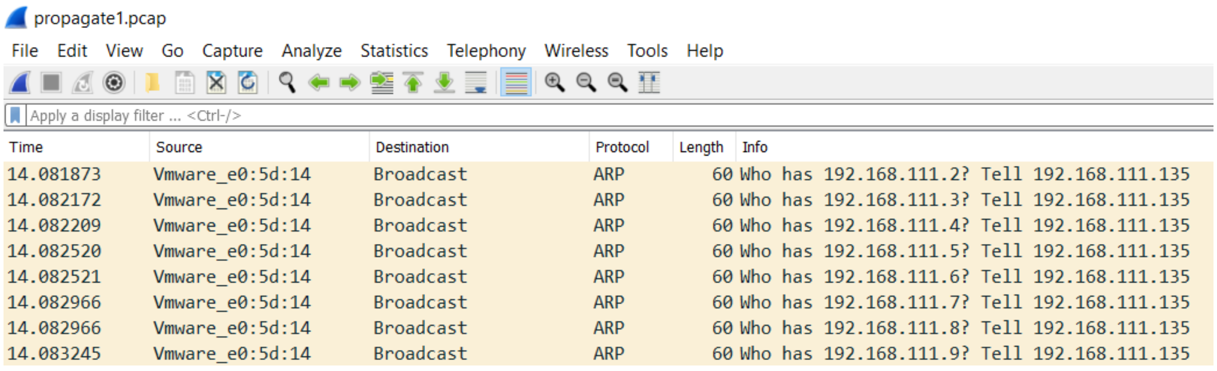


Fig. 7.6: The number of connected components after removing a fraction of nodes

7.5.2 Empirical experiments

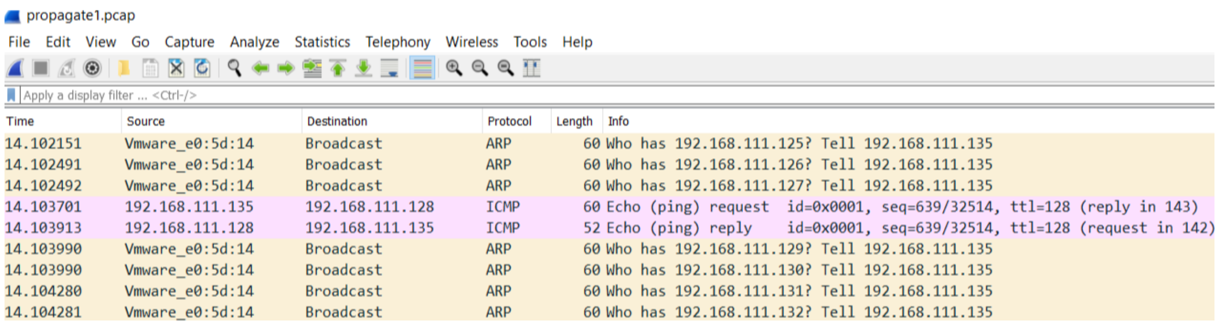
In this part, we execute the X-sWarm in the isolated environment to monitor its behaviour. In the presented case study, we obtained the behaviour data composed of the malware actions on the system network until the termination of the process. It was noticeable that these data were all typical network operations. The propagation begins with the worm send a ping request to all computer in the network as shown in Fig. 7.7. This step provides the information on whether the machine it is attacking supports the Common Internet File System (CIFS) protocol and is likely to be a Windows machine.


propagate1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
Time        Source             Destination  Protocol  Length  Info
14.081873   Vmware_e0:5d:14    Broadcast    ARP       60 Who has 192.168.111.2? Tell 192.168.111.135
14.082172   Vmware_e0:5d:14    Broadcast    ARP       60 Who has 192.168.111.3? Tell 192.168.111.135
14.082209   Vmware_e0:5d:14    Broadcast    ARP       60 Who has 192.168.111.4? Tell 192.168.111.135
14.082520   Vmware_e0:5d:14    Broadcast    ARP       60 Who has 192.168.111.5? Tell 192.168.111.135
14.082521   Vmware_e0:5d:14    Broadcast    ARP       60 Who has 192.168.111.6? Tell 192.168.111.135
14.082966   Vmware_e0:5d:14    Broadcast    ARP       60 Who has 192.168.111.7? Tell 192.168.111.135
14.082966   Vmware_e0:5d:14    Broadcast    ARP       60 Who has 192.168.111.8? Tell 192.168.111.135
14.083245   Vmware_e0:5d:14    Broadcast    ARP       60 Who has 192.168.111.9? Tell 192.168.111.135
```

Fig. 7.7: Scan IP address

If a host reply with a ping success as illustrate in Fig. 7.8 then the worm uses the SMB protocol to enumerate the list of account names on the remote machine. It also establishes some basic properties about the user for guessing the password process and attempt to connect to them in Fig. 7.9.

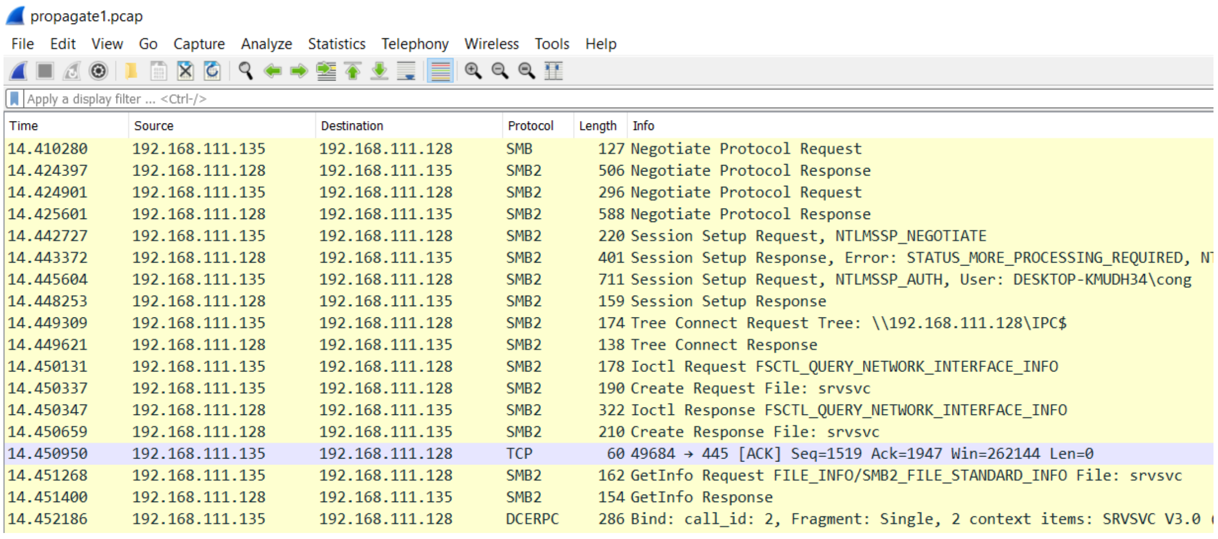

propagate1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
Time        Source             Destination        Protocol  Length  Info
14.102151   Vmware_e0:5d:14    Broadcast          ARP       60 Who has 192.168.111.125? Tell 192.168.111.135
14.102491   Vmware_e0:5d:14    Broadcast          ARP       60 Who has 192.168.111.126? Tell 192.168.111.135
14.102492   Vmware_e0:5d:14    Broadcast          ARP       60 Who has 192.168.111.127? Tell 192.168.111.135
14.103701   192.168.111.135    192.168.111.128    ICMP      60 Echo (ping) request  id=0x0001, seq=639/32514, ttl=128 (reply in 143)
14.103913   192.168.111.128    192.168.111.135    ICMP      52 Echo (ping) reply    id=0x0001, seq=639/32514, ttl=128 (request in 142)
14.103990   Vmware_e0:5d:14    Broadcast          ARP       60 Who has 192.168.111.129? Tell 192.168.111.135
14.103990   Vmware_e0:5d:14    Broadcast          ARP       60 Who has 192.168.111.130? Tell 192.168.111.135
14.104280   Vmware_e0:5d:14    Broadcast          ARP       60 Who has 192.168.111.131? Tell 192.168.111.135
14.104281   Vmware_e0:5d:14    Broadcast          ARP       60 Who has 192.168.111.132? Tell 192.168.111.135
```

Fig. 7.8: Ping reply

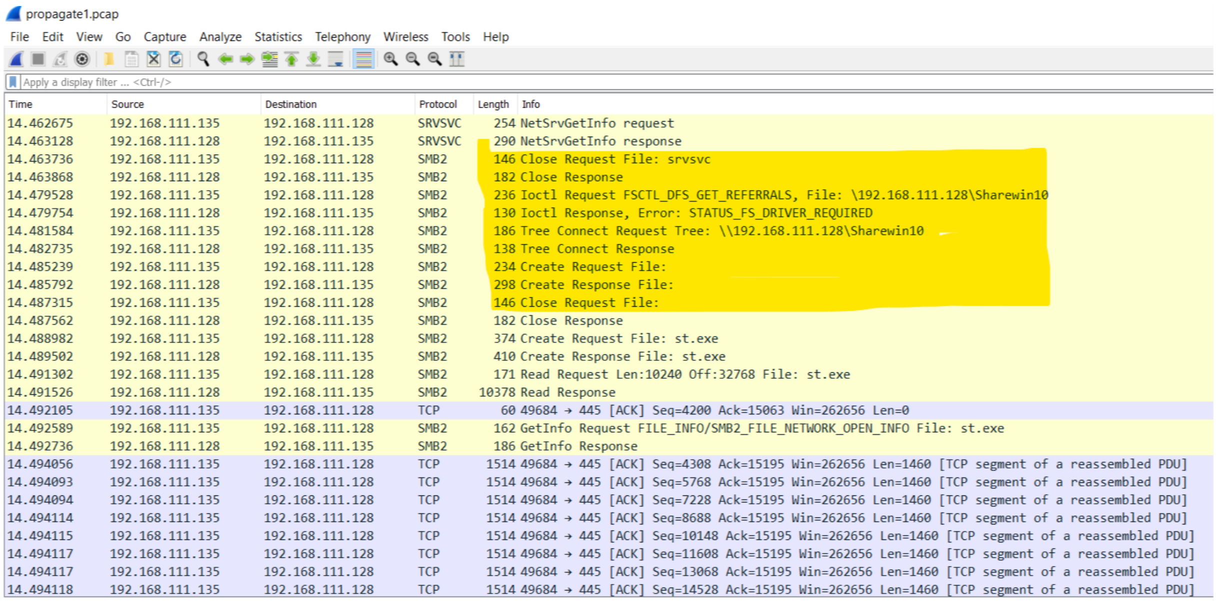
After that, it makes an SMB connection with the target computers, attempting to access the IPC$ connection.

If the worm successfully connects to the IPC$ share, it copies itself over to the remote machine. More precise, the worm will search the shared folder on victim, if a shared folder is



```
Time       Source           Destination      Protocol Length Info
14.410280  192.168.111.135  192.168.111.128  SMB        127 Negotiate Protocol Request
14.424397  192.168.111.128  192.168.111.135  SMB2       506 Negotiate Protocol Response
14.424901  192.168.111.135  192.168.111.128  SMB2       296 Negotiate Protocol Request
14.425601  192.168.111.128  192.168.111.135  SMB2       588 Negotiate Protocol Response
14.442727  192.168.111.135  192.168.111.128  SMB2       220 Session Setup Request, NTLMSSP_NEGOTIATE
14.443372  192.168.111.128  192.168.111.135  SMB2       401 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, N
14.445604  192.168.111.135  192.168.111.128  SMB2       711 Session Setup Request, NTLMSSP_AUTH, User: DESKTOP-KMUDH34\cong
14.448253  192.168.111.128  192.168.111.135  SMB2       159 Session Setup Response
14.449309  192.168.111.135  192.168.111.128  SMB2       174 Tree Connect Request Tree: \\192.168.111.128\IPC$
14.449621  192.168.111.128  192.168.111.135  SMB2       138 Tree Connect Response
14.450131  192.168.111.135  192.168.111.128  SMB2       178 Ioctl Request FSCTL_QUERY_NETWORK_INTERFACE_INFO
14.450337  192.168.111.135  192.168.111.128  SMB2       190 Create Request File: srvsvc
14.450347  192.168.111.128  192.168.111.135  SMB2       322 Ioctl Response FSCTL_QUERY_NETWORK_INTERFACE_INFO
14.450659  192.168.111.128  192.168.111.135  SMB2       210 Create Response File: srvsvc
14.450950  192.168.111.135  192.168.111.128  TCP         60 49684 → 445 [ACK] Seq=1519 Ack=1947 Win=262144 Len=0
14.451268  192.168.111.135  192.168.111.128  SMB2       162 GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO File: srvsvc
14.451400  192.168.111.128  192.168.111.135  SMB2       154 GetInfo Response
14.452186  192.168.111.135  192.168.111.128  DCERPC     286 Bind: call_id: 2, Fragment: Single, 2 context items: SRVSVC V3.0
```

Fig. 7.9: Connect with the victim

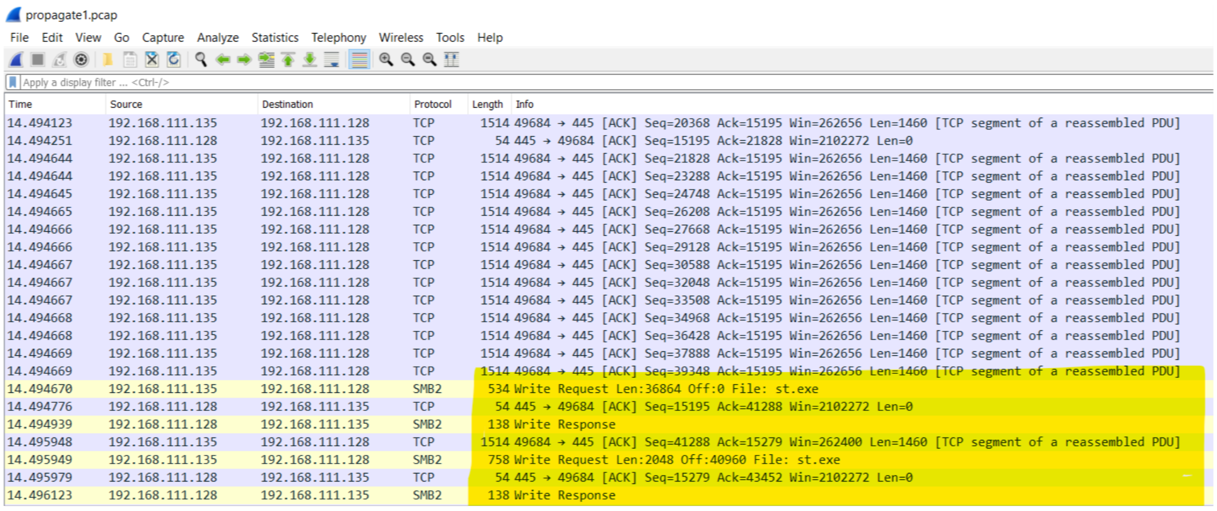


```
Time       Source           Destination      Protocol Length Info
14.462675  192.168.111.135  192.168.111.128  SRVSVC     254 NetSrvGetInfo request
14.463128  192.168.111.128  192.168.111.135  SRVSVC     290 NetSrvGetInfo response
14.463736  192.168.111.135  192.168.111.128  SMB2       146 Close Request File: srvsvc
14.463868  192.168.111.128  192.168.111.135  SMB2       182 Close Response
14.479528  192.168.111.135  192.168.111.128  SMB2       236 Ioctl Request FSCTL_DFS_GET_REFERRALS, File: \192.168.111.128\Sharewin10
14.479754  192.168.111.128  192.168.111.135  SMB2       130 Ioctl Response, Error: STATUS_FS_DRIVER_REQUIRED
14.481584  192.168.111.135  192.168.111.128  SMB2       186 Tree Connect Request Tree: \\192.168.111.128\Sharewin10
14.482735  192.168.111.128  192.168.111.135  SMB2       138 Tree Connect Response
14.485239  192.168.111.135  192.168.111.128  SMB2       234 Create Request File:
14.485792  192.168.111.128  192.168.111.135  SMB2       298 Create Response File:
14.487315  192.168.111.135  192.168.111.128  SMB2       146 Close Request File:
14.487562  192.168.111.128  192.168.111.135  SMB2       182 Close Response
14.488982  192.168.111.135  192.168.111.128  SMB2       374 Create Request File: st.exe
14.489502  192.168.111.128  192.168.111.135  SMB2       410 Create Response File: st.exe
14.491302  192.168.111.135  192.168.111.128  SMB2       171 Read Request Len:10240 Off:32768 File: st.exe
14.491526  192.168.111.128  192.168.111.135  SMB2     10378 Read Response
14.492105  192.168.111.135  192.168.111.128  TCP         60 49684 → 445 [ACK] Seq=4200 Ack=15063 Win=262656 Len=0
14.492589  192.168.111.135  192.168.111.128  SMB2       162 GetInfo Request FILE_INFO/SMB2_FILE_NETWORK_OPEN_INFO File: st.exe
14.492736  192.168.111.128  192.168.111.135  SMB2       186 GetInfo Response
14.494056  192.168.111.135  192.168.111.128  TCP       1514 49684 → 445 [ACK] Seq=4308 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494093  192.168.111.135  192.168.111.128  TCP       1514 49684 → 445 [ACK] Seq=5768 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494094  192.168.111.135  192.168.111.128  TCP       1514 49684 → 445 [ACK] Seq=7228 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494114  192.168.111.135  192.168.111.128  TCP       1514 49684 → 445 [ACK] Seq=8688 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494115  192.168.111.135  192.168.111.128  TCP       1514 49684 → 445 [ACK] Seq=10148 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494117  192.168.111.135  192.168.111.128  TCP       1514 49684 → 445 [ACK] Seq=11608 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494117  192.168.111.135  192.168.111.128  TCP       1514 49684 → 445 [ACK] Seq=13068 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494118  192.168.111.135  192.168.111.128  TCP       1514 49684 → 445 [ACK] Seq=14528 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
```

Fig. 7.10: Search shared folder

discovered then the worm will infect this victim as shown in Fig. 7.10 and Fig. 7.11. After the copy process, the worm start the a process to run itself on the target machine.

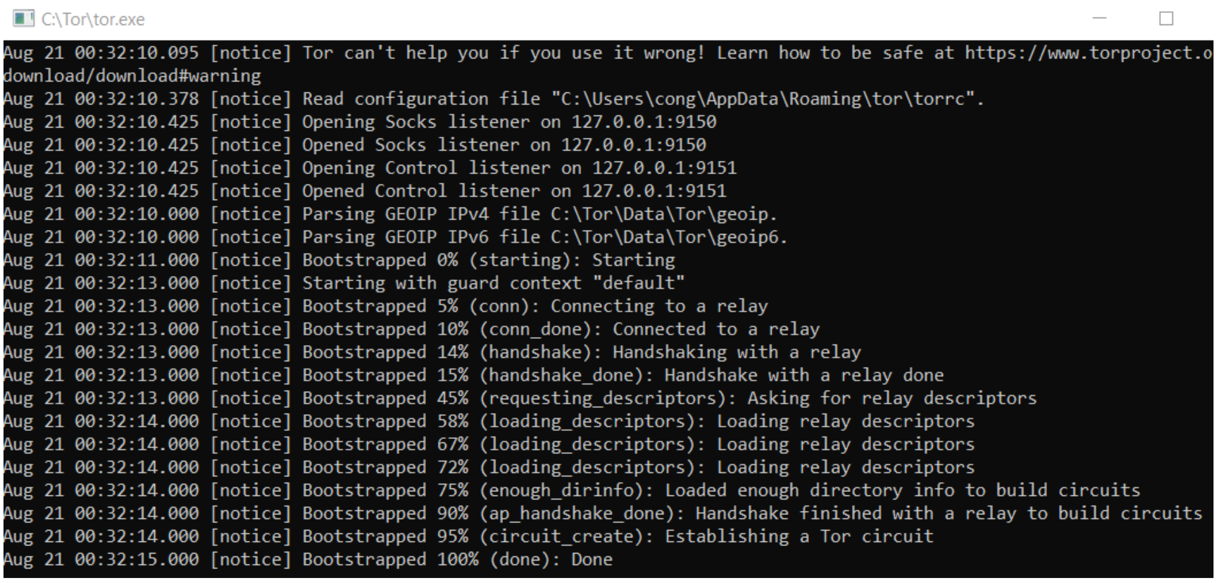
Tor allows to anonymously distribute services inside the network, which are called hidden services. In our experiments, the hidden services are established to allow the communication of the worm. By using the .onion address, one worm can contact with the others. First, it creates a new virtual circuit to a random relay asking it to act as a rendezvous point. Then it utilises the introduction points to notify the hidden service about the rendezvous point. Finally, the hidden service generates a virtual circuit to the rendezvous point and commences the communication


propagate1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

Time	Source	Destination	Protocol	Length	Info
14.494123	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=20368 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494251	192.168.111.128	192.168.111.135	TCP	54	445 → 49684 [ACK] Seq=15195 Ack=21828 Win=2102272 Len=0
14.494644	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=21828 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494644	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=23288 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494645	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=24748 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494665	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=26208 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494666	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=27668 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494666	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=29128 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494667	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=30588 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494667	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=32048 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494667	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=33508 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494668	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=34968 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494668	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=36428 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494669	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=37888 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494669	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=39348 Ack=15195 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
14.494670	192.168.111.135	192.168.111.128	SMB2	534	Write Request Len:36864 Off:0 File: st.exe
14.494776	192.168.111.128	192.168.111.135	TCP	54	445 → 49684 [ACK] Seq=15195 Ack=41288 Win=2102272 Len=0
14.494939	192.168.111.128	192.168.111.135	SMB2	138	Write Response
14.495948	192.168.111.135	192.168.111.128	TCP	1514	49684 → 445 [ACK] Seq=41288 Ack=15279 Win=262400 Len=1460 [TCP segment of a reassembled PDU]
14.495949	192.168.111.135	192.168.111.128	SMB2	758	Write Request Len:2048 Off:40960 File: st.exe
14.495979	192.168.111.128	192.168.111.135	TCP	54	445 → 49684 [ACK] Seq=15279 Ack=43452 Win=2102272 Len=0
14.496123	192.168.111.128	192.168.111.135	SMB2	138	Write Response

Fig. 7.11: Infect the victim

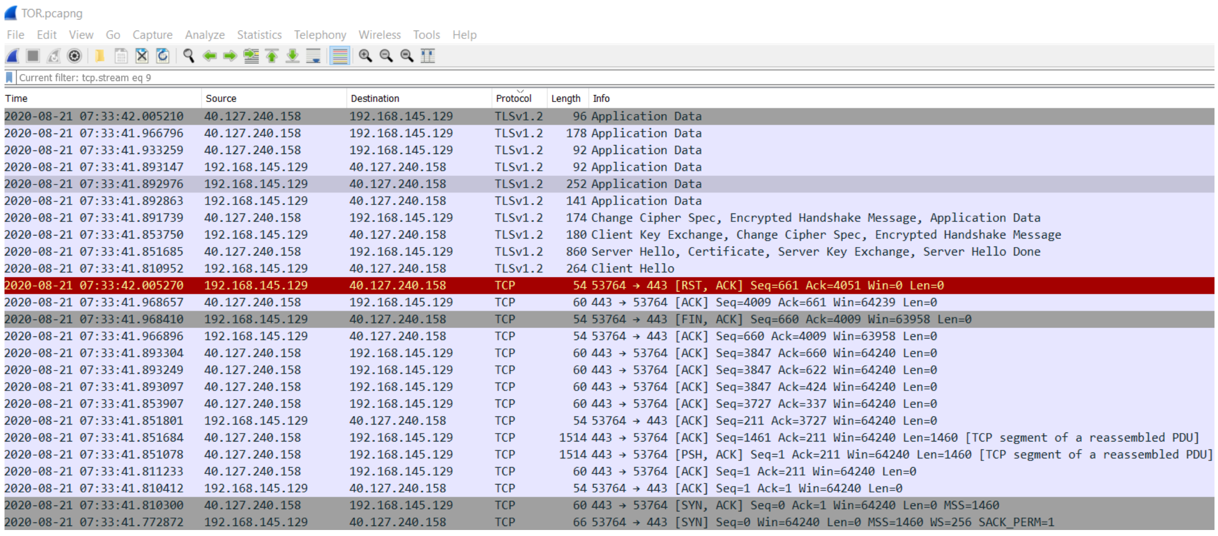
channel. The figure 7.12 demonstrates the process of creating virtual circuit.



```
Aug 21 00:32:10.095 [notice] Tor can't help you if you use it wrong! Learn how to be safe at https://www.torproject.o
download/download#warning
Aug 21 00:32:10.378 [notice] Read configuration file "C:\Users\cong\AppData\Roaming\tor\torrc".
Aug 21 00:32:10.425 [notice] Opening Socks listener on 127.0.0.1:9150
Aug 21 00:32:10.425 [notice] Opened Socks listener on 127.0.0.1:9150
Aug 21 00:32:10.425 [notice] Opening Control listener on 127.0.0.1:9151
Aug 21 00:32:10.425 [notice] Opened Control listener on 127.0.0.1:9151
Aug 21 00:32:10.000 [notice] Parsing GEOIP IPv4 file C:\Tor\Data\Tor\geoip.
Aug 21 00:32:10.000 [notice] Parsing GEOIP IPv6 file C:\Tor\Data\Tor\geoip6.
Aug 21 00:32:11.000 [notice] Bootstrapped 0% (starting): Starting
Aug 21 00:32:13.000 [notice] Starting with guard context "default"
Aug 21 00:32:13.000 [notice] Bootstrapped 5% (conn): Connecting to a relay
Aug 21 00:32:13.000 [notice] Bootstrapped 10% (conn_done): Connected to a relay
Aug 21 00:32:13.000 [notice] Bootstrapped 14% (handshake): Handshaking with a relay
Aug 21 00:32:13.000 [notice] Bootstrapped 15% (handshake_done): Handshake with a relay done
Aug 21 00:32:13.000 [notice] Bootstrapped 45% (requesting_descriptors): Asking for relay descriptors
Aug 21 00:32:14.000 [notice] Bootstrapped 58% (loading_descriptors): Loading relay descriptors
Aug 21 00:32:14.000 [notice] Bootstrapped 67% (loading_descriptors): Loading relay descriptors
Aug 21 00:32:14.000 [notice] Bootstrapped 72% (loading_descriptors): Loading relay descriptors
Aug 21 00:32:14.000 [notice] Bootstrapped 75% (enough_dirinfo): Loaded enough directory info to build circuits
Aug 21 00:32:14.000 [notice] Bootstrapped 90% (ap_handshake_done): Handshake finished with a relay to build circuits
Aug 21 00:32:14.000 [notice] Bootstrapped 95% (circuit_create): Establishing a Tor circuit
Aug 21 00:32:15.000 [notice] Bootstrapped 100% (done): Done
```

Fig. 7.12: The establishment of virtual circuit

After successfully establish the connection, the malware can connect with the others to retrieve the information. The communication between two members in the swarm is illustrated in Fig. 7.13. It should be noticeable that Tor uses the SSL/TLS protocol suite to establish encrypted connections between participating nodes, just as it is commonly used by web browsers, email clients and others.


TOR.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Current filter: tcp.stream eq 9

```
Time                        Source            Destination       Protocol  Length Info
2020-08-21 07:33:42.005210  40.127.240.158    192.168.145.129   TLSv1.2     96 Application Data
2020-08-21 07:33:41.966796  40.127.240.158    192.168.145.129   TLSv1.2    178 Application Data
2020-08-21 07:33:41.933259  40.127.240.158    192.168.145.129   TLSv1.2     92 Application Data
2020-08-21 07:33:41.893147  192.168.145.129   40.127.240.158    TLSv1.2     92 Application Data
2020-08-21 07:33:41.892976  192.168.145.129   40.127.240.158    TLSv1.2    252 Application Data
2020-08-21 07:33:41.892863  192.168.145.129   40.127.240.158    TLSv1.2    141 Application Data
2020-08-21 07:33:41.891739  40.127.240.158    192.168.145.129   TLSv1.2    174 Change Cipher Spec, Encrypted Handshake Message, Application Data
2020-08-21 07:33:41.853750  192.168.145.129   40.127.240.158    TLSv1.2    180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2020-08-21 07:33:41.851685  40.127.240.158    192.168.145.129   TLSv1.2    860 Server Hello, Certificate, Server Key Exchange, Server Hello Done
2020-08-21 07:33:41.810952  192.168.145.129   40.127.240.158    TLSv1.2    264 Client Hello
2020-08-21 07:33:42.005270  192.168.145.129   40.127.240.158    TCP         54 53764 → 443 [RST, ACK] Seq=661 Ack=4051 Win=0 Len=0
2020-08-21 07:33:41.968657  40.127.240.158    192.168.145.129   TCP         60 443 → 53764 [ACK] Seq=4009 Ack=661 Win=64239 Len=0
2020-08-21 07:33:41.968410  192.168.145.129   40.127.240.158    TCP         54 53764 → 443 [FIN, ACK] Seq=660 Ack=4009 Win=63958 Len=0
2020-08-21 07:33:41.966896  192.168.145.129   40.127.240.158    TCP         54 53764 → 443 [ACK] Seq=660 Ack=4009 Win=63958 Len=0
2020-08-21 07:33:41.893304  40.127.240.158    192.168.145.129   TCP         60 443 → 53764 [ACK] Seq=3847 Ack=660 Win=64240 Len=0
2020-08-21 07:33:41.893249  40.127.240.158    192.168.145.129   TCP         60 443 → 53764 [ACK] Seq=3847 Ack=622 Win=64240 Len=0
2020-08-21 07:33:41.893097  40.127.240.158    192.168.145.129   TCP         60 443 → 53764 [ACK] Seq=3847 Ack=424 Win=64240 Len=0
2020-08-21 07:33:41.853907  40.127.240.158    192.168.145.129   TCP         60 443 → 53764 [ACK] Seq=3727 Ack=337 Win=64240 Len=0
2020-08-21 07:33:41.851801  192.168.145.129   40.127.240.158    TCP         54 53764 → 443 [ACK] Seq=211 Ack=3727 Win=64240 Len=0
2020-08-21 07:33:41.851684  40.127.240.158    192.168.145.129   TCP       1514 443 → 53764 [ACK] Seq=1461 Ack=211 Win=64240 Len=1460 [TCP segment of a reassembled PDU]
2020-08-21 07:33:41.851078  40.127.240.158    192.168.145.129   TCP       1514 443 → 53764 [PSH, ACK] Seq=1 Ack=211 Win=64240 Len=1460 [TCP segment of a reassembled PDU]
2020-08-21 07:33:41.811233  40.127.240.158    192.168.145.129   TCP         60 443 → 53764 [ACK] Seq=1 Ack=211 Win=64240 Len=0
2020-08-21 07:33:41.810412  192.168.145.129   40.127.240.158    TCP         54 53764 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
2020-08-21 07:33:41.810300  40.127.240.158    192.168.145.129   TCP         60 443 → 53764 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
2020-08-21 07:33:41.772872  192.168.145.129   40.127.240.158    TCP         66 53764 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
```

Fig. 7.13: Communication channel

7.6 Countermeasures

In this section, we discuss some different mitigation strategies to counter against the X-sWarm. Mitigation and detection can take place at different levels, such as host level or network level.

7.6.1 Countermeasures for host level

Tor services frequently listen to several specific ports: ports 80, 443, 9001 and 9030, the default ports of the Tor protocol while it is running on an infected device. The communication is easily blocked by filtering network traffic if there is no application based on the Tor protocol on the infected device since X-sWarm relies on Tor for communicating. In principle, there are two feasible ways to do this. The first approach involves with control the traffic outbound to the Internet by the ports being used, such as block outbound traffic to specific ports, or limit the permitted outbound traffic to certain ports. The latter approach concerns with using network inspection techniques to try and determine which is legitimate traffic and which is malicious traffic. Nevertheless, blocking some commonly used ports may affect the user experience. Therefore, traffic filtering can be a temporary countermeasure.

7.6.2 Network level countermeasure idea

“Sybil attack“ is referred as a small number of entities forge multiple peer identities so as to compromise the peer-to-peer distributed systems [35]. In many P2P networks, the peers are feasible to join the network without authentication or validation of their identities. As a consequent, these P2P networks are vulnerable to Sybil attack.

In our work, we leverage the Sybil attack to form a countermeasure idea for the X-sWarm. To attack and break network down, the peer's onion addresses need to be obtained. This can be done either by detecting and reverse engineering an already infected host or by using a set of honeypots. After identifying the peer address, we run many hidden services, disclosing a subset of these as neighbours to each peer we encounter, so gradually over time our clone nodes dominate the neighbourhood of each peer and contain it.

7.7 Summary

In this chapter, we present X-sWarm, a novel design of malware with swarm characteristics, in which communication utilizes the Tor network. This design has shown that the combination between SI and Tor network producing a robust and stealthy malware that has the ability to evade detection, measurement, scale estimation and observation. Additionally, X-sWarm relies on a resilient self-healing network formation that is simple to implement, yet robust to partitioning, even up to 90% node removal the network is feasible self-recover. The results demonstrate the feasibility, stealthiness, and robustness of this new type of malware. More importantly, we suggest the countermeasure approach as the host and network level for this upcoming threat. These findings contribute in several ways to our understanding of X-sWarm and provide a basis for further research.

This research has also opened research directions. One potential research area is that the capability to extend the malware into a model based on multi-agent systems, in which the agents are embedded with more advance AI so that they can perform the task more effectively. On the other hand, based on the X-sWarm idea, we could develop the autonomous anti-malware technology in complex and large systems.

8 MULTI AGENT ANTIVIRUS SYSTEM: A PARADIGM TO DESIGN INTELLIGENT ANTI-MALWARE APPLICATIONS

8.1 Introduction

X-Ware's ideas are fully applicable to designing future anti-malware solutions. For instance, we can create adaptive, autonomous AI agents that collaborate with each other to achieve common tasks. Instead of getting guidance from a single, centralized AI model, agents will be smart and robust enough to communicate with each other and work together to achieve common goals. This will form a Multi agent anti-virus system (MAAS).

Agents will learn how to protect systems depend on what they inspect from their networks and local hosts. Furthermore, their strength is further enhanced by observations and behaviours learned across different industries and majors. Generally speaking, we will have a swarm of rapid response local AIs that accommodate to their environment while collaborating with each other, instead of one big AI system delivering decisions. This will improve the IT performance of organizations by saving resources, and also help them to avoid sharing confidential, potentially sensitive information through the cloud or other means.

8.2 Architecture

The proposed paradigm consists of several cooperating agents, which are classified by one of these roles: sensing, analysis, planning, and action execution. Furthermore, the system allows multiple agents for each role. Figure 8.1 illustrates the various roles, and gives an example of communication between the various agents.

On the other hand, the proposed system is intended to be dynamic: when the whole system operates, the agents that occupying the roles can be included or excluded. Regardless of the role of the agents, each agent must first announce its presence to the other agents existing in the system through a broadcast message. The message comprises a unique identifier of the new agent and an indicator of what role it will act. After that, existing agents in the system respond to the message, indicating their unique identifier and role, allowing agents to establish direct communication.

The ability for agents to join or leave the system in this way enhance the system's performance and resilience in the following ways.

- First, it allows the system to cover the sophisticated network topology. For example, in a network, specific segments may be covered by a variety of agents. Each agent can analyse network traffic independently, providing the network's information to other agents based on their demands.
- Next, it enables resource management. The proposed system can be deployed on several different devices in the protected network. Nevertheless, each device may have other processing priorities. When processing resources are limited, the agents can leave the

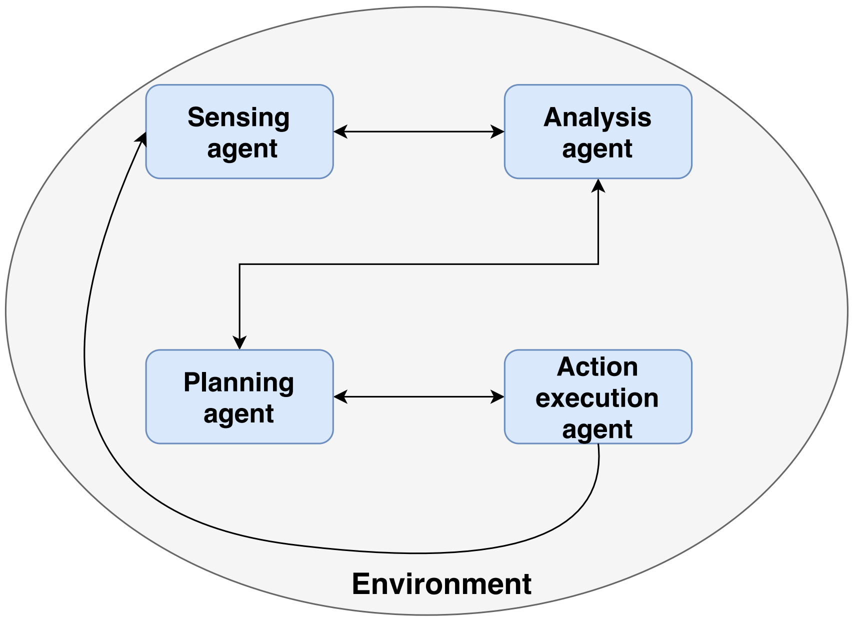


Fig. 8.1: Architecture of the Multi agent antivirus system

system in order to return the resources to the host device. And when the host device's resources are available, the agent can re-join the network.

- Next, multiple agents provide redundancy. If one agent malfunctions, or even becomes compromised by the adversaries, the other agents can be instantiated to take its place dynamically.
- Finally, this mechanism allows for designing and implementing a new type of agent, deploying it into the existing system, and proactively excluding the agent if it does not perform as supposed.

8.3 Type of agents

8.3.1 Sensing agents

This type of agent can acquire data from the environment and systems in which it operates, as well as from itself, to obtain the knowledge of the current state of the environment and trigger the specific analysis agent. Sensing agents operate based on data sources from external and internal sources. The internal data derive from agent-related information, while the external data are gathered from the protected system's resources like memory, file system, and others. Additionally, the external data also comes from observing the environment outside the agent.

Technically, sensing agents act as the roles of system and network monitoring tools. These agents collect logs and metrics from the agent's other internal systems, the underlying host system, and relevant applications running on the host. Sensing agents is also capable of capturing network traffic from the host network interfaces. For the sensory data to be beneficial to the rest of the system, they should go through normalization process so that way, unique and relevant information are transferred.

8.3.2 Analysis agents

These agents process data obtained from sensing to assess the environment. When identifying an abnormal behavior, the analysis agents trigger the planning agents for appropriate response plans. Based on the sensing data and the knowledge of environment, it identifies anomalies in data that are derived from the sensing agents. In case of threat detection, the analysis agent will take appropriate steps for its actions.

8.3.3 Planning agents

These agents elaborate one to several action proposals and propose them to the action selection phase, which decides the action or set of steps to execute to resolve the threats previously identified by analysis agents. It may be possible that the planning agents may interact with other agents to proceed up with an optimal set of actions forming a global response strategy.

In the action selection phase, the proposed response plan is analyzed based on the data from the agent's current goals and the execution constraints and requirements. After that, an executable response plan is submitted to the Action Execution agent.

8.3.4 Action prosecution agents

Action prosecution agents decide on an executable response plan, monitor its execution and its effects, and provide other agents with the means to adjust the performance of their part of the response plan as and when needed.

8.4 Generic work flow

The MAAS workflow is summarized in the following graph Fig. 8.2 that shows the agent's generic process flow.

In this diagram, each component of the MAAS has its principal tasks as follows:

- Sensing and monitoring component: acquire data from the systems as well as data exchanged with other entities (i.e., other agents or human experts). Furthermore, this component monitors the execution of action plans that are deployed by action prosecution components.
- Analysis component: processes the collected data as follows: The current state of the environment and the agent itself. It also identifies the adversarial or suspicious events and anomalies in collected data.
- Planning component: proposes the response plans that are corresponded with the contemporary state identified before. After that, this component evaluates and chooses the most appropriate strategy.

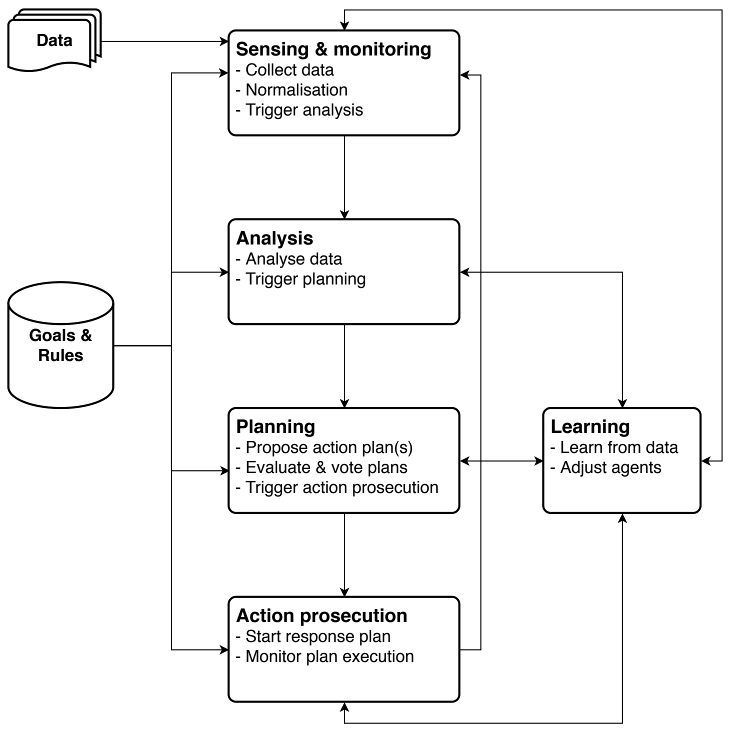


Fig. 8.2: MAAS generic workflow

- Action prosecution component: deploys the sequence actions corresponding to the plan. This component also notices the sensing component to monitor the execution of the action plan.
- Learning component: learns from the data acquired and stored by the agent and has an adjustment mechanism to improve the agent's performance.

8.5 Learning

Due to the continuous changing of the environment and the rapid evolvement of cyber-threats, the agent must be equipped with autonomous learning to adapt to the dynamic environment. The learning source of agents comes from the feedback data from the environment and the from the agent actions itself. To be more specific, the agent learns from experiences that could be obtained through actual confront with the cyber-threats or in simulations environment. There are several approaches for the learning process, one of such is by applying the concepts of Reinforcement Learning (RL) [117]. Next, we illustrative examples of learning from experience.

Let t denote the time, a denote the actions that are performed by the agent, and o denote the observations when the agent perform the actions. Let V denote the value of the state of the environment. Therefor the experiences of the agent can be represented by the sequence 8.1:

$$\sum_{t=1}^{n} a_t o_t V_t \tag{8.1}$$

We propose to utilise the experiences of the agent to train an ANN. In which, the inputs are the actions and observations for several preceding time points. On the other hand, the outputs are the value associated with taking that action as the next action. The next action will be calculated based on the trained ANN.

The following is an example scenario to help explain how the ANN work. Supposedly, at a specific time point, the agent can perform four actions: $a1, a2, a3$, and $a4$. At any specific time points, the agent can receive five observations: $o1, o2, o3, o4$, and $o5$. Alongside that, in this scenario, only these time points are considered: the most recent time when the agent takes action and the previous time. Note that in practical implementations, there could be hundreds of actions, observations, and multiple time points. Supposedly, the most recent action is $a1$ and the corresponding observations $o3$. In the previous time point, the agent performed $a3$ and received $o5$. These serve as the input data. After that, the ANN calculates value associated with the actions. Accordingly, the rewards for the possible next action are: $(a1, 0.05), (a2, 0.04), (a3, 0.14), (a4, 0.74)$. In this case, the agent will choose the action with highest reward, action $a4$.

8.6 Summary

In this chapter we propose a paradigm of a Multi agent anti-virus system based on the concept idea of the swarm malware. We believe that using the MAAS allows to build more robust, adaptive and flexible defense system. This may be considered a promising aspect of an insight for the autonomous anti-malware technology (i.e. Deux ex Machina) in complex and large systems.

In future work, we will continue to investigate more details about our paradigm for developing it and evaluating with much realistic examples. This will strengthen our intended approach towards intelligent anti-malware solution and will enable us to better exploit the design process for mapping requirements at run-time.

9 CONCLUSIONS AND FUTURE WORK

9.1 Summary of Results and Insights

There are four main goals of the thesis as follows. Parts of the thesis, which are dedicated to Goal 1, we can find in Chapter 5, where the application of the AI-based technique in cybersecurity issues was examined. Specifically, the application of AI in malware detection, intrusion detection, APT, and other domains such as spam detection, phishing detection was discussed. Furthermore, a vision of how AI could be adopted for malicious use was offered. Goal 2 and Goal 3 was presented in chapter 6, where the methodology to develop a prototype of X-Ware that combine the SI principle and the ANN was described. Furthermore, practical experiments to visualise, measure, and analyse the behaviour of the X-Ware under the form of a complex network were performed. Goal 4 was presented in Chapter 8, where the structure and the principles of the possible swarm anti-malware solution were introduced.

All four goals of the dissertation specified in Chapter 2 were fulfilled. Goal 1 was published in [rel1], [rel2], [rel3], [rel4], [rel6], in which we discussed the impact of AI, SI in cybersecurity domain. Goal 2, Goal 3 and Goal 4 was published in [rel5], [rel7], in which we proposed methodology to develop a prototype of X-Ware, X-sWarm as well as suggested a possible anti-malware solution.

This dissertation was aimed at exploring the possibilities of employing artificial intelligence in cybersecurity. In contemporary research, the primary targets for AI application in cybersecurity are network intrusion detection, malware analysis and classification, phishing, and spam emails. In those areas, the adoption of DL gradually become the primary trend. Furthermore, the combination of other intelligent techniques, such as bio-inspired methods, together with ML / DL, also attracted the attention of researchers. Such combinations yield very promising results and continue a trend for further research. Although the role of AI in resolving cybersecurity matters continues being researched, some of the problems that exist around the deployment of AI-based defences are also striking. For instance, the adversarial attack against the AI models or the emergence of autonomous intelligent malware.

By combining the swarm intelligence principle, ANN and a convention virus, the author proposed X-Ware, which was an improvement of the work in [145]. The results yielded from this work offer a better understanding of the behaviour of a possible new generation of malware in order to protect future computer technology. As this work has shown, the X-Ware prototype is a swarm one in which all individual viruses are capable of communicating amongst themselves as swarm less-more do. Base on the X-Ware, the author modified it as a worm, named X-sWarm, in which communication utilized the Tor network. This design had shown that the combination between SI and Tor network producing a robust and stealthy malware that had the ability to evade detection, measurement, scale estimation and observation.

More importantly, the concept idea of the swarm malware inspired us to proposed a paradigm of a Multi agent anti-virus system. Instead of getting guidance from a single, centralized AI model, agents would be smart and robust enough to communicate with each other and work together to achieve common goals. We believe that using the proposed paradigm allowed to build more robust, adaptive and flexible defense system. This may be considered a promising aspect of an insight for the autonomous anti-malware technology

9.2 Future work

To conclude the thesis, some works in the future are mentioned with the details as follows.

In the dissertation, we conduct a comprehensive survey about how AI can be used in cyber-security. Accordingly, the combination of several AI-based techniques in a defence solution is an exciting research direction.

In chapter 6, we proposed the X-Ware and extend it as X-sWarm in Chapter 7. Nevertheless, there is an emerging trend that malware resides in volatile system areas such as the system registry, in-memory processes and service areas and often know as the name “fileless“ malware. On this point, transforming the current X-Ware and X-sWarm into the form of “fileless“ is the next step for further research.

BIBLIOGRAPHY

[1] 27032, I.: Information technology–security techniques–guidelines for cybersecurity (2012)

[2] Ab Razak, M.F., Anuar, N.B., Othman, F., Firdaus, A., Afifi, F., Salleh, R.: Bio-inspired for features optimization and malware detection. Arabian Journal for Science and Engineering 43(12), 6963–6979 (2018)

[3] Al-Yaseen, W.L., Othman, Z.A., Nazri, M.Z.A.: Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system. Expert Systems with Applications 67, 296–303 (2017)

[4] Alejandre, F.V., Cortés, N.C., Anaya, E.A.: Feature selection to detect botnets using machine learning algorithms. In: 2017 International Conference on Electronics, Communications and Computers (CONIELECOMP). pp. 1–7. IEEE (2017)

[5] Ali, M.H., Al Mohammed, B.A.D., Ismail, A., Zolkipli, M.F.: A new intrusion detection system based on fast learning network and particle swarm optimization. IEEE Access 6, 20255–20261 (2018)

[6] Altaher, A., Barukab, O.M.: Intelligent hybrid approach for android malware detection based on permissions and api calls. International Journal of Advanced Computer Science and Applications 8(6), 60–67 (2017)

[7] Anderson, H.S., Kharkar, A., Filar, B., Evans, D., Roth, P.: Learning to evade static pe machine learning malware models via reinforcement learning. arXiv preprint arXiv:1801.08917 (2018)

[8] Anderson, H.S., Kharkar, A., Filar, B., Roth, P.: Evading machine learning malware detection. Black Hat (2017)

[9] Anderson, H.S., Woodbridge, J., Filar, B.: Deepdga: Adversarially-tuned domain generation and detection. In: Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security. pp. 13–21. ACM (2016)

[10] Apruzzese, G., Colajanni, M., Ferretti, L., Guido, A., Marchetti, M.: On the effectiveness of machine and deep learning for cyber security. In: 2018 10th International Conference on Cyber Conflict (CyCon). pp. 371–390. IEEE (2018)

[11] Ashfaq, R.A.R., Wang, X.Z., Huang, J.Z., Abbas, H., He, Y.L.: Fuzziness based semi-supervised learning approach for intrusion detection system. Information Sciences 378, 484–497 (2017)

[12] Aswani, R., Kar, A.K., Ilavarasan, P.V.: Detection of spammers in twitter marketing: a hybrid approach using social media analytics and bio inspired computing. Information Systems Frontiers 20(3), 515–530 (2018)

[13] Aycock, J.: Computer viruses and malware, vol. 22. Springer Science & Business Media (2006)

[14] Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science 286(5439), 509–512 (1999), `https://science.sciencemag.org/content/286/5439/509`

[15] Beni, G., Wang, J.: Swarm intelligence in cellular robotics systems. In: Proceedings of NATO Advanced Workshop on Robots and Biological System. pp. 703–712 (1989)

[16] Berman, D.S., Buczak, A.L., Chavis, J.S., Corbett, C.L.: A survey of deep learning methods for cyber security. Information 10(4), 122 (2019)

[17] Bhattacharya, A., Goswami, R.T., Mukherjee, K.: A feature selection technique based on rough set and improvised pso algorithm (psors-fs) for permission based detection of android malwares. International Journal of Machine Learning and Cybernetics 10(7), 1893–1907 (2019)

[18] Bianconi, G., Darst, R.K., Iacovacci, J., Fortunato, S.: Triadic closure as a basic generating mechanism of communities in complex networks. Physical Review E 90(4), 042806 (2014)

[19] Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.U.: Complex networks: Structure and dynamics. Physics reports 424(4-5), 175–308 (2006)

[20] Bonabeau, E., Marco, D.d.R.D.F., Dorigo, M., Theraulaz, G., et al.: Swarm intelligence: from natural to artificial systems. Oxford university press (1999)

[21] Botes, F.H., Leenen, L., De La Harpe, R.: Ant colony induced decision trees for intrusion detection. In: 16th European Conference on Cyber Warfare and Security. pp. 53–62 (2017)

[22] Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials 18(2), 1153–1176 (2015)

[23] Burnap, P., French, R., Turner, F., Jones, K.: Malware classification using self organising feature maps and machine activity data. computers & security 73, 399–410 (2018)

[24] Cani, A., Gaudesi, M., Sanchez, E., Squillero, G., Tonda, A.P.: Towards automated malware creation: code generation and code integration. In: SAC. pp. 157–160 (2014)

[25] Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., Song, D.: The secret sharer: Evaluating and testing unintended memorization in neural networks. In: 28th {USENIX} Security Symposium ({USENIX} Security 19). pp. 267–284 (2019)

[26] Chen, S., Xue, M., Fan, L., Hao, S., Xu, L., Zhu, H., Li, B.: Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. computers & security 73, 326–344 (2018)

[27] Chen, W., Liu, T., Tang, Y., Xu, D.: Multi-level adaptive coupled method for industrial control networks safety based on machine learning. Safety Science 120, 268–275 (2019)

[28] Chowdhury, M., Rahman, A., Islam, R.: Malware analysis and detection using data mining and machine learning classification. In: International Conference on Applications and Techniques in Cyber Security and Intelligence. pp. 266–274. Springer (2017)

[29] Christian, B., Daniel, M.: Swarm intelligence introduction and application. Natural Computing Series, Springer (2008)

[30] Chu, S.C., Tsai, P.W., Pan, J.S.: Cat swarm optimization. In: Pacific Rim international conference on artificial intelligence. pp. 854–858. Springer (2006)

[31] Cohen, F.: Computer viruses: theory and experiments. Computers & security 6(1), 22–35 (1987)

[32] Craigen, D., Diakun-Thibault, N., Purse, R.: Defining cybersecurity. Technology Innovation Management Review 4(10) (2014)

[33] Curtin, R.R., Gardner, A.B., Grzonkowski, S., Kleymenov, A., Mosquera, A.: Detecting dga domains with recurrent neural networks and side information. arXiv preprint arXiv:1810.02023 (2018)

[34] Dorigo, M., Bonabeau, E., Theraulaz, G.: Ant algorithms and stigmergy. Future Generation Computer Systems 16(8), 851–871 (2000)

[35] Douceur, J.R.: The sybil attack. In: International workshop on peer-to-peer systems. pp. 251–260. Springer (2002)

[36] Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. pp. 39–43. Ieee (1995)

[37] Engelbrecht, A.P.: Computational intelligence: an introduction. John Wiley & Sons (2007)

[38] Erdős, P., Rényi, A.: On the evolution of random graphs. Publ. Math. Inst. Hung. Acad. Sci 5(1), 17–60 (1960)

[39] Faghani, M.R., Matrawy, A., Lung, C.: A study of trojan propagation in online social networks. In: 2012 5th International Conference on New Technologies, Mobility and Security (NTMS). pp. 1–5 (2012)

[40] Faghani, M.R., Saidi, H.: Malware propagation in online social networks. In: 2009 4th International Conference on Malicious and Unwanted Software (MALWARE). pp. 8–14 (2009)

[41] Faris, H., Ala'M, A.Z., Heidari, A.A., Aljarah, I., Mafarja, M., Hassonah, M.A., Fujita, H.: An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks. Information Fusion 48, 67–83 (2019)

[42] Fatima, A., Maurya, R., Dutta, M.K., Burget, R., Masek, J.: Android malware detection using genetic algorithm based optimized feature selection and machine learning. In: 2019 42nd International Conference on Telecommunications and Signal Processing (TSP). pp. 220–223. IEEE (2019)

[43] Feng, F., Zhou, Q., Shen, Z., Yang, X., Han, L., Wang, J.: The application of a novel neural network in the detection of phishing websites. Journal of Ambient Intelligence and Humanized Computing pp. 1–15 (2018)

[44] Feng, W., Sun, J., Zhang, L., Cao, C., Yang, Q.: A support vector machine based naive bayes algorithm for spam filtering. In: 2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC). pp. 1–8. IEEE (2016)

[45] Fernandes, G., J., Carvalho, L., Rodrigues, J., ProenĂ§a, M.L., J.: Network anomaly detection using ip flows with principal component analysis and ant colony optimization. Journal of Network and Computer Applications 64, 1–11 (2016), cited By 28

[46] Filiol, E.: Strong cryptography armoured computer viruses forbidding code analysis: The Bradley virus. Ph.D. thesis, INRIA (2004)

[47] Filiol, E.: Computer viruses: from theory to applications. Springer Science & Business Media (2006)

[48] Freeman, L.C.: Centrality in social networks conceptual clarification. Social networks 1(3), 215–239 (1978)

[49] Gardner, M., Dorling, S.: Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric Environment 32(14), 2627 – 2636 (1998), `http://www.sciencedirect.com/science/article/pii/S1352231097004470`

[50] Garg, S., Batra, S.: Fuzzified cuckoo based clustering technique for network anomaly detection. Computers & Electrical Engineering 71, 798–817 (2018)

[51] Garg, S., Kaur, K., Kumar, N., Kaddoum, G., Zomaya, A.Y., Ranjan, R.: A hybrid deep learning based model for anomaly detection in cloud datacentre networks. IEEE Transactions on Network and Service Management (2019)

[52] Ghafir, I., Hammoudeh, M., Prenosil, V., Han, L., Hegarty, R., Rabie, K., Aparicio-Navarro, F.J.: Detection of advanced persistent threat using machine-learning correlation analysis. Future Generation Computer Systems 89, 349–359 (2018)

[53] Ghanem, T., Elkilani, W., Abdul-kader, H.: A hybrid approach for efficient anomaly detection using metaheuristic methods. Journal of Advanced Research 6(4), 609–619 (2015), cited By 30

[54] Grosse, K., Papernot, N., Manoharan, P., Backes, M., McDaniel, P.: Adversarial examples for malware detection. In: European Symposium on Research in Computer Security. pp. 62–79. Springer (2017)

[55] Gu, T., Chen, H., Chang, L., Li, L.: Intrusion detection system based on improved abc algorithm with tabu search. IEEJ Transactions on Electrical and Electronic Engineering (2019)

[56] Guan, Z., Bian, L., Shang, T., Liu, J.: When machine learning meets security issues: A survey. In: 2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR). pp. 158–165. IEEE (2018)

[57] Hajisalem, V., Babaie, S.: A hybrid intrusion detection system based on abc-afs algorithm for misuse and anomaly detection. Computer Networks 136, 37–50 (2018)

[58] Hamamoto, A.H., Carvalho, L.F., Sampaio, L.D.H., Abrão, T., Proença Jr, M.L.: Network anomaly detection system using genetic algorithm and fuzzy logic. Expert Systems with Applications 92, 390–402 (2018)

[59] Hashemi, H., Azmoodeh, A., Hamzeh, A., Hashemi, S.: Graph embedding as a new approach for unknown malware detection. Journal of Computer Virology and Hacking Techniques 13(3), 153–166 (2017)

[60] Haykin, S.: Neural Networks and Learning Machines. Pearson Education (2010)

[61] Hettich, S., Bay, S.: The uci kdd archive [http://kdd. ics. uci. edu]. irvine, ca: University of california. Department of Information and Computer Science 152 (1999)

[62] Holme, P., Kim, B.J., Yoon, C.N., Han, S.K.: Attack vulnerability of complex networks. Phys. Rev. E 65, 056109 (May 2002), `https://link.aps.org/doi/10.1103/PhysRevE.65.056109`

[63] Hu, W., Tan, Y.: Generating adversarial malware examples for black-box attacks based on gan. arXiv preprint arXiv:1702.05983 (2017)

[64] Jain, A.K., Gupta, B.B.: Towards detection of phishing websites on client-side using machine learning based approach. Telecommunication Systems 68(4), 687–700 (2018)

[65] Kabir, E., Hu, J., Wang, H., Zhuo, G.: A novel statistical technique for intrusion detection systems. Future Generation Computer Systems 79, 303–318 (2018)

[66] Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of global optimization 39(3), 459–471 (2007)

[67] Karbab, E.B., Debbabi, M., Derhab, A., Mouheb, D.: Maldozer: Automatic framework for android malware detection using deep learning. Digital Investigation 24, S48–S59 (2018)

[68] Khan, M.A., Karim, M., Kim, Y., et al.: A scalable and hybrid intrusion detection system based on the convolutional-lstm network. Symmetry 11(4), 583 (2019)

[69] Kolosnjaji, B., Demontis, A., Biggio, B., Maiorca, D., Giacinto, G., Eckert, C., Roli, F.: Adversarial malware binaries: Evading deep learning for malware detection in executables. In: 2018 26th European Signal Processing Conference (EUSIPCO). pp. 533–537. IEEE (2018)

[70] Krishnanand, K., Ghose, D.: Glowworm swarm optimisation: a new method for optimising multi-modal functions. International Journal of Computational Intelligence Studies 1(1), 93–119 (2009)

[71] Kudo, T., Kimura, T., Inoue, Y., Aman, H., Hirata, K.: Stochastic modeling of self-evolving botnets with vulnerability discovery. Computer Communications 124, 101–110 (2018)

[72] Kumaresan, T., Palanisamy, C.: E-mail spam classification using s-cuckoo search and support vector machine. International Journal of Bio-Inspired Computation 9(3), 142–156 (2017)

[73] Kushner, D.: The real story of stuxnet. ieee Spectrum 3(50), 48–53 (2013)

[74] Lazfi, S., Lamzabi, S., Rachadi, A., Ez-Zahraouy, H.: The impact of neighboring infection on the computer virus spread in packets on scale-free networks. International Journal of Modern Physics B 31(30), 1750228 (2017)

[75] Li, J.h.: Cyber security meets artificial intelligence: a survey. Frontiers of Information Technology & Electronic Engineering 19(12), 1462–1474 (2018)

[76] Li, J., Sun, L., Yan, Q., Li, Z., Srisa-an, W., Ye, H.: Significant permission identification for machine-learning-based android malware detection. IEEE Transactions on Industrial Informatics 14(7), 3216–3225 (2018)

[77] Li, P., Liu, Q., Zhao, W., Wang, D., Wang, S.: Bebp: an poisoning method against machine learning based idss. arXiv preprint arXiv:1803.03965 (2018)

[78] Li, X.l.: An optimizing method based on autonomous animats: fish-swarm algorithm. Systems Engineering-Theory & Practice 22(11), 32–38 (2002)

[79] Li, Y., Yang, Z., Chen, X., Yuan, H., Liu, W.: A stacking model using url and html features for phishing webpage detection. Future Generation Computer Systems 94, 27–39 (2019)

[80] Lison, P., Mavroeidis, V.: Automatic detection of malware-generated domains with recurrent neural models. arXiv preprint arXiv:1709.07102 (2017)

[81] Manku, G.S., Naor, M., Wieder, U.: Know thy neighbor's neighbor: The power of lookahead in randomized p2p networks. In: Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing. pp. 54—63. STOC '04, Association for Computing Machinery, New York, NY, USA (2004), https://doi.org/10.1145/1007352.1007368

[82] McLaughlin, N., Martinez del Rincon, J., Kang, B., Yerima, S., Miller, P., Sezer, S., Safaei, Y., Trickel, E., Zhao, Z., Doupé, A., et al.: Deep android malware detection. In: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. pp. 301–308. ACM (2017)

[83] Meng, G., Xue, Y., Mahinthan, C., Narayanan, A., Liu, Y., Zhang, J., Chen, T.: Mystique: Evolving android malware for auditing anti-malware tools. In: Proceedings of the 11th ACM on Asia conference on computer and communications security. pp. 365–376. ACM (2016)

[84] Microsoft: (Dec 2019), https://docs.microsoft.com/en-us/sysinternals/downloads/procmon, accessed: 2019-12-30

[85] Mirjalili, S., Lewis, A.: The whale optimization algorithm. Advances in engineering software 95, 51–67 (2016)

[86] Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Advances in engineering software 69, 46–61 (2014)

[87] Moon, D., Im, H., Kim, I., Park, J.H.: Dtb-ids: an intrusion detection system based on decision tree using behavior analysis for preventing apt attacks. The Journal of supercomputing 73(7), 2881–2895 (2017)

[88] Newman, M.E.: The structure and function of complex networks. SIAM review 45(2), 167–256 (2003)

[89] Ney, P., Koscher, K., Organick, L., Ceze, L., Kohno, T.: Computer security, privacy, and {DNA} sequencing: Compromising computers with synthesized {DNA}, privacy leaks, and more. In: 26th {USENIX} Security Symposium ({USENIX} Security 17). pp. 765–779 (2017)

[90] Noreen, S., Murtaza, S., Shafiq, M.Z., Farooq, M.: Evolvable malware. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation. pp. 1569–1576. ACM (2009)

[91] Olariu, S., Zomaya, A.Y.: Handbook of bioinspired algorithms and applications. Chapman and Hall/CRC (2005)

[92] Otero, F.E., Freitas, A.A., Johnson, C.G.: Inducing decision trees with an ant colony optimization algorithm. Applied Soft Computing 12(11), 3615–3626 (2012)

[93] Pan, W., Jin, Z.: Edge-based modeling of computer virus contagion on a tripartite graph. Applied Mathematics and Computation 320, 282–291 (2018)

[94] Panigrahi, B., Shi, Y., Lim, M.: Handbook of Swarm Intelligence. Series: Adaptation, Learning, and Optimization. Springer-Verlag Berlin Heidelberg (2011)

[95] Parsaei, M.R., Javidan, R., Kargar, N.S., Nik, H.S.: On the global stability of an epidemic model of computer viruses. Theory in Biosciences 136(3-4), 169–178 (2017)

[96] Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE control systems magazine 22(3), 52–67 (2002)

[97] Procdot: (Dec 2019), `https://www.procdot.com`, accessed: 2019-12-30

[98] Rad, B.B., Masrom, M., Ibrahim, S.: Camouflage in malware: from encryption to metamorphism. International Journal of Computer Science and Network Security 12(8), 74–83 (2012)

[99] Ren, J., Xu, Y.: A compartmental model for computer virus propagation with kill signals. Physica A: Statistical Mechanics and its Applications 486, 446–454 (2017)

[100] Rigaki, M., Garcia, S.: Bringing a gan to a knife-fight: Adapting malware communication to avoid detection. In: 2018 IEEE Security and Privacy Workshops (SPW). pp. 70–75. IEEE (2018)

[101] Rosenberg, I., Sicard, G., David, E.O.: Deepapt: nation-state apt attribution using end-to-end deep neural networks. In: International Conference on Artificial Neural Networks. pp. 91–99. Springer (2017)

[102] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. nature 323(6088), 533–536 (1986)

[103] Rurik: Rurik/noriben (Nov 2019), `https://github.com/Rurik/Noriben`

[104] Sahingoz, O.K., Buber, E., Demir, O., Diri, B.: Machine learning based phishing detection from urls. Expert Systems with Applications 117, 345–357 (2019)

[105] Selvakumar, B., Muneeswaran, K.: Firefly algorithm based feature selection for network intrusion detection. Computers & Security 81, 148–155 (2019)

[106] Seymour, J., Tully, P.: Weaponizing data science for social engineering: Automated e2e spear phishing on twitter. Black Hat USA 37 (2016)

[107] Seymour, J., Tully, P.: Generative models for spear phishing posts on social media. arXiv preprint arXiv:1802.05196 (2018)

[108] Sharma, P.K., Moon, S.Y., Moon, D., Park, J.H.: Dfa-ad: a distributed framework architecture for the detection of advanced persistent threats. Cluster Computing 20(1), 597–609 (2017)

[109] Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q.: A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence 2(1), 41–50 (2018)

[110] Sikora, L., Zelinka, I.: Swarm Virus, Evolution, Behavior and Networking, pp. 213–239. Springer Berlin Heidelberg, Berlin, Heidelberg (2018), `https://doi.org/10.1007/978-3-662-55663-4_11`

[111] Singh, J., Kumar, D., Hammouch, Z., Atangana, A.: A fractional epidemiological model for computer viruses pertaining to a new fractional derivative. Applied Mathematics and Computation 316, 504–515 (2018)

[112] Smadi, S., Aslam, N., Zhang, L.: Detection of online phishing email using dynamic evolving neural network based on reinforcement learning. Decision Support Systems 107, 88–102 (2018)

[113] Sohrabi, M.K., Karimi, F.: A feature selection approach to detect spam in the facebook social network. Arabian Journal for Science and Engineering 43(2), 949–958 (2018)

[114] Spafford, E.H.: Computer viruses as artificial life. Artificial life 1(3), 249–265 (1994)

[115] Stoecklin, M.P.: Deeplocker: How ai can power a stealthy new breed of malware. Security Intelligence 8 (2018)

[116] Šustr, J.: Malware and the Possibilities of its Evolution. Ph.D. thesis, Vysoká škola báňská-Technická univerzita Ostrava (2019)

[117] Sutton, R.S., Barto, A.G., et al.: Introduction to reinforcement learning, vol. 135. MIT press Cambridge (1998)

[118] Syarif, A.R., Gata, W.: Intrusion detection system using hybrid binary pso and k-nearest neighborhood algorithm. In: 2017 11th International Conference on Information & Communication Technology and System (ICTS). pp. 181–186. IEEE (2017)

[119] Szor, P.: The Art of Computer Virus Research and Defense. Pearson Education (2005)

[120] Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE symposium on computational intelligence for security and defense applications. pp. 1–6. IEEE (2009)

[121] Thanh, C.T., Zelinka, I.: A survey on artificial intelligence in malware as next-generation threats. In: MENDEL. vol. 25, pp. 27–34 (2019)

[122] Torres, J.M., Comesaña, C.I., García-Nieto, P.J.: Machine learning techniques applied to cybersecurity. International Journal of Machine Learning and Cybernetics pp. 1–14 (2019)

[123] Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction apis. In: 25th {USENIX} Security Symposium ({USENIX} Security 16). pp. 601–618 (2016)

[124] Tran, D., Mac, H., Tong, V., Tran, H.A., Nguyen, L.G.: A lstm based framework for handling multiclass imbalance in dga botnet detection. Neurocomputing 275, 2401–2413 (2018)

[125] Truong, T.C., Diep, Q.B., Zelinka, I., Tran, T.D.: X-swarm: The upcoming swarm worm. MENDEL 26(1), 15–21 (Aug 2020), `https://mendel-journal.org/index.php/mendel/article/view/112`

[126] Truong, T.C., Zelinka, I., Senkerik, R.: Neural swarm virus. In: Zamuda, A., Das, S., Suganthan, P.N., Panigrahi, B.K. (eds.) Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing. pp. 122–134. Springer International Publishing, Cham (2020)

[127] VMware: (Dec 2019), `https://www.vmware.com/`, accessed: 2019-12-30

[128] Von Solms, R., Van Niekerk, J.: From information security to cyber security. computers & security 38, 97–102 (2013)

[129] Wang, P., Lin, H.T., Wang, T.S.: An improved ant colony system algorithm for solving the ip traceback problem. Information Sciences 326, 172–187 (2016), cited By 23

[130] Wang, P., Sparks, S., Zou, C.C.: An advanced hybrid peer-to-peer botnet. IEEE Transactions on Dependable and Secure Computing 7(2), 113–127 (2010)

[131] Wang, W., Zhao, M., Wang, J.: Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. Journal of Ambient Intelligence and Humanized Computing 10(8), 3035–3043 (2019)

[132] Wang, Z., Dong, H., Chi, Y., Zhang, J., Yang, T., Liu, Q.: Dga and dns covert channel detection system based on machine learning. In: Proceedings of the 3rd International Conference on Computer Science and Application Engineering. p. 156. ACM (2019)

[133] Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’networks. nature 393(6684), 440–442 (1998)

[134] Wickramasinghe, C.S., Marino, D.L., Amarasinghe, K., Manic, M.: Generalization of deep learning for cyber-physical system security: A survey. In: IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society. pp. 745–751. IEEE (2018)

[135] Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., Wang, C.: Machine learning and deep learning methods for cybersecurity. IEEE Access 6, 35365–35381 (2018)

[136] Xu, W., Qi, Y., Evans, D.: Automatically evading classifiers. In: Proceedings of the 2016 network and distributed systems symposium. pp. 21–24 (2016)

[137] Xu, Z., Ray, S., Subramanyan, P., Malik, S.: Malware detection using machine learning based analysis of virtual memory access patterns. In: Proceedings of the conference on design, automation & test in Europe. pp. 169–174. European Design and Automation Association (2017)

[138] Yang, L., Zhai, J., Liu, W., Ji, X., Bai, H., Liu, G., Dai, Y.: Detecting word-based algorithmically generated domains using semantic analysis. Symmetry 11(2), 176 (2019)

[139] Yang, X.S.: Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms. pp. 169–178. Springer (2009)

[140] Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010), pp. 65–74. Springer (2010)

[141] Ye, Y., Chen, L., Hou, S., Hardy, W., Li, X.: Deepam: a heterogeneous deep learning framework for intelligent malware detection. Knowledge and Information Systems 54(2), 265–285 (2018)

[142] Yu, B., Pan, J., Hu, J., Nascimento, A., De Cock, M.: Character level based detection of dga domain names. In: 2018 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2018)

[143] Zelinka, I.: Soma:self-organizing migrating algorithm. In: New optimization techniques in engineering, pp. 167–217. Springer (2004)

[144] Zelinka, I., Chen, G.: Evolutionary Algorithms, Swarm Dynamics and Complex Networks: Methodology, Perspectives and Implementation, vol. 26. Springer (2017)

[145] Zelinka, I., Das, S., Sikora, L., Šenkeřík, R.: Swarm virus-next-generation virus and antivirus paradigm? Swarm and Evolutionary Computation 43, 207–224 (2018)

[146] Zhu, H.J., You, Z.H., Zhu, Z.X., Shi, W.L., Chen, X., Cheng, L.: Droiddet: effective and robust detection of android malware using static analysis along with rotation forest model. Neurocomputing 272, 638–646 (2018)

LIST OF PUBLICATIONS

Candidate's research cited in the thesis

[rel1] Truong, T.C.; Diep, Q.B.; Zelinka, I. :Artificial Intelligence in the Cyber Domain: Offense and Defense. In: Symmetry 12(3):410, (2020), Q2-2019, IF-2019 = 2.645

[rel2] Truong, T.C., Zelinka, I.: A Survey on Artificial Intelligence in Malware as Next-Generation Threats. In MENDEL (Vol. 25, No. 2, pp. 27-34). (2019), Q4, SJR 2019 = 0.15

[rel3] Truong, T.C., Zelinka, I., Plucar, J., Candick, M., Sulc, V.: Artificial Intelligence and Cybersecurity: Past, Presence, and Future. In: Dash, S.S., Lakshmi, C., Das, S., Panigrahi, B.K. (eds) Artificial Intelligence and Evolutionary Computations in Engineering Systems. Advances in Intelligent Systems and Computing, vol 1056. Springer, Singapore (2020)

[rel4] Truong, T.C, Quoc Bao Diep, and Ivan Zelinka. : Swarm intelligence in Cybersecurity Swarm Intelligence: From Social Bacteria to Human Beings. CRC Press (2020)

[rel5] Truong, T.C., Zelinka, I., Senkerik, R. : Neural Swarm Virus. In: Zamuda, A., Das, S., Suganthan, P.N., Panigrahi, B.K. (eds.) Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing. Communications in Computer and Information Science, vol 1056. Springer, Singapore (2020)

[rel6] Truong, T.C., Huynh, T-P., Zelinka, I.: Applications of swarm intelligence algorithms countering the cyber threats In: GECCO '20: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion (2020)

[rel7] Truong, T.C., Diep, QB., Zelinka, I., Tran, TD.: X-Swarm: The Upcoming Swarm Worm. In MENDEL (Vol. 26, No. 1, pp. 15-21). (2020), Q4, SJR 2019 = 0.15

Candidate's research out of the scope of the Thesis

[oth1] Huynh T-P, Ha D-H, Thanh CT, Fazio P, Voznak M. :Secrecy Outage Probability of a NOMA Scheme and Impact Imperfect Channel State Information in Underlay Cooperative Cognitive Networks. In: Sensors 20(3):895, (2020), Q1-2019, IF-2019:3.275.

[oth2] Cong, T.T., Quoc Bao Diep, and Ivan Zelinka. Pareto-based Self-Organizing Migrating Algorithm solving 100-Digit Challenge. In: Zamuda, A., Das, S., Suganthan, P.N., Panigrahi, B.K. (eds.) Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing. Communications in Computer and Information Science, vol 1056. Springer, Singapore (2020)

LIST OF RESEARCH RESULTS AND ACTIVITIES

Publication activities

I provide the following list indexed results in relevant scientific databases in order to document my research activities within the entire period of my doctoral study:

- ORCID: https://orcid.org/0000-0001-6603-392X
- ResearcherID: AAK-4338-2020
- Scopus Author ID: 57214223150
- Articles indexed in Web of Science, Socpus: 4
- Papers in Conf. Proc. Indexed in Web of Science, Scopus: 4
- Book chapter: 1

Participating in projects of specific research

1. Extension and creation of new support for PVBPS (Computer Viruses and Computer System Security). 62/2019/RPP-TO - 1/a
2. Artificial intelligence in computer security. 71/2019/RPP-TO - 2
3. Digital forensic analysis - modern methods and tools in investigating cyber crimes. 142/2020/RPP-TO - 1/a

Passed Courses

- 712-0191/02 English Language Dr.
- 460-6018/02 Social Networking
- 460-6016/02 Data Analysis
- 460-6021/02 Computer Security
- 460-6017/02 Bio-Inspired Computing
- 460-6006/02 Neural Networks