Imperial College
London

MSC INDIVIDUAL PROJECT

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Dynamic Network Segmentation for Cyberdefense

Author:
Sohaïb Ouzineb

Date: May 30, 2021

Contents

1 Introduction **3**
- 1.1 Motivations . . . 3
- 1.2 Objectives . . . 4
- 1.3 Research Issues . . . 4

2 Related Work **5**
- 2.1 Modelling . . . 5
  - 2.1.1 Enclave model . . . 5
  - 2.1.2 Network model . . . 8
  - 2.1.3 Loss function . . . 9
- 2.2 Optimisation algorithms . . . 10
- 2.3 Conclusion . . . 11

3 Experimentation **12**
- 3.1 First modelling . . . 12
  - 3.1.1 Network Model . . . 12
  - 3.1.2 Enclave Model . . . 13
  - 3.1.3 Attacker Model . . . 14
  - 3.1.4 Simulation and Optimization Frameworks . . . 14
  - 3.1.5 Experiment and Results . . . 16
  - 3.1.6 Conclusion . . . 17
- 3.2 Second modelling . . . 17
  - 3.2.1 Penalty term . . . 17
  - 3.2.2 Experiment and Results . . . 18
  - 3.2.3 Conclusion . . . 18

4 Plan for future work **19**

Bibliography **20**

A Appendix **21**

Chapter 1

Introduction

1.1 Motivations

Every company needs networks to store and exchange data between the companies' machines and the internet. Recent devastating cyber attacks show the urgent need for companies to strengthen the security of their networks. By secure networks, we mean that each layer of the network has to be secure. Even if software security is essential, the topology of the network also is.

Indeed, the company may host critical infrastructure that should not be compromised. If compromised, this incident might incur mission delay for the case of a web server, or it might incur a data breach for the case of database server. We might want, for performance issues, to put these machines close to the Internet. However, if these valuable machines face the Internet directly, they are more exposed to being compromised than if they were far from the internet with barriers in between. This alone shows the importance of network topology and the need to optimise it in terms of security and performance.

This performance/security trade-off is necessary to address but not sufficient. This alone might give us architectures having unrealistic characteristics like certain routers being connected to more ports that they can be connected to. The issues of constraints and business requirements also need to be taken into consideration. We may need for example certain enclaves (sub networks) to be connected directly. We may also have certain enclaves that do not have the capability of routing other enclaves' traffic : a printers enclave may not be able to route traffic from a web servers enclave.

The main problem that thus needs to be addressed is the optimization of a network architecture in terms of security and performance while respecting certain conditions.

To solve this problem, we will use a strategy that has been widely proposed (and that is mainly used in practice) to solve this problem, which is called network segmentation. It refers to the partitioning of devices into distinct enclaves that can be thought as sub-networks between which communications are restricted.

The main motivation behind this strategy is that the attacker may need to go through many barriers to get to his objective, which he might not be able to do so if he doesn't have the necessary strength or time budget. This strategy effectively reduces the attack surface and can be used to protect critical infrastructure. This strategy also restricts intelligence gathering, as machines in other enclaves are less visible from a network perspective and the attacker may hence not know the path that he has to take to get to his objective. Moreover, recent technologies such as micro-services make it easy to adapt a segmentation.

Beside the main motivations of security, performance and business requirements, the second main motivation of our work is the lack of realistic solutions in the state of the art.

It is indeed a difficult task to give the best network segmentation from a security and business point of view. We know general guidelines like enforcing the rules of least privileged, multi-layer control or grouping similar systems. However these are just principles and don't propose a best architecture, rather they propose candidate architectures that need to be tried out individually at a high computational and time cost.

Researchers have tried out different approaches to solve this problem, but to the best of our knowledge, none seems to cover all the mentioned objectives : security, performance and business requirements. In fact, none got into the issue of business requirements : proposed architectures only minimised the risk. However, proposed solutions are unrealistic and thus inoperable.

1.2 Objectives

The main objective of our thesis is to propose a framework that gives the best network architecture in terms of performance and security while respecting certain conditions.

This objective can be decomposed into more specific objectives. The first of which is to propose a general, detailed and realistic model and a corresponding efficient simulation algorithm implementation that will be able to model an attacker infiltrating a network having some defensive procedures.

The second main objective is to propose and implement an optimization algorithm working in the space of constrained network segmentations that gives the best architecture with respect to a certain loss function. This algorithm needs to be efficient and to propose a global optimum rather than to be stuck on a local optimum.

1.3 Research Issues

Some researchers attempted to solve the problem of network segmentation and proposed testing and optimisation frameworks to get an architecture that minimises a defined loss function representing the risk like in (1). However, the different proposed models (network model, enclave model,..) that define how an attacker spreads in a network and how a defender prevents the attacker from spreading are too simple and not representative of real networks behaviors.

For example, in (2), (1) or (3), researchers consider that the attacker will spread following an epidemic model at rate β in each enclave and compromise each infiltrated machine. This model is suited for untargeted attacks but fails to consider targeted attacks that are more prominent. These attacks consist in targeting specific critical infrastructure systems in big companies or government networks and often make the headlines. It is also very dangerous to assume that specifically the attacker's strategy when designing a network, because when one day an attacker follows a different strategy, the network topology will not be suited for it.

The most important issues in past works are associated with the proposed optimisation procedures: they suffer from exploitation. This issue has been specifically pointed out in (3). Exploitation is when an optimisation algorithm will keep on trying new solutions that are in a neighborhood of the currently best solution, instead of exploring the whole search space. This means that the proposed solution can be just a local optimum and not a global optimum.

We first address the modelling issue in our thesis by implementing a time based probabilistic simulation algorithm running on a more general model that includes several realistic behaviors and characteristics compared to the state of the art such as: more sophisticated incident response procedures, reconnaissance, regular updates, devices having different values, attacker compromise and data appetite, and more.

To this end, we also propose a new loss function that takes into account an attack term measuring the damage done on the network. The damage has been measured in past papers in terms of mission delay due to the compromise of machines (a mission being on of the main goals of a network like running a web application). We extend this modelling by including the damage in terms of data breach due to the infection of database servers hosting classified information. We model the stealing of data as it is one of the main attacker's goals in targeted attacks. We also take into account a penalty term that will measure the latency incurred by the network segmentation as a function of the distances of mission devices from the internet.

We then deal with the optimization issue by proposing a novel approach which is to use the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) algorithm as an optimisation algorithm. By illuminating a search space whose dimensions are custom features that represent different types of solutions (for example the number of connections or the standard deviation of the valuable machines distribution), we can mitigate the exploitation issue and thus propose high performing and diverse solutions.

The business requirements are incorporated into our optimisation procedure to constraint our search space. To that end, we propose a novel mutation operator that will, in a constrained search space, mutate a graph into another one with a probability conversely proportional to the graph edit distance between the two graphs.

Chapter 2

Related Work

In this section, we present related work with regards to modelling and optimisation algorithms. In the modelling section, we discuss the different enclave models starting with the machine models, then some known enclave models and their attacker and defender strategies : how does an attack spread in an enclave and how does an enclave defends itself from it. Then we discuss similarly the different network models and their attack and defense models. To conclude this modelling part, we will see the goals and different kinds of loss functions. In the optimisation section, we will present and analyse the main optimisation procedures proposed in the literature.

2.1 Modelling

Many different network modellings have been proposed in the literature. Researchers tend to distinguish between two types of modellings : network modelling and enclave modelling. Enclave modelling models the attacker's spread in the enclave or across enclaves and the enclave response to it. Network modelling operates at a higher level: it models how the attacker spreads in the network. For an attacker to spread in the network, he has to spread through enclaves : this is the purpose of attacker propagation models that we will discuss at different layers of the network.

2.1.1 Enclave model

An enclave is a set of machines that operate in a sub-network of the main network with common security policies. Let's see more specifically how they have been modelled in the literature.

Machine model In the literature (see (4) or (3)), the enclave is modeled as a set of machines having homogeneous reachability. Which means that each device can open a TCP/UDP connection to any other machine in the enclave and to on any port on the machine. This modelling is reasonable because machines in the same sub-network tend to trust themselves, port blocking is usually the job of enclave firewalls.

Enterprise networks have missions (tasks) like running web applications. From this notion of mission, researchers have distinguished between two types of devices : mission devices and non mission devices. (See (2), (1), (3)). By mission devices, we mean that they are necessary for the main mission to function. When compromised, mission devices incur mission delay that is reflected on the total loss.

However, these compromise incidents are modelled as having the same impact value on the total loss, which is not realistic. Indeed, a shut down web server should have much more impact on the total loss than a shut down employee computer.

Moreover, this model is satisfying when we only want to protect ourselves from availability attacks. However, they are not pertinent when considering a larger class of attacks, including data breaches. This is due to the simple distinction between mission and non mission devices. There are in reality non mission devices, like backup servers hosting classified information, that can be infiltrated and have their data stolen. Such devices are not modelled and can thus not reflect the impact of data breaches.

We address this issue by mapping each device in the network to a tuple of values : the compromise and information value. The compromise value is high if the machine being turned down incur a high mission delay.

Similarly, an information value is high if the exfiltrated data of the infected machine is of high classification level.

Markov chain model In (4), the authors use Markov chains to model their enclaves and their corresponding services.

Services Services states (number of vulnerabilities and exploit developed) are modeled as a Markov chain (see Figure 2.1 cited from (4)) via multiple transition probabilities based on real world data. Δ_{vuln}^{-1} is the probability of a service getting an additional vulnerability, it is computed as the average vulnerability arrival rate of the most common services in (5). Δ_{patch}^{-1} is the probability of the service vulnerabilitie(s) being patched, it is derived from the patch arrival times after vulnerability disclosure of some 15K past vulnerabilities given in (6) . Δ_{dev}^{-1} is the probability of developing an exploit based on one vulnerability, it is again derived in a similar way from known exploit availability dates given in (6). Finally, $\Delta_{exploit}^{-1}$ is the probability of the exploit being used in real life, this parameter varies according to the importance of the network : an attacker may wait more before attacking a high value network.

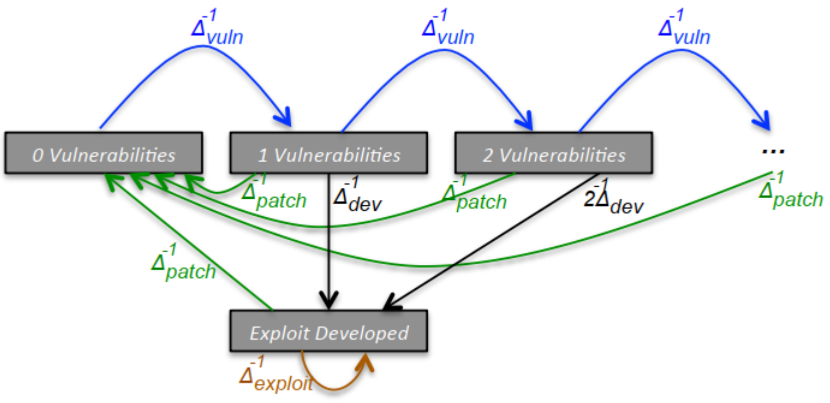

Figure 2.1: Markov chain for services, cited from (4)

However, the core problem of this modelling is that, even if detailed and based on real data, it non explicitly implies that these probabilities are the same for all services, which is usually not the case. In fact, the probability of getting an additional vulnerability may be higher for certain common services compared to others.

Enclave The two possible enclave states, compromised and clean, are modeled by a Markov chain (See Figure 2.2 cited from (4)). When compromised, an enclave can become clean with a probability Δ_{clean}^{-1}, this parameter varies from once every month to once a year. When clean, an enclave can become compromised with probability Δ_{comp}^{-1}. The latter is the sum of compromise probabilities of each service running on this enclave.

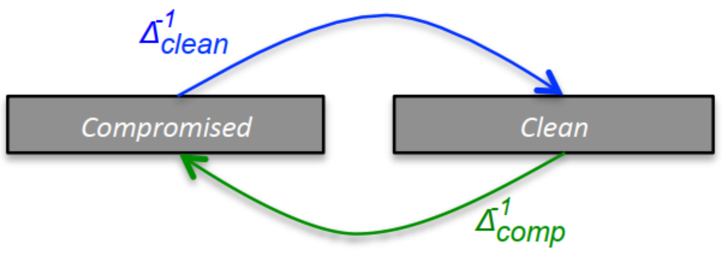

Figure 2.2: Markov chain for enclaves, cited from (4)

This enclave model is too simple because it assumes that when compromised, enclaves will be cleansed with the same probability. However, this is not realistic as an enclave with more compromised devices should have a higher chance of being cleansed.

Incident response The incident response approach defined above is not specific to (4). In fact, in (1), (2) and (7), the main defensive approach when detecting device compromise is to cleanse the whole enclave with a certain probability. While being effective to deter the attacker from further spreading, it has a potentially high impact on the total loss. If you cleanse a whole web server enclave for just one infiltrated guest machine, you will get a high mission delay.

We address this first issue by proposing a more proportional incident response procedure. First of all, we introduce the notion of device level IDS. Each device has an IDS that will prevent (with a certain accuracy) attacker lateral movement through the enclave. Secondly, if machines IDS detect too many infiltration attempts, the enclave will be cleansed. Then with regards to machines being shut down, if there are too many of them, then, with a higher probability, the enclave will be cleansed (this is the partial mitigation that was implemented in (3) by using the notion of sensitivity). Else, this shut down incident will be detected, investigated, and solved under a certain time dependent on the shut down machine.

Hierarchical modelling The second issue with the enclave Markov chain modelling is that it is a high-level model that ignores the infected machines, their numbers and the types of infections.

These issues have been partially addressed in (2) and (1) by using the Markov chain model not as the only enclave model, but in a hierarchical modelling (see Figure 2.3 cited from (1)). By hierarchical, we mean that in order to run an attacker/defender simulation, we need to do in sequence different simulations operating at different levels of modelling, with the output of the previous simulation being used as a parameter for the next simulation. The Markov chain is used in a testbed environment to measure the vulnerability probability of each enclave. Then, with these probabilities in mind, another enclave model simulates a blind epidemic spread at rate β inside the enclaves to get the probabilities of device compromise, the mean and standard deviation of their compromise time. From these, a network model runs to measure a unified metric and feed it to the optimisation algorithm that will propose a new segmentation. Then the loop restarts. It is a very computationally effective process as it runs simplified high-level models on top of each other and gather their result to get a global loss. The downside is its assumption that the intra-enclave attacker spread will not impact the high level enclave model, which is not true in reality : when too many devices are compromised, you should have a higher chance of triggering enclave cleansing and not just a constant Δ_{clean}^{-1} enclave cleansing probability.

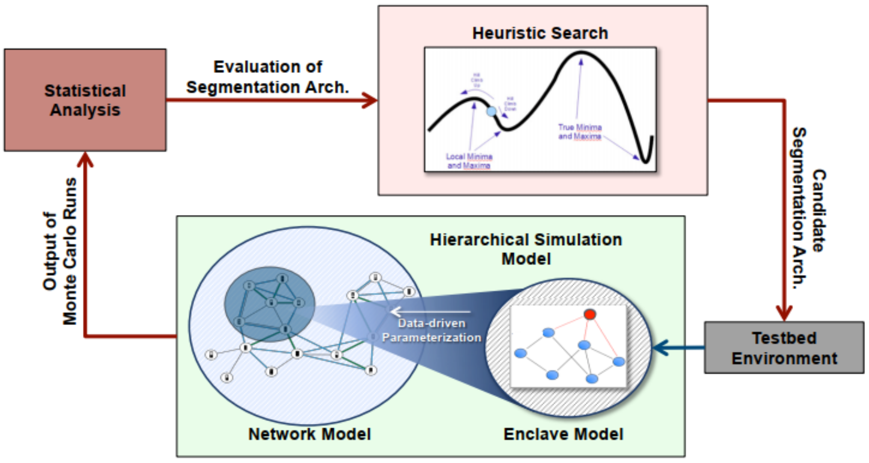
Statistical Analysis
Evaluation of Segmentation Arch.
Heuristic Search
Local Minima and Maxima
True Minima and Maxima
Candidate Segmentation Arch.
Testbed Environment
Hierarchical Simulation Model
Data-driven Parameterization
Network Model
Enclave Model
Output of Monte Carlo Runs

Figure 2.3: Hierarchical model, cited from (1)

Attacker model in enclaves When it comes to the attacker model, the issue with the epidemic spread (that (3) will also use) is that it assumes that the attacker strategy is to compromise as many machines as possible. This is a correct strategy when considering untargeted attacks. However, targeted attacks, that are more common, are not modelled. This is a very dangerous hypothesis as an attacker has much more choice than just compromising systems. First of all, he could just use infected machines as stepping stones without inflicting any damage to the running mission services. He could also steal data on an infiltrated database server without shutting it down. The latter incident is representative of a data breach, which has not been mentioned in cited papers, despite the fact that it is the main goal of targeted attacks.

We address this issue by first modelling the attacker information appetite and the attacker compromise appetite. It means that an attacker will be more attracted by targets with high compromise values if the attacker has more compromise appetite and vis-versa. These appetites are thus used to represent mainly targeted attacks. However, they can also be used to represent untargeted attacks. We just need to set the values of the different devices to a common real value and the information and compromise appetite to a same value. Hence, the attacker will not rank its targets, he will just infect as many devices as possible. If we want to only model availability untargeted attacks, this is also possible by setting the information appetite to zero and all the devices values to the same real number. Thus, untargeted and targeted attacks are included in our proposed modelling.

The second issue with this epidemic spread model is that it is a blind epidemic spread: the attacker will compromise random machines in the enclave without making any choice. While it is true that the attacker will most likely not know a priori the entire network segmentation, he will discover some of it through reconnaissance. This important step in the kill chain has not been modeled in cited papers.

In this paper, we model reconnaissance as the process of an attacker waiting for some time in the enclave and discovering some random machines (and their compromise/information value) in the process.

2.1.2 Network model

Network models tend to be very similar in the literature, with only little variation.

General considerations In (4), the network is modeled as set of enclaves (Internet included) connected via services. The authors in (4) emphasise that these connections are not meant to be physical, rather they are only logical connections representing the flow of information. The example that the authors take is that of the email that goes through many places (Internet, DMZ, email server, then downloaded and read by a LAN user) but can be represented as a link between the Internet and the LAN user.

The Internet is represented as the starting point of the attacker. This enclave is initially modeled as being compromised (as in (4)), which is reasonable because it models external threats. Some examples of network models are given in Figure 2.4 cited from (4). We can see in (a), one enclave E1 connected to the compromised Internet (in red). One can connect to E1 through the services S1, S2, .. S10. This simple architecture can be expended by adding some enclaves in a tail fashion : we get the (b) architecture. We can also have a topology with parallel branches all connected to an enclave : we get the (c) architectures.

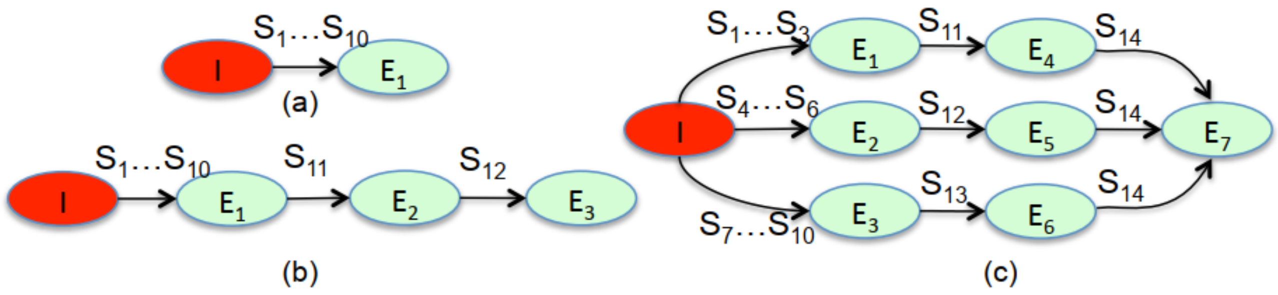

Figure 2.4: Network models, cited from (4)

We can be more general by including internal threats and considering some internal enclaves as being compromised.

Attacker model in the network Then come the issue of modelling the spread through the network. In (4), the attacker starts from the Internet and tries to infiltrate each connected enclave one by one by trying to exploit a vulnerability in a service connecting his current enclave to the remote enclave. This is a reasonable modelling as it models the fact that the attacker does not know where to spread to reach his objective, so he might try a breadth first search to explore the whole network.

Where papers differ is the probability of enclave infiltration. In (4), an infiltration is successful when the Markov chain for services running in a destination enclave reaches the "Exploit developed" state and when the attacker successfully uses it with probability $\Delta^{-1}_{exploit}$. The advantages and disadvantages of this approach have been described when discussing the Markov chain model.

Other papers like (8) used another approach which is to model enclave infiltration as the process of successfully exploiting a vulnerability with probability p_{v_i} sampled from the Common Vulnerability Scoring System distribution (CVSS, see (9)), and avoiding to get detected by the network level IDS with probability p_e. The advantages of this approach are its simplicity and its realistic behaviors. The main disadvantage however is that this model does not differentiate between services, and just give a probability drawn from the CVSS distribution, which is a distribution over all known vulnerabilities. We can add to it the fact that it doesn't take into consideration the number of services running in the enclave as opposed to the Markov chain services model, even though this can be easily fixed by drawing a probability for each service, and modelling enclave infiltration as the event of exploiting a vulnerability of a service connecting to that enclave.

The Markov chain approach is more suited to long term simulations and is effectively making the probability of enclave compromise not constant over time. The approach in (8) on the other hand supposes that these probabilities are constant. Here, we will suppose that the attack is fast enough to consider that these probabilities are constant and thus use the approach in (8) to that end.

2.1.3 Loss function

There is a large variety of loss functions in the literature and we will discuss some of them.

The goal of a loss function is to measure the performance of the network in terms of security. The goal of a cost function is to penalize the loss and encourage better performance. Loss and cost need to be combined to solve our objectives of getting the best network segmentation in terms of security, performance, while respecting certain business requirements.

The first simple loss function used in (4) is the expected probability of enclave compromise. If it is low, it means that enclaves are not likely to be compromised. Therefore, it is indicative of a secure network. However, this simple approach has obvious limitations. First of all, it doesn't take into account the fact that all enclaves are not the same. Some enclave may host critical infrastructure like web servers. Their compromise will not have the same impact as compromising a printers enclave.

This first issue has been addressed in (2), (1) and (3). It is done by introducing the notion of mission devices that, when compromised, will incur mission delay. (2) and (1) even go a bit further by introducing the notion of system security index, which is the expected normalized device uptime. Both system security index and mission delay are used to produce a normalized unified metric that is the total loss.

Even though this metric is useful when measuring the impact of compromise incidents, it lacks two things. The first of which is the distinction between low value mission devices and high value mission devices. The second thing that it fails to model is data breach: an attacker infiltrating a database server holding classified information should have an impact on the total loss. We have already explained how we dealt with this issue by using compromise and information values (See paragraph 2.1.1). The total loss is modeled in our paper as the sum of compromise values of compromised machines and the sum of information values of infiltrated machines.

The second issue that has been partially addressed in some papers is the lack of a cost function. In (7), a cost function is used in a weighted sum with expected probability of enclave compromise to get the total loss. This cost function is a normalized exponential function that is high when the number of enclaves is high. The idea is that a lot of enclaves will incur a high maintenance cost. A more detailed cost function that is even more suited to compromise incidents has been proposed in (10) which includes different types of costs that are associated with : node compromise, countermeasures (like patching) and recovery.

It is indeed pertinent to penalize our network architecture if it has too many enclaves because of the incurred maintenance cost. It is also reasonable to assume that the cost associated with node enclave compromise will not be the same than that of recovery. However, these cost functions do not capture the incurred mission delay of putting critical mission devices far from the internet. We address this issue by modelling the penalty term as the sum of the distances from the internet of each device.

2.2 Optimisation algorithms

The second main algorithm to develop after that of the simulation is the optimisation algorithm. Again, there is a variety of ways to do this.

In some papers like (4), there is not an optimisation procedure per say, authors just try out different architectures and compare the losses to make comments. An optimisation procedure should try to produce as an output the best network segmentation.

In (7) and (1), an optimisation procedure is defined as follows. There is a base architecture that is fed to the simulation which in return feeds a total loss to the optimisation procedure to produce another architecture. This new architecture will be accepted with a high chance if it is better than the currently best architecture.

What is interesting in this framework is how a new architecture is proposed. This is done by generating what authors call a neighbor of the current best solution so far. A neighbor is produced by either changing its services or its enclaves. Changing services is the process of randomly removing, adding or altering some services between random enclaves. Changing enclaves is the process of randomly merging or splitting enclaves.

The problem of this procedure is that it may suffer from exploitation. Indeed, this optimisation may be stuck in a local optimum by trying only neighbors of the currently best solution. This means that this optimisation process may not explore the whole search space of all possible network segmentations.

The characteristic of our problem is that we want to minimise a loss function that is costly to evaluate. There is no simple analytic description of this function (black-box function), we have to run a simulation many times and take the average of the losses to get a result. Thus, we also do not have any gradient information on this loss function. We also have to take into account the fact that simulations are costly and we want to find a global optimum with only a few number of evaluations.

Several algorithms have been designed to solve such a problem. For instance, Bayesian optimization uses a computationally cheap proxy (usually a Gaussian process) to approximate the function we want to optimise based on the points we have so far. This proxy is optimized to determine which solution to evaluate next.

Another algorithm is the random search algorithm. It consists in evaluating a certain number of points drawn pseudo randomly from a search space. This simple procedure has been proven powerful for black-box problems such as hyper-parameter tuning for neural networks.

A new approach that is becoming increasingly popular is that of evolutionary algorithms. The main idea of evolutionary algorithms is to generate a population of individuals. To select the best of them who will be parents and generate offsprings through cross-over and mutation. These offsprings will constitute the new population. There are many variants to this but these are the main guidelines.

(3) used a co-evolutionary algorithm (Figure 2.5 cited from (3) represents its main framework), the Grammar evolution algorithm, to solve the network segmentation problem. The contribution of this paper is that it tries to optimise the defender and attacker parameters. This is a pertinent approach because when we want to evaluate our network, we need to fix the attacker parameters like the strength or the time budget. However, our network segmentation will only be optimized for these specific parameters. In (3), the network defensive parameters tend to be optimized with respect to the best attacker parameters. This significant progress is however constrained by the fact that authors in this paper did not attempt to optimise the topology of the network, which is what we will do in this paper.

The other problem with this coevolutionary algorithm is that it only seeks to optimise fitness and not novelty. However, this approach may suffer again from exploitation due to the algorithm getting stuck in a local optimum. To deal with that approach, we will explore quality diversity (QD) algorithms that try to produce high-performing and diverse solutions.

The MAP-Elites algorithm, introduced in (11), is one of these QD algorithms which include diversity promotion. This algorithm consists in representing the search space as a grid with several dimensions that are behavioral descriptors. They are used to represent different types of solutions. For the case of a robot trying to walk, it is in (12), for each leg the percentage of its contact time with the ground. An example of an illuminated grid is showin in Figure 2.6 cited from (12). The color scale from blue to red represents the fitness of the robot in terms of walking speed (We can see that there are multiple high performing and diverse solutions). For the case

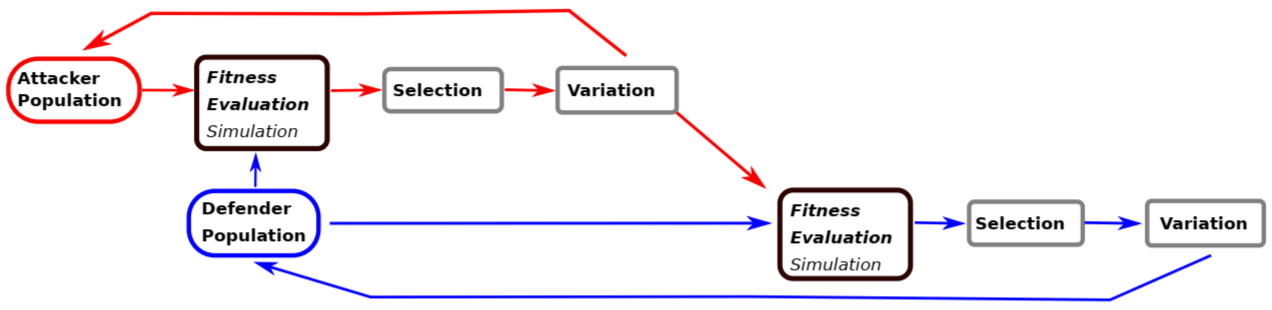

Figure 2.5: Coevolutionary algorithm, cited from (3)

of a molecule, it could simply be its weight or dimensions.

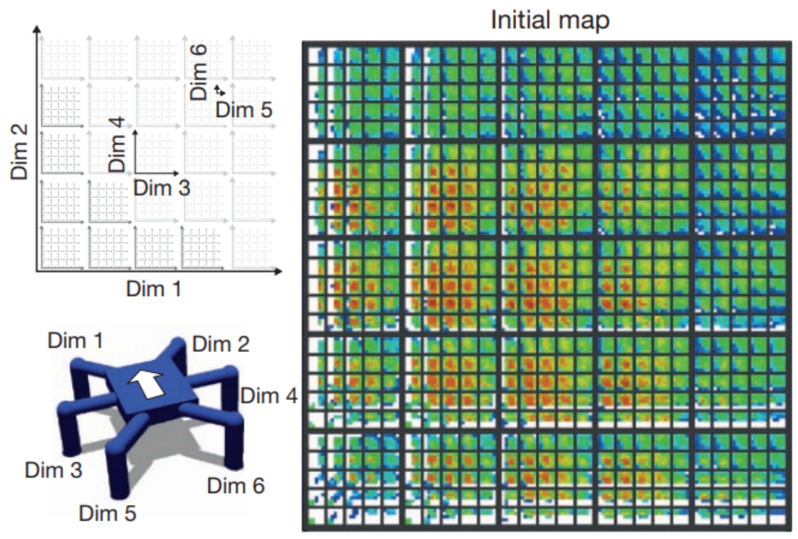

Figure 2.6: MAP-Elites grid for robot evolution, dimensions represent the leg contact percentage with the ground, red boxes represent high fitness solutions, cited from (12)

Initially, this grid is fed with an initial population of solutions drawn at random. Each solution is put to its corresponding box in the grid. If many solutions are in the same box, the best in fitness is kept. Then, some individuals in the grid are picked at random and used to generate randomly modified copies via mutation and/or crossover. These new individuals are in turn placed into their corresponding boxes. Again, only the best individual is kept if there are many individuals in the same box. This process runs for a certain number of iterations until which we can analyse the grid and get the best found individuals.

An example of its successful application is that of robotics. In (12), the MAP-Elites algorithm has been successfully used to optimise the walking speed of a robot. This powerful algorithm managed to produce different ways of walking that were all effective.

2.3 Conclusion

In conclusion, the problem of optimizing a network segmentation with respect to security, performance and business requirements has been partially addressed in the past, with some researchers implementing complete simulation and optimization frameworks to that end. However, first of all, many researchers had their model simplified, lacking some key attacker and defender behaviors, and only suited for certain class of attacks. Secondly, their optimization frameworks had limitations due to the lack of diversity promotion which lead to their algorithm producing satisfying results but not the best results. We intend to solve these issues in our thesis.

Chapter 3

Experimentation

In this chapter, we intend to propose some architecture modellings that include the network and enclave modellings, but also the attacker and defender strategies. After modelling, we run an optimization process to produce the best network segmentation with respect to our model.

3.1 First modelling

In this first modelling, we intend to propose a general and detailed model that fits a network architecture with respect to security under certain simple connectivity conditions, but without taking into account the more complex business requirements and performance issues. The goal of this section is to compare our results with that of past papers with a more general model and a different approach but with the same goal.

3.1.1 Network Model

Simulation related model As mentioned in the related work, network models are fairly similar. The main guideline of our modelling is that it needs to be realistic, complete, and detailed. We will use the network modelling presented in (8) because it successfully represents realistic behaviors and components that have not been modelled in other papers, especially the network level IDS. Our network model is summarized as follows.

- The Internet is initially compromised, each enclave has a firewall with vulnerability p_{v_i} sampled from the CVSS distribution.
- There is a Network level IDS that prevents intrusions to enclaves with an error probability of p_e.
- In order for an attacker to compromise an enclave, he needs to first be on a compromised enclave, then to exploit a vulnerability, then to evade the Network level IDS.

Optimization related model With regards to the optimization procedure, we first fix the number of enclaves N to avoid searching over a space that will be too large.

We then need to represent the network topology in a way that will be easy to handle when optimizing with the MAP-Elites algorithm. To to it, we will, in this first modelling, model the network topology as a vectorized version of an $N \times N$ symmetric matrix of length $\frac{N(N-1)}{2}$. For example, the adjacency matrix $\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ is represented by the vector : $\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$ by taking the upper part of the matrix (what is strictly above the diagonal) and reading it from top to bottom and from left to right.

Constraints For now, we only add a simple connectivity constraint : a network architecture is valid when its graph representation is connected and when its graph without the internet node is also connected.

3.1.2 Enclave Model

Enclave model is where researchers have differed the most. We will keep in this part some parts of past modellings which were reasonable. However, the main problem with past modellings is that they lacked key concepts with regards to enclave behaviors and characteristics. The main additional concepts that we model are : the concept of device values (compromise and information values), the concept of device level IDS and multi-level incident response. The enclave is modelled as follows.

Machine Model

- All devices within the enclave are reachable between themselves because devices within the same enclave tend to trust themselves.
- An enclave has at least one machine representing its firewall, some valuable devices and a pseudo-random number of ”junk machines”. These valuable devices could be an email or a web server. A ”junk machine” can be a guest computer with no intrinsic value for an attacker.
- There are M valuable devices. They are distributed across enclaves according to a global valuable devices partition. This is a list of lists that puts each valuable device in a pseudorandomly chosen enclave (but not the internet). For example, for M = 10 and N = 5 (5 enclaves including the internet, there is no machine in the internet), we could have a partition of [[5,8,4],[1,3],[],[2,0,7,9,6]]. Which means that in enclave E1, you have the valuable machines 5,8,4. In E3, you have no valuable device.
- Each machine is associated with a tuple (m, i) representing its value. The first value m is a real number between 0 and 1 and represents the impact if this machine is turned down. The second value i is again a real between 0 and 1 that represents the impact if this machine is infiltrated. A web server will have a high compromise value because if down, there will be a denial of service for a certain time. An Active Directory server will have a high information value because it hosts some sensitive data related to the network resources in a certain domain.
- The M valuable devices tend to have greater value compared to the junk machines that tend to have lower values. The value of the firewall is defined independently.

Update model

- Each enclave has a probability of update p_{update} that represents the probability that at each turn, the enclave is updated. These updates are not modelled to have any impact on the loss because they are supposed to be planned in advance. However, updates kick the attacker out of the enclave and restores every compromised machine: the enclave is no longer compromised. The attacker can try to infiltrate the enclave again with the same vulnerability sampled from the CVSS distribution at the beginning.

Incident Response model

- Each device has an integrated IDS that prevents lateral movement with probability of error p_e. When such incidents are detected by the devices, they are reported to an enclave level IDS whose role is to decide how to react when facing different situations.
- Enclave cleansing is the process of updating/re-imaging each machine in the enclave. This impacts the loss depending on the different compromise values of the devices in the enclave.
- Each enclave has an enclave level IDS with sensitivity s that indicates the probability of the enclave triggering an enclave cleansing when it detects too many alerts. A low s may result in an attacker being able to spread without being detected. However, a high s may result in a false alert that will trigger an enclave cleansing incurring mission delay for no reason.
- If a lateral movement is detected, it is blocked, but if there are too many of them, an enclave cleansing is triggered with sensitivity s. If a machine is turned down, it gets back up after some time depending on its compromise value, but if there are too many of them, an enclave cleansing is triggered with sensitivity s.

3.1.3 Attacker Model

In the literature, the attacker goal is just to compromise as many machines as possible. This is adapted only to model availability untargeted attacks. We address this issue by modelling a greater class of attacks : targeted and untargeted attacks, which will include availability attacks and data breaches. The attacker model is defined as follows.

- The attacker starts from the internet to model external threats.
- At each time step, from each compromised enclave, the attacker tries to spread to its neighbors (enclaves that are connected to the enclave the attacker is in). He succeeds if he exploits a vulnerability and if he is not detected by the Network level IDS : this idea has been presented in (8) and we use it again because it is a realistic model.
- We also add the following modelling. If an attacker infiltrates an enclave, he can get compromise and information reward. But if he is kicked out of the enclave, he can again try to infiltrate the enclave for compromise reward, but he can not get extra information reward because we assume he can not get more reward for getting the same information twice.
- When an attacker successfully infiltrates an enclave, he infiltrates a pseudo-random chosen machine from the enclave's machines.
- The attacker spends some time in the enclave. This time is defined by a global time shares list.
- When infiltrating an enclave, the attacker knows at first no machine except the one he first infiltrated.
- When an attacker discovers a machine, he discovers its value (m, i).
- The attacker has a compromise appetite c which is a real value between 0 and 1 representing its appetite for shutting down machines with high m. He also has an information appetite a which is a real value between 0 and 1 representing its appetite for infiltrating machines with high i.
- The attacker spends some time doing reconnaissance : this important aspect has not been modelled in past papers despite being one of the most important steps of an attacker. It is represented by a real value r between 0 and 1 showing how much time the attacker will spend in reconnaissance.
- When doing reconnaissance, the attacker discovers some machines index and their corresponding value.
- For undiscovered machines, the attacker has an a priori regarding their values.
- The attacker may not spend an eternity doing reconnaissance because he also has to spread within his time frame in the enclave. He also has to be fast enough to avoid getting thrown out due to enclave update.
- The attacker chooses his next machine targets in the enclave by ranking them according to their prior value and according to his appetite.
- When infiltrating a device, he may do nothing or shut down the main mission service according to his compromise appetite. This larger set of actions is more representative of what an attacker can do in general without making too specific assumptions.
- An attacker may lose his foothold in the enclave if the latter is updated or cleansed.
- An attacker may lose his foothold in the enclave if he no longer has a foothold on any device in the enclave (may happen if he turns down a machine and fails to spread to other machines before his machine getting back up).
- The total loss is

$$L = -\sum_{infiltrated\ device} a * i_{device} - \sum_{shut\ down\ device} c * m_{device}$$

3.1.4 Simulation and Optimization Frameworks

In this part we want to show how we implement the simulation and optimization procedures. The main framework is the following. A time-based probabilistic simulation algorithm is used to simulate an attacker spreading in a network segmentation with various parameters. This algorithm outputs a loss function. The optimization function tries to optimize this loss function and find the corresponding best network segmentation.

Simulation The time-based probabilistic simulation is based on the model we have defined in the previous sections. The simulation pseudo-code can be found in the Appendix. First of all, we have the initialization algorithm at 2 that initiates each enclave with their vulnerability, sensitivity, machines and enclave neighbors. Then comes the attack algorithm at 3, the one that simulates the attacker spread in the network. To do it, we follow the network model. Finally, there is the spread algorithm at 4 , the one that simulates the attacker spread in the enclave. Again, we follow the enclave model.

Optimization For the optimization algorithm, we will use the MAP-Elites algorithm (introduced in (11)). In the original version of this algorithm, we start by initializing an empty grid whose dimensions are behavioral (or feature) descriptors representing different types of solutions. The mapping between the feature descriptor and the best fitness is called $\mathcal{P}$, the mapping between the feature descriptor and the best solution is $\mathcal{X}$. Then, a random solution $\mathbf{x}'$ is put in the grid for G iterations, if many solutions are found to be in the same box in the grid, only the best performing solution is kept. After the Gth iteration, some solutions are randomly selected from the grid. A copied modified version of each selected solution is created. These new modified versions are put in the grid by again only taking the best performing solution when there is a competition between many solutions for the same grid box. The initial pseudo code taken from (11) can be see in Algorithm 1.

Algorithm 1 MAP-Elites algorithm, taken from (11)

```
$\mathcal{P} \leftarrow \emptyset, \mathcal{X} \leftarrow \emptyset$
for $iter = 1 \rightarrow I$ do
    if $iter < G$ then
        $\mathbf{x}' \leftarrow$ random_solution()
    else
        $\mathbf{x} \leftarrow$ random_selection($\mathcal{X}$)
        $\mathbf{x}' \leftarrow$ random_variation($\mathbf{x}$)
    $\mathbf{b}' \leftarrow$ feature_descriptor($\mathbf{x}'$)
    $p' \leftarrow$ performance($\mathbf{x}'$)
    if $\mathcal{P}(\mathbf{b}') = \emptyset \vee \mathcal{P}(\mathbf{b}') > p'$ then
        $\mathcal{P}(\mathbf{b}') \leftarrow p'$
        $\mathcal{X}(\mathbf{b}') \leftarrow \mathbf{x}'$
return $\mathcal{P}$ and $\mathcal{X}$
```

First, we need to define how do we encode the network segmentation for it to be easily manipulated by the operators. A network segmentation is encoded as a list of 3 lists being :

- The topology list, defined in the Network Model part.
- The partition list, defined in the Enclave Model part.
- The sensitivities list, a list consisting of the sensitivities of each enclave.

Behavior descriptors will be : the number of nodes connected to more than two other nodes, and the standard deviation of the valuable machine distribution.

We now need to define appropriate operators, namely **random_solution**, **random_selection**, and **random_variation**.

random_solution is used to generate a random network segmentation, this is done by generating a random list of zeros and ones for the topology, then by generating a random partition of the M valuable devices in the N enclaves, then by generating a random list of sensitivities selected at random between 0 and 1.

random_selection is used to select random solutions in the grid. We will use a uniform distribution to that end: each solution is selected with the same probability.

random_variation is used to create a mutated solution from a past one. To do it, we first need to mutate the topology. To that end, we will use the bit flip operator : flipping each bit with a certain probability. However, the new mutated topology may not be valid because it may not be connected (see constraints in the Network Model). To solve this issue, in this part, we will use the following approach: if the mutated solution is valid, then we keep it and stop there, Else, we will discard the mutation and try it again until we get a valid mutation. This approach works for the simple connectivity constraint of this modelling, but we intend to use a more sophisticated mutation operator for more complex constraints including degree or edge constraints. To mutate the partition, we will follow the approach in (13) which consists in randomly selecting a device and putting it in a randomly selected enclave. To mutate the sensitivities, we will use the polynomial operator presented in the NSGA-II algorithm. This operator uses a polynomial probability distribution to produce a new solution that is close to the past solution, within a certain range which is [0,1] in our case. The closeness between the new and the past solution is measured with the η parameter : a high η will produce a mutant similar to its parents, but a low η will produce a mutant that is more different.

3.1.5 Experiment and Results

In this part, we intend to run the simulation and optimization procedures with some predefined parameters in the experimental setup. Then, we will analyse the results and see how does it compare with past papers results.

Expected Results We only try in this part to find the architecture that minimises the risk. We expect to get a linear architectures with all valuable devices at the end. This is the intuitive naive architecture that seems to minimize the risk because an attacker has to go through all enclaves before being able to break into the enclave with valuable devices. Moreover, we expect low vulnerabilities first to prevent the attacker spread in the network. With regards to sensitivities, we expect high sensitivities first to kick the attacker out of the enclave with a high chance. The enclave with the valuable devices should have a medium sensibility to solve the trade-off issue between undefended enclave and false alerts.

Experimental Setup In this first experiment, we want to choose parameters that reflect realistic scenarios. To model this architecture, we will thus choose the following parameters.

- $N = 5$ enclaves, which a reasonable number of enclaves for medium scale enterprise networks.
- $M = 10$ valuable devices : Web and Web Proxy Servers, DNS server, FTP server, E-mail server, App Server, SQL servers, DHCP server , Authentication Server and a Syslog server. We model these high value devices as having a high compromise and information value of 0.8 because these types of servers host sensitive information and, if compromised, they can disrupt some missions.
- A pseudo-random number of low value devices between 0 and 20 in each enclave, which represents approximately 7 employees in each team, each employee having one laptop and maybe a smartphone connected to the Wi-fi.
- vulnerabilities are (in enclave order): 0.5 , 0.6 , 0.8 , 0.7 as sample from the CVSS distribution as done in (8).
- The simulation time is 24 time units, and the attacker spends 6 time units in each enclave (other than the internet), which means that he will spend the same time in each enclave because he doesn't have an a priori knowledge about the network architecture and its valuable device distribution.
- The attacker reconnaissance parameter r is equal to 0.7, which means that he will spend 70% of his time doing reconnaissance, which is reasonable because of the importance of such a task.
- We model the attacker as having an information appetite of 0.4 and a compromise appetite of 0.9, which represents an attacker whose main goal is to compromise as much device as possible, but who is also interested in retrieving high value data.
- Features for the MAP-Elites algorithms are : the number of nodes with degree greater or equal to 3, and the standard deviation of the machines distribution.
- The loss is the attack loss defined in the modelling without a penalty term.

Results Here are the results for the first modelling in Figure 3.1. We can see that it is a linear architecture, enclaves are connected in a tail fashion as expected at the beginning of this section. The first 3 enclaves act like barriers : they have high sensitivities and no valuable device in them. High sensitivities prevent attacker spread within the enclave and also frequently triggers enclave cleansings which kick the attacker out of the enclave. The attacker then has to start all over again from the internet or the last compromised enclave. Then the last enclave has all of the valuable devices. These results are all expected, expect for two things. The first unexpected thing is that this last enclave has a pretty low sensitivity to avoid false alerts which may cost too much due to the fact that if there is a false alert, all servers will be down, which will incur a very high loss. The second unexpected thing is that enclave barriers are not the one with the lowest vulnerabilities. In fact, the servers enclave has the second lowest vulnerability, which sounds like a trade-off between : using low vulnerabilities to protect the servers enclave, and protecting the servers enclave itself to prevent attacker from infiltrating if he succeeds to get as far as E3.

For 7 enclaves, we reach similar results that can be found in the Appendix at A.2. This shows that this type of architecture is not an isolated case for just 5 enclaves.

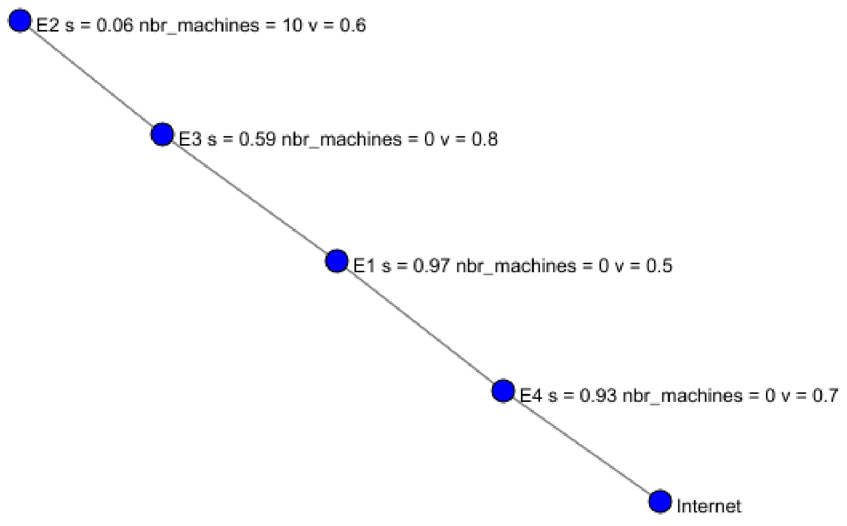
E2 s = 0.06 nbr_machines = 10 v = 0.6
E3 s = 0.59 nbr_machines = 0 v = 0.8
E1 s = 0.97 nbr_machines = 0 v = 0.5
E4 s = 0.93 nbr_machines = 0 v = 0.7
Internet

Figure 3.1: 5 enclaves, no penalty, best configuration

3.1.6 Conclusion

These linear architecture results are consistent with the results of past papers and match greatly with our expectations. In fact, in (4) and (1), the best network segmentation is a linear architecture, with valuable devices at the end of the network in (1). This first modelling intended to pursue the same goal of past papers, which was to find the best network segmentation in terms of security with only a simple connectivity constraint. Even if our approach was different, with a more general and detailed model, we still got similar results. This confirms that the linear architecture is a strong candidate to consider when designing networks and when performance is not key. However, in reality, performance is important to optimize, and constraints are much more numerous and complex than just connectivity constraints. We intend to model penalty in a second modelling to see how does the best segmentation changes.

3.2 Second modelling

In this second modelling, the goal is to make our model more realistic by adding a new performance constraint by adding a penalty term to the loss and see how does this affect the topology of the network segmentation compared to when we did not fit with respect to performance.

3.2.1 Penalty term

In this new modelling, we want to add a penalty term to the loss. Indeed, putting all the valuable machines at the end of the network seems intuitive from a risk management perspective. However, such configuration will incur a mission delay due to the high distance with the internet. We add a penalty term to capture this information and prevent tail architectures.
Mathematically, we compute :

$$\textbf{penalty} = \sum_{enclave\ e} \#\{machines \in e\} \times d(\textbf{Internet}, e)$$

Having many machines at a high distance from the distance results in a high penalty. High number of machines close to the Internet results in a low penalty. Let's define **attack** as the first loss:

$$\textbf{attack} = -\sum_{infiltrated\ device} a * i_{device} - \sum_{shut\ down\ device} c * m_{device}$$

The total loss is thus :

$$\textbf{Loss} = \textbf{attack} - \textbf{pcoeff} \times \textbf{penalty}$$

With **pcoeff** being a coefficient to express how much weight we put to the penalty term (0.01 seems like a great trade-off).

3.2.2 Experiment and Results

Expected Results Valuable devices can't be too far from the network, so we might expect the best segmentation to put these devices close to the internet but in very protected enclaves with low vulnerability and medium sensitivity. As devices have to be close to the internet, we could also expect the devices to be spread out in different enclaves to minimise the risk.

Experimental Setup We will run two experiments with the same parameters as in the first modelling. We only add a penalty coefficient, which we set to 0.01.

Results For 5 enclaves, we reach a new optimized architecture. These results contradict our expectations. This new architecture has a barrier with high sensitivity and no valuable machine, then a node that acts like another barrier and as a super router with 2 branches. One branch has, at its end, most valuable machines. The other branch acts like a diversion branch to divert the attacker into a path with a low number of valuable machines, but where he might get stuck and lose time. We can also note that the architecture is shortened. The maximum distance from the internet is 3 hops compared to 4 hops without penalty. In terms of vulnerability distribution, the enclave with the large number of valuable devices has the lowest vulnerability. The second two lowest vulnerabilities are appointed to the 2 barriers. Then the diversion enclave has a much higher vulnerability (0.8) than the enclave with 8 valuable devices (0.5 : the lowest vulnerability). The attacker, when at at the super node (E2), will then most likely in the next time unit infect this diversion enclave. This confirms that the optimisation indeed found a diversion strategy.

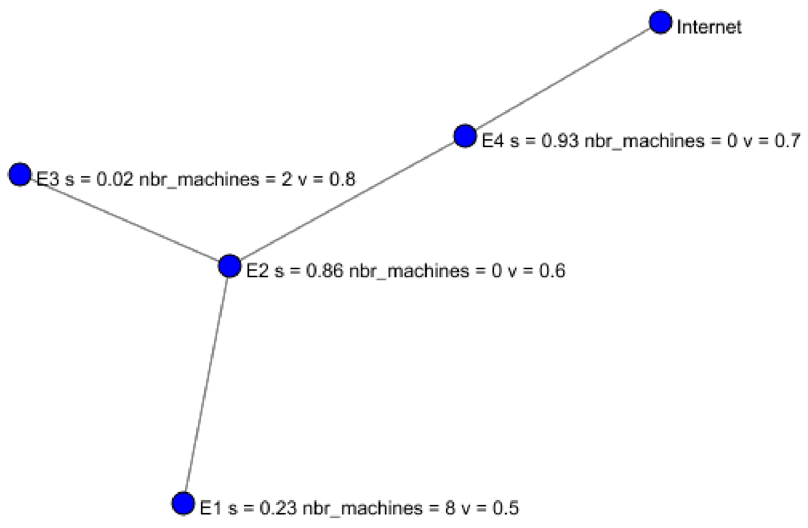
Internet
E4 s = 0.93 nbr_machines = 0 v = 0.7
E3 s = 0.02 nbr_machines = 2 v = 0.8
E2 s = 0.86 nbr_machines = 0 v = 0.6
E1 s = 0.23 nbr_machines = 8 v = 0.5

Figure 3.2: 5 enclaves, penalty term, best configuration

For 7 enclaves, we reach a similar shortened architecture that can be found in the Appendix at A.1.

3.2.3 Conclusion

This new modelling shows that when we want to optimize our network with respect to security and performance with simple constraints, we get a completely new architecture that has not been found in the literature. In fact, this new best architecture, which we may refer to as a diversion architecture, shows that the diversion strategy needs to be considered by network architects when security and performance have to be optimized with simple connectivity constraints. We intend in a further part to explore more complex constraints that include degree and edge constraints.

Chapter 4

Plan for future work

We intend to complete this thesis by doing the following tasks in order.

- Model degree and edge constraints and implement a new mutation operator that mutates a network segmentation within a constrained search space using the graph edit distance : Estimated time : 1 week.
- Refine the network and enclave model by modelling even more realistic network characteristics, such as (ideas for further modelling): maximum enclave capacity, vulnerabilities for each service, multiple types of missions and mission dependencies, time between attack and detection, more counterattacks. Estimated time : 2 weeks.
- Model a broader variety of attackers including insider threats. Estimated time : 2 weeks.
- Adapt our evolutionary algorithm to a co-evolutionary algorithm to also optimize the attacker strategy. The attacker and defender fitness functions are not the same: the attacker only sees what it gets, while the defender sees all true damage done to the network. Estimated time : 2 weeks.
- Propose a dynamic framework where the proposed network segmentation is an (as simple as possible) adaptation of an already existing baseline architecture, and where the network is adapted as the attacker spread (trade-off between containing the attacker and maintaining as many services as possible). Estimated time : 2 weeks.
- Writing and preparation for the final presentation. Estimated time : 3 weeks.

Bibliography

[1] Wagner N, Şahin C Winterrose M, Riordan J, Pena J, Hanson D, et al. Towards automated cyber decision support: A case study on network segmentation for security. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI); 2016. p. 1–10. pages 4, 5, 7, 9, 10, 17

[2] Wagner N, Sahin CS, Winterrose M, Riordan J, Hanson D, Peña J, et al. Quantifying the mission impact of network-level cyber defensive mitigations. The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology. 2016 08;14. pages 4, 5, 7, 9

[3] Hemberg E, Zipkin JR, Skowyra RW, Wagner N, O'Reilly UM. Adversarial Co-Evolution of Attack and Defense in a Segmented Computer Network Environment. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. GECCO '18. New York, NY, USA: Association for Computing Machinery; 2018. p. 1648–1655. Available from: `https://doi.org/10.1145/3205651.3208287`. pages 4, 5, 7, 9, 10, 11

[4] Wagner N, Sahin CS, Peña J, Riordan J, Neumayer S. Capturing the Security Effects of Network Segmentation via a Continuous-Time Markov Chain Model; 2017. p. 17. pages 5, 6, 7, 8, 9, 10, 17

[5] Frei S, Schatzmann D, Plattner B, Trammell B. Modeling the Security Ecosystem - The Dynamics of (In)Security. In: Moore T, Pym D, Ioannidis C, editors. Economics of Information Security and Privacy. Boston, MA: Springer US; 2010. p. 79–106. pages 6

[6] Frei S, May M, Fiedler U, Plattner B. Large-Scale Vulnerability Analysis. In: Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense. LSAD '06. New York, NY, USA: Association for Computing Machinery; 2006. p. 131–138. Available from: `https://doi.org/10.1145/1162666.1162671`. pages 6

[7] Wagner N, Sahin C, Peña J, Streilein W. A nature-inspired decision system for secure cyber network architecture. 2017 IEEE Symposium Series on Computational Intelligence (SSCI). 2017:1–8. pages 7, 9, 10

[8] Muñoz González L, Sgandurra D, Paudice A, Lupu EC. Efficient Attack Graph Analysis through Approximate Inference. ACM Trans Priv Secur. 2017 Jul;20(3). Available from: `https://doi.org/10.1145/3105760`. pages 9, 12, 14, 16

[9] Common Vulnerability Scoring System, V3. 2015;. `https://www.first.org/cvss`(2015). pages 9

[10] Soikkeli J, Muñoz-González L, Lupu EC. Efficient attack countermeasure selection accounting for recovery and action costs. CoRR. 2019;abs/1904.03082. Available from: `http://arxiv.org/abs/1904.03082`. pages 9

[11] Mouret J, Clune J. Illuminating search spaces by mapping elites. CoRR. 2015;abs/1504.04909. Available from: `http://arxiv.org/abs/1504.04909`. pages 10, 15

[12] Cully A, Clune J, Mouret J. Robots that can adapt like natural animals. CoRR. 2014;abs/1407.3501. Available from: `http://arxiv.org/abs/1407.3501`. pages 10, 11

[13] Greene W. Partitioning Sets with Genetic Algorithms.; 2000. p. 102–106. pages 15

Appendix A

Appendix

Algorithm 2 Initialization algorithm

```
Require: sol, encoded network segmentation as topology, partition and sensitivities, global_vulnerabilities list of $N$ probabilities sampled from CVSS distribution
  function INITIALIZE(sol)
      Enclaves = empty list of length N
      for $i = 1 \to N$ do
          e = Enclaves[i]
          e.vulnerability = global_vulnerabilities[i]
          e.sensitivity = sol.sensitivities[i]
          e.machines = sol.partition[i]
          e.machines += random number of low value devices
          e.neighbors = sol.topology[i]
      loss = attack(Enclaves,T,times)
      return loss
```

Algorithm 3 Attack algorithm

```
Require: enclaves: list of enclaves, T: the simulation time, times: mapping between enclave and the time to spend in it, $p_e$: probability of network level IDS error
  function ATTACK(enclaves,T, times)
      compromised_enclaves (at the moment)$\leftarrow \emptyset$
      already_compromised (at least once in the past)$\leftarrow \emptyset$
      loss = 0, spent_time = 0
      while spent_time $\leq$ T do
          for Enclave e $\in$ compromised enclaves do
              for Neighbors next $\in$ e.neighbors do
                  if next not compromised then
                      infect = random(0,1)
                      IDS_intercept = random(0,1)
                      if infect $\leq$ next.vulnerability and IDS_intercept $\leq p_e$ then
                          loss += spread(next, times[next], not next in already_compromised)
                          already_compromised += next
                          if next.compromised then
                              compromised_enclaves += next
                          spent_time += times[next]
                          if spent_time $\geq$ T then
                              return loss
          spent_time += 1 ▷ (trying to infiltrate enclaves costs 1 time unit)
      return loss
```

Algorithm 4 Spread algorithm

```
Require: Enclave e: enclave being infected, time: the time to spend infiltrating the enclave, info_loss_available: can an
    attacker get information reward (not if he already infiltrated the enclave in the past)
    function SPREAD(Enclave e, time, info_loss_available):
        loss = 0, e.compromised = True, discovered_machines ← $\emptyset$
        recon_time = r×time ▹ (r is the reconnaissance percentage)
        for t = 1 → recon_time do Reconnaissance
            update = random(0,1)
            if update ≤ $p_{update}$ then ▹ Updates kick the attacker out of the enclave
                Set all enclaves machines as uncompromised
                e.compromised = False
                return loss
            not_discovered = e.machines - discovered_machines
            if length(not_discovered) ≥ 0 then ▹ If there are some machines left to discover
                discovered_machine = random_choice(not_discovered)
                discovered_machines += discovered_machine
                Update prior value of the discovered machine
                return loss
        for t = recon_time → time do Spread
            update = random(0,1)
            if update ≤ $p_{update}$ then ▹ Updates kick the attacker out of the enclave
                Set all enclave machines as uncompromised
                e.compromised = False
                return loss
            to_infect = machines that have not yet been infected, ranked according the their prior values.
            IDS_ALERT = 0 ▹ number of prevented lateral movements
            for Machine m ∈ to_infect do
                Device_IDS_intercept = random(0,1)
                if Device_IDS_intercept ≤ $p_e$ then ▹ Lateral Movement succeeded
                    m.infected = True, m.already_infected = True
                    loss -= info_loss_available×a×m.info_value ▹ update information loss if it is available
                    compromise = random(0,1) ▹ compromise attempt with probability c
                    if compromise ≤ c then ▹ compromise success
                        m.turned_down = True ▹ it is up after a certain time dependent on its compromise value
                        loss -= c×m.compromise_value ▹ update compromise loss
                        threshold = e.nbr_turned_down×e.sensibility
                        if random(0,e.nb_machines) ≤ threshold then ▹ Too many down machines trigger enclave cleansing
                            Set all enclave machines as uncompromised
                            loss -= e.total_compromise_value/2 ▹ divided by 2 because there is no detection or investigation time
                            return loss
                else ▹ Lateral Movement prevented by the device level IDS
                    IDS_ALERT += 1
                    if random(0,e.nb_machines) ≤ IDS_ALERT×e.sensibility then ▹ Too many detected lateral movements trigger
    enclave cleansing
                        Set all enclave machines as uncompromised
                        e.compromised = False
                        loss -= e.total_compromise_value/2
                        return loss
        return loss
```

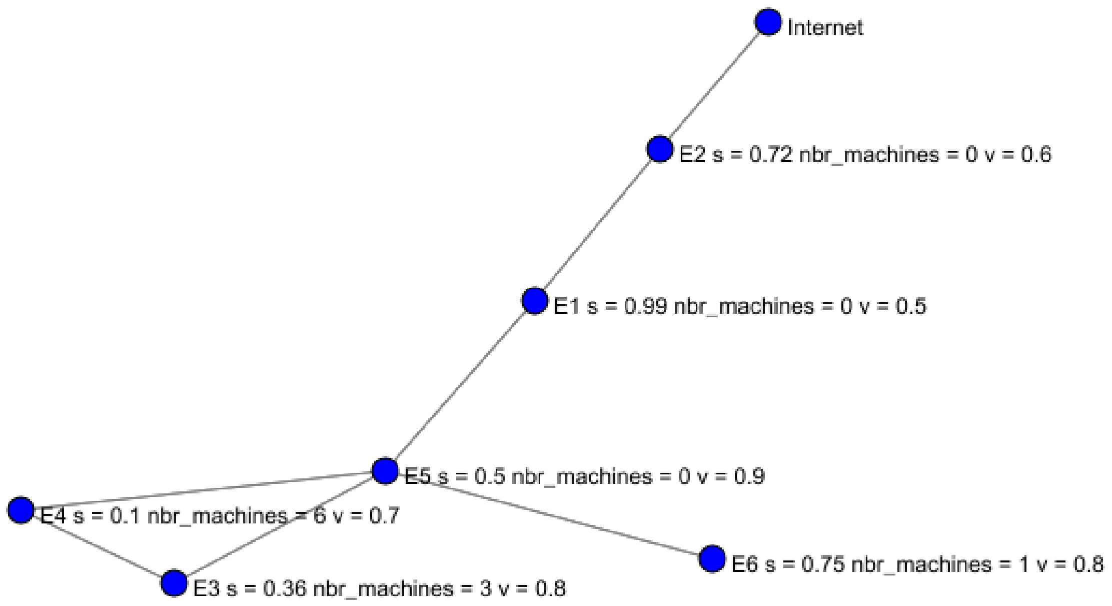
Internet
E2 s = 0.72 nbr_machines = 0 v = 0.6
E1 s = 0.99 nbr_machines = 0 v = 0.5
E5 s = 0.5 nbr_machines = 0 v = 0.9
E4 s = 0.1 nbr_machines = 6 v = 0.7
E6 s = 0.75 nbr_machines = 1 v = 0.8
E3 s = 0.36 nbr_machines = 3 v = 0.8

Figure A.1: 7 enclaves, penalty term, best configuration

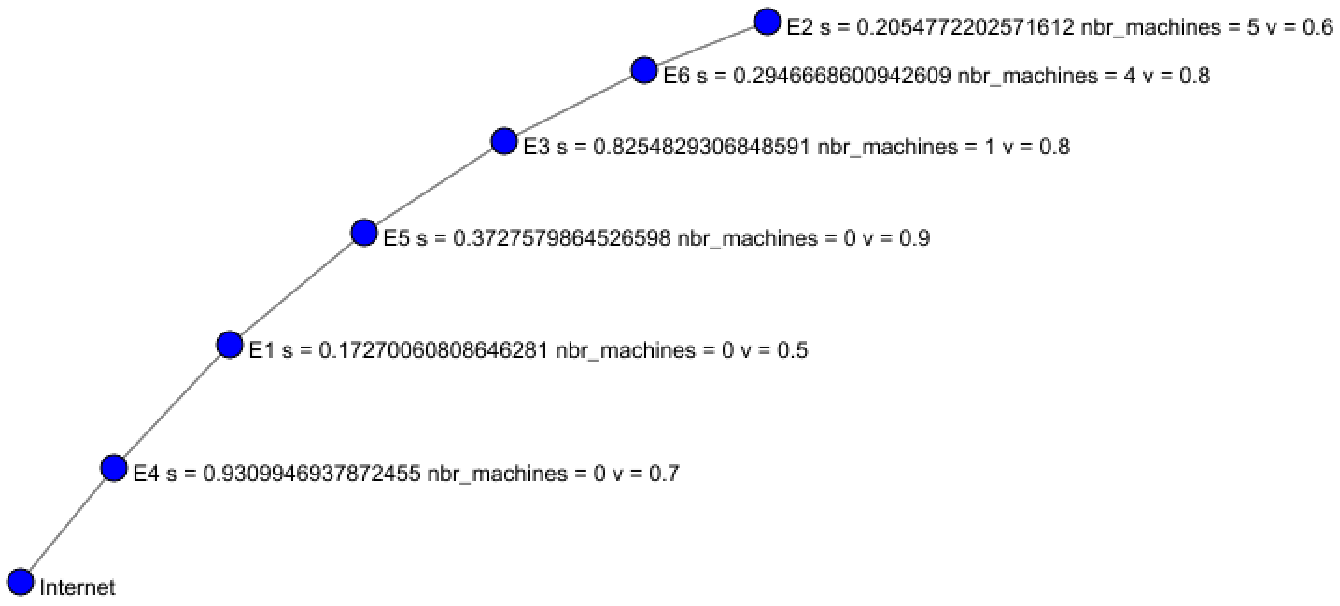
E2 s = 0.2054772202571612 nbr_machines = 5 v = 0.6
E6 s = 0.2946668600942609 nbr_machines = 4 v = 0.8
E3 s = 0.8254829306848591 nbr_machines = 1 v = 0.8
E5 s = 0.3727579864526598 nbr_machines = 0 v = 0.9
E1 s = 0.17270060808646281 nbr_machines = 0 v = 0.5
E4 s = 0.9309946937872455 nbr_machines = 0 v = 0.7
Internet

Figure A.2: 7 enclaves, no penalty, best configuration