Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Dynamic Network Segmentation for Cyber-Defence

MSc Individual Project Final Report

Author:
Thachphon Sirimongkolkasem

Supervisor:
Prof Emil Lupu

Second Marker:
Dr Soteris Demetriou

Submitted in partial fulfillment of the requirements for the MSc degree in Computing Science of Imperial College London

October, 2020

Abstract

Rapid growth and development in technologies have significantly increased the number of devices connected to the internet. Not only these developments improve quality of life, they also enhance efficiency in all sectors, including transport, electricity, health, entertainments and among others.

Technological advancements, however, can also come with unforeseen vulnerabilities. Such vulnerabilities can be fatal when they pose risks to individuals, organisations, critical infrastructures, and government functions etc. With the rise of cyber threats and evolution of cyber-attack techniques, of which these vulnerabilities can be exploited, these issues must be addressed.

Network segmentation, a cyber-defensive strategy that reduces the risk of a network by separating a network into different segments, thereby slows down the rate at which an attacker can progress. Network segmentation has proven to be effective in enhancing network security, but it is not properly understood, with only vague guidance. This paper aims to answer the following questions: how can an organisation apply network segmentation? How do we evaluate the security of a network? What is the cost of implementing security?

We propose an automated system that generates segmentation architectures optimised for cost and security. The system utilises probabilistic cyber risk modelling and an optimisation algorithm that simulates social behaviour, to construct segmentation architectures, evaluate them, and efficiently search the solution space for better architectures. The results we obtained indicate that the approach is feasible and can prove useful for future research.

Acknowledgements

The motivation for the creation, development and completion of this thesis would not have been possible without the support and guidance from my supervisor, **Prof. Emil Lupu**. His invaluable insights, comments and suggestions throughout the project had helped me with organising my thoughts, clearing my doubts, and ultimately designing and implementing the model. I am extremely grateful for his time and patience throughout the process.

In addition, I would like to express my gratitude and appreciation to:

- **Dr. Soteris Demetriou**, for the project progress discussion in June. Although short, it was very helpful in understanding methods of conducting a literature review.
- **Michelle Landgan** and **Jasmin Han**, for their support and counselling advice, which guided me to think positively and work effectively during this difficult time.
- **My friends**, for their encouragements and support throughout my time at Imperial.

Finally, I would like to thank **my family** for their unconditional love and support, and for providing me the opportunity to study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In an ever-growing digital landscape, we witness not only more technological innovations [1], but also the emergence of hackers' new abilities to compromise company networks. Security is a growing problem that cannot be ignored; the average cost of cybercrime for each company increased from US$11.7m in 2017 to US$13.0m in 2018 [3].

A similar trend is also observed in the number of new software vulnerabilities being published each year, with a record-breaking number of vulnerabilities (16,556) published in 2018 [4] (See **Figure 1.1**). However, as pointed out by [4], more published vulnerabilities do not necessarily mean that organisations are more vulnerable. One major factor is that more resources are spent on vulnerability research (from software vendors) and publishing abilities (from the MITRE organisation and National Vulnerability Database, NVD) [4]. Other factors, such as increase in opportunities for attackers as a result of the growing complexities in the digital world, also contribute to the increase in more vulnerabilities being discovered.

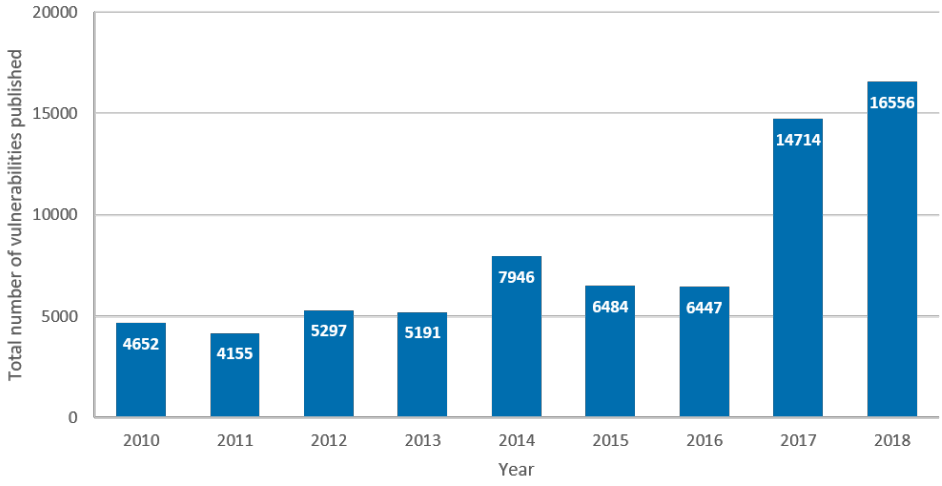

Figure 1.1: Number of vulnerabilities published between 2010 and 2018.
Data taken from Common Vulnerability and Exposures (CVE) website [5].

Such opportunities can arise from major global events. The prime example of this is the recent COVID-19 pandemic; many organisations were forced to adapt and operate remotely; schools' and universities' assessments were either cancelled [7] or done remotely [8, 9], and meetings were changed to video conferences [10]. The pandemic created the perfect opportunity for video conferencing applications such as Zoom to flourish, but these applications also became easy targets for cyber-attacks [11].

Another example is the increased utilisation of smart grid technologies (such as smart meters) to improve efficiency and savings for both energy producers and consumers. Cyber-attacks on critical infrastructures such as the electricity sectors can have serious impacts on a nation [12, 13]. A successful attack means that electricity cannot be delivered, thus stopping almost all other sectors from operating normally. This includes heating systems, transportation, healthcare, etc. leading to potential loss of many lives [12].

In order to protect critical systems from cyber-attacks, organisations (such as Google [16]) have proposed framework/ guidelines in order to mitigate these threats. It is expensive and difficult to protect a system from cyber-attacks as it requires protecting all potential attack paths, whilst an attacker only need one successful attempt to gain access into the organisation. Therefore, it is important to combine different defensive strategies to maximise security. Network segmentation is one of the core defensive measures against cyber-attacks.

Network segmentation is a defensive measure that is widely regarded as critical for protecting an organisation from network attacks. Network segmentation refers to splitting a network topology into different segments called enclaves. Each segment serves as an isolation unit, having communication restrictions between them [51]. By correctly controlling segment-to-segment communications and also internet-to-segment communication, network segmentation can effectively restrict an attacker from moving laterally. *Lateral movement* is a concept that describes the techniques and tools that an attacker uses to move deeper into a network after gaining initial foothold in the the network. Each successful step grants the attacker more privilege and access to more resources. The attacker attempts to avoid detection in order to search for critical assets. An example of an attack path that can be taken by attacker for lateral movement is shown in **Figure 1.2**.

In **Figure 1.2**, the attacker takes the following path (shown in red) which results in them obtaining access to sensitive information in the database. The attack's path involves the following steps:

1. The attacker first launches a spear phishing attack, where a specially crafted malicious email is sent to an employee in the LAN enclave. The employee opens the email and the malicious attachment, resulting in the computer being compromised. The attacker then gains initial access to the network.

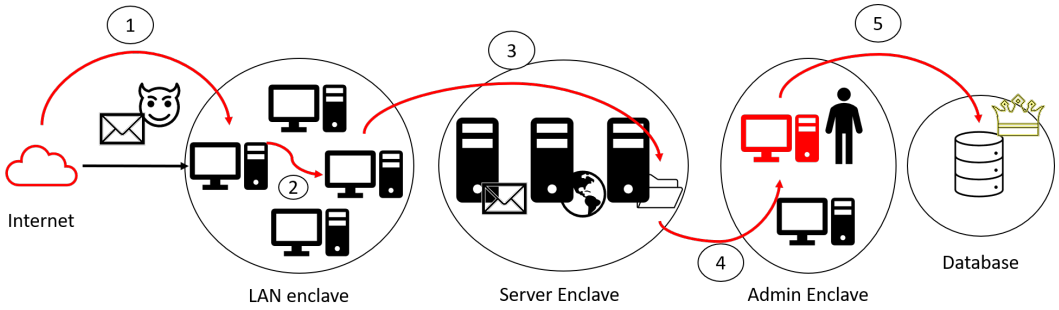

Figure 1.2: An illustration of how an attacker can move laterally within the network after initial access (**step 1**). Ultimately obtaining the goal of accessing sensitive information in the database.

2. The attacker then infects another computer in the same LAN enclave.
3. The attacker finds a vulnerability in the file server which he/she can exploit. This results in the file server being compromised.
4. The attacker then infects all the files within the file server, compromising an admin computer.
5. With admin privilege, the attacker has the highest privilege in the network, thus gaining access to the database where sensitive information is stored. The attacker achieves his/her goal.

Security principles that are closely tied to network segmentation are *defense in depth* and *principle of least privilege*. The concept of defense in depth lies on *layered protection*, where an organisation puts multiple defensive measures both inside and outside the network to protect the organisation's assets. This defensive technique essentially divides the organisation's network into segments, where each segment represents a security point that has to be overcome. Defense in depth makes sure that if one of the security points fails, the attacker would need to penetrate more in order to progress. If an attacker were to attack a network, this would mean that the attack path (similar to the one in **Figure 1.2**) that he/she has to take can be a very long and difficult one.

Principle of least privilege refers to the process of granting access to resources and communication only when it is necessary to carry out the task. For example, an employee who works in a Sales department should not be able to communicate or have access to payroll computers from the company's Finance department, since the operations of the employee from Sales department do not rely on having access to or communicating with Finance department. Therefore, the employee in this case should be segmented from the Finance department.

Practices that implement network segmentation include: segmenting a network into different

virtual local area networks (VLANs); implementing filtering rules; introducing multi-factor authentication; segmenting network based on functionalities and sensitivity levels etc. Network segmentation aims to increase the security of a network in several ways [48]:

1. Reducing the number of entry points into a network.
2. Limiting an attacker's network access who has already breached the network.
3. Slowing down an attacker's ability to spread to other machines.
4. Providing more time for the defender to detect and stop the attack.

The question of what the best network segmentation architectures are is an ongoing area of research. One of the reasons is due to the complex relationships between security, cost, and operation requirements. Too much security may affect the normal operations of an organisation. For example, if an organisation were to completely block all communications to the internet, it can lead to emails not being exchanged between employees and customers. Additionally, applying more security measures come with great cost [50]. Yet, too little security in a network can lead to cyber-attacks, which can potentially cost an organisation a fortune. To complicate matters further, it can be difficult for organisations to measure the impact of segmenting a network would have on the smoothness of their operations [49], on top of growing concerns regarding cyber threats.

1.2 Contributions

The goal of this project is to design a network architecture generation system that outputs architectures that optimise based on the level of importance between security and cost, whilst satisfying traffic requirements.

The main contributions of this paper are as follows:

1. We review current cyber risks and the importance of network segmentation in cyber-defense.
2. We review the current state of the art on network segmentation research.
3. We combine Wagner et al. approach on network segmentation with methods that use attack graph for cyber risk assessment to evaluate a given segmentation architecture.
4. We closely follow Wagner et al. approach on investigating the trade-off between cost and security in applying network segmentation.
5. We propose a novel approach to optimising network segmentation architecture using particle swarm optimisation, which intelligently explores the search space to find near optimal network segmentation architectures.

6. We evaluate the effectiveness of particle swarm optimisation on network segmentation and discuss potential ways to tune the hyperparameters.

1.3 Outline

The rest of the paper is broken down as follows:

- **Chapter 2** describes the background of network attacks and mitigation strategies and principles, an overview of problems in network segmentation, a literature review on network segmentation research, and the mathematical backgrounds required for implementing our model.
- **Chapter 3** provides an overview of the design of the network model, our network architecture evaluation metrics and the optimisation algorithm we utilise.
- **Chapter 4** summarises the overall flow of the program, and explains the implementation of our model and algorithms with pseudocode.
- **Chapter 5** details our experiments and discusses the implications behind the results we obtained.
- **Chapter 6** discusses future work that can expand the model further.
- **Chapter 7** concludes the findings of the paper.

Chapter 2

Background

This chapter describes several cyber risks, the challenges behind detecting and preventing them. Then we introduce the security principles that aim to implement network segmentation, the problems that currently exist, and literature papers that study network segmentation and risk assessment. Finally, we describe the mathematical background that is needed to understand the design of our model.

2.1 Cyber Risks

It is important for organisations to be aware of and understand the types of threats that they may face in order to properly mitigate or prevent when an attack occurs. In this section we introduce various types of cyber risks, their anatomies, and why they can be difficult for organisations to detect and respond.

2.1.1 Vulnerabilities and Exploits

Vulnerabilities are software bugs that can be exploited in a way that allows the software to operate outside of designed boundaries. An *exploit* is a piece of software that is developed to take advantage of a vulnerability in order to gain information and also elevate privilege.

A *zero-day* vulnerability is a type of vulnerability that is only known to the attackers, and thus impossible for vendors to fix since they are not aware of it. The damage that zero-day exploits cause can be devastating [13]. Once a vulnerability is known, and/or that it has been allowed to disclose because the vendor has released a patch to fix it, it will be published in online security databases such as the National Vulnerability Database (NVD) [14] and CVE [15]. Then, proof of concept exploits may become available in *Exploit Database*. The existence of such databases offers a safe access to codes that will help testing, patching, and thus safeguarding against vulnerability-related attacks.

A straightforward solution would be to simply patch the vulnerability. However, it may not

always be a viable option for various reasons. One major obstacle for patching is that the process consumes a lot of resources. It requires a thorough understanding of the vulnerability (potentially involves hundreds of programs) and correcting it. Then, the patched version of the software needs to be distributed to all computers that contain the outdated version, which can lead to some downtime. Another reason would be that if the vulnerability is not known (zero-day), then it is impossible to patch. Even if the vulnerability is discovered, it takes an average of 65 days to fix [6], during which critical infrastructures are exposed to attacks. A timeline of vulnerability discovery to patching is shown in **Figure 2.1**.

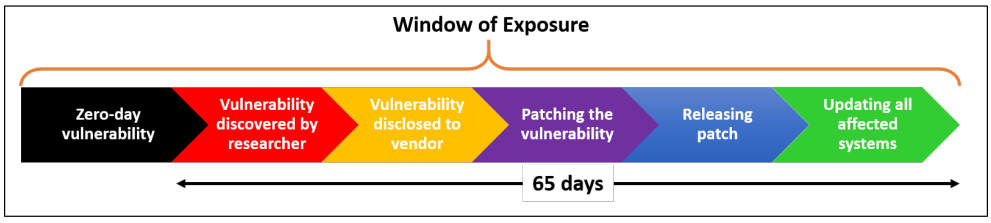

Figure 2.1: Timeline of vulnerability discovery to patching to updating all systems. Window of exposure shows the period in which the systems are vulnerable [6].

2.1.2 Distributed Denial-of-Service Attacks

A *denial-of-service* attack (DoS) is a type of attack in which the attacker paralyses a company's operations by flooding its network with requests, overloading the system, and thus preventing legitimate users from accessing the resources [21].

A distributed denial-of-service attack (DDoS) occurs when multiple machines carry out attacks on one target simultaneously. These machines are often compromised machines, called *botnets*. DDoS attackers are difficult to identify as a result of this; machines which launch the attacks may not necessarily be the true source of the attackers [21]. **Figure 2.2** visualises a DDoS attack against a web server.

There is a growing concern around DDoS attacks as they are becoming more common and sophisticated. From 2017-2019, more than 40% of unique DDoS targets were financial services organisations [24]. When looking at the number of DDoS attack events, the gaming industry accounts for at least 70% of attacks [24]. This is because botnets can be rented out, providing "attack-for-hire" services, allowing even novice users to launch DDoS attacks; players are willing to pay in order to disrupt and decide the results of matches [25].

Critical infrastructures, such as the electricity sector, are frequent targets of DDoS due to the damage it can cause, and also the ease of launching a successful attack in comparison to other areas (such as financial services and gaming industries). As stated by [27], the electricity sector has a lot of constraints as it heavily relies on delay-free communication net-

works. Most grid applications require 4, 30, 40 and 100ms latency for communication; most protection mechanisms cannot meet this requirement due to their long process time [26].

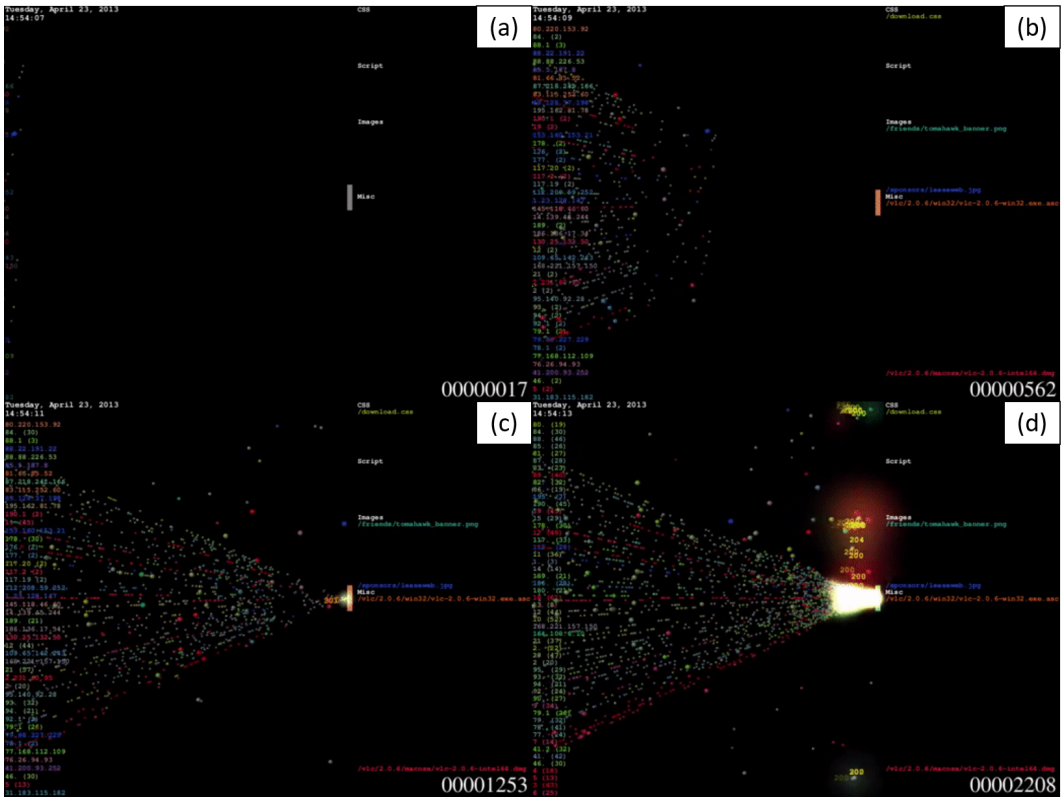

Figure 2.2: A visualisation of an DDoS attack by [22] (chronologically from (a) to (d)). Coloured balls are requests which travel across screen to the victim machine. Successful requests are hit by the paddle and bounce back whilst unsuccessful ones simply pass through. Software made by [23].

DDoS attacks can be broadly categorised into three types [38]: *volumetric attacks*, *protocol attacks*, and *application layer attacks*. Volumetric attacks focus on paralysing the network by saturating the bandwidth of the site, for example, by initiating multiple synchronise (SYN) requests for transmission control protocol (TCP) connections, which is a protocol that if the host was to answer then the host would have to then wait for the requester to respond with an acknowledegment (ACK) of the connection (commonly known as a three-way handshake). However, the requester never responds with the acknowledgement, making the host wait and thus consuming the host's resources. This particular volumetric attack is called SYN flood.

Protocol attacks involve overloading the server resources, such as sending malformed/oversized packets, which causes memory overflow and crashes in the system (ping of death) [38]. Application layer attacks involve crashing an application (usually web servers) by exploiting a vulnerability in the application, thus making it unavailable to the users.

In the case of a DDoS attack, network segmentation can be referred to as having multiple security measures that aim to detect and mitigate the damage. For example: (1) by

having multiple data centres in place and reroute the traffic to these data centres when a volumetric attack happen; (2) using intelligent detection system to detect and implements filtering rules which block malicious traffic that tries to launch a protocol attack [38]; and (3) mitigating application layer attacks via analysing the nature of the incoming packets (deep packet inspection) and subsequently dropping all suspicious packets.

2.1.3 Insider Threats

Even if we assume that a system is completely immune to outside threats, it does not stop a cyber-attack that is launched within the network. Insider threats are threats that originate from within an organisation, regardless of whether they are intentional or not. Insiders include current/ former employees, contractors and business associates. Any entities who have inside information (or access) about the organisation such as data, security practices, computer systems etc. are also considered insiders [28].

Insider threats are extremely dangerous for an organisation and the damage caused by insider threats can be more destructive than attacks launched from outside. This is because insiders have legitimate access to an organisation's network and sensitive information, making detection and prevention difficult [28].

Two common types of insider threats are summarised in **Table 2.1** below. A detailed survey about insider threats can be found in [33], which carries out a systematic literature review on insider threats and includes detailed sub-categories of insider types.

Types	Description
Negligent Insiders	Negligent insiders are people who have bad security practice/awareness and willfully violate policies and protocols. They are more likely to make errors such as misconfiguring firewall, falling for phishing attacks, storing sensitive data in insecure personal device [29] etc, which exposes the organisation to damage such as data breach [30].
Malicious Insiders	Malicious insiders abuse the privilege and access they have to commit acts that have negative impacts to the organisation for personal, financial or political gains. Recent examples of malicious insiders are Facebook security engineer who abused his power to stalk women, and also disgruntled employee at Tesla who exported sensitive data to third parties just because he was reassigned to a new role [31, 32].

Table 2.1: A general description of insider types and the extent of damage they can cause.

Network segmentation frameworks, particularly *zero trust policy* (also discussed in **Section 2.3.1**), aim to address the growing concerns of insider threats. The core idea of zero trust policy is to only trust the user when the user has been truly verified, for example, via multi-

factor authentication. The user is logged out after a certain period of time to force the user to authenticate again (continuous authentication). This prevents an attacker who happens to obtain one-time access to the user from having continuous access to them. Zero trust policy also implements the principle of least privilege, which prevents an attacker from gaining further privilege after compromising one user (further discussed **Section 2.2**).

2.1.4 Advanced Persistent Threats

Advanced persistent threat (APT) is a targeted attack, in which it is done by highly skilled individuals (advanced) and the aim is to stay undetected in an organisation for as long as possible (persistent), whilst monitoring the target and slowly gathering information. APT attacks use multiple *vectors* (such as cyber, physical and social) in order to achieve their objectives [35].

APT attacks are resource-intensive since they require extensive research and individuals with specific skill sets, with multiple techniques and customised tools developed solely for the target. Therefore, traditional targets of APTs are often large organisations, government facilities, and critical infrastructures mostly due to the cost of hiring a group of experts in different areas and long duration of the operation.

However, as suggested in a report by Symantec [34], APT techniques and tools that are used by a few elite individuals are likely to be adopted by wider threat actors, including organised criminals, to target other organisations and steal sensitive information such as intellectual property [34, 36].

APT attacks generally involve the following steps:

Step 1: Research and Infiltration

The first step for an APT attack is to gain a foothold in the organisation. This involves thorough research into the organisation's structure, systems, processes, people etc. Once sufficient information is gathered, tools are developed and attackers would attempt to break into the organisation's network. Methods that are used include, but not limited to, social engineering, exploitation of zero-day vulnerabilities, spear phishing attacks, SQL injection etc. [34]. It is also possible that a DDoS attack is launched to distract network security administrators, and to weaken the security system, which eases the process of breaching [37].

Once initial access has been obtained, a *backdoor* is installed to monitor the machine. Unlike traditional attacks that aim to be quick, sudden and opportunistic, APT attacks aim to stay undetected for as long as the operation requires [37].

Step 2: Discovery and Expansion

The next step is to map out the organisation's structures and systems, and scan for sensitive data, vulnerabilities in software and hardware, pathways to other resources etc. [34]. The goal is to gather more information and continue monitoring the organisation. Obfuscation techniques are often used to avoid detection and the tools used in infiltration stage will be used or modified to gain access to other systems.

Step 3: Capture and Data Exfiltration

At this stage, sensitive information is accessed. Attackers can also attempt to escalate their privileges to capture the systems via installing a *rootkit*. The main priority is again to stay hidden in the system for as long as possible [35]. Symantec [34] reports that on average, a host is actively infected by an APT for 145 days, with the longest infection span being 660 days. The data that is collected is sent to the adversary in a secure channel to avoid detection [35]. A visualisation of an APT attack cycle is shown in **Figure 2.3** below.

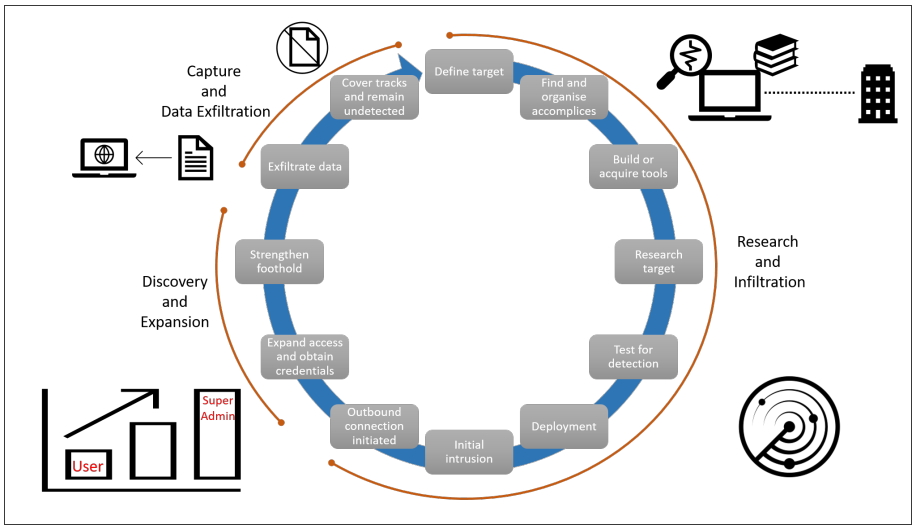

Figure 2.3: The Life Cycle of APT attack.
Adapted from [37].

A simplified, cartoon depiction of an APT attack progression is also shown in **Figure 2.4**. In stage 1, the attacker distracts and weakens the defensive mechanisms by launching a DDoS attack, whilst simultaneously attacking another group of devices with the aim of gaining initial access to the network. In stage 2, the attacker tries to explore critical assets of the network in an attempt to escalate his/her privilege. The task is done carefully to avoid detection. Finally, the attacker sees an opportunity and launches another DDoS attack whilst exfiltrating sensitive data out of the network.

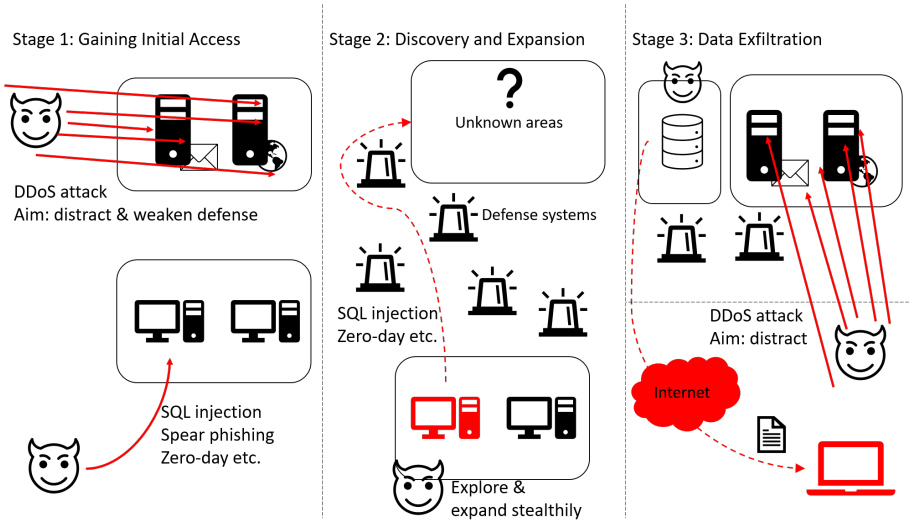

Figure 2.4: Visualisation of APT attack progression.

2.2 Core Security Principles

There are many security principles that security experts follow. All aim to strengthen the network of an organisation, minimise the damage caused by cyber-attacks, and maintain the functionality of the organisation. We will discuss the two most important security concepts: defense in depth and principle of least privilege.

2.2.1 Defense in Depth

The concept of defense in depth originates from a military strategy where barriers are placed to slow down intruders from progressing, which provides time for defender to monitor and develop an effective response [54]. In cybersecurity, defense in depth follows the idea of layered protection, where security of a system is enhanced through combining multiple security controls. This provides multiple layers of defense, instead of relying on one single access control point [41] where one successful attack could lead to huge damage to the organisation. A very common implementation of defense in depth is the application of Demilitarised Zone (DMZ), where resources (usually public facing server and systems) that are more permissive for outside access are placed [41].

Not only the concept of defense in depth is applied for defending unwanted intruders from entering the network, but also in preventing an attacker who has already gained a foothold in the network from spreading and further damaging the organisation. With layered protection within the network, an attacker would need to overcome multiple defensive measures in order to progress further into the network. An example of defense in depth within a network would be by having an intrusion detection system (IDS) to monitor and inspect suspicious network traffic. In this example, the attacker would need to first compromise a device within the network (the attacker may need to overcome many defensive layers to achieve this).

Then the attacker would need to avoid detection by the IDS (or they may even need to compromise the IDS) whilst carrying out their operations.

2.2.2 Principle of Least Privilege

Principle of least privilege specifies that communication is only allowed when it is necessary for the task, and the user is granted minimum privilege and access that is needed for the task [47]. Principle of least privilege enhances network security by making it more difficult for an attacker to escalate privileges and spread within the network after he/she has compromised devices associated with low-level user accounts.

2.3 Utilising Security Principles

Several techniques and frameworks aim to use the idea of defense in depth and principle of least privilege to enhance the security of an organisation. In this section we will cover zero trust policy, *segregate networks and functions* and *limit workstation-to-workstation communication*.

2.3.1 Zero Trust Policy

Zero trust policy is a security framework that is closely tied to principle of least privilege. Traditional security model uses the moat and castle approach (see **Figure 2.5**), where the main objective is to protect the ”castle” (the organisation) from outside intruders from gaining access to the network, and automatically trust everyone who is within the network [40].

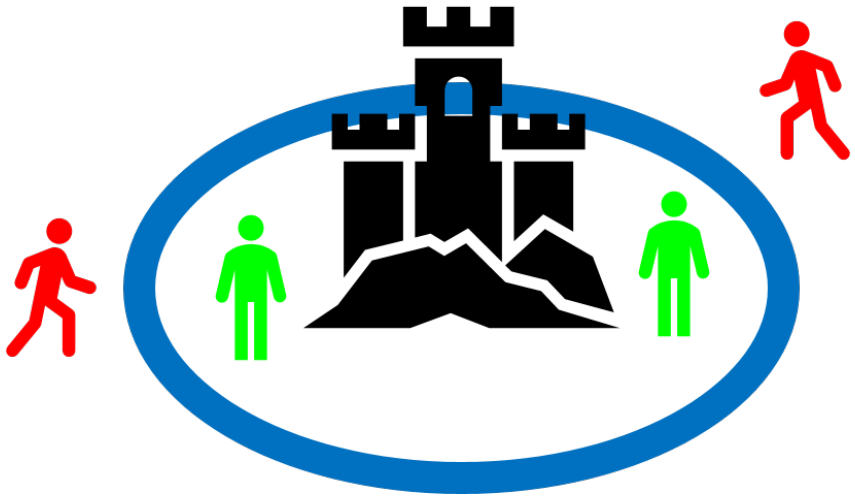

Figure 2.5: The castle and moat security approach aims to make it as hard as possible to gain access to the network. Everyone inside the network is automatically trusted.
Adapted from [40].

The obvious weakness in this approach, as we learn from the previous section (**Section 2.1.3**), is that it is completely ineffective against attacks which originate within the network. The castle and moat approach cannot prevent insider-type cyber-attacks that take advantage of excess trust given to certain devices/individuals whom access should not have been granted in the first place. It is unreasonable to automatically trust user based only on the fact that the user is inside the organisation's network. The framework of zero trust aims to address this issue.

Zero trust aims to strengthen the security of an organisation by making it more difficult for attackers to move laterally by always assuming that the network is hostile, and there are attackers from outside and also within the network. Zero trust zone is where no user should be trusted unless the user is properly authenticated (for example, via multi-factor authentication) and that the authentication should be continuous, where the user would have to re-authenticate after a certain period of time to verify their identity. Zero trust also implements principle of least privilege to prevent an attacker who has already compromised a user from moving laterally within the network and gaining access to critical assets.

Zero trust policy provides an effective defensive measure, as an attacker would need to either continuously spoof the identity of the victim, or exploit a vulnerability in the authentication process in order to gain initial access to the network. The attacker would then need to get through multiple security points without getting undetected. These security points are placed due to principle of least privilege.

2.3.2 Segregate Networks and Functions

The principle behind segregate networks and functions (SNF) is that different cyber assets have different organisational functions (for example, finance databases, public-facing web servers, network admins etc.). These functions do not hold the same sensitivity levels and security requirements [47].

An example of a network that implements defense in depth but not SNF is shown in **Figure 2.6**. Note that although there are two firewalls for a traffic to get through, it does not properly protect the resources (such as the engineering workstations). This is because the web server and mail server would have allowed almost all traffic from the internet, or that no emails can be received since Engineering Workstations would require stricter policies.

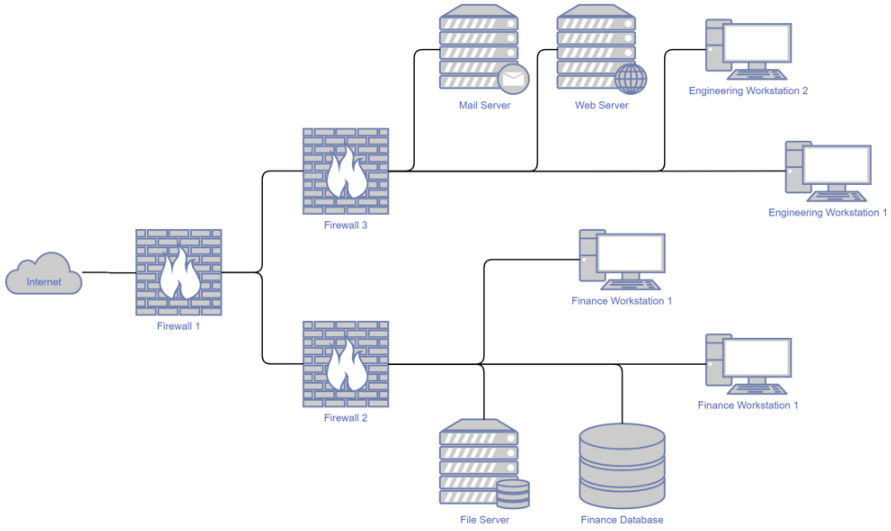

Figure 2.6: An example of a network that attempts to implement defense in depth but do so without considering SNF.
Adapted from [41].

SNF aims to restrict the ability of an attacker (who has already set foot in the network) to move laterally and compromise more devices. This is done by separating unrelated functions, grouping similar ones, and then restrict communications between the separated groups. SNF is often implemented through firewalls configurations, network filters, application-level filters etc. [47]. A proper way to implement SNF in the case given in **Figure 2.6** is shown in **Figure 2.7**.

2.3.3 Limit Workstation-to-Workstation Communication

Limit Workstation-to-Workstation Communication (LWC) further enforces principle of least privilege by even restricting device-to-device communication. Communication is only allowed when it is necessary to perform the task for the device. Methods of implementing LWC includes setting device-level firewall rules, using private virtual LANs, disabling remote login etc. [47].

By comparing **Figure 2.6** and **Figure 2.7**, we can see the difference between good and bad network segmentation. Good network segmentation can effectively protect a network with multiple barriers which isolate functions that are different and group those that are the same. The segmentation would not hugely impact the normal operations of the organisation. Bad network segmentation, on the other hand, tries to add extra security to the network, but it is as vulnerable to attacks as a network without segmentation. As seen in **Figure 2.7**, mail server and web server usually allow more traffic from outside, meaning that *Firewall 3* will have very soft policies and filtering rules, thus allowing almost anyone (including the

attacker) to easily gain access to engineering workstations.

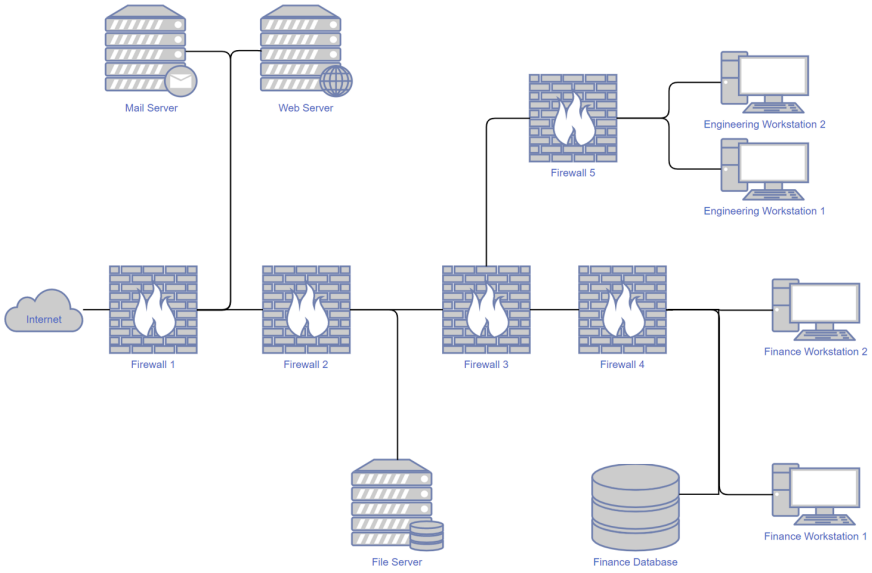

Figure 2.7: An example of a network that properly implements defense in depth and SNF, giving the most optimal security whilst making sure that communications are only granted when needed.
Adapted from [41].

2.4 The Problems

2.4.1 Misconfiguration of Filtering Rules

Firewall is one of the main methods of implementing network segmentation, and the primary defence line in the electricity sector [27]. Firewalls identify and discard suspicious packets by inspecting the properties of packets such as time delay, port numbers and IP addresses [27].

However, the difficulty arises when there are more connections in the network, the number of rules can have hundreds of configurable rules, which can often conflict [26]. Additionally, developing accurate firewall rules require the knowledge of all cyber assets and all authorised communications [26], to which the information is rarely available. This means that the more devices connected to the network, the harder and more complex network segmentation (such as configuring firewall rules) can be applied.

2.4.2 Unclear Mission Impacts

Not only in electrical sector where it is difficult to apply network segmentation, many organisations are unclear as to how and what cyber assets are being used to execute missions

[50]; it can be difficult to measure the mission impacts caused by changes in network architecture. As stated by [49], network segmentation is only easy for small networks. The number of potential network configurations grows exponentially with increase in network size [49].

2.4.3 Growing Cyber Threats

New vulnerabilities, advanced tools and techniques are being developed frequently (such as those discussed in **Section 2.1.4**). These tools and techniques are being adopted by other attackers as a result of the ease of sharing and discussing the effectiveness of those tools anonymously online. Over time, current network segmentation techniques (e.g. multi-factor authentication) might become less effective in defending against new types of attacks; new mitigation strategies are needed to prevent this from happening. Therefore, just as the attackers are constantly adapting, security experts would also need to do the same in order to effectively respond to new cyber threats.

2.5 Related Work

Network segmentation is the practice of partitioning a network into different segments, and placing communication restrictions between these segments to enhance security. Network segmentation plays an important role in cyber-attack mitigation and it is widely regarded as crucial for network security [16, 17, 18]. However, it is not clear what segmentation architectures are best to maximise the security of a network when taking into accounts of other elements such as cost of implementation, impacts of mission performance etc.

2.5.1 Modelling and Simulation

Studies that focus on network segmentation, particularly those done by Wagner et al. [47, 48, 49, 50, 51], and also by others [53, 55], involve modelling and simulation to investigate the effectiveness of network segmentation against cyber-attacks, the trade-off between cost, security and operational impacts with network segmentation etc. These studies provide great benefits and insights of how defensive measures can be implemented in practice. This includes assessment of security risk; optimising between security, cost and performance; automated intrusion detection and prevention; and many more [52].

In [47], Wagner et al. utilise agent-based simulation to capture the dynamics between attackers, defenders, and mission operators in a given network environment. The study uses testbed environment where real software vulnerabilities and the corresponding exploits are used to calculate the probability that an enclave is compromised [47]. An *enclave* is defined

as a set of devices with *homogeneous reachability*, meaning that all devices within an enclave are interconnected to each other. The network model, however, does not capture the dynamics of devices within an enclave. Instead, the study follows a very popular epidemic model [57, 58] to measure how infected devices spread within an enclave. The equation of the infection model is shown in **Equation 2.1**.

$$I(t) = I(0)e^{\beta N t} \tag{2.1}$$

where t is time, $I(0)$ is the number of infected devices at $t = 0$, β is the infection propagation rate and N is the number of all devices inside an enclave. $I(t)$ represents the total number of infected devices at a given time t. The study executes with only 7 iterations and generate three solutions due to the intensive computation from testbed environment [47]. Therefore, it is limited to very small size networks.

In [48] Wagner et al. use continuous-time Markov chain (CTMC) to model the vulnerability of a given enclave. Security events, such as vulnerability arrival, patching, exploit development, exploit deployment, and enclave cleansing [48], are modelled with a Poisson process. Services represent software applications that run on a destination enclave. These services can have vulnerabilities that attackers can exploit to compromise the destination enclave [48].

A service can be in any of the three states at a given time: patched, vulnerable (with n number of vulnerabilities), and exploited. Then the study uses three Poisson rates to capture the transition between the three states [48]: vulnerability arrival rate Δ_{vuln}^{-1}, exploit development rate $\Delta_{exploit}^{-1}$, and patch rate Δ_{patch}^{-1}. The Markov process state for software service is shown in **Figure 2.8(a)**.

The Markov state for an enclave is shown in **Figure 2.8(b)**, where an enclave can be in any of the two states in a given time: compromised and clean. The transition between states is characterised by enclave cleansing rate Δ_{clean}^{-1} and enclave compromise rate Δ_{comp}^{-1}, where enclave compromise rate is the sum of all service compromise rates running on the enclave [48].

The main goal in [48] is to replace the testbed experiment in [47] with a continuous-time Markov chain model to efficiently compute the probability of compromising an enclave, to model the dynamics between attack progression and enclave cleansing.

Hemberg et al. investigate network segmentation as a co-evolution between attackers and defenders in [51]. The study follows a similar network model as Wagner et al. Specifically, Hemberg et al. follow the same method of computing the vulnerability of enclaves of a given network architecture from a CTMC simulation done by [48].

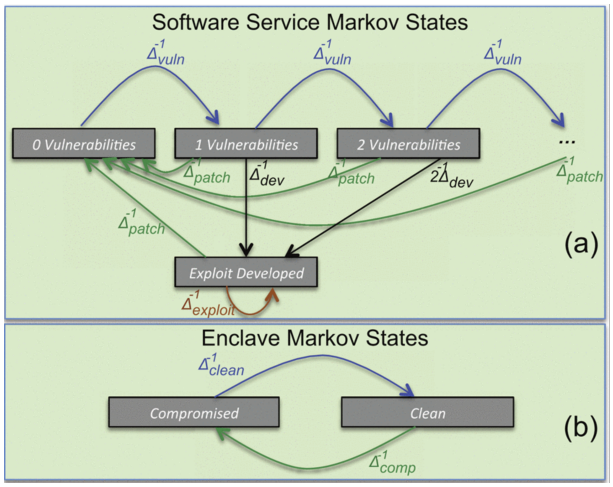

Figure 2.8: Markov process states for (a) software service (b) enclave.
Taken from [48].

The main differences between the two studies are as follow: (1) Hemberg et al. modify the infection model in **Equation 2.1** to **Equation 2.2**. **Equation 2.2** shows that the initial infection growth is exponential, and slows down (flattens out) as the number of infected devices approaches the maximum number of devices in the enclave [51]; (2) the objective function is to minimise "mission delay", which correlates to the amount of time that a mission cannot be carried out as a result of a mission device being either compromised or unavailable (due to enclave cleansing). In contrast, Wagner et al. include the security index, cost function and also mission delay for the objective function. (3) Hemberg et al. include a *tap* sensitivity for each enclave. A tap can be seen as similar to an alert system, illustrating that attack detection and prevention are imperfect. When the tap detects device compromise, it will trigger enclave cleansing which eliminate all infections inside the enclave but also introduce mission delay.

$$I(t) = \frac{nI(0)e^{\beta t}}{n + I(0)(e^{\beta t} - 1)} \tag{2.2}$$

2.5.2 Metaheuristics for Network Segmentation

An optimisation problem is about finding the best values for a set of variables in order to minimise (or maximise) an objective function with a given set of constraints [64]. Optimisation methods can be broadly divided into two classes: exact and approximate algorithms. Exact algorithms aim to solve optimisation problems which guarantee optimal solutions [64], whereas approximate methods provide good quality solutions that are usually near optimal. Metaheuristics fall into the category of approximate methods.

Real-life optimisation problems are often complex and difficult to solve, where the problems are highly complex, contain non-linear constraints, and the solution space is too large [64]. Exact algorithms for these problems require colossal amount of computational resources, and may generate poor quality solutions due to the complexity of the problems.

Metaheurstics, often inspired by natural process, have proven to deliver good solutions with reasonable amount of time [64]. Metaheuristics are becoming more popular in the recent years.

One of the problems with optimising network segmentation is that there is only vague guidance for how to apply network segmentation [49], and it is reasonable since it depends on the objectives of organisations. Network segmentation can be applied to even small size networks, with numbers of potential architectures. However, the complexity of the problem grows exponentially when the number of network increases [49]. This problem is further complicated by constantly evolving technologies, with more Internet of Things (IoT) devices being introduced to the networks, combining with continuously progressing cyber threats (for example, those we mention in **Section 2.1.3**), then the problems of securing the network whilst maintaining smooth operations can become overwhelming.

Simulated Annealing

[47, 49] use simulated annealing (SA) technique to optimise network segmentation. Simulated annealing is a metaheuristic that originates from annealing in metallurgy [49]. The general steps for SA algorithm are as follow:

1. Start with the initialisation of a basic solution.
2. Generate new solution neighbouring to the current solution.
3. Evaluate the fitness score of the two solutions (with fitness function).
4. If the new solution is superior to the current solution, accept the new solution.
5. If the new solution is inferior to the current solution, probabilistically accept new solution.
6. Repeat **step 2** until stopping criterion met.

SA imitates the annealing process by initially setting high probability (the temperature) of accepting inferior solution. The probability is then lowered after each iteration [49]. This process avoids staying at local maxima whilst exploring around the search space.

Specifically, Wagner et al. first generate a baseline architecture. The algorithm then explores alternative network segmentation architecture by making small changes to the current architecture. The small changes can be applied to either services or enclaves, involving either adding, merging, or removing them. Then the new architecture is evaluated. If the analysis of the new architecture is shown to be better than the current one, then this new architecture is accepted and it is used for the next iteration. If the new architecture is evaluated as inferior to current solution, then depending on the current number of iterations, probabilistically

accept the new inferior solution (the lower the number of iterations, the higher the probability of accepting inferior solution) [49].

Genetic/Evolutionary Algorithms

Inspired by the natural laws of evolution, genetic algorithms (GA) revolve around the idea of ”genes”, which define the characteristics of the individual. GA involves presenting each individual as a binary string [62] and mimics the process of natural selection, where the genes of stronger individuals get passed on to the next generation whilst weaker individuals perish. The implementation of GA can be generally summarised as the following:

1. Randomly initialise population.
2. Compute fitness score .
3. Selection of most ”fit” individuals, discarding the rests (survival of the fittest)
4. Crossover and mutation of individuals' genes (genes inheritance and random mutation).
5. If termination criterion not met (e.g. max iterations not reached, fitness score threshold not reached), then randomly initialise new individuals to replace those that were discarded in **step 3**. Repeat **step 2**.

In [51], Hemberg et al. use co-evolutionary algorithms to model the dynamics and the co-evolutions between attackers and defenders, where the goal of the attackers is to maximise mission delay by employing availability attacks whilst the goal of the defender is to minimise mission delay by using network segmentation [51]. The paper attempts to capture how attackers' and defenders' strategies grow with respect to each other. The paper also explores how certain network segmentation architectures might be effective for a period of time, before the attacker evolves and renders the architecture ineffective. This is equivalent to how cyber-attacks evolve and how network security experts attempt to predict and prevent those attacks from succeeding.

The papers by Wagner et al. and Hemberg et al. emphasise on the effects of mission delay, which correlates to the amount of time mission devices are compromised [49, 50, 51]. The studies do not include metrics which quantify the importance of devices/enclaves, where even one successful attack should be avoided at all cost (for example, a database with sensitive information).

2.5.3 Attack Graph

Shyner et al. propose using *Attack Graphs* (AGs) to show the attack paths that an attacker can take to move deeper into the network by successive exploitation of vulnerabilities [43].

AGs are directed graphs where the nodes represent the security states of the network after a successful attack [42]. However, this approach of risk analysis has ”unrealistic” data requirements [56]; for each node added to the graph, the computation requirement increases exponentially, making such analysis impossible for large networks, where there are hundreds of thousands of nodes. An example of AG is shown in **Figure 2.9**.

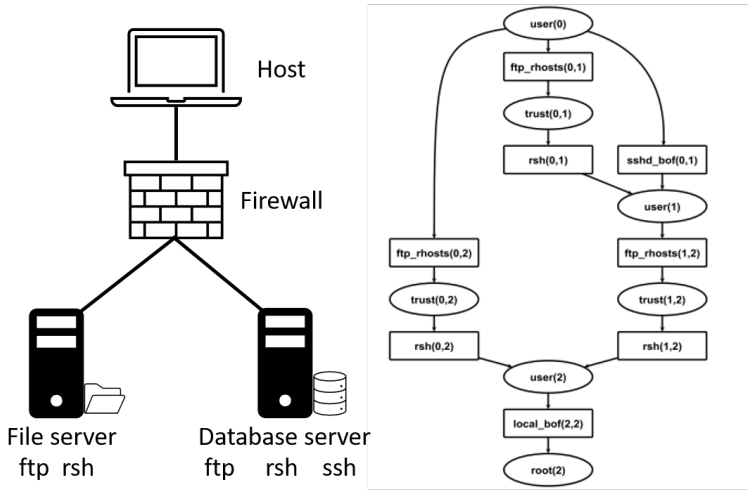

Figure 2.9: An example network configuration and the corresponding attack graph. *Taken from [45].*

Bayesian Attack Graph (BAG) model has been proposed by Liu and Man [44], providing a more compact representation of attack paths [44], allowing more scalable analysis of AGs. In [42], Muñoz-González et al. propose a revised Bayesian Attack Graph (BAG) model where the analysis of AGs is done using approximate inference techniques [42], particularly Loopy Belief Propagation, for scalable analysis of BAGs.

In these studies, each node in the Bayesian attack graph model represents a single security property violation state [44]; the edges that connect the nodes correspond to the possible exploits that can be used to progress to the next security state. The weights of the edges represent the probabilities of progressing to the next security state, which are also the probabilities of successful exploitation of the vulnerabilities that exist in the nodes [44]. These probabilities of exploitation can be used as a measure of risk to identify weak spots of a network.

2.6 Mathematical Background

This section describes the mathematical background required for the project. We use theories in this section to model the attack propagation, calculating the marginal probability of compromising an enclave and measuring the security score of a network segmentation architecture.

2.6.1 Probability Theory

Sum rule

The marginal probability of a variable A can be calculated by summing out all instances of B in joint probability $p(A, B)$ [46].

$$p(A) = \Sigma_B p(A, B) \tag{2.3}$$

Product rule

Product rules describes the relationship between joint probability of A and B in terms of the marginal and conditional probabilities [46].

$$p(A, B) = p(A|B)p(B) = p(B|A)p(A) \tag{2.4}$$

Bayes theorem

By rearranging **Equation 2.4**, we obtain the following equation, which is known as Bayes rule [46].

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \tag{2.5}$$

2.6.2 Bayesian Networks

A Bayesian network (BN) is a directed graph that is used to describe probability distributions. The nodes represent random variables and the directed edges represent the dependencies between the random variables [42]. For example, given joint distribution:

$$p(A, B, C) = p(A)p(C|A)p(B|A, C) \tag{2.6}$$

Equation 2.6 can be represented as a Bayesian Network shown in **Figure 2.10**

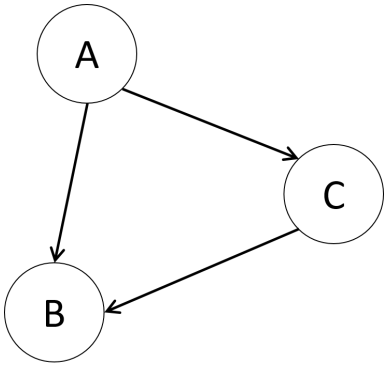

Figure 2.10: An example of directed graphical model showing the joint probability distribution over variable A,B and C. *Adapted from [46].*

2.6.3 Markov Random Fields

Markov random fields (MRF), also known as a Markov network, is the opposite of BN. Similar to BN, the nodes in MRFs represent random variables. However, the edges are undirected, representing potential functions between the nodes. For example, in **Figure 2.11**, we can represent the joint distribution as:

$$p(A, B, C, D, E) \propto \phi(A, B)\phi(B, C)\phi(B, E)\phi(C, D)\phi(D, E) \tag{2.7}$$

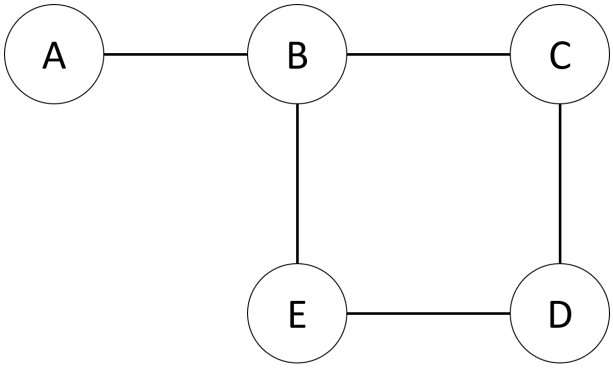

Figure 2.11: An example of Markov random fields with five variables A, B, C, D and E.

However, for the project, we currently only consider MRF that is a tree, where it represents a graph with no loops (**Figure 2.12**). A MRF that is a tree is also a pairwise MRF (but not the other way round), where any two non-adjacent variables are conditionally independent given all other variables [59].

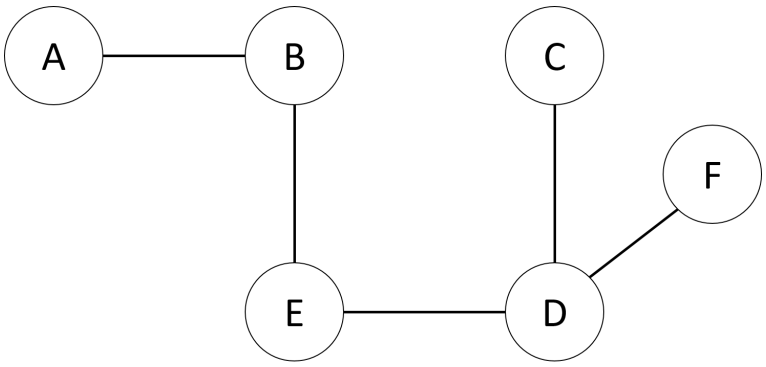

Figure 2.12: An example of MRF that contains no loops.

We can change **Figure 2.12** in a way that represents a network segmentation architecture (**Figure 2.13**), where the nodes represent enclaves and the edges are the services that are used between two enclaves. The term service represents any programs that allow communication between two enclaves, and that these programs contain executions that can lead to attacker control [56].

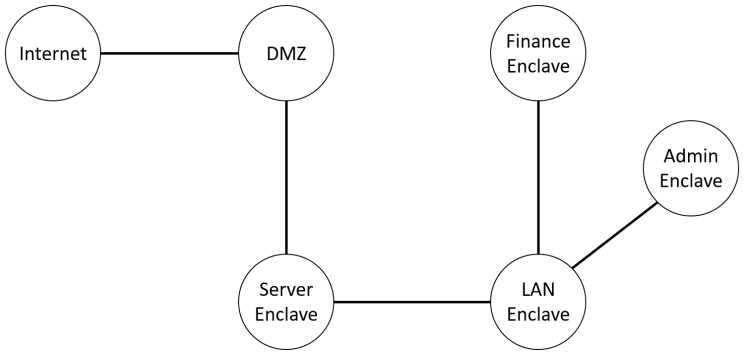

Figure 2.13: An example of network architecture that is represented as a MRF with no loops.

2.6.4 Belief Propagation

Belief propagation, is a special case of sum-product algorithm, allows the calculation of marginal probabilities on BN and MRF [46]. Before going into belief propagation, we have to first know the creation of a graph that can be used for belief propagation – a factor graph.

We can first obtain the following joint distribution from **Figure 2.12**:

$$p(A, B, C, D, E, F) \propto \phi(A, B)\phi(B, E)\phi(D, E)\phi(C, D)\phi(C, F) \tag{2.8}$$

Belief propagation allows the calculation of marginal probabilities in both BN and a MRF when the graph is a tree [42]. To first understand belief propagation, we first need to convert MRF to what is known as a *factor graph* [46]. A factor graph contains a node for every variable, but also additional nodes for each factor (usually depicted as squares) in the joint distribution. If we convert our graph in **Figure 2.12**, we would obtain the following factor graph: The joint distribution can be factorised as:

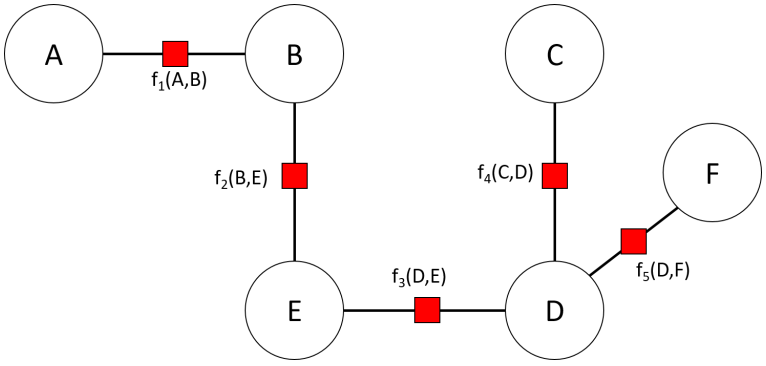

Figure 2.14: The factor graph representation of **Figure 2.12**.

$$p(A, B, C, D, E, F) = \Pi_{i=1}^{5} f_i(\mathbf{X}_i) \tag{2.9}$$

If we put **Figure 2.12** in the context of a network architecture in **Figure 2.13** (where A = Internet, B = DMZ etc.), we would obtain the follow factors:

$$\begin{aligned} f_1(A,B) &= p(A)p(B|A) \; f_3(D,E) = p(E)p(D|E) \\ f_2(B,E) &= p(B)p(E|B) \; f_4(C,D) = p(D)p(C|D) \\ & \qquad\qquad f_5(D,F) = p(D)p(F|D) \end{aligned} \tag{2.10}$$

Here we introduce the idea of message passing between variables and the message from factor f_i to variable X_j has the follow form:

$$m_{f_i \to X_j}(X_j) \equiv \sum_{X_k \in \mathbf{X}_s} f_i(X_j, \mathbf{X}_s) \prod_{X_k \in \mathbf{X}_s} m_{X_k \to f_j}(X_k) \tag{2.11}$$

where X_s is the set of nodes that are neighbours of f_i except X_j. The idea behind **Equation 2.11** is that to compute the message from f_i to X_j, we first collects all of messages from neighbours of f_i (except the message from X_j), and send them to X_j [42].

Take the graph below as an example (**Figure 2.15**). To pass a message from factor $f_1(A, B, E)$ to node B (denote as $m_{f_1 \to B}(B)$), we first collect messages that $f_1(A, B, E)$ receives, which is from (1) node A and (2) node E (red arrows). We do not include message from node B since we are sending message to node B. We multiply messages from (1) and (2), then we (3) summarise for all the variables that are not associated with B and sends the resulting message to node B (blue arrow).

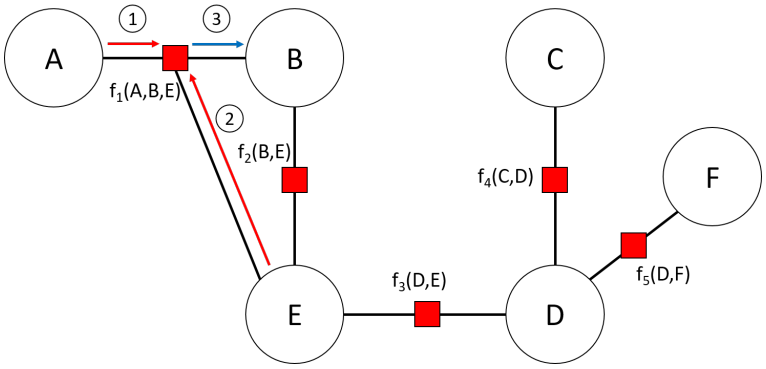

Figure 2.15: Message passing from factor $f_1(A, B, E)$ to node B

The marginal probability, or the belief, for a variable X_i can then be obtained by multiplying all adjacent incoming messages:

$$p(X_i) = \prod_{f_j \in neighbours(X_i)} m_{f_j \to X_i}(X_i) \tag{2.12}$$

where $neighbours(X_i)$ are the factor nodes that are adjacent to node X_i.

Chapter 3

Design

Our aim is to investigate how network segmentation can be implemented to enhance the security of the network, whilst also minimising the corresponding cost that comes with maintaining such architecture. Additionally, we also aim to design a system that explores around the solution space and finds network segmentation architectures that fit best into our evaluation metrics.

To achieve this, we first define the network model, which characterises the dynamics between attacker and defender; we also need to consider how we evaluate a given network architecture, which requires an evaluation method; and finally we need an optimisation technique that explore different network architectures in an attempt to find a network architecture that maximises security and minimises cost. This chapter is divided into three main parts: the network model, evaluation metrics, and the optimisation algorithm.

3.1 Network Model

We describe the network model in terms of the network environment, enclaves, services and *functional information flows*. The network environment describes the underlying assumption of the network model as a whole. The enclaves, services and functional informational information flows all together describe the reachability and security of the network.

3.1.1 Network Environment

We model the network environment similar to Wagner et al. 2018 [48] and Riordan et al. 2016 [56], where there exists a *communications topology*. The communication topology consists of a set of enclaves that are connected by communication pathways that are called functional information flows (FIFs) [48]. FIF includes any direct and indirect connections between source and destination enclaves.

Intermediate enclaves between source and destination enclave can be modelled as a single

FIF as long as an attacker from a source enclave can compromise a destination enclave without first compromising any of the intermediate enclaves between the two enclaves [48]. An example of this is shown and explain in **Figure 3.1**.

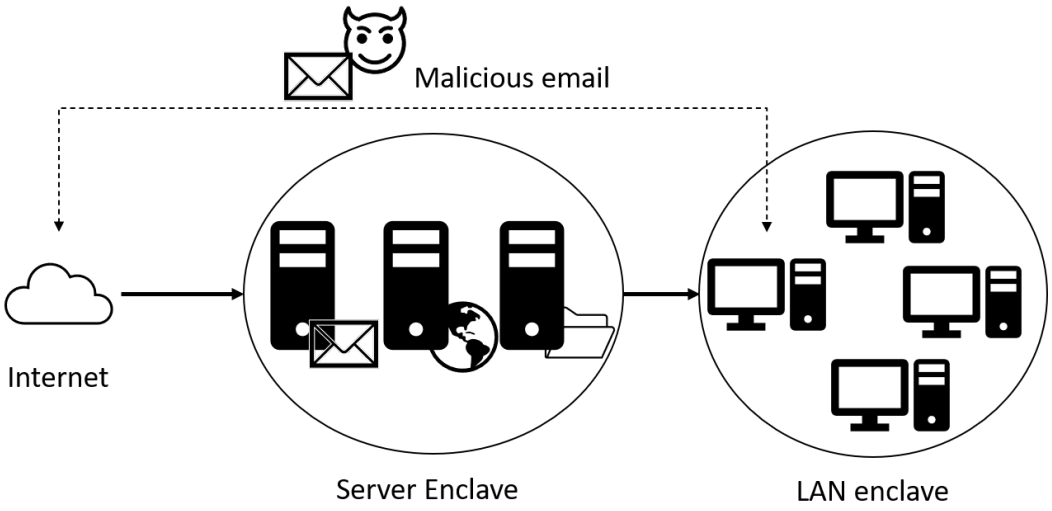

Figure 3.1: An attacker sends a malicious email to the to the DMZ, where the mail server picks it up in the server enclave, then the email is read by the user in the LAN enclave [48]. In this case a single FIF can be represented between the internet and LAN enclave (dotted line), since the attacker does not need to compromise the server enclave, or the DMZ.
Adapted from [56].

3.1.2 Enclaves

An enclave consists of a set of devices with homogeneous reachability [48]. We define homogeneous reachability as complete interconnection between devices; if the enclave is compromised, then we assume that all devices within the enclave are also compromised. Enclaves are connected via FIFs, and the destination enclave may run several software services that allow communication between the source enclave and the destination enclave. We consider the attack surface of an enclave as the union of the services running on the enclave.

An enclave E can be compromised under the following conditions: (1) There is a FIF connection between E and another enclave that is already compromised E_{comp} via a set of services; (2) when the attacker in E_{comp} successfully exploits a vulnerability in the service running in E.

3.1.3 Services

Services are defined as programs that are used for enclave-to-enclave or internet-to-enclave communication. These programs may contain executions that can lead to attacker control [56]. Our approach to computing service vulnerability differs to Wagner et al. approach with CTMC [48]. Although the CTMC model overcomes the problems with resource intensive

process of the generating a BAG, it omits details that a BAG provides. This includes all the possible attack paths at a particular time, and the probability of a successful exploitation for each attack path.

3.1.4 Attacker Model

We assume that an attacker resides in the internet, where it is always compromised. The attacker can only launch his/her very first attack when there is a FIF connection between the internet and an enclave via certain services. The vulnerability of the enclave is defined by the union of the vulnerability scores of all services that are running in the enclave. The more services that are running between a source enclave and a destination enclave (given that an attacker has already compromise the source enclave), the higher the probability for an attacker to exploit the vulnerabilities in the service and compromise the enclave.

3.2 Network Architecture Evaluation

3.2.1 Service Vulnerability

For each of the service s_i, if no known vulnerability is assigned, we assume that there is a zero-day vulnerability exists for an attack to exploit. In that case, We use CVSS score distribution (see **Figure 3.2**) to model the probability of compromising an enclave that uses the service. More precisely, we first obtain a random score (as integer) from the distribution (e.g. 7). Then we divide the value by 10 to obtain an estimate probability of compromising a service (0.7). However, since the value does not reflect the fact that there is a score range (0.6 to 0.7), we uniformly sample a value from that range. This also avoids the problem of obtaining a value that equals to exactly 1.0, which implies that the service is already compromised.

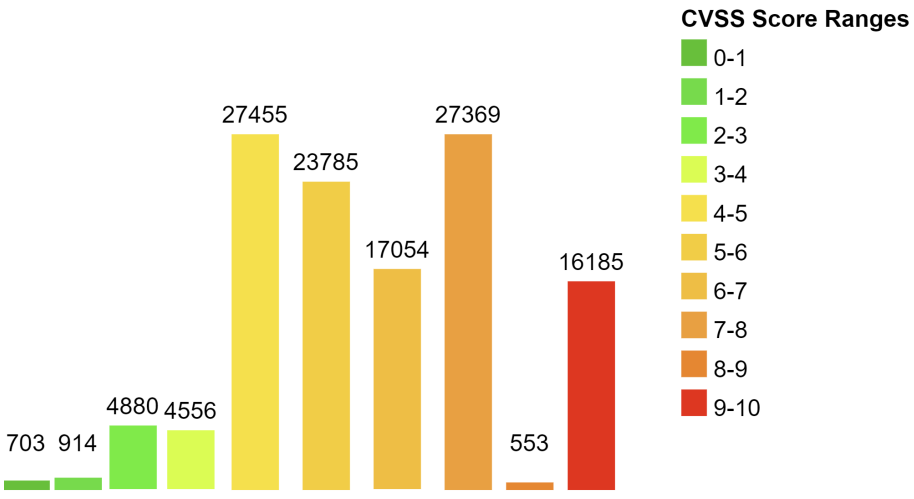

Figure 3.2: CVSS score distribution [60].

3.2.2 Enclave Vulnerability

Consider two enclaves e_1 and e_2, connected by a set of services s_1, ... ,s_n. Assuming that an attack is launched from e_1 to e_2, given that e_1 is already compromised, the probability of compromising e_2 would equal to the probability of compromising the service(s) that connects the two enclaves. This can be represented by **Equation 3.1** below:

$$p(e_2|e_1) = 1 - \Pi_i^n(1 - p(s_i)) \tag{3.1}$$

Enclaves Connected to the Internet

For an enclave e_1 that is connected to the internet e_0, we can calculate the marginal probability of compromising e_1 by utilising Bayes rule in **Equation 2.5** , the resulting equation is shown in **Equation 3.2**.

$$p(e_1|e_0) = \frac{p(e_0|e_1)p(e_1)}{p(e_0)} \tag{3.2}$$

Since e_0 is always compromised (as it represents the internet), we can assign $p(e_0) = 1$, and also $p(e_0|e_1) = 1$. This transforms **Equation 3.2** to **Equation 3.3**.

$$p(e_1|e_0) = p(e_1) \tag{3.3}$$

Finally, if we combine **Equation 3.1** with **Equation 3.3**, we can calculate the marginal probability of compromising e_1 in terms of the sum product of the services running between e_0 and e_1.

$$\begin{aligned} p(e_1|e_0) &= 1 - \Pi_i^n(1 - p(s_i)) \\ p(e_1) &= 1 - \Pi_i^n(1 - p(s_i)) \end{aligned} \tag{3.4}$$

The graphical illustration of the relationship between internet enclave e_0 and enclave 1 e_1 is shown in **Figure 3.3** below.

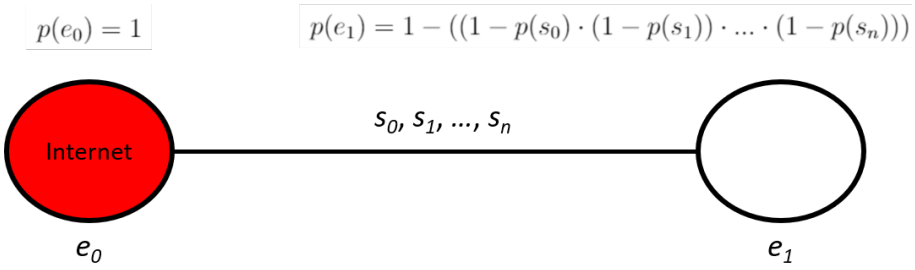

Figure 3.3: Graphical illustration of computing marginal probability of compromising enclave 1, e_1 from internet enclave e_0.

To put **Figure 3.3** in the context of factor graph that we previously mention in **Section 2.6.4**. We can create a factor node $f_1(e_0, e_1)$ between e_0 and e_1, where

$$f_1(e_0, e_1) = p(e_0)p(e_1|e_0)$$

Then we can deduce that

$$f_1(e_0, e_1) = p(e_1)$$

and the message would equal to

$$\begin{aligned} m_{f_1 \to e_1}(e_1) &= f_1(e_0, e_1) \prod_{X_k \in \mathbf{X}_s} m_{X_k \to f_j}(X_k) \\ &= p(e_1) \cdot (p(e_0)) \\ &= p(e_1) \end{aligned} \tag{3.5}$$

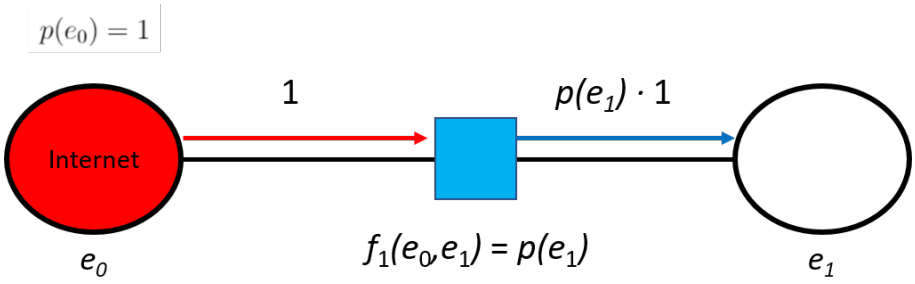

Figure 3.4: The message from factor $f_1(e_0, e_1)$ to node e_1 is equal to summing all messages to $f_1(e_0, e_1)$ (which equals to 1) then multiply the sum with $f_1(e_0, e_1)$, which is equal to $p(e_1)$.

3.2.3 Security Index

We follow Wagner et al. approach [50] to define the security index as the expected probability of compromising an enclave over all enclaves (**Equation 3.6**) [50]. The metric is normalised to [0,1]. The lower the security score, the more secure overall for the network.

$$Sec = \frac{\Sigma_{e \in Enclaves} p(e)}{|Enclaves|} \tag{3.6}$$

where $p(e)$ represents the marginal probability of compromising an enclave and $|Enclaves|$ is the total number of enclaves in the network.

3.2.4 Cost Function

We introduce the cost metric to model the cost of adding more enclaves to the network. It represents the cost of implementing and maintaining N number of enclaves, given a maximum M number of enclaves an organisation can handle. k represents the steepness constant which is used to characterise relationship between implementing more security (enclaves) and cost [49] (see **Figure 3.5**).

$$C = \frac{e^{\frac{N \cdot k}{M}} - 1}{e^k - 1} \tag{3.7}$$

As shown in **Equation 3.7**, the cost is exponential, with N number of enclaves that are implemented being part of the exponent. This represents the trade-off between security and cost; the more secure the network (more enclaves implemented), the more IT cost there is to maintain [49]. The function is then normalised such that the cost is in range between [0,1], which is also the same range as security score. Normalisation avoids a particular score from dominating the fitness function.

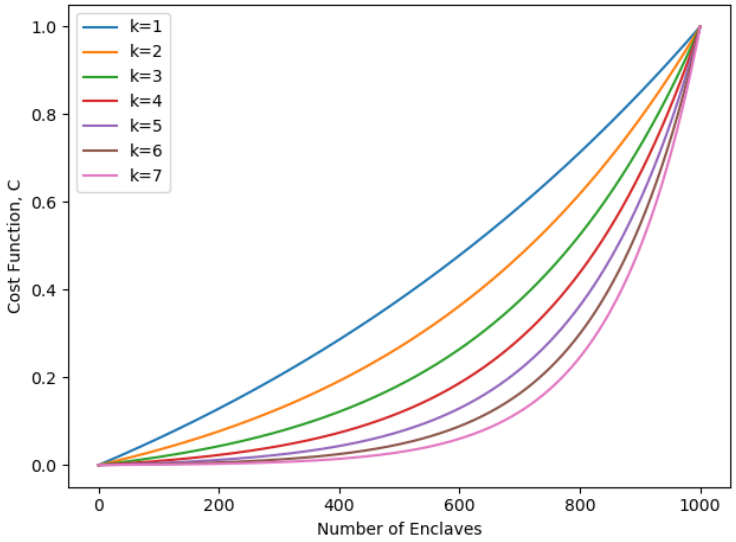

Figure 3.5: Cost function for maximum number of enclaves $M = 1000$ with different values of k

As shown in **Figure 3.5**, the value of k can be used flexibly, where low value of k (e.g. $k = 1$ defines a linearly increase in cost against number of enclaves N, whilst high value of k (e.g. $k = 7$) characterises a exponential increase in cost with respect to N [50].

3.2.5 Fitness Function

Fitness function is a special type of objective function for evaluating how close a given solution is to achieving certain goals. The difference between objective function and fitness function is that an objective function is a function that aims to optimise a solution, whereas fitness function is a function that acts as a guide towards optimal solution. Fitness function is often used in evolutionary algorithms.

We combine the two metrics (security score and cost score) linearly to form our fitness function, L (**Equation 3.8**). Since the higher the security score means enclaves are more vulnerable, and the higher cost score means more resources are needed to maintain the enclaves, we put a negative sign to both scores so that the aim is to maximise the fitness function, which fits into the context of GA (and other population-based metaheuristics), where the higher the fitness score, the better fit the individual, thus the higher chance of survival.

$$L = -(w_{sec} \cdot Sec + w_{cost} \cdot C) \tag{3.8}$$

The values of w_{sec} and w_{cost} are weights in range [0,1] that are determined by the practitioner, representing the prioritisation between security and cost, subject to constraint w_{sec} + w_{cost} = 1.0 [49]. It also makes sure that L is normalised to [-1,0].

3.3 Optimisation

In this section we describe the optimisation algorithm that we use to obtain near optimal network architecture.

3.3.1 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is a metaheuristic that is based on swarm behaviour. It is inspired by the natural process [62] of bird flocking, fish schooling, and swarming theory [63], where the movement of each individual is affected by the social behaviour of the flock as a whole (**Figure 3.6**).

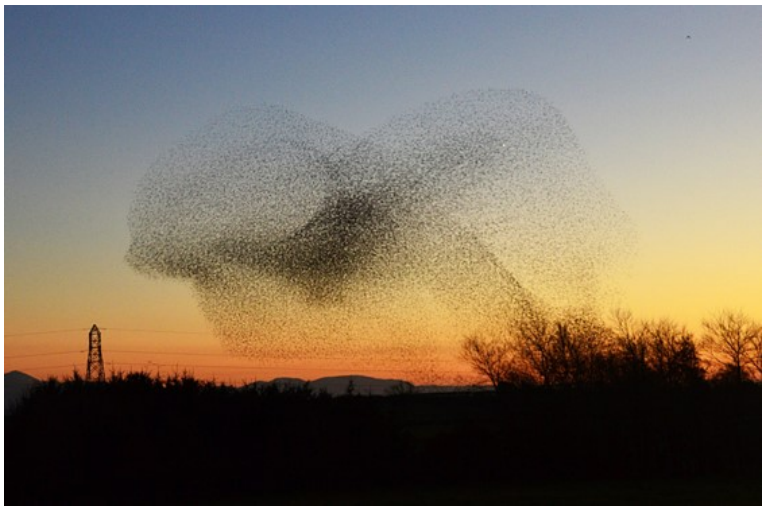

Figure 3.6: Illustrating the movements of bird flock, where the movements of the birds are affected by the direction of the swarm as a whole.

PSO starts with randomly generating a population of solutions. The solutions are called particles and the population is referred to as a swarm [62]. Each particle has its own velocity and position, and the velocity is determined by the particle's current velocity, the particle's best position, and the swarm's best position. The equation for updating a particle's velocity is given in **Equation 3.9** below:

$$v_{i,d} \leftarrow w \cdot v_{i,d} + \phi_p r_p (p_{i,d} - x_{i,d}) + \phi_g r_g (g_d - x_{i,d}) \tag{3.9}$$

where w is the inertia weight, d is the dimension, ϕ_p and ϕ_g are the acceleration constants, r_p and r_g are random uniform numbers between [0,1], and $p_{i,d}$ and g_d are the particle's best position and the swarm's best position respectively [62]. The intuition behind **Equation 3.9**

is explained in **Figure 3.7**

The ”main” character **A** (person in solid red) is trying to achieve the goal of getting to the crown as close as possible. His current position is denoted as **x**. Lets say **A** used to be at the position **p**, and it was his closest to the crown (his personal best position). The difference between **p** and **x** will determine part of **A**'s velocity $v_{\mathbf{p}}$. Lets say someone in the population used to be at the position where it is the closest to the crown out of **everyone** (global best position) denoted as **g**. **A**'s velocity will also take into account of the difference between **g** and **x**, we denote this part of velocity as $v_{\mathbf{g}}$. Finally, his current velocity (denoted as $v_{\mathbf{I}}$) will also have an effect to his movement. Therefore, **A**'s overall velocity v would be summing up $v_{\mathbf{I}}$, $v_{\mathbf{p}}$ and $v_{\mathbf{g}}$ all together.

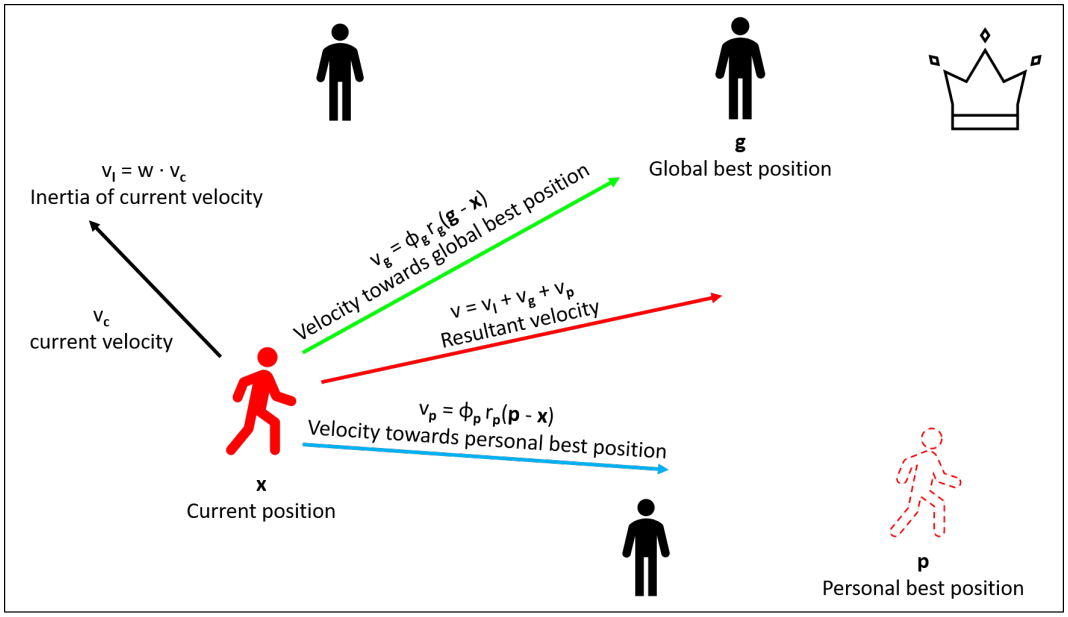

Figure 3.7: Intuitive illustration of particle swarm optimisation where each person represents one of the particles in the population. The person in red (and also other people) is trying to get to the crown as close as possible. His velocity is affected by his current proximity to his personal best score, his proximity to global best score, and his current velocity.

After computing the particle's velocity, the particle's position needs to be updated. The equation for updating particle's position is shown in **Equation 3.10** below:

$$x_i \leftarrow x_i + lr \cdot v_i \tag{3.10}$$

where lr is the learning rate. With reference to **Figure 3.7**, we can think of the update of position as person **A**'s new position after moving with velocity v.

3.3.2 Particle Swarm Optimisation for Network Segmentation

To put PSO into the context of our model. We consider each ”particle” as a distinct potential solution for solving the best network segmentation architecture; so if we have n particles then we would have n number of solutions. We consider the ”position” of the particle as the characteristics of the network segmentation architecture, such as the number of enclaves, connections between the enclaves, and the services used between the enclaves etc. However, for this experiment, we only model the ”velocity” of the particle as the change in number of enclaves in the network architecture.

To overcome the limitation of modelling only one feature as the velocity, we randomly reconfigure the connectivity each time the solution is updated (whilst satisfying traffic requirements), which allows more exploration options.

We have chosen PSO for optimisation since it differs to the approaches by Wagner et al. [49] and Hemberg et al. [51] where SA and GA are used. PSO is similar to GA in the sense that it is population-based a very intuitive algorithm that is very easy to follow, and studies have shown that PSO out performs SA overall; both in terms of time efficiency and proximity to optimised solution [61].

Although PSO does not guarantee to provide an optimal solution [64], in the context of network segmentation and cybersecurity where the problem and the search space are often complex, the efficient exploration of the search space of PSO can help finding good quality segmentation architectures.

Chapter 4

Implementation

This chapter describes the detail implementation of the network model, network architecture evaluation methods, and the optimisation technique (PSO) we describe in **Chapter 3**. We provide pseudocode also the external libraries we use to achieve this. All codes, algorithms and analysis are implemented with Python 3.7.5.

4.1 Flow of Control for Evaluating One Solution

An overview of the flow of control for evaluating one solution (hence excluding optimisation step) is given in **Figure 4.1** below. The program starts with initialising services, and combine that with other inputs such as number of enclaves N and traffic requirement F. After obtaining all three factors, the program generates a graph with only connections that are specified by the traffic requirements. Then randomly joining unconnected nodes to complete the graph. After that a factor graph is created and the marginal probability of compromising each enclave is calculated using belief propagation. Finally, the fitness score is calculated based on the metrics described in **Section 3.2.3** to **Section 3.2.5**.

4.2 Initialising Services

The first step to build our model is the initialisation of all services (**Algorithm 1**). For all services we assign a random CVSS score according to the distribution shown in **Figure 3.2**. Specifically, we use Python "random" library and also **Numpy** [1]for uniform distribution to achieve this (see **Figure C.1** in **Appendix C** for more details). If the score comes out to be 10, we reduce it by 0.5. It is also possible to specify the vulnerability score of a particular service if necessary.

[1] **https://numpy.org/**

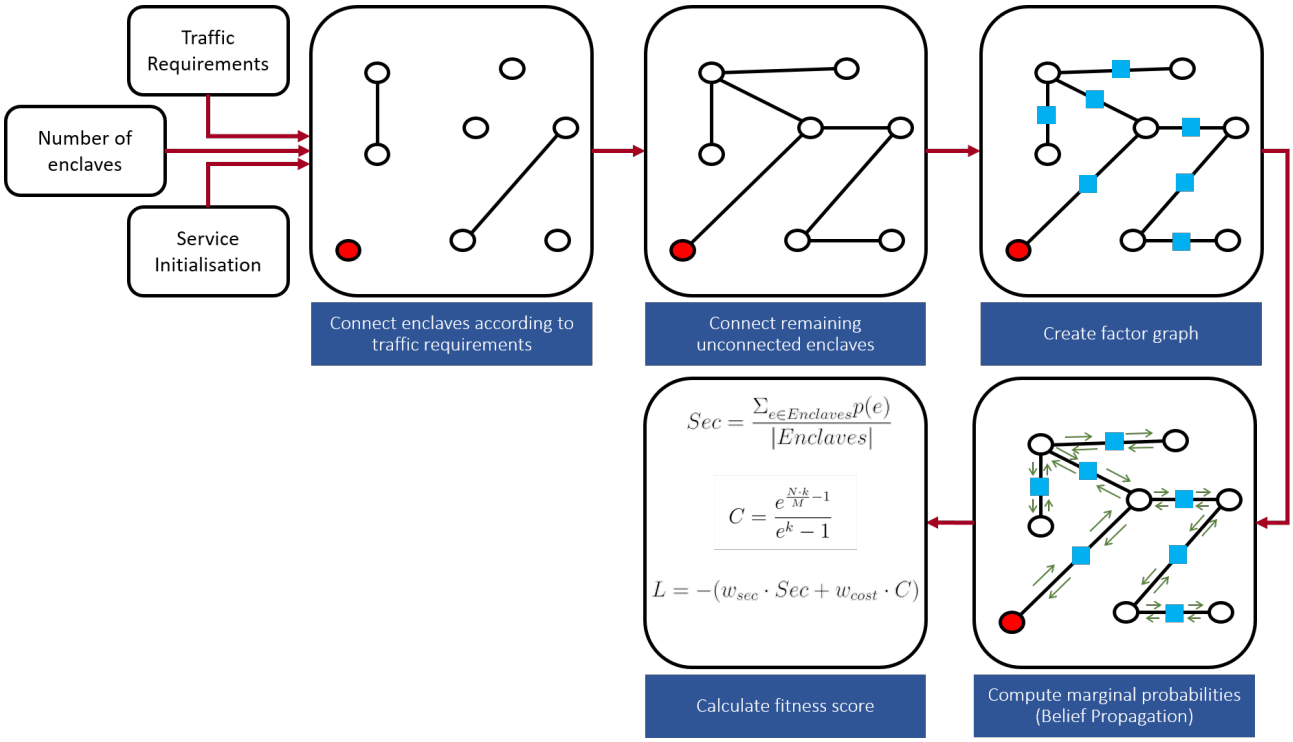

Figure 4.1: The control flow of the program (excluding optimisation), from initial inputs to graph creation, edges connection, factor graph creation, marginal probability computation, and fitness score calculation.

Algorithm 1: Service Vulnerability Initialisation

Input: S, services
for *each service* s_i ***in*** *S* **do**
  cvss_score ← CVSS distribution ▷ Draw integer 1-10 from distribution
  if *cvss_score == 10* **then**
    cvss_score ← cvss_score – 0.5
  end
  cvss_score ← cvss_score/10
  $p(s_i) \sim U$ (cvss_score – 0.1, cvss_score) ▷ Assign service compromise probability
end

4.3 Initialising Network Model

We represent our model as a graph G: $< E, C, W >$ where E represents a set of enclaves (vertices), C represents a set of connections between enclaves (edges), and the weights of the edges are represented as W We also show what services are running between any two enclaves.

Given traffic requirements F as a set of n-tuples: $< U, V, S >$, where U and V ($U \neq V$) represent enclaves that need to be connected via services S. For example: $F = \{(0, 1, 4), (3, 5, 2)\}$ represent the traffic requirements where enclave 0 and enclave 1 have to be connected via service 4, and enclave 3 and enclave 5 have to be connected via service 2. We can compute the minimum number of enclaves required to satisfy the traffic requirements (**Figure C.2** in **Appendix C**).

After calculating the minimum number of enclaves required for the network mode, we can check whether the number of enclaves N we want for our model meets the requirements. Then we start initialising our network model (**Algorithm 2**). There we initialise our connections (**Algorithm 3**). After completing the connections between nodes, we move on to create factor graph and compute the marginal probability of compromising each enclave in the network (**Algorithm 5**).

Algorithm 2: Network Model Initialisation

Input: G, graph; N,number of enclaves; F, traffic_requirement; S, services
Initialise Enclaves Connections ▷ Algorithm 3
Compute Marginals(G) ▷ Algorithm 5

4.4 Initialising Enclave Connections

Since we are not modelling at device level, the only information that each enclave contain is the enclave vulnerability score, the node that it needs to connect to, and via the corresponding service(s), and the ID of the enclave.

The connections between enclaves can be represented as the weight of the edge joining the two enclaves, where the weight corresponds to the probability of compromising the services running between the two enclaves. We have implemented this with **NetworkX** [2], which is a Python library for studying complex networks.

Algorithm 3 starts with adding $N + 1$ nodes to the graph, where node 0 represents the

[2](**https://networkx.github.io/**)

internet, and the other nodes represent enclaves in the network. Then we join the enclaves with services according to the traffic requirements. The function add_edge joins node u and node v with service s. Then we check whether the graph is connected using breadth-first search (BFS) in **Algorithm 4**. We consider the node around node 0 as the ”main” graph, meaning that unconnected nodes are nodes that do not have a path that lead to node 0. connected is a list of length $N + 1$ where each element i is either True or False, representing whether node i is connected. If i is False, then we would randomly join node i to the connected graph, prioritising services that have not been used. We use BFS again to update the list of connected. We iteratively connects unconnected node until i reaches N. Note that we only join the nodes together with only one service (unless traffic requirements specify) to follow the principle of least privilege, where communication and access to resource are not necessary unless they are essential for the business' operation.

Algorithm 3: Initialise Enclave Connections

```
Input: G, graph; N, number of enclaves; F, traffic_requirement; S, services
for i = 0,1, ...,N do
    G.add_node(i)                                   ▷ Internet is enclave 0
end
for u, v, s in F do
    G.add_edge(u,v,s)                ▷ Add edges according to traffic requirements
                                                    and assign service s to the edge
end
connected ← BFS(G,0)                  ▷ Look for connections around the internet
iter_list ← 0,1,...,N
shuffle(iter_list)                      ▷ Shuffle the list for random connection
for i in iter_list = {4,1,0,N, ...} do
    if i ∉ connected then
        us ←unused_service(S)                       ▷ Check for unused service
        r ← random integer in [0,length(connected)-1]
        G.add_edge(i,connected[r],us)   ▷ Join unconnected node to a connected
                                                    node randomly, then assign unused
                                                    service us to the edge
        connected ← BFS(G,0)          ▷ Recompute the connections from node 0
    end
end
```

We utilise breadth-first search (BFS), shown in **Algorithm 4**, to check whether our graph is fully connected. If there are unconnected nodes, then we would assign the unconnected nodes to a randomly connected node (we start BFS from the node 0, the internet).

Algorithm 4: Breath First Search (BFS)

```
Input: G, graph; u, starting node
visited ← ∅
queue ← {u}
while queue do
    vertex ← queue.pop
    if vertex ∉ visited then
        visited.add(vertex)
        for v in G.edges(vertex) do
            queue.append(v)
        end
    end
end
return visited
```

4.4.1 Marginal Probabilities

For computing marginal probabilities for each enclave, we utilise a Github repository[3]which implements belief propagation. The library helps creating factor graph and computing marginal probabilities of compromising an enclave (using belief propagation). As mention in **Section 2.6.3**, we currently only consider MRF that is a tree, we would raise error if it is not the case.

Algorithm 5: Compute Marginals

```
Input: G, graph;
fg ← create_factor_graph(G)
if fg is a tree then
    bp = belief_propagation(fg)
else
    Raise error
end
for i = 0,1, ...,N do
    e_i.vulnerability ← compute_marginal(i)      ▷ Obtain marginal probability of
                                                   compromising enclave i
end
```

[3]**https://github.com/krashkov/Belief-Propagation**

4.5 Particle Swarm Optimisation

As previously discussed in **Section 3.3.2**, we define each ”particle” as a solution of a network segmentation architecture. The ”position” defines the features of the network architecture, such as number of enclaves, connection between enclaves, and the services used for the connection. We define ”velocity” as the change in number of enclaves. After initialising services in **Algorithm 1**, **Algorithm 6** is a pseudocode that describes the rest of the process of the program.

At the start of the algorithm, we use discrete uniform distribution to randomly initialise the number of enclaves N. Then we input this, along with services, traffic requirement to build and also evaluate the network architecture of each particle. Then, for each particle, we set the current network architecture of the particle as the best personal position (since it is the first one). We then calculate the fitness score of each particle (**Algorithm 7**) and set the network architecture of the particle that has the highest fitness score as the global best position. We then randomly initialise the velocity of the particle (corresponds to the change in number of enclaves). This completes the initialisation step (the first for loop).

Then, for each iteration until T, and for each particle i:

1. Compute number of enclaves N_i for the particle. This means that given the change in number of enclaves, add that to the current number of enclaves with learning rate lr. $N_i \leftarrow N_i + lr \cdot v_i$ (**Equation 3.10**).

2. Input N_i, services S, and traffic requirements F into **Algorithm 2**. This creates a new network architecture with new number of enclaves. This also calculates the marginal probabilities of compromising the enclaves for the current network model of the particle (**Algorithm 3**, **Algorithm 4**, **Algorithm 5**).

3. Pick a random uniform number: r_p, $r_g \sim U(0, 1)$

4. Calculate the fitness score for the particle (the security score, cost score and fitness function discussed in **Section 3.2.3** to **Section 3.2.5**). The calculation is also shown in **Algorithm 7**.

5. If the particle’s current fitness score is higher than the particle’s previous personal best score, update the position of the particle as the new personal best position. And if the particle’s current fitness score is higher than the global best score, then update make the current particle’s position as the global best position.

6. Update particle’s velocity v_i by following **Equation 3.9**. Again, the velocity represents change in number of enclaves.

7. Repeat **Step 1**.

Algorithm 6: Particle Swarm Optimisation

Input: *n*, population size; P, population of network models; S, services; T, iterations
for *particle i = 1, ..., n* ***in*** *population* **do**
  Initialise particle's position: $\mathbf{x}_i \sim U$ (min enclaves, max enclaves) ▷ Algorithm 2
  Set the current particle position as the particle's best position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
  if *fitness(**p**$_i$)* $>$ *fitness(**g**)* **then**
    update swarm's best position: $\mathbf{g} \leftarrow \mathbf{p}_i$
  end
  difference $\leftarrow$ min enclaves – max enclaves
  Initialise particle's velocity: $\mathbf{v}_i \sim U$ (– |difference|, |difference|)
end
while *t = 0,1,...,T* **do**
  for *particle i = 1, ..., n* ***in*** *population* **do**
    Update particle's position: $\mathbf{x}_i \leftarrow \mathbf{x}_i + lr\ \mathbf{v}_i$ ▷ Equation 3.10 and Algorithm 2
    Pick random numbers: $r_p, r_g \sim U(0,1)$
    if *fitness(**x**$_i$)*$>$*fitness(**p**$_i$)* **then**
      update particle's best position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
      if *score(**p**$_i$)* $>$ *score(**g**)* **then**
        update swarm's best position: $\mathbf{g} \leftarrow \mathbf{p}_i$
      end
    end
    Update velocity: $\mathbf{v}_i \leftarrow w \cdot \mathbf{v}_i + \phi_p r_p(\mathbf{p}_i - \mathbf{x}_i) + \phi_g r_g(\mathbf{g} - \mathbf{x}_i)$ ▷ Equation 3.9
  end
end

Algorithm 7 implements the fitness function and the evaluation metrics we describe in **Section 3.2.3** to **Section 3.2.5**. $p.E$ represents all enclaves in the particle (excluding internet enclave), $p.|E|$ represents the number of enclaves, e_i corresponds to individual enclave i in the particle, and e_i.vulnerability is the marginal probability of compromising enclave i.

Algorithm 7: Compute Particle Fitness Score

Input: p, particle; p.E, enclaves of the particle
security_score $\leftarrow$ 0
for *each enclave e_i in p.E* **do**
  security_score $\leftarrow$ security_score + e_i.vulnerability
end
security_score $\leftarrow \frac{security\_score}{p.|E|}$
cost_score $\leftarrow \frac{e^{\frac{N \cdot k}{M}} - 1}{e^k - 1}$
p.fitness $\leftarrow$ – (w_{sec}· security_score + w_{cost}· cost_score)

4.6 Program Execution Overview

An overview of the program flow of control is shown in **Figure 4.2**. To summarise, the program:

1. First initialises service, takes in traffic requirements and number of enclaves as inputs.
2. Constructs a graph that has connection which satisfies the traffic requirements.
3. Connects the remaining unconnected nodes randomly.
4. Creates a factor graph.
5. Computes message passing (belief propagation) to obtain marginal probabilities of compromising enclaves.
6. Evaluates the cost score, the security score, and finally the fitness score.
7. Compares the solution to other better solutions to obtain the velocity for particle swarm optimisation.
8. Calculates new position (number of enclaves).
9. Repeats **Step 2** until the maximum number of iterations has reached.

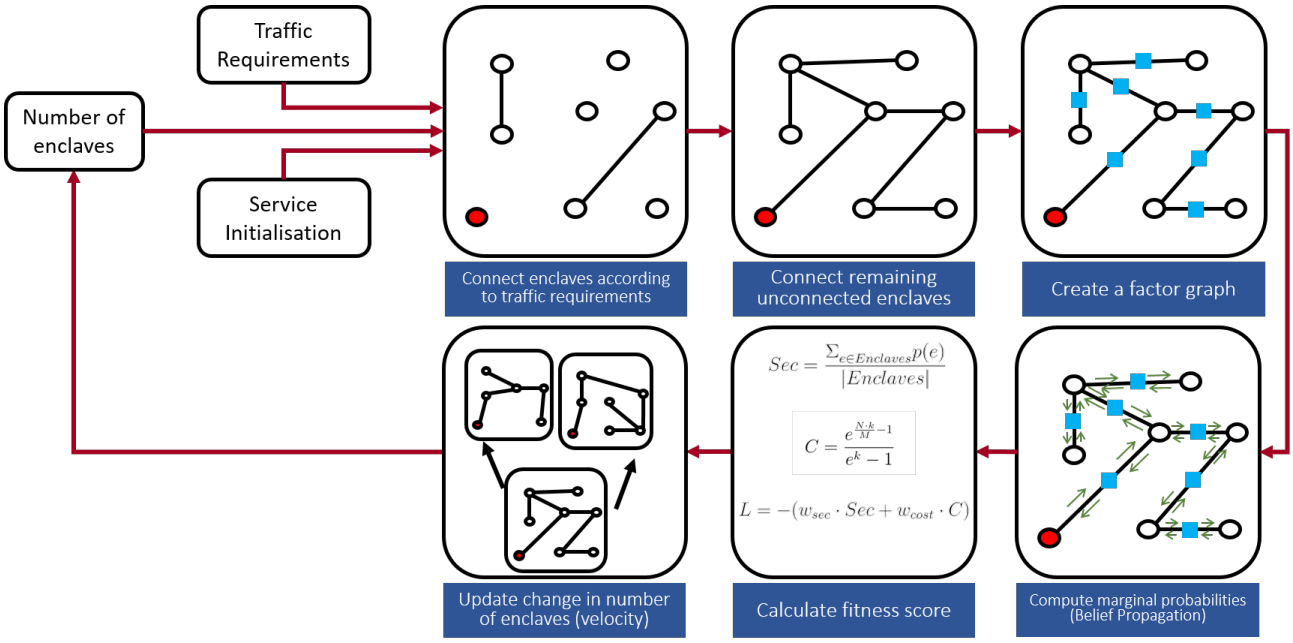

Figure 4.2: An overview of the overall control flow of the program, including optimisation.

Chapter 5

Results and Evaluation

This chapter describes and evaluates the results we obtained for the experiment. We first investigated and explored different hyperparameters values for PSO to find the set of hyperparameters that suits our experment. Then we explored the ”best” network segmentation architectures that would be produced under three different scenarios, one where security and cost were equally weighted ($w_{sec} = w_{cost} = 0.5$), one where security was prioritised more than cost ($w_{sec} = 0.8, w_{cost} = 0.2$), and one where cost was prioritised more than security ($w_{sec} = 0.2, w_{cost} = 0.8$). We then discuss the effectiveness of PSO, evaluate the time complexity of our simulation model, and finally, the strengths and limitations of our experiment.

Here we include a table of common symbols that we use in the section for reference.

Symbol	Description	Symbol	Description
N	Number of enclaves	n	Number of particles
M	Maximum number of enclaves	ϕ_p	Acceleration constant towards personal best position
w_{sec}	Weights of security score	ϕ_g	Acceleration constant towards global best position
w_{cost}	Weights of cost score	w	Inertia weight
k	Steepness constant in cost function	lr	Learning rate for updating particle's position
		T	Maximum iterations for PSO

Table 5.1: Symbols and their descriptions for reference and clarity. Left-hand side of the table shows symbols associated with network architecture and evaluation metrics. Right-hand side of the table represents symbols associated with PSO.

All of the plots and graphs are created using **Matplotlib** [1], which is a Python library for creating static, animated, and interactive visualisations [65].

[1] **https://matplotlib.org/**

5.1 PSO Hyperparameters

Recall from **Equation 3.9**, that the equation for updating particle's velocity has the following form:

$$v_{i,d} \leftarrow w \cdot v_{i,d} + \phi_p r_p(p_{i,d} - x_{i,d}) + \phi_g r_g(g_d - x_{i,d})$$

and the equation for updating a particle's position is:

$$x_i \leftarrow x_i + lr \cdot v_i$$

The hyperparameters for the above two equations are: ϕ_p, ϕ_g, w and lr.

We conducted a few experiments to investigate the effects of hyperparameters on the quality of solutions. The motive behind this is that some hyperparameters in PSO share similarity with the concept of step size; when these hyperparameters are too large, the particles would explore the search space quickly and would constantly skip over the optimal solution. This corresponds to step size being too large and it can take a long time to converge.

For this experiment, we set the following parameters $k = 7, w_{sec} = 0.5, w_{cost} = 0.5, lr = 0.5, w = 1, \phi_p = 1.5, \phi_g = 1.5, n = 50, T = 100$ and $M = 50$. We also set the following simple traffic requirements:

1. Enclave 0 (internet) has to be connected to enclave 1 via service 0. Service 0 has vulnerability score of 0.950.
2. Enclave 2 has to be connected to enclave 3 via service 1, also with vulnerability score of 0.950.

The traffic requirements make sure that solutions cannot just be one or a few enclaves with service that has the lowest vulnerability score. The connection between enclave 0 and enclave 1 can also be interpreted as connection between the internet and DMZ. As DMZ is public facing and has very soft firewall rules, it is easily one of most vulnerable enclaves in the network. Connection between enclave 2 and 3 can be interpreted as a two-way trust relationship in the network, meaning that if any one of the enclaves are compromised, it is not difficult for an attacker to compromise the other enclave as a result of the trust relationship.

Figure 5.1 shows the different states of the particles at different number of iterations. Note that even at iteration 100, all the particles do not narrow down their search space; the particles still explore the extreme ends of the search space, where either minimum security is implemented ($N = 4$), or adding as many enclaves as possible ($N = 100$). This is also supported by inspecting the average fitness score of the population per iteration, as shown in **Figure 5.2**, where the average fitness score across population is still fluctuating in the same range after 100 iterations. Although both figures suggest that the hyperparameters are poor,

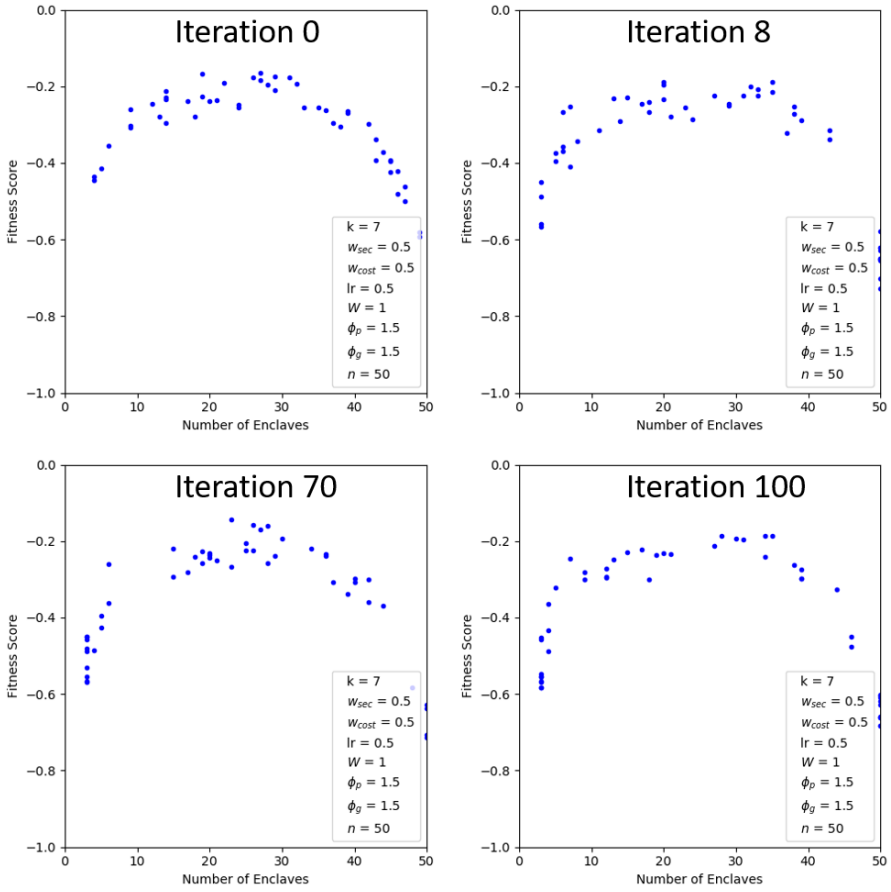

Figure 5.1: Shows the particles (50) at 4 instances; iteration 0 (initialisation), iteration 8, iteration 70 and iteration 100.

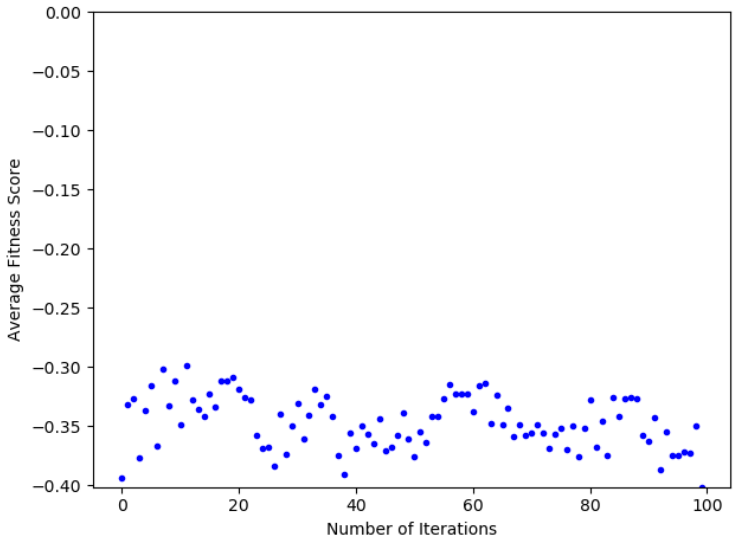

Figure 5.2: The average fitness score for the population, as seen in the figure, does not improve even after 100 iterations.

it is still possible for particles to reach the optimal solution with large step size, it is just a matter of time. Additionally, average fitness score does not imply that a particle does not reach a good quality solution; we show the ”best” segmentation architecture for this particular experiment in **Figure 5.3**. The network architecture has 28 enclaves, with -0.119 fitness

score, which was significantly higher than the average that we observed from the above two figures. The reason can be due to the fact that there are particles at the extreme ends of the search space, which act as outliers and pull the average fitness scores down.

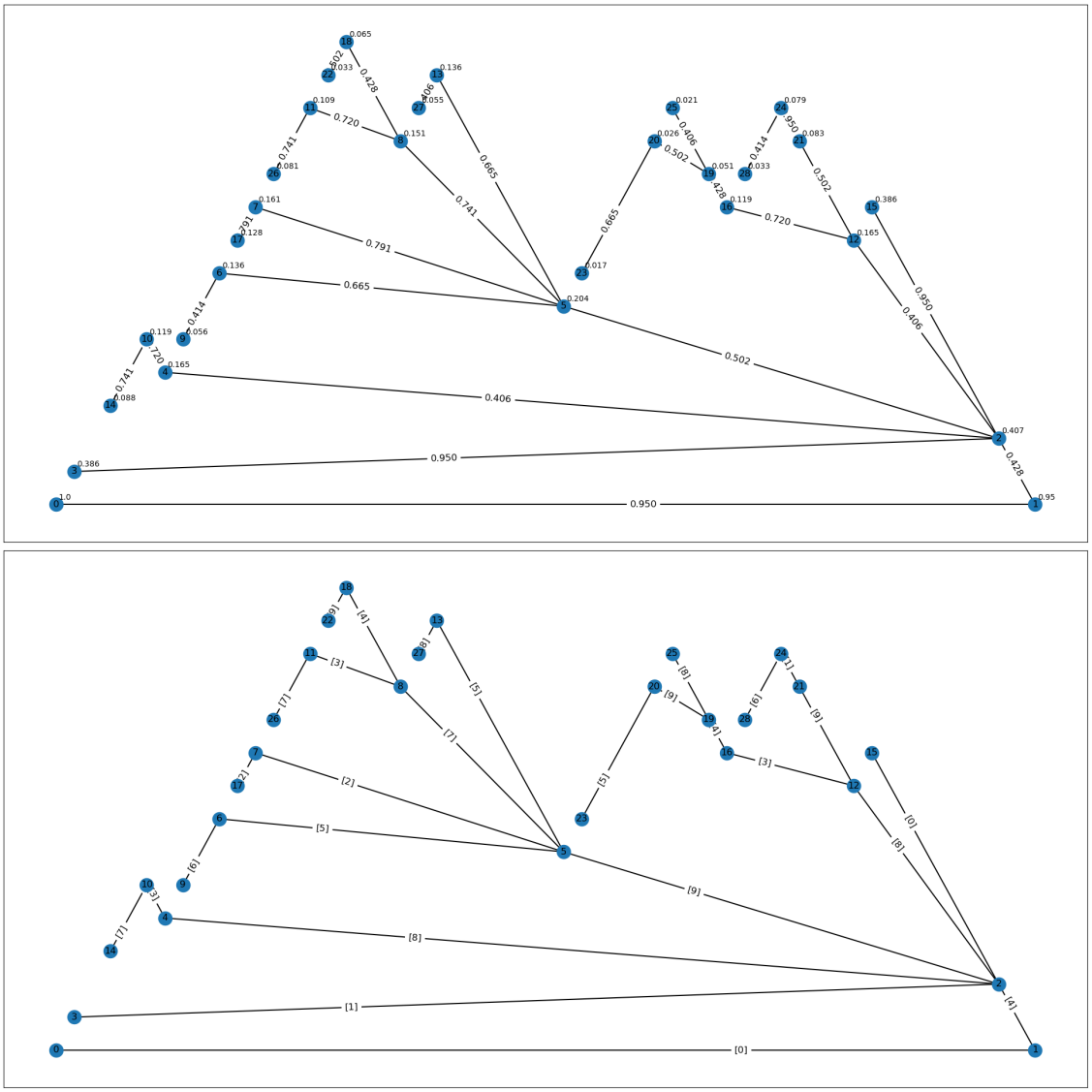

Figure 5.3: The ”best” network segmentation architecture output by PSO. Top figure shows the network architecture where the edge represents the conditional probability of compromising an enclave, and the number next to each node is the marginal probability of compromising the enclave. Bottom figure shows the services that are used between enclaves.

For our main experiment, we set the hyperparameters as follow: $lr = 0.5, w = 0.7, \phi_p = 1, \phi_g = 1, n = 50, T = 60$. This is because due to the simplicity of our model (the fitness score is strongly correlated to number of enclaves implemented, as seen in **Figure 5.2**), exploration of the search space can therefore be limited; we can quickly narrow down the search space.

We also observed that for a balanced weighting between security score and cost score (where $w_{sec} = w_{cost} = 0.5$), good quality network architectures situate around 25 enclaves. This is due to the steepness constant k has a value of 7, indicating that it does not cost too

much to implement more enclaves (refer to **Figure 3.5**).

5.2 Main Experiment

For our main experiment, we aim to investigate how optimum network segmentation can be implemented in three different scenarios: balance security and cost ($w_{sec} = w_{cost} = 0.5$), security focused ($w_{sec} = 0.8, w_{cost} = 0.2$), and cost focused ($w_{sec} = 0.2, w_{cost} = 0.8$). We focus our experiment on a network with a capacity of 30 number of enclaves ($M = 30$), and a cost function that linearly scales with number of enclaves ($k = 1$). Referring to **Figure 3.5**, the value $k = 1$ shows that the cost of implementing additional enclaves is high.

5.2.1 Parameters and Services Initialisation

As mentioned in the previous section, we set the hyperparameters as $lr = 0.5, w = 0.7, \phi_p = 1, \phi_g = 1, n = 50, T = 60$. We also model 10 services and manually assign each service its vulnerability to make sure that the services are the same across the three experiments. We initialise the services as shown in **Figure 5.4** below.

```
Service ID = 0 Vulnerability score = 0.95
Service ID = 1 Vulnerability score = 0.95
Service ID = 2 Vulnerability score = 0.502
Service ID = 3 Vulnerability score = 0.898
Service ID = 4 Vulnerability score = 0.113
Service ID = 5 Vulnerability score = 0.25
Service ID = 6 Vulnerability score = 0.362
Service ID = 7 Vulnerability score = 0.945
Service ID = 8 Vulnerability score = 0.442
Service ID = 9 Vulnerability score = 0.671
```

Figure 5.4: Instances of services used in experiment and the corresponding vulnerability score.

Similar to the scenario we describe in **Section 5.1**, we define the following traffic requirements:

1. Enclave 0 (internet) has to be connected to enclave 1 via service 0.
2. Enclave 2 has to be connected to enclave 3 via service 1.

The motive behind this is the same as **Section 5.1**, where we aim to model a trust relationship between enclave 2 and 3, and enclave 0 to enclave 1 can be seen as the connection between the internet and DMZ. We also try to avoid scenario where only a few enclaves are implemented with services that have the lowest vulnerability scores.

5.2.2 Results

We observed that in a balance scenario where the cost and security are equally weighted, the best architecture generated by the system is one that implements 4 enclaves (out of 30) with fitness score of – 0.1802. Interestingly, enclave 2 and enclave 3, where we manually specified as two enclaves connected via a two-way trust relationship, have very probabilities of getting compromised by an attacker, with 0.027 and 0.025 probabilities respectively. This indicates that although the trust relationship between enclave 2 and 3 can be dangerous, this is mitigated by putting multiple enclaves before them, which is equivalent to using defense in depth strategy, one of the techniques that implement network segmentation. An attacker would have to first overcome security measures (e.g. via exploitation of vulnerability) between enclave 0 and enclave 1, then exploit another weakness in connection between enclave 1 and enclave 4, and finally further compromise service running between enclave 2 and enclave 4 in order to reach enclave 2 (**Figure 5.5**).
On the other hand, when looking at the scenario where cost has higher weight than security ($w_{cost} = 0.8, w_{sec} = 0.2$), the result in **Figure 5.6** shows that the best architecture generated by the system is one that implements 4 enclaves. The result network architecture similar to the solution for balanced weighting scenario. The number of enclaves is near minimum number of enclaves (3). Again, we can see that the system generates network architecture that puts the two most vulnerable enclaves (2 and 3) away from the internet, separated by two services, service 4 and service 2. As seen from the figure, enclave 2 only has 0.051 probability of being compromised.

The result also supports the idea that it is possible to implement network segmentation even in small network. The fitness score for the network architecture in **Figure 5.6** is – 0.1245, which is higher than the balanced scenario. This is because more weight was put on cost, thus in order to maximise the fitness score, fewer number of enclaves are implemented, leading to lower cost score. Another interesting point is that by comparing **Figure 5.5** and **Figure 5.6**, we can see that although **Figure 5.5** is indeed more secure than **Figure 5.6**, the heavy weightings on cost mean that the solution in **Figure 5.6** has higher fitness score.
Finally, in the scenario where more importance is placed on the security than cost ($w_{sec} = 0.8, w_{cost} = 0.2$), we observed that the best segmentation architecture generated by the system implements more enclaves (9 enclaves) than the previous two scenarios (4), but not significantly more (maximum is 30). This leads to one important finding in this experiment: that strategic placements and proper implementation of network segmentation can effec-

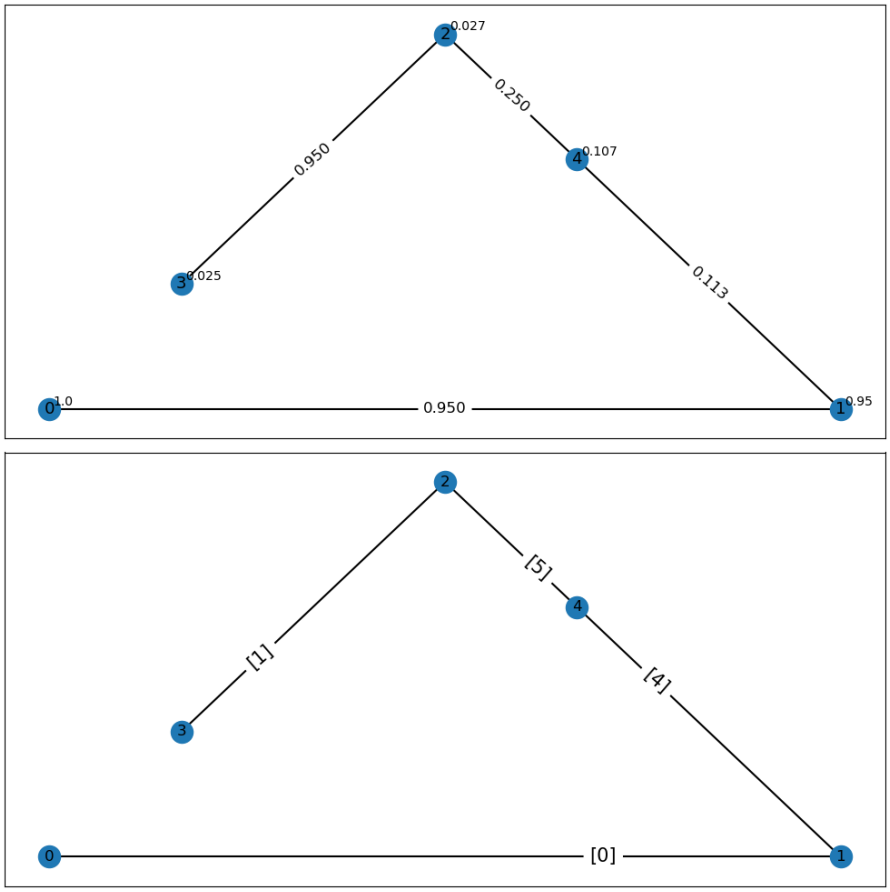

Figure 5.5: The best segmentation architecture when the cost weight and security weight are equal to 0.5.

tively enhance the security of the network without raising the cost too high.

The solution has a fitness score of – 0.1439. This is lower than the scenario where $w_{cost} = 0.8$. This is to be expected. In scenario where security is not a big concern ($w_{cost} = 0.8, w_{sec} = 0.2$), simply do not implement network segmentation and the cost is significantly reduced.

Note that in all three scenarios, they have two things in common: (1) all the best solutions have service 4 between enclave 1 and enclave 4, which is the most secure service. (2) enclave 2 and enclave 3 are always the placed far away from the internet, since they are the most vulnerable. These results suggest that PSO can generate good quality network segmentation architectures, and that the positioning of the security points is crucial to the overall security of the network.

5.3 Evaluation

There are a few important observations that are seen from our results. First, our model successfully captures the relationship between cost and security. Experiments have shown that network segmentation is an effective method in enhancing security of network and protecting

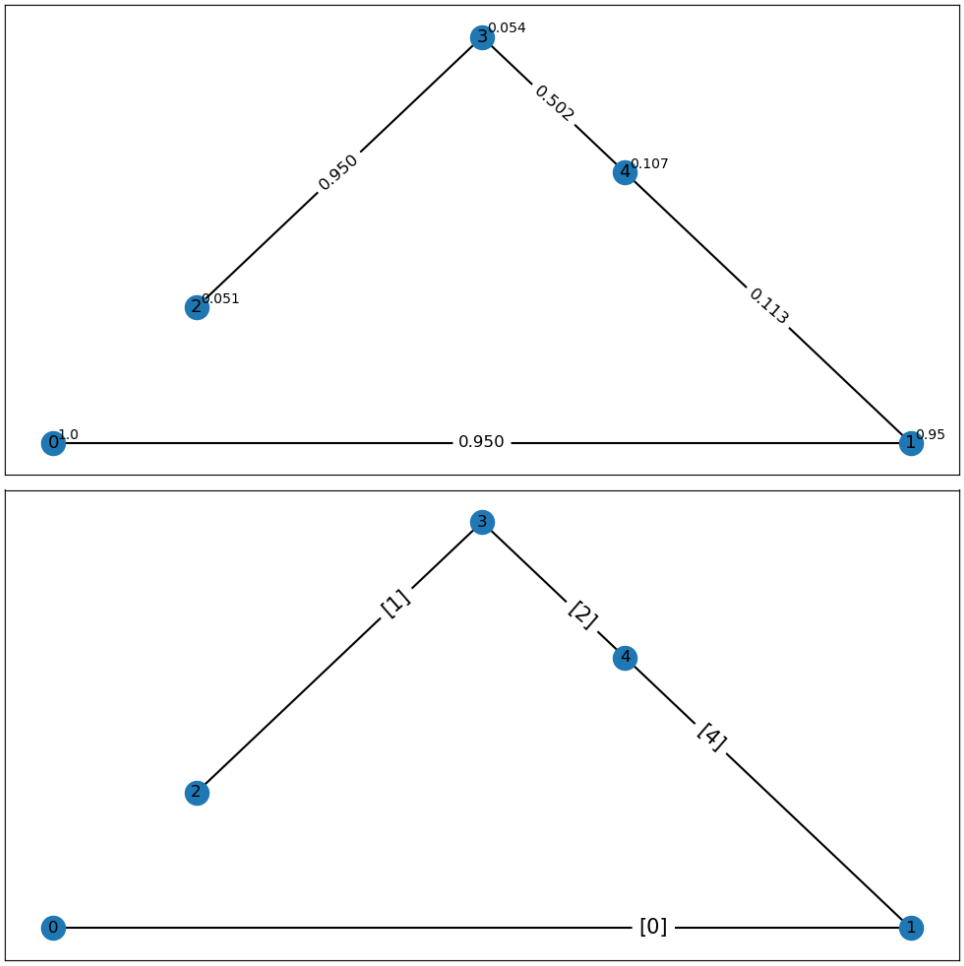

Figure 5.6: The best segmentation architecture when the cost weight is 0.8 and the security weight is 0.2.

vulnerable assets. We also observed that in an environment where implementing enclaves can be costly ($k = 1$), it is still possible to efficiently implement network segmentation, and it is also possible to apply network segmentation to small network.

We briefly summarise our findings from experiments as follow:

1. The most secure defense should be placed in ”front” of the network to protect enclaves that are vulnerable to attack. Network segmentation can be utilised to protect services with vulnerabilities yet to be patched (the idea of window of exposure is discussed in **Section 2.1.1**.

2. Adding more enclaves to the network does not automatically enhance the security of a network. Strategic placing of enclaves is considered more crucial in protecting against network attacks.

3. To bring out the full potential of PSO's ability to efficiently and intelligently explore the solution space, a intuitive, yet impactful fitness function is required; there is also a need to quickly find the best hyperparameters values.

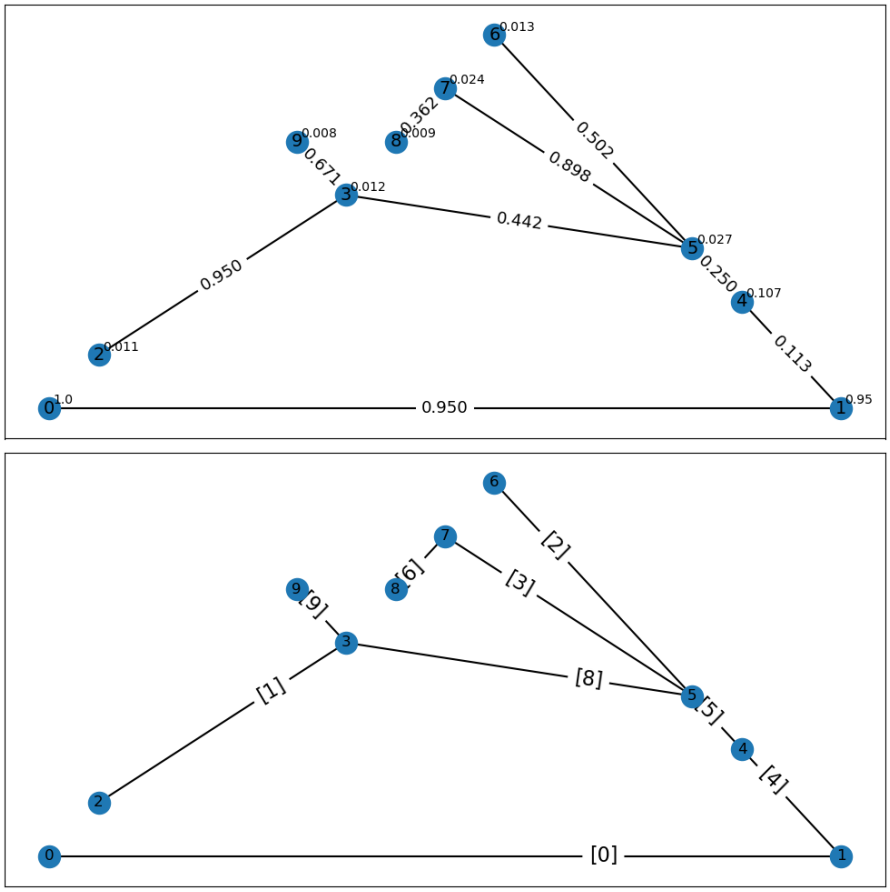

Figure 5.7: The best segmentation architecture when the cost weight is 0.2 and the security weight is 0.8.

5.3.1 Particle Swarm Optimisation

PSO explores the solution space with great efficiency. The ”best” architectures output by PSO that we observed show that by only guiding the particle with fitness function and influences from other particles, PSO output near optimal solution. Our results also show that PSO implements defense in depth strategy, where the algorithm prioritises the most secure enclave to face the internet whilst putting the most vulnerable one behind several security points.

5.3.2 Computational Complexity

We investigate how increasing the number of enclaves/ nodes affects computation time of the program for **1 iteration of 1 particle**. Our results are shown in **Table 5.2**, and our graph is plotted in **Figure 5.8**.

Number of nodes	Computation time/s	Number of nodes	Computation time/s
0	0	5000	76.4
1000	3.9	6000	110.25
1500	9.5	7878	237

2000	13	10000	372
3000	30.9		

Table 5.2: Shows the results we obtained from experimenting on how increase in number of nodes (enclaves) increases the computation time

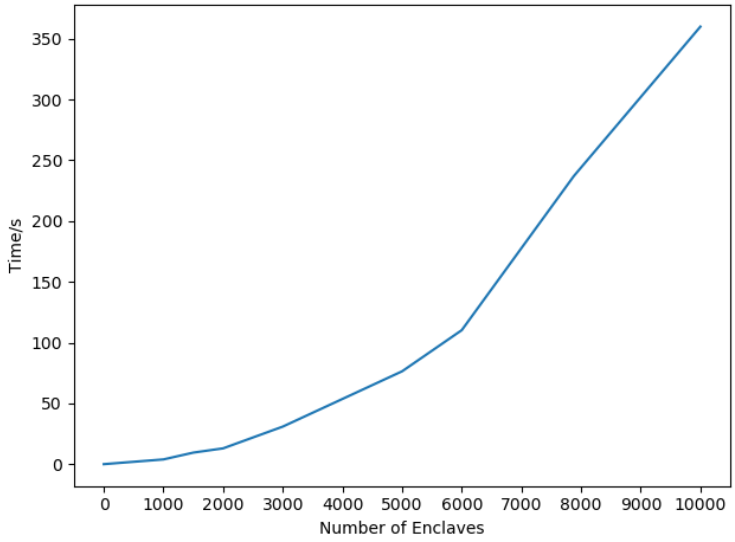

Figure 5.8: Computation time with respect to number of enclaves (nodes)

Here, we estimate the time complexity of the program. Taking the data we obtained for 1000 nodes (3.9 s) and 10000 (372 s), and let x be the exponent.

$$\begin{aligned} 3.9(\frac{10000}{1000})^{x} &= 372 \\ x &= 1.979 \\ x &\approx 2 \end{aligned} \tag{5.1}$$

Our calculation suggests that the worst-case complexity for evaluating one network architecture is $\mathcal{O}(M^2)$, where M is the maximum number of enclaves.

Particle Swarm Optimisation

For PSO, we have n number of particles. Each particle has to evaluate the network architecture. Additionally, for each iteration until T, the program requires evaluating all n particles. Therefore, the worst-case complexity for PSO is $\Theta(nT)$.

Overall Computational Complexity

Since each particle in PSO will evaluate the network, which means that each particle evaluation will have a computational complexity of $\mathcal{O}(M^2)$. Since this process is done $(n \cdot T)$ times, this gives us the overall worst-case computational complexity of our program: $\mathcal{O}(nTM^2)$.

5.3.3 Limitations

Our current model has the following limitations:

- **Limited scope**: our model currently only captures the trade-off between cost and security, and we also assume the segmentation architecture consists of a MRF with no loops. This means that our model currently does not capture attacks that can take multiple paths.
- **Insider threats**: we do not specifically model insider threats. It is however, possible to treat an insider attack as a path from the internet that skips multiple enclaves and into the organisation's network.
- **Complex metrics**: our current evaluation metrics only take into account of cost and security. We did not take into account of how a given network segmentation can affect the operations of the organisation. Currently, the most optimal solution to the network architecture is a network that is a chain (assuming no traffic requirements), where the internet enclave is the root node. This is because the probability of compromising subsequent enclave is lowered each time via repeated multiplication of probabilities, which minimises the security risk score (hence maximises fitness score).
- **Device level impact**: currently we do not model, or make assumptions at device level. Hence, it is not possible to quantify the effect of enclave compromise on devices within the enclave.
- **Redundancy**: for each iteration of PSO, since we randomly reconfigure the whole network architecture of each particle, it is likely that a particle would explore a segmentation architecture that has already been explored, especially for small networks, since the segmentation configurations are limited.
- **Scalability**: the current computational complexity of our model is $\mathcal{O}(nTM^2)$, meaning that it does not scale well for large number of nodes. However, this is also part of the reasons for the need to model at enclave level, where devices are grouped to reduce size of the networks, which ultimately reduces the computational complexity.

Chapter 6

Conclusion

"All models are wrong, but some are useful."

—- *George Edward Pelham Box*

This project provides a context on current cyber issues, and how network segmentation is one of the defensive measures that is constantly evolving to combat against rising cyber threats.

We present how Wagner et al. network model can be combined with Bayesian attack graph probabilistic approach to evaluate the risk of a given network architecture. The system we propose attempt to generate a network segmentation architecture that optimises security and cost.

The system has proven to perform particularly well as it manages to output architecture that is close to optimal solution. This shows the effectiveness of PSO as a population-based optimisation algorithm. Our model also confirms the validity and the importance of a well-known defensive technique – network segmentation.

Although numerous limitations exist in our current model, our experiments and results provide valuable information about the validity of our assumptions (assuming attacker originates from the internet), the effectiveness of the optimisation algorithm we used (PSO), the impacts of network segmentation (defense in depth), our evaluation metrics (fitness function), the feasibility of our risk assessment method (belief propagation) and among others.

Chapter 7

Future Work

7.1 Device Level Modelling

Future work could extend to model at device level, in which there are traffic requirements for some devices (for example, device 1 must be connected to device 2 via service 4). Then a grouping strategy can be utilised to group those devices into enclaves, and compute the required traffic requirements. Afterwards, an enclave-level simulation (e.g. an improved version of our model) can be applied alongside of PSO to generate optimised network architectures.

7.2 Additional Evaluation Metrics

As previously discussed in **Section 5.3.3**, our current model needs to abstract device level information in order to further quantify and observe the impacts of network segmentation on security, cost, mission impacts. This is equivalent to searching for a fitness function that better guides the particles in PSO towards optimal solution.

7.3 Automatic Hyperparameters Tuning

Our experiment has suggested that it can be unclear what values of hyperparameters (such as $\phi_p, \phi_g, w, lr,$ and T) are best for PSO. The answer to the question however, depends on the complexity and nature of the problem. Are the search space large or small? Complex or simple? Setting the values of ϕ_p, ϕ_g, w and lr too high for the problem, then the particles may "overfly" the optimal solution. Setting those values too low, then the particles may get stuck in one area and never approach anywhere near the optimal solution.

Future work can investigate automation of PSO hyperparameters tuning in the context of optimising network segmentation which can significantly improves the efficiency of experiments. It is also possible to experiment hyperparameters with damping ratio, where these

hyperparameters are set high at the beginning of the experiment to explore the search space as much as possible. Then slowly decay per iteration in order for the ”flock” to start concentrating to find certain maxima/minima.

7.4 Comparison to Other Metaheuristics

Several studies suggest that PSO outperforms SA in overall [61]. The results obtained in Soltani-Mohammadi et al. indicate that PSO approaches to optimised solution faster than SA. Future work can investigate which metaheuristics (or other optimisation algorithms) are more efficient in accurately evaluating the risk of a given network, suggesting recommended actions against cyber threats (such as how to segments the network, what services to use etc.).

7.5 Scalable Risk Evaluation

As we have empirically proven that our risk evaluation (particularly beleif propagation) has worst-case computational complexity $\mathcal{O}(M^2)$. Future work can utilise approximate inference techniques, such as loopy belief propagation [42], which can potentially reduce the worst-case computational complexity to $\mathcal{O}(M)$.

Bibliography

[1] Griffiths, R. 5G for Beginners. (2019) *Huawei BLOG*. Weblog. Available from: https://blog.huawei.com/2019/07/16/5g-for-beginners/ [Accessed 6th June 2020].

[2] Palmer, D. (2019) *Cybersecurity warning: 10 ways hackers are using automation to boost their attacks.* March 25 2020. Available from: https://www.zdnet.com/article/cybersecurity-warning-10-ways-hackers-are-using-automation-to-boost-their-attacks/ [Accessed 6th June 2020].

[3] Accenture and Ponemon Institute. (2019) *The Cost of Cybercrime: Ninth Annual Cost of Cybercrime Study*. Accenture.

[4] Skybox Security. (2019) *Vulnerability and Threat Trends*. Skybox Security.

[5] Common Vulnerabilities and Exposures, Browse Vulnerabilities By Date. (2019) *CVE Details*. Available from: https://www.cvedetails.com/browse-by-date.php [Accessed 19th August 2020].

[6] Edgescan. (2019) *Vulnerability Statistics Report*. Skybox.

[7] The Department for Education. (2020) *Coronavirus (COVID-19): cancellation of GCSEs, AS and A levels in 2020*. Available from: https://www.gov.uk/government/publications/coronavirus-covid-19-cancellation-of-gcses-as-and-a-levels-in-2020/coronavirus-covid-19-cancellation-of-gcses-as-and-a-levels-in-2020 [Accessed 6th June 2020].

[8] Gast, A. & Walmsley, I. (2020) *COVID-19 Update to students - 13 March 2020*. Available from: http://www.imperial.ac.uk/about/leadership-and-strategy/president/writing-and-speeches/community/covid-19-update-to-all-staff-and-students—15-march-2020/ [Accessed 6th June 2020].

[9] Batty, D. (21 March 2020) Oxbridge to replace summer exams with online assessments due to coronavirus. *The Guardian*. Available from: https://www.theguardian.com/education/2020/mar/21/oxbridge-to-replace-summer-exams-with-online-assessments-due-to-coronavirus [Accessed 6th June 2020].

[10] Warrington, J. (2020) *Zoom revenue surges as video conferencing app cashes in on Covid-19.* Available from: https://www.cityam.com/zoom-revenue-surges-as-video-conferencing-app-cashes-in-on-covid-19/ [Accessed 6th June 2020].

[11] Ikeda, S. (3 June 2020) Half a Million Zoom Accounts Compromised by Credential Stuffing, Sold on Dark Web. *CITYA.M.*. Available from: https://www.cpomagazine.com/cyber-security/half-a-million-zoom-accounts-compromised-by-credential-stuffing-sold-on-dark-web/ [Accessed 18th August 2020].

[12] Chester, D. (2020) *How and Why Power Grid Cyberattacks are Becoming Terrorists' Go-To.*. Weblog. Available from: https://energycentral.com/c/iu/how-and-why-power-grid-cyberattacks-are-becoming-terrorists-go [Accessed 19th August 2020].

[13] Zetter, K. (3 November 2014) An unprecedented Look at Stuxnet, the World's First Digital Weapon. *Wired.* Available from: https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/ [Accessed 20th August 2020].

[14] National Vulnerability Database. (2016) *National Institute of Standards and Technology*. Available from: https://nvd.nist.gov [Accessed 20th August 2020].

[15] Common Vulnerabilities and Exposures, *The MITRE Corporation*. Available from: https://cve.mitre.org/.

[16] Google. (2012) *Google's approach to IT security.* Google. Available from: https://static.googleusercontent.com/media/1.9.22.221/en//enterprise/pdf/whygoogle/google-common-security-whitepaper.pdf [Accessed 23th August 2020].

[17] Microsoft. (2015) *Enterprise Security Best Practices.* Avaialble from: https://docs.microsoft.com/en-us/previous-versions/tn-archive/dd277328(v=technet.10) [Accessed 22th September 2020].

[18] SANS Institute. (2013) *IAD's Top 10 Information Assurance Mitigation Strategies.* Available from: https://www.sans.org/security-resources/IAD_top_10_info_assurance_mitigations.pdf?msc=securityresourceslp [Accessed 22nd September 2020].

[19] Morrow, D. & Young, B. (2016) *ICS Network Segmentation SANS Institute*. Available from: https://www.sans.org/webcasts/ics-network-segmentation-101347 [Accessed 22th September 2020].

[20] Wikipedia. (2020) *Cyberattack*. Available from: https://en.wikipedia.org/wiki/Cyberattack [Accessed 19th August 2020].

[21] Cybersecurity and Infrastructure Security Agency. (2019) *Security Tip: Understanding Denial-of-Service Attacks.* Department of Homeland Security. Available from: https://us-cert.cisa.gov/ncas/tips/ST04-015 [Accessed 19th Aug 2019].

[22] Fauvet, L.(2013) *DDOS attack on the VudeoLAN downloads infrastructure.* Available from: https://www.youtube.com/watch?v=hNjdBSoIa8k [Accessed 23th August 2020].

[23] Logstalgia. (2010) *Website traffic visualisation tool.* Available from: https://logstalgia.io/ [Accessed 28th August 2020].

[24] Akamai Technologies. (2020) *Financial Services – Hostile Takeover Attempts. 2020 State of the Internet/ Security Report*. 6 (1), 1-28. Available from: https://www.akamai.com/uk/en/multimedia/documents/state-of-the-internet/soti-security-financial-services-hostile-takeover-attempts-report-2020.pdf [Accessed 20th August 2020].

[25] Stahie, S. (2020) *Gaming Became Industry Most Affected by DDoS Attacks in 2019.* Available from: https://securityboulevard.com/2020/02/gaming-became-industry-most-affected-by-ddos-attacks-in-2019/ [Accessed 20th August 2020].

[26] Sun, C. C., Hahn, A. & Liu, C. C. (2018) Cyber security of a power grid: State-of-the-art. *International Journal of Electrical Power & Energy Systems*. 99, 45-56. Available from: https://doi.org/10.1016/j.ijepes.2017.12.020.

[27] Leszczyna, R. (2019) *Cybersecurity in the Electricity Sector: Managing Critical Infrastructure.* Switzerland. Springer Nature Switzerland AG. Available from: https://doi.org/10.1007/978-3-030-19538-0 [Accessed 26th August 2020].

[28] Lord, N. (2018) *What is an Insider Threat? An Insider Threat Definition.* Available from: https://digitalguardian.com/blog/what-insider-threat-insider-threat-definition [Accessed 26th August 2020].

[29] Henry, J. (2018) *These 5 Types of Insider Threats Could Lead to Costly Data Breaches.* Available from: https://securityintelligence.com/these-5-types-of-insider-threats-could-lead-to-costly-data-breaches/ [Accessed 26th August 2020].

[30] Reuters. (2006) *U.S. Says Personal Data on Millions of Veterans Stolen.* Available from: https://www.washingtonpost.com/wp-dyn/content/article/2006/05/22/AR2006052200690.html [Accessed 26th August 2020].

[31] Popken, B. (1 May 2018) Facebook fires engineer who allegedly used access to stalk women. *NBC News*. Available from: https://www.nbcnews.com/tech/social-media/facebook-investigating-claim-engineer-used-access-stalk-women-n870526 [Accessed 26th August 2020].

[32] Brook, C. (Jan 2020) Tesla Data Theft Case Illustrates the Danger of the Insider Threat. *Digital Guardian's Blog*. Available from: https://digitalguardian.com/blog/tesla-data-theft-case-illustrates-danger-insider-threat [Accessed 26th August 2020].

[33] Homoloiak, I., Toffalini, F., Guarnizo, J., Elovici, Y. & Ochoa, M. (2019) Insight into Insiders and IT: A Survey of Insider Threat Taxonomies, Analysis, Modeling, and Countermeasures. *Association for Computing Machinery*. 52 (2). Available from: https://dl.acm.org/doi/10.1145/3303771.

[34] Symantec. (2011) *Advanced Persistent Threats: A Symantec Perspective.* Symantec. Available from: https://web.archive.org/web/20180508161501/https://www.symantec.com/content/en/us/enterprise/white_papers/badvanced_persistent_threats_WP_21215957.en-us.pdf.

[35] Chen, J., Su, C., Yeh, K. H. & Yung, M. (2018) Special Issue on Advanced Persistent Threat. *Future Generation Computer Systems*. 79 (1), 243-246. Available from: https://doi.org/10.1016/j.future.2017.11.005.

[36] Mimran, M. (2017) *The Long-term Threats Posed by the Vault 7 Leaks.* Weblog. Available from: https://www.cybereason.com/blog/vault-7-leaks-long-term-threats [Accessed 28th August 2020].

[37] Imperva. (n.d.) *Advanced Persistent Threat (APT).* Available from: https://www.imperva.com/learn/application-security/apt-advanced-persistent-threat/ [Accessed on 28th August 2020].

[38] Imperva. (n.d.) *DDoS Attacks.* Available from: https://www.imperva.com/learn/ddos/ddos-attacks/ [Accessed on 29th September 2020].

[39] McAfee Labs and McAfee Foundstone Professional Services. (2010) *Protecting Your Critical Assets: Lessons Learned from "Operation Aurora".* McAfee. Available from: https://www.wired.com/images_blogs/threatlevel/2010/03/operationaurora_wp_0310_fnl.pdf [Accessed 28th August 2020].

[40] Cloudflare. (n.d.) *Zero Trust Security — What's a Zero Trust Network?.* Available from: https://www.cloudflare.com/learning/security/glossary/what-is-zero-trust/ [Accessed on 22nd September 2020].

[41] Khedri, R., Mhaskar, N. & Alabbad, M. (2019) *On the Segmentation of Networks*. Department of Computing and Software, McMaster University.

[42] Muñoz-González, L., Sgandurra, D., Paudice, A. & Lupu, C. E. (2017) Efficient Attack Graph Analysis through Approximate Inference. *ACM Transactions on Privacy and Security*. 20 (3). Available from: https://doi.org/10.1145/3105760.

[43] Jha, S., Sheyner, O. & Wing, J. (2002) Two formal analyses of attack graphs. In *Proceedings ofthe Workshop on Computer Security Foundations*. 49-63. Available from: https://doi.org/10.1109/CSFW.2002.1021806.

[44] Liu, Y. & Man, H. (2005) Network vulnerability assessment using Bayesian networks. In *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security*. 61-71. Available from: https://doi.org/10.1117/12.604240.

[45] Wang, L., Islam, T., Long, T., Singhai, A. & Jajodia, S. (2008) An Attack Graph-Based Probabilistic Security Metric. In *Atluri V. (eds) Data and Applications Security XXII*. 5094. Available from: https://doi.org/10.1007/978-3-540-70567-3_22.

[46] Bishop, C. M. (2006) *Pattern Recognition and Machine Learning (Information Science and Statistics)*. New York, Springer-Verlag New York.

[47] Wagner, N., Sahin, C. S., Winterrose, M., Riordan, J., Pena, J., Hanson, D. & Streilein, W. W. (2016) Towards Automated Cyber Decision Support: A Case Study on Network Segmentation for Security. *2016 IEEE Symposium Series on Computational Intelligence(SSCI)*. 1-10. Available from: https://doi.org/10.1109/SSCI.2016.7849908.

[48] Wagner, N., Sahin, C. S., Pena, J., Riordan, J. & Neumayer, S. (2017) Capturing the security effects of network segmentation via a continuous-time Markov chain model. In *Proceedings of the 50th Annual Simulation Symposium*. 1-12. Available from: https://dl.acm.org/doi/10.5555/3106388.3106405.

[49] Wagner, N., Sahin, C. S., Pena, J. & Streilein, W. W. (2017) A nature-inspired decision system for secure cyber network architecture. *2017 IEEE Symposium Series on Computational Intelligence(SSCI)*. 1-8. Available from: https://doi.org/10.1109/SSCI.2017.8285297.

[50] Wagner, N., Sahin, C. S., Pena, J. & Streilein, W. W. (2019) Automatic Generation of Cyber Architectures Optimized for Security, Cost, and Mission Performance: A Nature-Inspired Approach. In: Shandilya S., Shandilya S., Nagar A. (eds) *Advances in Nature-Inspired Computing and Applications.* EAI/Springer Innovations in Communication and Computing. Springer, Cham. Available from: https://doi.org/10.1007/978-3-319-96451-5_1.

[51] Hemberg, E., Zipkin, J. R., Skowyra, R. W., Wagner, N. & O'Reilly, U. M. (2018) Adversarial co-evolution of attack and defense in a segmented computer network environment. *2018 Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1648-1655. Available from: https://doi.org/10.1145/3205651.3208287.

[52] Damodaran, S. & Wagner, N. (2020) Modeling and simulation to support cyber defense. *The Journal of Defense Modeling and Simulation*. 17 (1), 3-4. Available from: https://journals.sagepub.com/doi/10.1177/1548512919856543.

[53] Winterrose, M. L., Carter, K. M., Wagner, N. & Streilein, W. W. (2016) Balancing security and performance for agility in dynamic threat environments. *2016 46th annaual IEEE/I-*

FIP international conference on dependable systems and networks (DSN). 607-617. Available from: https://doi.org/10.1109/DSN.2016.61.

[54] The Department of Homeland Security (DHS)'s National Cybersecurity. (2016) *Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies.* Available from: https://us-cert.cisa.gov/sites/default/files/recommended_practices/NCCIC_ICS-CERT_Defense_in_Depth_2016_S508C.pdf [Accessed on 22nd September 2020].

[55] Knezevic, K., Stjepan, P., Domagoj, J. & Hernandex, C. J. (2020) What is Your MOVE: Modeling Adversarial Network Environments. In: Castillo P., Jiménez Laredo J., Fernández de Vega F. (eds) *Applications of Evolutionary Computation. EvoApplications 2020.* Lecture Notes in Computer Science, vol 12104. Springer, Cham. pp. 260-275. Available from: https://link.springer.com/chapter/10.1007%2F978-3-030-43722-0_17.

[56] Riordan, J. F., Lippmann, R. P., Neumayer, S. J. & Wagner, N. (2016) *A Model of Network Porosity.* Massachusetts Institute of Technology, Lincoln Laboratory. Report number: IA-4.

[57] Daley, D. J. & Gani, J. (1999) *Epidemic Modelling: An Introduction.* Cambridge: Cambridge University Press. Available from: https://www.cambridge.org/core/books/epidemic-modelling/6F7376322E00A98D6801B97D9429A0CF.

[58] Yu, S., Gu, G., Barnawi, A., Guo, S. & Stojmenovic, I. (2015) Malware propagation in large-scale networks. *IEEE Transaction on Knowledge and Data Engineering*. 27 (1), 170-179. Available from: https://doi.org/10.1109/TKDE.2014.2320725.

[59] Wikipedia. (2020) *Markov Random Field.* Available from: https://en.wikipedia.org/wiki/Markov_random_field [Accessed on 28th September 2020].

[60] CVE Details. (2016) *Current CVSS Score Distribution For All Vulnerabilities.* Available from: https://www.cvedetails.com/ [Accessed on 20th September 2020].

[61] Soltani-Mohammadi, S., Safa, M. & Mokhtari, H. (2016) Comparison of particle swarm optimization and simulated annealing for locating additional boreholes considering combined variance minimization. *Computers & Geosciences*. 2016, 146-155. Available from: https://doi.org/10.1016/j.cageo.2016.07.020.

[62] Sahab, M. G., Toropov, V. V. & Gandomi, A. H. (2013) A Review on Traditional and Modern Structural Optimization. In: Gandomi, A. H., Yang, X. Y., Talatahari, S. & Alavi A. H. (eds) *Metaheuristic Applications in Structures and Infrastructures*. Elsevier. pp. 25–47. Available from: https://www.sciencedirect.com/science/article/pii/B9780123983640000024.

[63] Kennedy, J. & Eberhart, R. (1995) Particle Swarm Optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*. 4, 1942-1948. Available from: https://doi.org/10.1109/ICNN.1995.488968.

[64] Gogna, A. & Tayal, A. (2013) Metaheuristics: Review and application. *In Journal of Experimental and Theoretical Artificial Intelligence*. 25 (4), 503-526. Available from: https://doi.org/10.1080/0952813X.2013.782347.

[65] Hunter, J. D. (2007) Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*. 9 (3), 90-95. Available from: https://doi.org/10.1109/MCSE.2007.55.

[66] Rivard, K. (2011) *Belief Propagation.* Carnegie Mellon University. Available from: http://curtis.ml.cmu.edu/w/courses/index.php/Belief_Propagation [Accessed 30th September 2020].

Appendix A

Legal and Ethical Considerations

The ”network architecture” that is generated from the program is purely abstract, thus does not reflect nor contain any confidential information about real organisations’ networks.

Copyright and Licensing Implications

As this project uses several libraries to perform the computations. It is therefore crucial that these libraries are properly documented and referenced.

NetworkX is a Python package for the study of the dynamics, structure and functions of complex networks. NetworkX is released under 3-Clause BSD License (**https://github.com/networkx/networkx)**.

Matplotlib is a Python package for plotting. Matplotlib is distributed under Matplotlib License, which is based on the Python Software Foundation (PSF) license (**https://matplotlib.org**/).

NumPy is a Python package that supports multi-dimensional arrays and matrices, and has many mathematical functions to assist numerical computations. NumPy is distributed under BSD-3-Clause License (**https://numpy.org**/).

krashkov/Belief-Propagation is a GitHub repository that implements sum-product algorithm (Belief Propagation) and also Loopy Belief in Python. krashov/Belief-Propagation is licensed under GNU General Public License v3.0 (**https://github.com/krashkov/Belief-Propagation**).

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
Section 2: HUMANS		
Does your project involve human participants?		✓
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		✓
Section 5: ANIMALS		
Does your project involve animals?		✓
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		✓
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals taking part in the project at risk?		✓
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		✓
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans, including project staff?		✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		✓
Section 8: DUAL USE		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?		✓
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		✓
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		✓
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		✓
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		✓
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		✓
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		✓
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?	✓	
Will your project use or produce goods or information for which there are data protection, or other legal implications?		✓
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		✓

Figure A.1: Legal and ethical considerations checklist

Appendix B

User Guide

For reproducibility of the experiment:

1. Download the code repository.
2. Install the required Python libraries (networkx == 2.5, numpy == 1.19.2 , matplotlib == 3.3.2 , scipy == 1.5.2 , pyvis == 0.1.8.2 , python-igraph == 0.8.2) .
3. Go to line 81 in main.py where you can tweak the value of w_{sec} (denote as *alpha* in the program and w_{cost} (denote as *beta*) and also k for cost function. The other hyperparameters are set:
   ($lr = 0.5, w = 0.7, \phi_p = 1, \phi_g = 1, n = 50, T = 60$)
4. Run main.py.
5. Change *alpha* and *beta*, and repeat **Step 4**.

Options:

1. To customise hyperparameters, go to line 85 and 86 in main.py.
2. To set traffic requirements, go to line 61 in main.py.
   (2,3,1) means enclave 2 has to be connected to enclave 3 via service 1.
   (Currently the model will raise an error with loops)
3. To change the number of services, go to line 20 in main.py. To customise the vulnerability score of a particular service, have a look between line 49 to line 58 in main,py.

Appendix C

Code Snippet

```python
def assign_prob_of_exploit():
    """
    Assign a random CVSS score from CVSS distribtuion
    Return a probability that an enclave is compromised
    (Conditionally depends on the enclave before being compromised)
    """

    population = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    prob = [0.0060, 0.0070, 0.0400, 0.0370, 0.2220,
            0.1930, 0.1380, 0.2220, 0.0040, 0.1310]
    score = random.choices(population, prob)[0]
    if score == 10:
        score -= 0.05

    # uniform distribution U~(score-1,score)
    score = np.random.uniform(score - 1, score)
    return score / 10
```

Figure C.1: Python code snippet which assigns CVSS score randomly according to CVSS score distribution in [60].

```python
def compute_min_enclave(traffic_requirements):
    """
    :param traffic_requirements:
    :return: minimum number of enclaves require
             for the network model
    """
    if not traffic_requirements:
        return 1
    required_enclaves = []
    for u, v, s in traffic_requirements:
        required_enclaves.append(u)
        required_enclaves.append(v)
    required_enclaves = set(required_enclaves)
    min_enclaves = len(required_enclaves)
    if max(required_enclaves) >= min_enclaves:
        min_enclaves = max(required_enclaves)
    if 0 in set(required_enclaves):
        min_enclaves -= 1
    return min_enclaves
```

Figure C.2: Calculating minimum number of enclaves require for the network model, given traffic requirements F.