University of Piraeus

School of Information and Communication Technologies

Department of Digital Systems

Postgraduate Program of Studies

MSc Digital Systems Security

Solving Zero-Sum Strategic Games for Dynamic Honeypot Allocation over Attack Graphs.

MSc Thesis

Supervisor Professor: Dr. Stefanos Gritzalis

Name-Surname	E-mail	Student ID.
Ioannis Koufos	mte2014@unipi.gr	MTE2014

Piraeus

18/03/2024

Περίληψη

Αυτή η διατριβή διερευνά τη στρατηγική τοποθέτηση honeypots για την ενίσχυση της κυβερνοασφάλειας μέσω του φακού της θεωρίας παιγνίων. Ξεκινώντας με μια εισαγωγή στο τι είναι τα honeypots, τη σημασία τους και τα διάφορα επίπεδα αλληλεπίδρασης, η μελέτη εμβαθύνει σε θεμελιώδεις έννοιες της θεωρίας παιγνίων, συμπεριλαμβανομένων των ιδιωτών πρακτόρων, της ισορροπίας Nash και των μικτών στρατηγικών. Στη συνέχεια, η έρευνα προχωρά στην πρόταση ενός πλαισίου μοντελοποίησης για την τοποθέτηση honeypot, που ενσωματώνει θεωρητικές αρχές παιχνιδιών και το Common Vulnerability Scoring System (CVSS) ενσωματώνοντας τη δημιουργία προσαρμοσμένων μοντέλων γραφημάτων επίθεσης, κανόνων, και συσχετισμών. Επιπλέον αναφέρονται τεχνικές οπτικοποίησης εμπλουτίζουν την κατανόηση του προτεινόμενου πλαισίου. Στη συνέχεια, διερευνώνται οι βέλτιστες στρατηγικές κατανομής honeypot μέσω της διαμόρφωσης παιχνιδιών μηδενικού αθροίσματος, από την άποψη του αμυνόμενου. Παρουσιάζεται μια μεθοδική προσέγγιση για τη διατύπωση πινάκων απόδοσης και επίλυση μοντέλων παιχνιδιών χρησιμοποιώντας τη revised simplex τεχνική, γραμμικής πολυπλοκότητας. Ολοκληρώνοντας, ένα παράδειγμα επίδειξης απεικονίζει την πρακτική εφαρμογή του προτεινόμενου πλαισίου σε ένα παράδειγμα τοπολογίας με 5 υπολογιστές. Συμπερασματικά, αυτή η εργασία υπογραμμίζει τη σημασία της ενσωμάτωσης της θεωρίας παιγνίων στις στρατηγικές ασφάλειας στον κυβερνοχώρο.

Abstract

This thesis explores the strategic deployment of honeypots for enhancing cybersecurity defenses through the lens of game theory. Beginning with an introduction to honeypots, their significance, and various interaction levels, the study delves into fundamental concepts of game theory, including self-interested agents, Nash equilibrium, and mixed strategies. The research then progresses to propose a honeypot allocation modeling framework, incorporating game theoretic principles and the Common Vulnerability Scoring System (CVSS) including discussions on creating custom attack graph models, rules, associations and also visualization techniques to enrich the understanding of the proposed framework. Subsequently, optimal honeypot allocation strategies is investigated through the formulation of zero-sum games, on the defender perspective. A methodical approach to formulating payoff matrices and solving game models using linear revised simplex methods is presented. Furthermore, an example demonstration illustrates the practical application of the proposed framework in a 5 Host topology scenario. In conclusion, this work underscores the importance of integrating game theory into cybersecurity strategies,.

Table of Contents

Table of Tables

Table of Figures

Chapter 1: Introduction

The ever-evolving nature of cyber-attacks is task which is constantly tackled by security professionals. Security administrators try to protect those systems by identifying all vulnerabilities, exploits and ways in which a machine can be compromised. While individual vulnerabilities might seem manageable on their own, malicious actors can often chain exploits together so as to bypass security measures, posing a significant threat that traditional security approaches and related controls often severely underestimate. Sophisticated cyber-attacks necessitate innovative approaches to safeguard digital assets. Among the arsenal of defensive measures, honeypots emerge as a strategic tool for cybersecurity professionals, offering insights into malicious activities and augmenting threat detection capabilities, when placed correctly.

Chapter 2 of this work focus on an exploration of honeypots, providing a comprehensive overview of their role in cybersecurity defense strategies. Furthermore, the chapter, provides a brief introduction to honeypots, setting the stage for a more detailed examination of their characterization by interaction level. Understanding the intricacies of honeypots, from their inception to their evolution as sophisticated decoy systems, lays the foundation for delving deeper into their usage and why strategic deployment is needed. Chapter 3 delves into the realm of game theory, a discipline renowned for its applicability in modeling strategic interactions among rational agents. Self-interested agents, Nash equilibrium, and Pareto optimality are among the fundamental concepts that are depicted in this chapter, serving as theoretical background for the subsequent exploration of honeypot allocation modeling. Chapter 4 focuses on a novel approach to honeypot deployment, informed cybersecurity risk assessment methodologies as occur from graphical network security models. The integration of the Common Vulnerability Scoring System (CVSS) and attack graph modeling techniques enriches the framework, enabling a more nuanced understanding of the cybersecurity landscape and facilitating informed decision-making. The rest of the chapters delve into the practical application of the proposed framework, explaining the optimal honeypot allocation strategy through the lens of zero-sum games.

In essence, this thesis endeavors to bridge the realms of cybersecurity and game theory, offering a framework for optimizing honeypot deployment and enhancing cybersecurity resilience. Through rigorous analysis and practical demonstrations, this research seeks to empower organizations with the tools and insights needed to navigate the complex cybersecurity landscape and safeguard their digital assets effectively.

Chapter 2: A Brief Introduction on Honeypots

Honeypots are fake systems/hosts that are placed inside all types of networks, so as to fool intruders/attackers into thinking that they are legitimate devices with a purpose. A honeypot serves as a strategic tool designed to attract and potentially fall victim to malicious attacks. To do so, most of the times these devices imitate valuable known services that are often met in regular business networks and personal networks [1]. These services may vary from regular SSH servers to HTTP servers, monitoring systems, cloud servers, platform controllers, VPN servers and many more. Each one of the aforementioned services resides in different ports meaning that a honeypot can actually run multiple services at the same time. However, in order to be seen as a target from an attacker's view, it must imitate a realistic host. That means no business networks that consider themselves to have a proper and realistic architecture will ever place a cloud server, and a VPN server in the same host, as it's inevitable to preserve the availability in the system in case of an accident. Apart from imitating a dummy target that lures attackers, Honeypots are also an important tool for Forensic Analysis [2]. As it has been mentioned before, honeypots are particularly useful when an attack in not yet known to the cybersecurity world. It is feasible for a Honeypot to host network and forensic analysis tools, keeping logs of all the actions that are carried out to the host. This enables cybersecurity penetration testers, security admins, bug-hunters, researchers and analysts to detect and discover new attacks by analysing logs gathered by the honeypot. Moreover, honeypots provide the following advantages and indicatively some of them are as follows:

- A honeypot is never connected to other assets in the system. Meaning that all collected logs are exclusively malicious logs.
- Honeypots are helpful defensive countermeasures that are flexible enough to adapt to various environments.

- Honeypots usually have small resources requirements.

2.1: Honeypot Characterization by purpose

[2], [3] state that honeypots can be characterized in many ways. Among them, the literature focuses on usage, distinguishing 2 types, namely research honeypots and production honeypots.

2.1.1 Honeypots used for research

A research honeypot serves the purpose of gathering insights into the activities of the an attacker and does not directly contribute to an organization's operational value [4]. Its primary objective lies in acquiring intelligence on potential threats that organizations might encounter, enhancing their defensive capabilities. By carefully watching the attackers' methodologies and attack vectors, they can achieve a deeper level of understanding their motives and behaviors. Deploying and maintaining research honeypots is a complex task, demanding considerable time and effort due to the extensive data they capture.

Although research honeypots offer minimal direct security benefits to organizations, the knowledge gleaned from them can be leveraged to enhance measures for preventing, detecting, and responding to attacks. They play an important role in advancing research by providing a platform to observe cyber threats. Furthermore, research honeypots contribute to the development of analytical and forensic skills, occasionally leading to the identification of new malware.

2.1.2 Honeypots used for production

Production honeypots have as a primary functionality to protect an organization from attacks act as security controls that can potentially mitigate risks from incoming attacks [2], [5]. They deliver immediate security benefits, require fewer functionalities than research honeypots and are simpler to develop as they are not built with CTI gathering in mind. However, while they pose significant benefits in identifying attacks, such honeypots are limited to insights about attackers and the do not provide detailed information about the attacker’s identities or tools. They are typically deployed within production networks or and emulate real-world environments. According to [4], the main scope of usage also includes the detection of zero-day vulnerabilities, acting as an early warning system.

2.2: Honeypot Characterization by Interaction Level

The interaction level of a honeypot is essentially the degree to which an adversary can access the system/host services. Three interactions levels are defined: (1) Low interaction, (2) Medium interaction and (3) High interaction [2], [3], [5].

Table 1: Factors associated with Honeypots, based on their interaction level [1]

	Low Interaction	**Mid Interaction**	**High Interaction**
Involvement	Low	Mid	High
Acts as an OS	No	No	Yes
Risk	Low	Mid	High
CTI Gathering	Connections on services	Connections on services + Requests	Possibly ALL
Compromise	No	No	Yes
Develop	Low	High	High
Maintenance	Low	Low	Very High

2.2.1 Low Interaction Honeypots

A low interaction Honeypot is described as a host where the interaction with the rest of the system is kept to the minimum, meaning that it cannot be used to launch attacks to an external system and is usually an ending destination for an attacker. These honeypots are most likely to be a target and it is common for an attacker to easily tell the difference between a legitimate host and a honeypot as they do not host an operating system. Mokube et al. in [5] explain that a low interaction honeypot can also be compared with a passive intrusion detection system (IDS), as there is no possible way to directly interact with the attacker. While, this option does not provide any implementation risks, the expected reward from using such systems is also quite limited. Moreover, the authors in [6], highlight that there is high usage of low interaction honeypots in analyzing spam and worms.

2.2.2 Medium Interaction Honeypots

A medium interaction honeypot is described as a honeypot that implements a concrete but not full set of services but at the same time do not come with an operated system installed. These services are more complicated and gather cyber-threat intelligence up to specific level, but they can be almost immediately recognized as honeypots. Mokube et al. [5] highlight that medium interaction honeypots provide the attacker with a better illusion since they allow more interactions, leading to more cyber-threat information gathered. Similarly, to low interaction honeypots, the same logic is considered. While the usage of such systems, has a higher risk in comparison to low interaction honeypots [7], their usage is considered to be a more versatile and useful.

2.2.3 High Interaction Honeypots

A high interaction honeypot is described as a full operating system with a plethora of functionalities available, even functionalities that allow consequent actions to take place, such as launch attacks or install backdoors or applications [3]. Their goal is to provide an interactive system where nothing is simulated and at the same time collect various types of information of all available actions and not only from logs [7]. Due to the freedom level in high interaction honeypots being particularly high, these devices are highly possible to not be easily distinguished from a regular system from an attacker.

Chapter 3: Game Theory Fundamentals

To set the foundation for the upcoming analysis of strategic honeypot placement, this chapter will provide a brief introduction to the game theory. Game theory studies strategic decision-making by combining a set of disciplines such as mathematics, psychology, philosophy and nowadays computer science which provides an efficient tool for solving, evaluating, and simulating various games. Several fields seen the application of game theory to be an important tool for enhancing and incorporating reasoning and decision making in complex and multi-factor situations. Some of those fields are business, finance, economics, political science, psychology and since the early 20s cybersecurity is a field that benefits considerably from game theoretic approaches in many ways.

3.1 Self-interested Agents and Utility Theory

This short introduction starts with the notion of self-interested agents which is a common concept in game theory, describing individuals, groups or entities that primarily act in their own best interest and have a specific goal. A self-interested agent particularly prioritizes her/his interest above others and make decision based on maximizing their own utility in terms of some short of profit. Profit is usually interpreted differently and has to do with the nature of the game and the goal set. A self-interested agent has his/her own description of states of the world that are preferable, compared to others and acts based on the aforementioned description. An utility function of a self-interested agent can:

- Quantify degree of preference
- Explain the impact of uncertainty
- Be a Decision-theoretic rationality (act towards the maximization an expected utility)

For example, the weather can be 25 degrees Celsius with a probability of 0.4 and 25 degrees Celsius with probability of 0.3, $where\, p(w = 24) + p(w = 25) = 1$. The agent is expected to have an opinion on which is the preferable temperature according to his tastes.

The authors in [8], create the axiom of the utility function. Precisely, Von Neumann and Morgenstern's theorem states that under certain conditions, preferences can be represented by a utility function that assigns a real number to each possible outcome of a decision. Key conditions for that theorem are completeness, transitivity, and continuity of preferences, as well as the independence of irrelevant alternatives:

- **Completeness** - The decision-maker can compare any two alternatives and has a preference for one over the other, or is indifferent between them.
- **Transitivity -** If the decision-maker prefers option A over option B, and option B over option C, then the decision-maker also prefers option A over option C.
- **Continuity** - The decision-maker's preferences are continuous, meaning that if option A is preferred to option B, there exists some probability at which the decision-maker would be indifferent between receiving option A for sure and taking a gamble with a chance of receiving option B.
- **Independence of irrelevant alternatives solutions** - If the decision-maker prefers option A to option B, introducing a third option that is worse than A but better than B should not change the preference between A and B.

Von Neumann and Morgenstern in their fundamental book of game theory distinguish two games: (i) Normal Form Games (NFG - also called Strategic form), (ii) Extensive Form Games (EFG). The first list the payoff that agents get as function of their own actions (e.g., Matrix Games – Prisoner Dilemma), while the latter indicate games in which agents move sequentially and the actions taken are represented as tree (e.g., Poker, Chess, …). EFG further keep track of what each agent knows when he/she makes each decision.

A finite $n - person$ NFG is defined as $< N, A, u >$:

- Players: $N = \{1, \dots, n\}$ is a finite set of n, indexed by i
- A player i, has a set of possible actions A_i,where:

$a = (a_1, \ldots, a_n) \in A = A_1 \times \ldots, \times A_n$ is an action profile

- u is an utility of payoff function for player $i: u_i: A \rightarrow \mathbb{R}$, where:
  $u = (u_1, \ldots, u_n)$ is a profile of utility functions and $u_i(\alpha)$ is the payoff of player i if α is the profile of actions chosen.

NFG are often represented by a table and are the most famous and basic games. While one indicative example is the Prisoners Dilemma, in this part of the Master Thesis a more suitable example will be described: the TCP Backoff Game of non-cooperative game theory.

TCP Backoff Game

The game analyzes the behavior of a competing network flows in a shared network environment in context of congestion control [9]. In the game, multiple TCP flows, represented as players, are contending for bandwidth in shared network. Each TCP flow has a goal of maximizing its own throughput and at the same time consider network congestion and fairness.

Each TCP flow starts by transmitting data packets at an initial rate which is determined by a congestion control algorithm. While packets are successfully transmitted without loss the sending rate is said to increase. If a network congestion occurs, routers can drop packets and trigger congestion control mechanisms in the TCP protocol. While detecting packet loss, TCP flows react by reducing the transmission rate to avoid packet loss. When packet loss happens, TCP enter in a backoff state where temporarily reduces the transmission rate before gradually starts increasing them again and this process helps in the prevention of further congestion, allowing the network to slowly recover. When multiple TCP flows compete for bandwidth provision, they continuously adjust their transmission rates based on the network conditions (e.g., bandwidth availability, delay, packet loss). Flows that experience greater congestion may reduce their transmission rates is a more aggressive manner, while those with less congestion can instead increase their transmission rates faster. The game is depicted in the following example[1]:

- Both TCP flows use a correct implementation give 1ms delay in the network

[1] Example is from the citated source, as well as the given payoffs.

- One Effective and One Defective implementation give 4 ms delay and 0 ms delay respectively
- Both TCP flows defective results in 3 ms delay.

The above raise the question on if packets should be sent using a correctly implemented TCP or an incorrect implementation?

The matric below the TCP game is written in normal form:

- Rows are the actions of player: A_1
- Columns are the actions of player: A_2
- Cells define outcomes and are depicted as tuples of utility values for each player N

A \ B	Effective	Defective
Effective	**(-1, -1)**	**(-4, 0)**
Defective	**(0, -4)**	**(-3, -3)**

Figure 1: TCP Strategic Game

In the above example, both players have a dominant strategy, which is to choose the effective implementation, regardless of the other player's choice. This is because, no matter what the other player does, choosing the effective solution always results in a higher payoff for each player individually.

3.2 Best Response, Nash Equilibrium, Strict Domination

Nash equilibrium serves as a pivotal concept in game theory, and helps in predicting, analyzing, and understanding strategic interactions among rational decision-makers (players). John Nash in [10] defines Nash equilibrium as a profile (set) of strategies in which no player has an incentive to deviate from his/her chosen strategy, while taking into account the possible actions of all the other players. Nash equilibrium is valuable because it offers a fundamental framework for grasping strategic decision-making processes. This comprehension helps in the creation of more effective and dependable systems across diverse domains such as cybersecurity in our case. In detail:

3.2.1 Best Response (BR)

If there was knowledge and information of what every player is going to choose, in a game, it would be easy to choose our own actions, as well. Let:

$$a_{-i} = < \alpha_{1,}, \dots, \alpha_{i-1}, \alpha_{i+i,}, \dots, \alpha_n > \ and\ \alpha = (\alpha_{-i}, \alpha_i)$$

A strategy α_i is a best response for player i to a profile of strategies $\alpha_{-i} \in \alpha_{-i}$ for the other players if:

$$u_i(\alpha_i, \alpha_{-i}) \geq u_i(\alpha'_i, \alpha_{-i}), \forall\, \alpha'_i$$

A best response of player i to a profile of strategies of the other plays is strict best response if it is also the only best response.

3.2.2 Nash Equilibrium (NE)

A profile of strategies $\alpha \in A$ is a pure strategy Nash Equilibrium if α_i is best response to the set $\alpha_{-i}\ \forall\ i$. Meaning that α is a NE if:

$$u_i(\alpha_i, \alpha_{-i}) \geq u_i(\alpha'_i, \alpha_{-i}), \forall\, i, \alpha'_i$$

The difference is the definition of NE with respect to that of BR where in a pure strategy NE, each player's action is optimal considering the actions of other players at equilibrium, rather than considering all potential actions those players might take. As the authors in [11] state, sometimes, a player's best response to the actions of others is singular. When all players are playing unique best responses, it constitutes a strict NA. It is also highlighted that a profile of dominant strategies qualifies as a Nash equilibrium, but the reverse isn't always true.

3.2.3 Strict Domination (SD)

Let S_i and S'_i be two strategies for player i, and let S_{-i} be the set of all possible strategy profiles for other players. S_i strictly dominates S'_i if:

$$u_i(S_i, S_{-i}) > u_i(S'_i, S_{-i}), \forall\, s'_i \in S'_i$$

Only if:

$$u_i(S_i, S_{-i}) \geq u_i(S'_i, S_{-i}), \text{ implying weak domination}$$

A strategy profile consisting of dominant strategies for every player must be a NE. An equilibrium in strictly dominant strategies must be unique

3.3 Pareto Optimality

From the point of view of an outside observer, some outcomes of a game can be said to be better than others. It can’t be said that one player’s interests are more important than those of others. Sometimes players try to find the revenue that maximizes the outcome when it’s not known what currency is used to express the players’ payoff. For that reason, in [12] it is explained how an outcome o is at least as good for every player as another outcome o' and there are some players who strictly prefer o to o'. In this case, it is reasonable to say that o is better than o'. The state-of-the-art defines the previous case as the Pareto-domination of outcome o to outcome o'. Thus, an outcome $o*$ is Pareto-optimal if there is no other outcome that Pareto-dominates it.

3.4 Mixed Strategies and Mixed Strategy NE

There are many applications and situation in game theory, where players may not have complete information about their opponent’s strategies and most importantly there is no unique pure strategy Nash equilibrium for the participating players. Furthermore, pure strategies can lead to predictable behavior, which can be exploited by other players in order to win. For that reason, Nash and Morgenstern introduce the concept of mixed strategies to ensure that players can address this uncertainty by randomizing their actions, reflecting the lack of information and to break symmetries in games that contain by nature symmetrical payoffs, preventing players from exploiting predictable patterns and/or symmetrical advantages [13]. In detail, the state-of-the art defines a mixed strategy s_i for player I as any probability distribution over the actions A_i. In pure strategy, only one action is played with positive probability but in mixed strategies more than one actions are played with positive probability. The actions in mixed strategies are called the support of mixed strategy.

- Let the set of all strategies for i be S_i
- Let the set of all strategy profiles be $S = S_1 \times \ldots \times S_n$

To define a mixed strategy, the expected utility from decision theory is used [11], [14]:

$$u_i(s) = \sum_{\alpha \in A} u_i(\alpha) Pr(\alpha|s)$$

, where:

$$Pr(\alpha|s) = \prod_{j \in N} s_j(\alpha_j)$$

A mixed strategy for a player i is a distribution s_i on α_i, where $s_i(\alpha_i)$ is the probability that α_i is chosen. A profile of mixed strategies constitutes a mixed-strategy NE if:

$$\sum_{\alpha \in A} u_i(\alpha) \prod_{j \in N} s_j(\alpha_j) \geq \sum_{\alpha \in A} \left(\prod_{j \neq i} s_j(\alpha_j) \right) u_i(\alpha'_{i,}, \alpha_{-i}) \qquad , \forall\, i', \alpha'_i$$

A set of mixed strategies forms an equilibrium when no player can improve their payoff by deviating from their chosen mixed strategy in response to the mixed strategies of other players. This implies that each player must be equally satisfied with every strategy they employ with a positive probability within their mixed strategy. Additionally, players' randomizations are done independently. However, due to the mixed strategy practically be probability distribution, when a strategy happens with probability equal to 1, then that strategy is a pure strategy.

3.5 Computing a Nash Equilibrium

The approach described in Section 5 assumes that the game is played with 2 players: an attacker and a defender. For that reason, this section will consider solutions for solving only 2 player games. The first solution that is described here is formulated as a Linear Complementarity Problem (LCP) [15]pap. Precisely:

- s_n: $mixed\ strategies\ for\ players\ A_1\ or\ A_2$
- r_n: $slack\ variables$
- u_i: $is\ the\ payoff\ to\ player\ 1\ in\ NE$

Where the player 1 expected reward can be calculated as:

$$u_1^* = \sum_{k \in A_2} u_1(\alpha_1^j, \alpha_2^k) \cdot s_2^k + r_1^j \quad , \forall\, j \in A_1$$

And player 2 respectively:

$$u_2^* = \sum_{j \in A_1} u_2(\alpha_1^j, \alpha_2^k) \cdot s_1^j + r_2^k \quad , \forall\, k \in A_2$$

Considering that that the probabilities sum to one: $\sum_{j \in A_1} s_1^j = 1\ and\ \sum_{k \in A_2} s_2^k = 1$

, where:

$$s_1^j \geq 0, s_2^k \geq 0, \ \ \forall\, j \in A_1, \forall\, k \in A_2$$

$$r_1^j \geq 0, r_2^k \geq 0, \ \ \forall\, j \in A_1, \forall\, k \in A_2$$

$$r_1^j \cdot s_1^j \geq 0, r_2^k \cdot s_2^k \geq 0, \ \ \forall\, j \in A_1, \forall\, k \in A_2$$

3.6 Maxmin and Minimax Strategies

Maxmin and minmax are important concepts in game theory because they provide strategies that players can use to optimize their outcomes in competitive settings. Particularly, in the context of zero-sum games (Subsection 3.9) [16], where one player's gain is directly affected by another player's loss, the maxmin and minmax strategies help players minimize their maximum possible loss and maximize their minimum guaranteed gain, respectively. These strategies are particularly useful when players have incomplete information about their opponents' strategies or when they seek to minimize risk.

3.6.1 Maxmin

A player's i's maxmin strategy is a strategy that maximizes i's worst-case payoff, in the situation where all other players (-i) happen to play their strategies that cause the greatest harm to player i. The maxmin strategy for player i is:

$$arg \max_{s_i} \min_{s_{-i}} u_i(s_i, s_{-i})$$

And the maxmin value for player i is defined as:

$$\max_{s_i} \min_{s_{-i}} u_i(s_i, s_{-i})$$

3.6.2 Minmax

A player's i minmax strategy against player $-i$, in a 2-player game, is a strategy that minimizes $-i\,'s$ best case payoff and the minmax value for i against $-i$ is payoff. The minmax strategy for player i against player $-i$ is defined as:

$$arg \min_{s_i} \max_{s_{-i}} (s_i, s_{-i})$$

And the minmax value for player i is defined as:

$$\min_{s_i} \max_{s_{-i}} (s_i, s_{-i})$$

3.7 Zero-Sum Games

The game that this thesis will use a test-case is zero-sum game. A zero-sum game is a type of situation in game theory where one participant's gain or loss is exactly balanced by the losses or gains of the other participant(s). In other words, the total utility or value remains constant among the participants. The term "zero-sum" comes from the fact that when you add up the gains and losses of all players, the total sum is zero. This doesn't mean that there's no value created overall, but rather that any value created is simply redistributed among the participating players. Von Neuman in 1928 [8], claimed and then proved that in any finite, two player zero-sum game, in any NE, each player receives a payoof that is equal to both his maxmin and minmax value.

For a 2-player game, minimax is solvable with linear programming approaches as described in [16], where the payoff to player N_1 in equilibrium is $\boldsymbol{u}_1^*$ and the scope is to minimize it, subject to:

$$\sum_{k\in A_2} u_1(\alpha_1^j, \alpha_2^k)\cdot s_2^k \le \boldsymbol{u}_1^* \quad ,\forall\, j \in A_1$$

Considering that that $\sum_{k\in A_2} s_2^k = 1$ and $s_2^k \geq 0,\ \forall\, k \in A_2$

<u>Example:</u>

One of the easiest to understand examples is the Attack-defend problem but in this example, it has cybersecurity twist to capture the basic notion. In this game, we consider two players: the Defender (Player 1) and the Attacker (Player 1). The company has to decide how much to invest in cybersecurity defenses, while the hacker must decide whether to attempt a cyberattack or not.

P1 \ P2	Attack	No Attack
Invest Low	**(0.6, 0.4)**	**(0.8, 0.2)**
Invest High	**(0.9, 0.1)**	**(0.7, 0.3)**

Figure 2: Example CyberAttack Strategic Game

In the above example, we demonstrate how the defender maximizes his minimum, as follows:

$$\max_{s_i}\min_{s_{-i}} u_i(s_i, s_{-i})$$

$$\rightarrow \max_{s_1}\min_{s_2}[s_1(I_L)s_2(A)0.6 + s_1(I_L)s_2(A')0.8 + s_1(I_H)s_2(A)0.9 + s_1(I_H)s_2(A')0.7]$$

, where:

- $s_1(I_L)$ = prob that attacker attacks the target system
- $s_2(A\ \ )$ = prob that attacker attacks the target system

The minimum that the defender should keep in mind is:

$$\min_{s_2}[s_1(I_L)s_2(A)0.6 + s_1(I_L)(1 - s_2(A))0.8 + (1 - s_1(I_L))s_2(A)0.9 \\ + (1 - s_1(I_L))(1 - s_2(A))0.7]$$

$\Longrightarrow\ s_1(I_L)\ =\ ½\ ,\ s_1(I_H)\ =\ ½$

Similarly, if the attacker wants to minimize the defender's maximum utility:

$\min_{s_i} \max_{s_{-i}}(s_i, s_{-i})$

$\rightarrow\ \min_{s_1} \max_{s_2}[s_1(I_L)s_2(A)0.6 + s_1(I_L)s_2(A')0.8 + s_1(I_H)s_2(A)0.9 + s_1(I_H)s_2(A')0.7]$

$\Longrightarrow \qquad \ldots \qquad \Longrightarrow s_2(A)\ =\ 1/4\ ,\ s_2(A')\ =\ 3/4$

Sorin, Sylvain, (1992), Repeated games with complete information, ch. 04, p. 71-107 in Aumann, R.J. and Hart, S. eds., Handbook of Game Theory with Economic Applications, vol. 1, Elsevier.

Chapter 4: Modeling for the Honeypot Allocation Game

The application of many Cyber Deception games is considered a growing topic in the current state of the art of cybersecurity games. Overall, cyber deceptions games, whether they are expressed as game theoretic or AI models, require a formal and concrete modeling of the scenario which portrays their application. Modeling usually refers to the expression of the environmental attributes as a connected multi-layer (and often hierarchical) composition of entities, actions, states, and results. The previous will be formalized in later sections. In our case, the model operates within the scope of deceiving one of the entities, leading to an equilibria state in which neither actor (entity) can get "on top of the other".

This Chapter will initially explain how we formulate the game over attack graphs. Following, a brief explanation of the usage of attack graphs will be conducted. Attack Graphs in this specific implementation depict models that are used to conduct vulnerability root-cause analysis. To quantify the vulnerabilities' capabilities on vulnerability attack graphs, a short introduction will be made on the CVSS metrics which are a widely known framework used especially in information security risk management. As the previous, constitute the theory needed for the creation of the attack graphs and the vulnerability interconnections, the rest of this chapter focuses on creating a custom attack graph model with a domain specific language, called Meta-Attack Language. The model is defined on a set of rules which are described in the last part of this section and then we use a proposed engine to generate the attack graph model that will be used for the Optimal Honeypot allocation, as seen in Chapter 5.

4.1 General Game Formulation

In detail, the Cyber Deception of game of optimal Honeypot placement which is described in this Master Thesis is the practical implementation of [17]which is enriched and applied on top of proper AG models. The described game theoretic framework is applied on top of network security graphical models and specifically in attack graphs. Attacks graphs are widely known for being an efficient way to model and explain multi-stage attacks. During the years multiple attack graphical models and types have been developed in order to accurately model different situations, environments, topologies, attack methods, etc.

4.2 The Common Vulnerability Scoring System

CVSS offers three primary advantages compared to other scoring systems. Firstly, it operates within an open framework, providing daily updates for existing entries as well as incorporating new ones. Secondly, vulnerability scores are standardized for both open source and commercial platforms, ensuring consistency across different systems. Major vulnerability databases like the National Vulnerability Database (NVD) integrate CVSS metrics into their feeds. Additionally, adopting a common scoring algorithm facilitates the establishment of a unified vulnerability management policy within organizations. Lastly, CVSS facilitates risk prioritization by enabling the computation of environmental scores (ES) for a more comprehensive understanding of overall risk. It achieves this through three sets of metrics: base, temporal, and environmental metrics.

The assessment of overall risk associated with identified vulnerabilities in an IT system, including the likelihood of exploitation and the impact of successful exploitation, is quantitatively measured using industry-standard vulnerability scoring systems. These systems, managed by various commercial and non-commercial entities, each offer unique advantages. Differences exist in what they measure and the ranges of scores they employ. For instance, SANS Institute's scales consider factors such as default configurations and server systems, while Microsoft's system reflects the level of exploitation and total impact of a vulnerability [18]. FIRST notes that while these systems are useful, they adopt a one-size-fits-all approach by assuming constant vulnerability impacts across individuals and organizations [19].This section focuses solely on the Common Vulnerability Scoring System 3.1 standard (CVSS), which provides a measure of vulnerability criticality to prioritize risk mitigation efforts.

Nevertheless, this section will focus on the Base Score and Metrics of the CVSS, as Chapter 5 utilizes these metrics to fill the reward matrix that will be defined.

4.2.1 Base Score Metrics

Precisely, base metrics constitute of two sub metric groups, namely the Exploitability metrics and the Impact Metrics. The exploitability metrics focus on assessing the exploitability of a vulnerability, consider characteristics of the vulnerable component and how they contribute to the potential of the successful exploitation, while impact metrics help the understanding of the potential consequences of the vulnerability being exploited.

Table 2: Exploitability Metrics and corresponding values

Exploitability Metric	*Value*
Attack Vector (AV)	• **Network (N)** – This value indicates that the vulnerability can be exploited remotely over a network connection, without requiring any prior access to the target system. An attack can exploit the vulnerability from anywhere, provided that he/she can reach the target system over the network. • **Adjacent (A)** – This value indicates that the vulnerability can be exploited by an attacker has access to the same network segment as the one in which the target system resides. This includes attack in which both the attacker and the target are in the same wireless network or LAN. • **Local (L)** – This value indicates that the attacker has limited access to the target system. This may be that the attacker exploits the vulnerability by having access to the target system through a keyboard or a console or has remote access

	through an SSH connection. This value may also imply that the attacker requires further actions to be conducted from other external entities in order to successfully exploit the vulnerability. • **Physical (P)** – This value indicates that the attacker has physical access to the vulnerable component.
Attack Complexity (AC)	• **Low (L)** – This value indicates that there are no particular access conditions that must be achieved first in order to attack the vulnerable component and that the exploitation has a deterministic nature. • **High (H)** – This value indicates that the attack relies of factors outside of the attacker's immediate control, leading to a certain level of effort to be applied or preparation**.** Such effort may involve acquiring knowledge about the target environment, a preliminary exploitation, other types of network attacks and/or overcoming obstacles that are prerequisites for the exploit such as a race condition.
Privileges Requires (PR)	• **None (N)** – This value indicates that the attacker does not require to be authorized in order to attempt the vulnerability exploitation nor have any access to settings, files or rights in the system. • **Low (L)** – This value indicates that the attacker must have user capabilities or rights to access

	files in the target system and is limited to only non-sensitive resources. • **High (H)** – This value indicates the attacker requires admin privileges or other significant control over files or executables and that the exploitation leads to getting access to most setting and files.
User Interaction (UI)	• **None (N)** – This value indicates that the target system can be exploited without any interaction form other users. • **Required (R)** – This value indicates that the vulnerability exploitation requires other users to conduct some actions before the exploitation takes place.

The following table maps the previous qualitative Metric Values to quantitative metrics, as FIRST indicates in the CVSS specification [19]:

Table 3: Exploitability Metrics mapping to their quantitative values

Exploitability Metric	***Mapping to Quantitative values***
Attack Vector (AV)	Network (N) → $\mathbf{0.85}$ Adjacent (A) → $\mathbf{0.62}$ Local (L) → $\mathbf{0.55}$ Physical (P) → $\mathbf{0.20}$
Attack Complexity (AC)	Low (L) → $\mathbf{0.77}$ High (H) → $\mathbf{0.44}$
Privileges Requires (PR)	None (N) → $\mathbf{0.85}$ Low (L) → $\mathbf{0.62}\ /\ \mathbf{0.68},\ if\ \boldsymbol{Scope}^{2}\ is\ Changed$

[2] The Scope metric in vulnerability assessment refers to the extent to which a vulnerability in one component can affect resources beyond its immediate security boundaries. It evaluates potential impact

	High (H) → $\mathbf{0.27}$ / $\mathbf{0.50}$, if $\boldsymbol{Scope}$ is $Changed$
User Interaction (UI)	None (N) → $\mathbf{0.85}$
	Required (R) → $\mathbf{0.62}$

4.2.2 Impact Metrics

Impact metrics like confidentiality (C), integrity (I), and availability (A) measure the immediate consequences of an exploit. Confidentiality relates to how much sensitive information could be revealed, integrity concerns the potential damage to data accuracy or trustworthiness, and availability deals with the accessibility of information resources during an attack. Higher impact levels, designated by "H" for high, "L" for low, and "N" for none, correspond to numerical values of 0.56, 0.22, and 0.00, respectively, leading to an increase in the Base Score (BS).

Table 4: Impact Metric mapping to their quantitative values

Impact Metric	***Mapping to Quantitative values***
Confidentiality (C)	None (N) → $\mathbf{0.00}$
	Low (L) → $\mathbf{0.22}$
	High (H) → $\mathbf{0.56}$
Integrity (I)	None (N) → $\mathbf{0.00}$
	Low (L) → $\mathbf{0.22}$
	High (H) → $\mathbf{0.56}$
Availability (A)	None (N) → $\mathbf{0.00}$
	Low (L) → $\mathbf{0.22}$
	High (H) → $\mathbf{0.56}$

4.2.3 Environmental Score Metrics – Security Requirements

These metrics enable adjustments to the base and temporal severity levels of. The CVSS within an organization's environment, reflecting the distinct characteristics of that environment. Additionally, they take into account the significance of a vulnerable

of a vulnerability on interconnected systems, networks, or components that may not be directly related but could be otherwise inderectly affected.

system within an infrastructure, considering factors such as the presence of security mechanisms and mitigation measures that could hinder or mitigate potential attacks.

Chapter 5 utilizes the Security Requirements of the Environmental Metrics to define the defender capture reward. Security Requirements are impact metrics that are established by the system administrator (in this thesis' case; the defender) in response to the specific needs arising within the target vulnerable system. In Chapter 6, these metrics are defined for the whole simulation as parameters.

4.3 Creating the Attack Graph Model

Attack Graphs are widely used as a tool that initially heled in visualizing cyber-attacks in on computers or complex ICT systems. Nowadays, attack graphs fall under the broader category of graphical security models and the main scope of usage lies in providing numerous benefits such as mitigation prioritization, understanding complex attack patterns, conducting probabilistic vulnerability risk analysis, moving target-based applications, intrusion response [20]. The bibliography gathers most of the applications under the following categories, as seen in Figure 3:

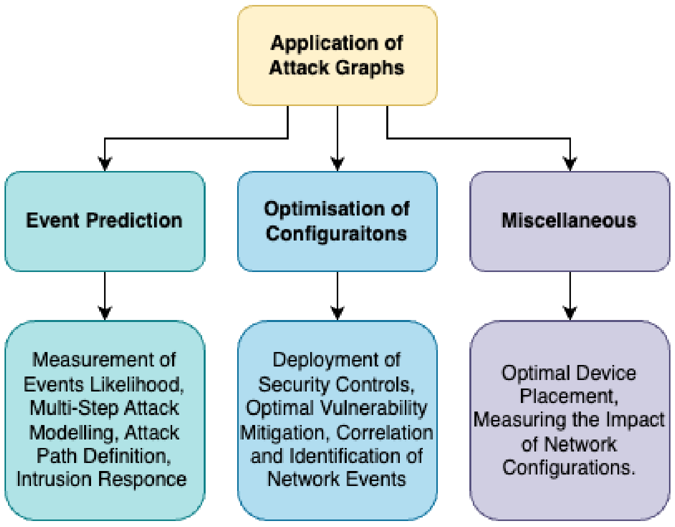

Application of Attack Graphs
Event Prediction
Optimisation of Configuraitons
Miscellaneous
Measurement of Events Likelihood, Multi-Step Attack Modelling, Attack Path Definition, Intrusion Responce
Deployment of Security Controls, Optimal Vulnerability Mitigation, Correlation and Identification of Network Events
Optimal Device Placement, Measuring the Impact of Network Configurations.

Figure 3: Attack Graphs and their applications

While there are many attack graphs representations, for this thesis we will follow the basic notations for creating graphical models as they have originally been described. A basic structure of a graph G is represented in the form: $G = (V, E)$, which comprises of vertices $v \in V$, and edges $e \epsilon E$, representing relationship between nodes in the graph structure. Particularly in Attack Graphs, there is a more complex structure that is expressed in the form of $G = (\{V_{pre}, V_{post}, V_{exploits}\}, E)$, where vertices, also referred as nodes, can be represented as three different types:

- V_{pre} is a set of nodes, representing exploit pre-conditions,
- V_{post} is a set of nodes, representing exploit post-conditions (they can be interpreted as derivations of exploit-related actions):

$$V_{post} \subseteq V_{pre} \mid \forall v \epsilon V_{post}, V_{pre}$$

- $V_{exploits}$ is set of nodes representing the exploits that occur from existing pre-conditions and have post-conditions as successors.

However, as the above definition in no way defines all the different attack graph models out in the wild, most available works capture this idea and built on top of it. In addition, such models are often visually represented as acyclic graphs. Some examples are attack trees, topological attack graphs, state-dependency attack graphs, Bayesian attack graphs.

State Enumeration Attack Graphs

A state attack graph [21][22] illustrates how a system progresses from one state to another by exploiting vulnerabilities. Each state, which can represent a host, privilege, exploit, or vulnerability, is depicted as a vertex, while connections between states are depicted as edges. Vertices are further classified into pre-conditions and post-conditions, delineating the stages before and after an attack. These pre and post conditions, along with their interconnected edges, form a directed attack graph. State-based attack graphs proliferate exponentially by detailing all possible combinations required for system compromise, without considering duplicate attack paths. However, it is stated that such models have significant issues with scalability as the number of nodes (states) grows exponentially since such graphical models do not capture and integrate the monotonicity assumption.

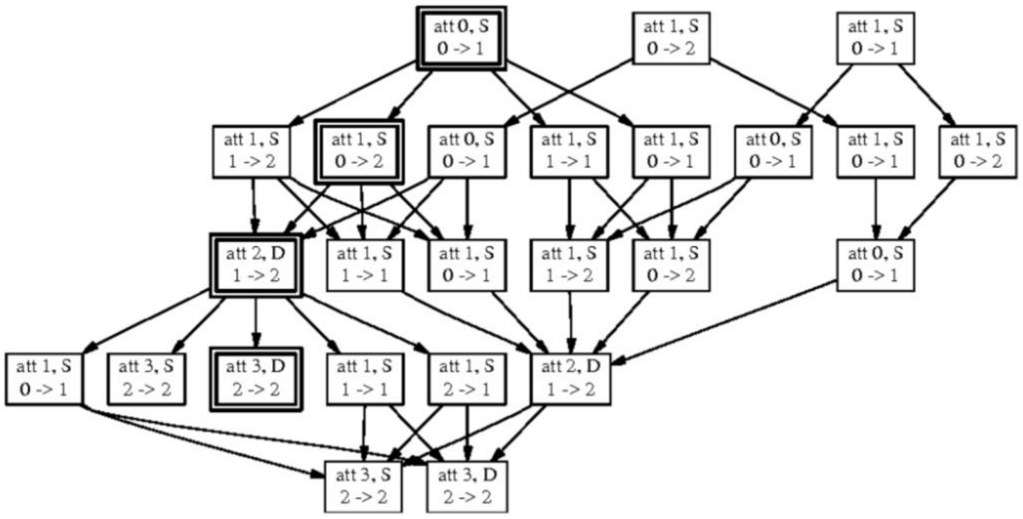


Figure 4: State Attack Graph from [21]

Exploit Dependency Graphs

The authors in [22] describe a framework that manages network attack graphs and their complexity by conducting hierarchical aggregation of the graph elements (nodes and vertices). The aggregation behaves recursively and practically operates by collapsing non-overlapping subgraphs of the initial model into a single attack graph, thus reducing the final complexity of the model. The final graph model is called an Exploit Dependency attack graph and inherit all the information from state-enumeration graphs (as they are considered an improvement over the previous) [21]. Exploit Dependency Attack Graphs contain all the required information for network vulnerability and attack analysis and are also exhaustive in terms of attack paths. The specific graphical model captures the attack graph representation of State Enumeration Attack graphs but instead transition to a more exploit-oriented representation. Exploits and conditions are aggregated to exploits sets and condition sets and vice versa**.** Moreover, set of exploits and conditions nodes are aggregated into abstract machines-exploits sets which are called protection domains and these are represented a fully connected subgraph models which represent the exploit dependency attack graph when merged altogether. An indicative example of an exploit dependency graph can be seen below:

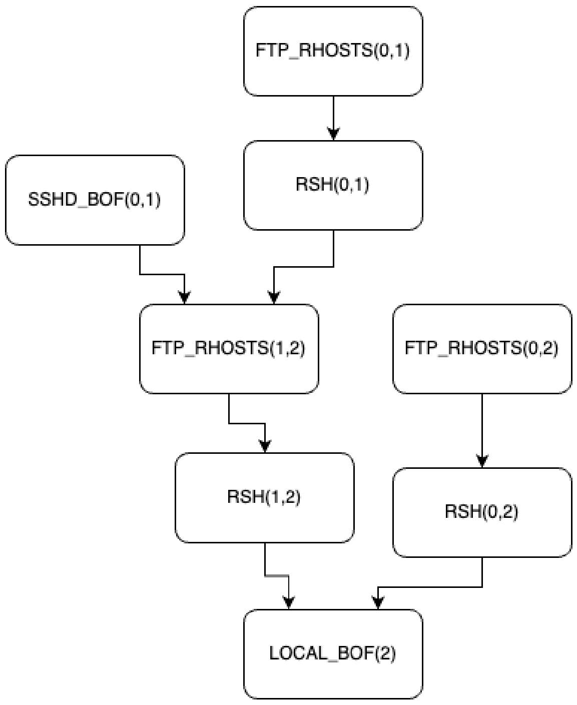


Figure 5: Example Exploit Dependency Attack Graph

4.3.1 Meta Attack Language

In order to create an attack graph model, we utilize the Meta Attack Language framework [23] which the authors introduce as a meta-domain specific language (DSL) for modeling infrastructures with domain-specific threats.

MAL Combines elements of UML classes and object diagrams with probabilistic attack-defense graphs. The language uses *Assets* which can be thought as grouping of classes that contain instances and represent objects. Assets can also be interpreted as set of nodes in an attack graph. Those subgraphs (Assets) relate to other subgraphs via *Associations.* Associations define relationships that are similar to class diagrams and feature roles and multiplicities. Furthermore, the language incorporates the subgraph logic behind Assets in the form of attack steps, following at the same time the formal definition of attack graphs from the bibliography. A MAL DSL enables the encoding of attack tress in assets.

Assets and Associations are defined in a “.mal” file which must follow a specific syntax. To compile a .mal file, the MAL compiler[3] must be used. The compiles uses Maven[4] to download all supplementary packages and but is easily accessible from the Github repository. In our case, we use the compiler without any modifications. More details are described in in the next section..

To showcase the language structure, the *exampleLang*[5] is presented below:

Table 5: ExampleLang syntax - Association Rules and Assets

```
/*
 * Copyright 2020-2022 Foreseeti AB <https://foreseeti.com>
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     https://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
#id: "org.mal-lang.examplelang"
#version: "1.0.0"

category System {
  asset Network {
    | access
      -> hosts.connect
  }
  asset Host {
    | connect
```

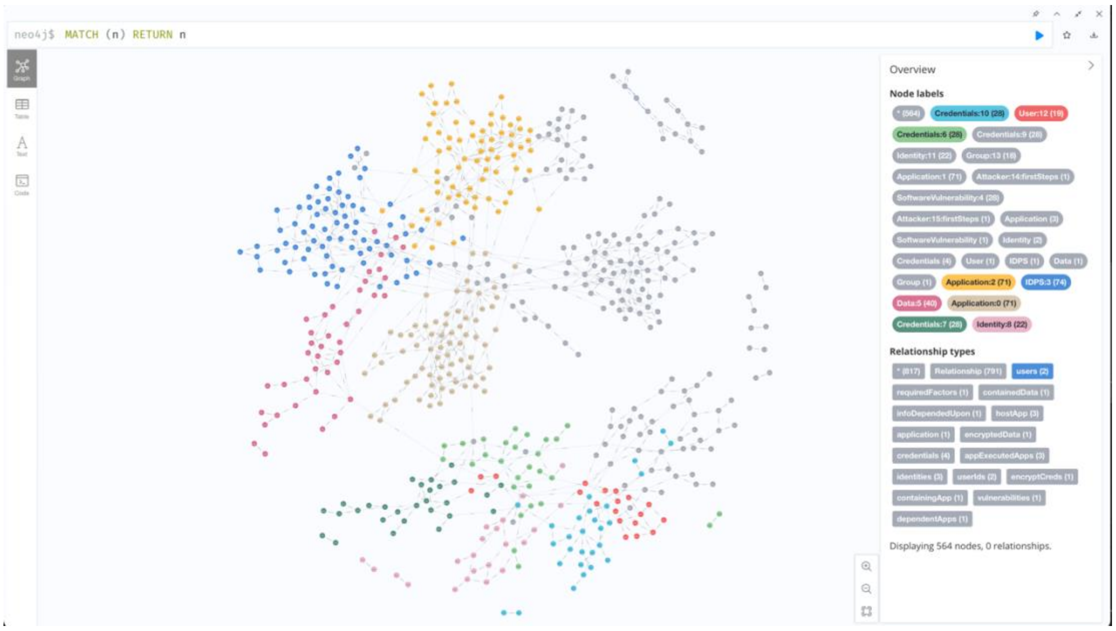
[3] https://github.com/mal-lang/malcompiler
[4] https://maven.apache.org
[5] https://github.com/mal-lang/exampleLang/blob/master/src/main/mal/exampleLang.mal

```
      -> access
    | authenticate
      -> access
    | guessPassword
      -> guessedPassword
    | guessedPassword
      -> authenticate
    & access
  }
  asset User {
    | attemptPhishing
      -> phish
    | phish
      -> passwords.obtain
  }
  asset Password {
    | obtain
      -> host.authenticate
  }
}
associations {
  Network [networks] * <-- NetworkAccess --> * [hosts] Host
  Host [host] 1 <-- Credentials --> * [passwords] Password
  User [user] 1 <-- Credentials --> * [passwords] Password
}
```

The above example starts by defining the category of the assets. The category practically acts as a wrapper to the assets that will be used together and also specifies which assets can be connected with the associations. In the **exampleLang** example, there are four assets: i) Network, ii) Host, iii) User and iv) Password. Each asset is operated as an attack path or a subgraph and the existing vertex within can show transitions other assets. For example, the Network asset contains a vertex “**access”** which, when created, initiates a connection with the **“connect”** node of the Host asset. Each vertex is characterized by a logical operator; “|” for the the **“OR”** logical operation and **“&”** for the **AND** logical operation.

- The **OR** logical operator allows connections to that node when at least one precondition is created from another asset.
- The **AND** logical operator allows connections to that node when all preconditions are met from either the same asset and/or when other assets are present. The present assets must lead to that specific node in the AND node, as well. Consider the example in Host -> & access. In that case, to create a node **“access”** with the AND logical operator, both nodes **“connect” & “authenticate”** must be present in the current asset.

As it has already been mentioned, each association in the Associations part of the code represent allowed connections between nodes. To define an association, the Class (Name) of the association exist in title case and have arrows towards the entities to be connected. The first present association explains that a **“NetworkAccess”** type connection is allowed between the **“Network”** and the **“Host”** assets. To do so, the associations define the variable which is shown in the square brackets for each of the assets and this variable name must also be present in the asset description when a connection is initiated to an another asset (e.g. “hosts.Connect”). To define the connection between assets, MAL uses “*” and “1” character to express associations relationships. Allowed relationships are one-to-one, many-to-many and one-to-many. The aforementioned process must be done for all associations in order to be able to further develop the MAL graph.



```
neo4j$ MATCH (n) RETURN n
```

Displaying 564 nodes, 0 relationships.

Figure 6: Indicative MAL Attack Graph in Neo4J

4.3.2 ViolenceLang

The attack model aims to capture a higher level of abstraction comparing to the original capabilities of MAL. For that reason, it is preferred to restrict the number of nodes in each asset and restrict the dynamic nature of the MAL framework. Such an approach is followed due to the reason that nodes and particularly the edges in Attack Graphs can scale exponentially while the number of hosts get bigger. However, the information depicted by the attack graph is expected to satisfy the needs of the models as the states described in the graph are going to represent the states a root-cause vulnerability analysis. Assets contain more than one vertex, in cases where the same action is described differently. For example, to capture the existence of different privileges, we define under the Privileges asset, both the rootPrivilegeEscalation and the UserPrivilegeEscalation vertices.

The following table shows the used assets used for building the core of the ViolenceLang:

Table 6: violenceLang Assets Definition

```
asset Scan
{
   & attemptReconnaiscance
     -> vulns.vulnerabilityExists
}

asset Vulnerability
{
   | vulnerabilityExists
     -> lexploits.localExploit,
        nexploits.networkExploit,
        aexploits.adjacentExploit
}

asset Network
{
   & networkExploit
     -> privesc.userPrivilegeEscalation,
        privesc.rootPrivilegesEscalation
}

asset Adjacent
{
   & adjacentExploit
     -> privesc.rootPrivilegesEscalation
}

asset Local
{
   & localExploit
     -> privesc.rootPrivilegesEscalation,
        insufs.insuficientAttack,
        ddosattacks.denialOfServiceAttack
}

asset Denial
{
   & denialOfServiceAttack
```

```
}

asset Privileges
{
   | rootPrivilegesEscalation
     -> compromises.hostCompromise
   | userPrivilegeEscalation
     -> lexploits.localExploit
}

asset Host
{
   & hostCompromise
     -> accesses.hostIsAccessible
}

asset Unsuccesfull
{
   | insuficientAttack
     -> mulattacks.complexAttack
}

asset Chain
{
   & complexAttack
     -> privesc.rootPrivilegesEscalation
}

asset Access
{
   | hostIsAccessible
     -> aexploits.adjacentExploit,
        scans.attemptReconnaiscance
}

asset Internet
```

```
  {
    # connectedToTheInternet
      -> nexploits.networkExploit
  }
```

The following table shows the associations used for building the core of the violenceLang:

Table 7: violenceLang Association Rules

```
associations {
    Scan [scans] 1 <-- ReconInit --> * [vulns] Vulnerability
      user info: "Reconnaissance -- Vulnerability Exists"

    Vulnerability [vulns] * <-- AvL --> 1 [lexploits] Local
      user info: "Vulnerability Exists -- Local Exploit"
    Vulnerability [vulns] * <-- AvN --> 1 [nexploits] Network
      user info: "Vulnerability Exists -- Network Exploit"
    Vulnerability [vulns] * <-- AvA --> 1 [aexploits] Adjacent
      user info: "Vulnerability Exists -- Adjacent Exploit"

    Local [lexploits] * <-- AcHl --> 1 [insufs] Unsuccesfull
      user info: "Local Exploit -- Insufficient Attack"
    Local [lexploits] * <-- VaHL --> 1 [ddosattacks] Denial
      user info: "Local Exploit -- Insufficient Attack"

    Adjacent [aexploits] * <-- VcHA --> 1 [privesc] Privileges
      user info: "Adjacent Network Exploit -- Root Privilege Escalation"

    Network [nexploits] * <-- ViHN --> 1 [privesc] Privileges
      user info: "Network Exploit -- USER PRIV/ ROOT PRIV"
    Local [lexploits] * <-- ViHLN --> 1 [privesc] Privileges
      user info: "Local Exploit -- Root Privilegae Escalation"

    Privileges [privesc] * <-- Standard --> 1 [compromises] Host
      user info: "Root Privilege Escalation -- Host Compromise"
    Unsuccesfull [insufs] * <-- Complex --> 1 [mulattacks] Chain
      user info: "Insufficient Attack -- Complex Attack"


    Host [compromises] 1 <-- Reachability --> * [accesses] Access
      user info: "Host Compromise -- HostIsAccessible *** ON NEW HOST *****"


    Access [accesses] 1 <-- DoReconOnReachableHost --> 1 [scans] Scan
      user info: "HostIsAccessible -- Reconnaisscance"
    Access [accesses] 1 <-- CanExploit--> * [aexploits] Adjacent
      user info: "HostIsAccessible -- Adjacent Network Exploit"


    Internet [globalconnections] 1 <-- HasInternet --> * [nexploits] Network
      user info: "Connected to the internet -- Network Exploit"


    Chain [scans] 1 <-- multistage --> * [privesc] Privileges
      user info: "Connected to the internet -- Network Exploit"
}
```

The allowed associations between assets for the current graph environment can be seen in Figure 7.

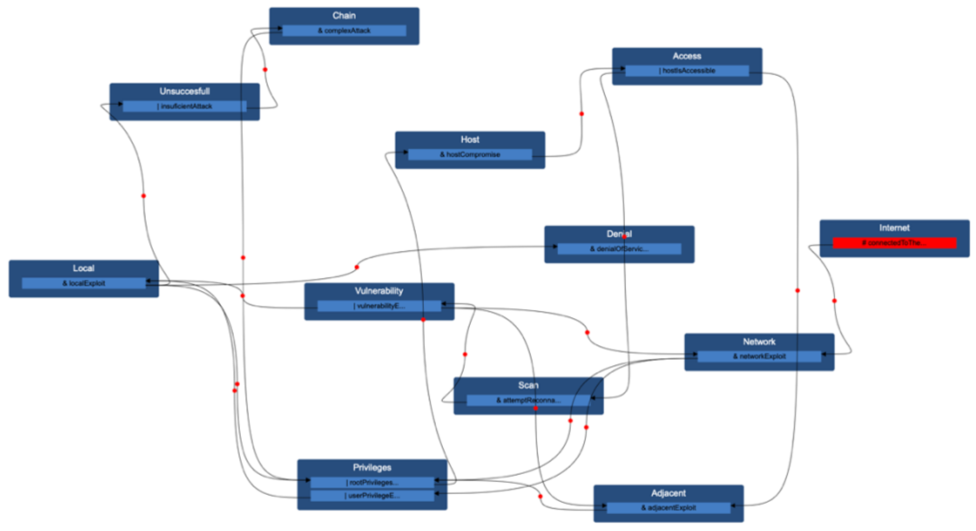


Figure 7: Association between assets in the violenceLang

4.2.3 Meta Attack Language Attack Graph Model Creation and Visualization

The generation of the MAL attack graph is a multi-step process. The high-level architecture that describes this process is depicted in Figure 8. In detail, to generate the Attack Graph model, a ". mal" file must be created according to the language described in Section 4.2.2. The previous compiles the MAL specification into ".java" source code files. When the compilation of the specification is finished, a .mar file (org.mal-lang.ViolenceLang-1.0.0.mar) is created in the local directory in which the command was executed.

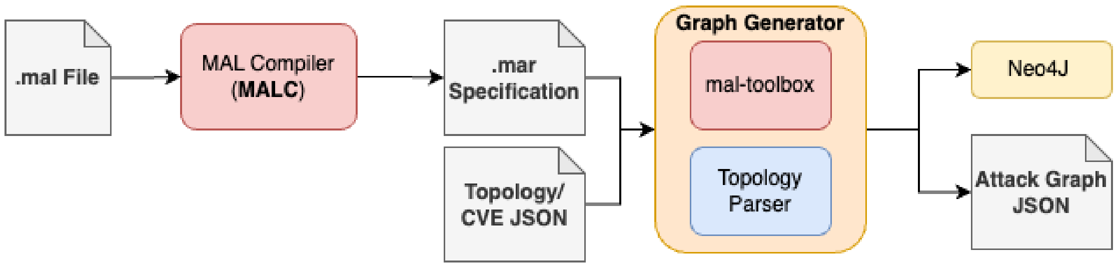


Figure 8: High-level overview of the MAL framework

The generated file is then fed to the Graph Generator module which utilizes the mal-toolbox[6] python library given by the mal-lang GitHub repository[7]. From the mal-toolbox module, the generator makes use of the Attack-Graph-MAL-Module which facilitates the generation and the analysis of attack graphs from MAL instance models. Specifically, the "generate_graph" function is used, which creates nodes for each attack step, given the instance module and the language specification. Node attributes such as children, parents and assets aid in referencing related nodes and associating attack steps with model objects. The mal-toolbox further allows the porting of the final graphical model to a neo4j[8] representation. Moreover, the generator outputs the attack graph in JSON format.

To create an attack graph based on the desirable infrastructure definitions, the generator uses a simple JSON file which defines the topology and the captured vulnerabilities for each host.

[6] https://github.com/mal-lang/mal-toolbox
[7] https://github.com/mal-lang
[8] https://neo4j.com

Table 8: Custom Example Input file for the MAL attack graph generator

```
{
   "configOrigins":"custom_example",
   "totalNumberOfHosts": N,
   "hosts":[
      {
         "id":1,
         "hostname": "Host 1",
         "vulnerabilities": [
         {
            "cve":"CVE-XXXX-XXXXX",
            "cvss":"CVSS:3.1/AV:X/AC:X/PR:X/UI:X/S:X/C:X/I:X/A:X"
         },
         {
            . . .

            Vuln 2, Vuln 3, …, Vuln N
            . . .

         }
         ],
         "connections":[{
            "source": 1,
            "destination": N
         },
         {
            "source": 1,
            "destination": N-x
         }]
      },
        . . .
       Host 2, Host 3, …, Host N
        . . .
    ]
}
```

The nature and the impact of the exploitation is portrayed from the CVSS string[9] of each CVE[10]. Meaning that in order to assess the exploitability of the vulnerabilities as a whole, a decomposition of the CVSS string is being conducted, breaking down the string into its constituent parts. The Base Score provides fundamental attributes of the vulnerability, such as exploitability and impact metrics. Metrics like Access Vector (AV), Access Complexity (AC), and Authentication (Au) gauge the ease of exploiting the vulnerability. A lower score indicates easier exploitation, while a higher score means more challenging exploitation conditions. The generator also takes into account the Impact metrics and specifically Confidentiality (C), Integrity (I), and Availability (A) to understand the severity of the vulnerability's consequences. Higher scores indicate more severe impacts on confidentiality, integrity, and availability, respectively. At the current development state, the MAL specification considers only vulnerability exploitations and not any other networks or host related attacks, such as credential brute-forcing, Man-in-the-Middle, DNS Spoofing, etc…

A 3-host example attack graph is presented in Table 9 and follow the MAL Specification of Section 4.2.2. Each color indicates the different type of vertex according to the specification.

Table 9: Custom 3-host Example Input for the MAL attack graph generator

```
{
   "configOrigins":"custom_example",
   "totalNumberOfHosts": 3,
   "hosts":[
      {
         "id":1,
         "hostname": "Host 1",
         "vulnerabilities": [
         {
            "cve":"CVE-2024-21851",
            "cvss":"CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H"
         },
```

[9] https://www.first.org/cvss/v3.1/specification-document
[10] https://cve.mitre.org

```
        {
          "cve":"CVE-2024-23109",
          "cvss":"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H"
        },
        {
          "cve":"CVE-2021-34527",
          "cvss":"CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"
        }
        ],
        "connections":[{
          "source" : 1,
          "destination": 3
        },
        {
          "source" : 1,
          "destination": 2
        }]
     },
    {
        "id":2,
        "hostname": "Host 2",
        "vulnerabilities": [
        {
          "cve":"CVE-2021-44228",
          "cvss":"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H"
        },
        {
          "cve":"CVE-2021-27190",
          "cvss":"CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"
        },
        {
          "cve":"CVE-2021-26084",
          "cvss":"CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H"
        }
        ],
        "connections":[
        {
            "source" : 2,
            "destination": 3
        }
        ]
    },
    {
        "id":3,
        "hostname": "Host 3",
        "vulnerabilities": [
        {
            "cve":"CVE-2024-21851",
            "cvss":"CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"
        },
        {
            "cve":"CVE-2024-23109",
            "cvss":"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:H"
        },
        {
            "cve":"CVE-2021-34527",
            "cvss":"CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"
        }
        ],
        "connections":[]
    }

  ]
}
```

The following Figure presents the Attack Threat Correlation graph in Neo4j representation. This graph indicates all the allowed connection between created assets and the subgraph formation.

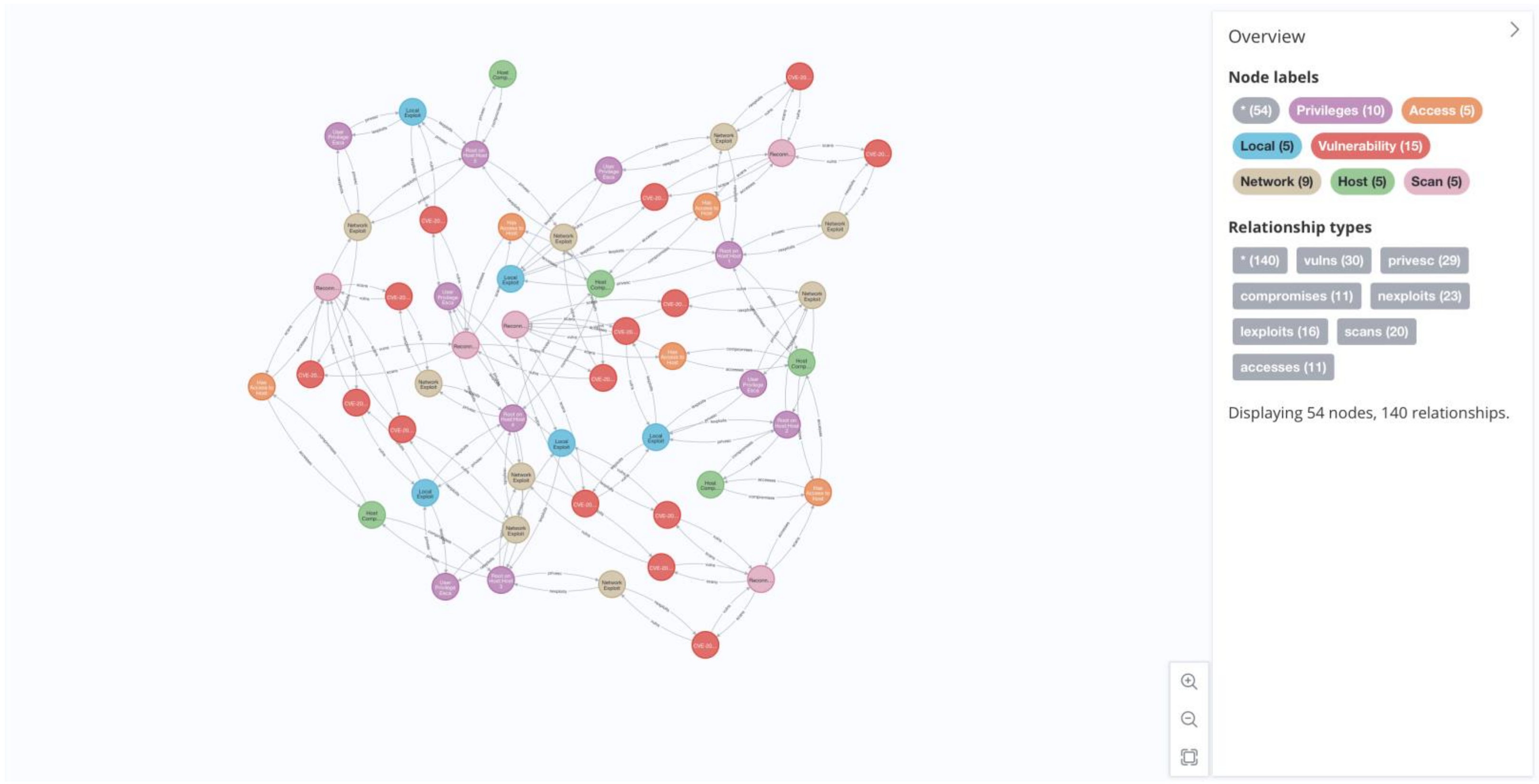


Figure 9: Attack Threat Correlation Graph

The following figure presents the Vulnerability Attack graph in Neo4j representation. This graph showcases the fully exploitable test network. Each small neighborhood that is formed in the attack graph is a host and the non-dense nodes indicate accessibility paths from each subgraph to another. The red-colored node represents the attacker entry point, and the rest of the big nodes represent ending points in the directed graph. Ending point signal that the attacker can no longer progress with her/his attacks. On the side of the Figure, the labels show the different created assets for each distinct host in the topology.

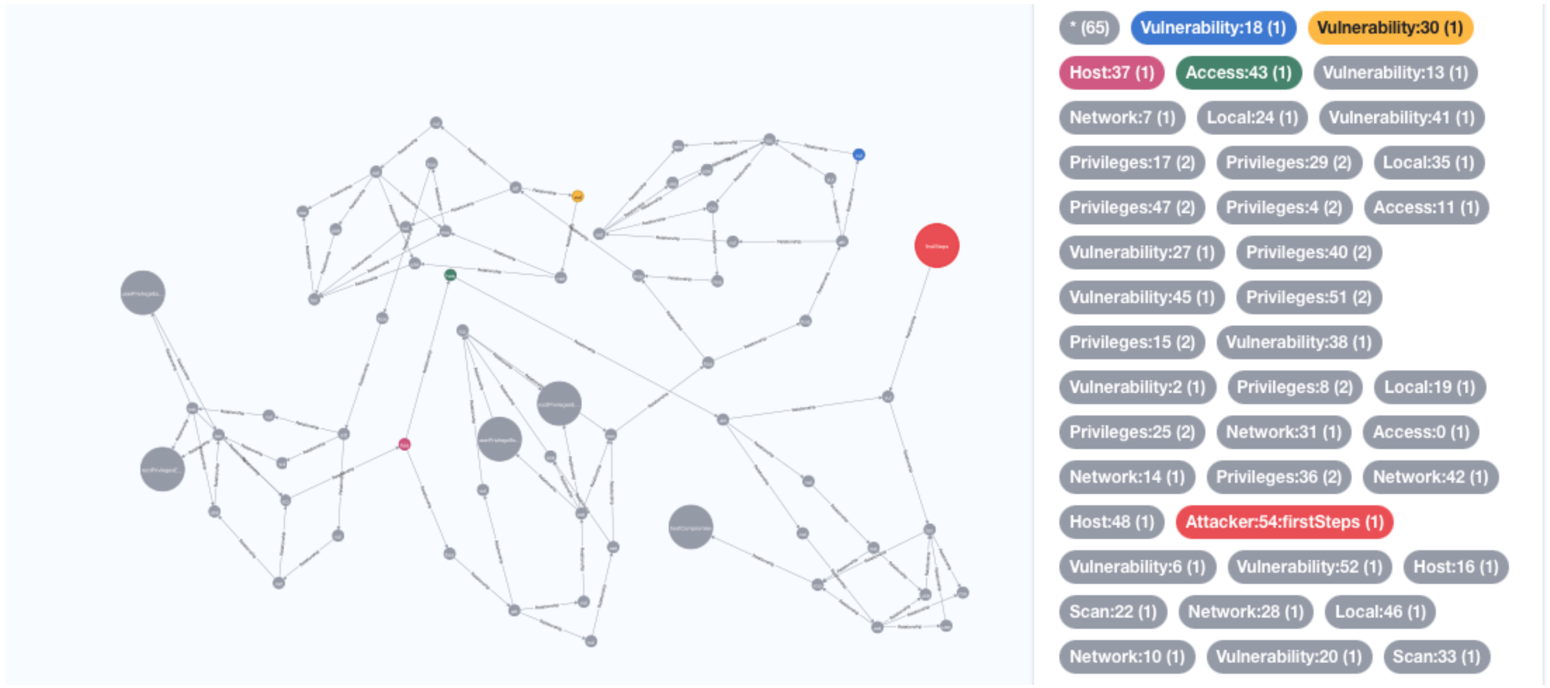


Figure 10: Example 5-Host Attack Graph

Chapter 5. Optimal Honeypot Allocation

5.1 Game Definition

This work considers the Attack Graph as defined in Chapter 4 for representing the attacker-defender game. In comparison to the original work in [17], this model takes into account multiple vulnerabilities that reside in many hosts, rather than assuming that each host in the target network has an exclusive vulnerability.

In contrast, to the original paper, where each node is assigned a weight $\boldsymbol{w_n}$, that indicates the importance of the host to the network administrator, this work assigns the weight $\boldsymbol{w_n}$ to a subgraph that represent a network machine. The network machine subgraph can potentially have the sets of edges and nodes were defined in Chapter 4 – Section 4.2. The nodes of a Host with higher importance are considered valuable asset to the network that contains important features and critical information, thus constituting a highly attractive target for an attacker. The attacker is a player who intends to maximize his expected reward by carefully selecting a victim host among the set of all reachable hosts.

A reachable considered a host which can logically by traversed and there exists a feasible attack path from the attacker's starting point. The attacker starting point as seen in Figure 10, is the red-colored node. For a host to be reachable, there must exist a host compromise chain from each host to another, creating a logical traversal towards attack goals. In the game, there is the assumption that the attacker has insider information regarding the importance of each subgraph (host) similar to the original work.

The defender has no full nor partial information about the attacker's whereabouts, so as to think of a realistic threat model, where the attacker is a sophisticated actor. However, the defender has access to the vulnerability attack graph mapping of his/her network and has complete knowledge on possible vulnerability-exploitation chains and paths that can happen in his system.

5.2 The Zero-Sum Game

For the game definition, this work follows the notation of the original work, slightly adjusted to match that of Chapter 3.

The Honeypot allocation game is defined as a two-player zero-sums strategic game $Z = (N, A, R)$, where:

- $N = \{1,2\}$ is the set of players. Player 1 is the network defender and player 2 is the network attacker.
- $A = A_1 \times A_2$ is action space of the game, where A_1 is the action space of the network defender and A_2 is the action space of the network attacker. The action profile is denoted by the tuple (a_1, a_2), being the joint action taken by the two players and determines their expected utility.
- $R = \{R_1, R_2\}$, where $R_1 + R_2 = 0 \; and \; R: A \rightarrow \mathbb{R}^2$ is the reward function for both players.

5.2.1 The Defender

The defender deploys a group of k honeypots along the attacker path to obstruct the attacker from reaching his/her target goals and to manipulate his/her actions. These honeypots represent with false vulnerabilities. When the attacker interacts with one of the honeypots, the defender gains knowledge of their current location and may gather valuable insights. This thesis’s focus lies on maximizing the defender's benefit derived from tracking the attacker in each instance of the game over the attack path.

Assuming the attacker initiates their attack from a captured node, the defender is faced with the option to deploy a single honeypot and thus must determine the optimal edge on which to place it. Given the uncertainty surrounding the attacker's precise location over the graph, the defender must also consider the option of taking no action to avoid incurring allocation costs.

5.2.2 The Attacker

The attacker tries to determine which host to target for his attack. The original model considers node instead, but the proposed change in the thesis consider that an attack plan will be based on multiple vulnerabilities. An Attacker must choose whether to exploit a vulnerability among one of all available "**access available**" edges. However, in order to not be detected from defensive security controls, the attacker has an expected cost which is associated with attempting each attack. This attack cost reflects the risk of detection with the attack action. As a result, the attacker may opt to avoid taking this cost, meaning that backing-off is also a viable choice.

5.3 Payoff Matrix Formulation

The network administrator is assumed to have a fixed cost for placing a honeypot each time over an *"access available"* edge in the attack graph. The honeypot placement host is denoted as $\boldsymbol{P_C}$. The expected cost of initiating a stealth attack by the attacker is denoted as $\boldsymbol{A_C}$.

If the defender has placed honeypot on the same edge the attacker exploits, the defender gains a capturing reward. Otherwise, if the attacker exploits the other safe edge, the attacker gains a successful attack reward. The defender capture reward is denoted as $\boldsymbol{Cap}$ and the attacker's reward when conducting a successful attack is denoted as $\boldsymbol{Esc}$.

As in the original work [17], the players' rewards are weighted by the importance fixed value w_n and:

$$\sum_{n \in N} w_n = 1$$

, where N is the number of hosts considered in the game. The reward matrix is defined as follows:

$$R_a^n(\alpha_1, \alpha_2) = \left\{ \begin{array}{cc} -P_C + A_C + Cap \cdot w_n \ ; & \alpha_1 = e_{\alpha,v}, \alpha_2 = v \quad \forall v \in V \\ -P_C + A_C - Esc \cdot w_n \ ; & \alpha_1 = e_{\alpha,v}, \alpha_2 = u \quad \forall\, u \neq v \in V \\ -P_C \ ; & \alpha_1 = e_{\alpha,v}, \alpha_2 = 0 \quad \forall\, v \in V \\ 0 \ ; & \alpha_1 = 0, \alpha_2 = 0 \end{array} \right\}$$

, where $\alpha_1 = 0$ and $\alpha_2 = 0$ indicate that the defender is not taking any action and the attacker decided to back-off respectively and $u, v \in V$ are nodes forming an edge in the subgraph of each network.

Due to the game being a zero-sum game, the attacker's payoff matrix is:

$$R_2 = -R_1 - 1$$

The metric values of the reward matrix are defined as follows (the values are normalized to be in the range of [0, 10]:

- $Ac = \frac{\sum_{v \in V_N} AV \times AC \times PR \times UI \times 21.10}{V_N}$
- $Esc = \frac{\sum_{v \in V_N} 1 - [(1-C) \times (1-I) \times (1-A)] \times 10.988}{V_N}$
- $Pc = [1, 10]$, defined by the defender
- $Cap = \frac{\sum_{v \in V_N} 1 - [(1-RC) \times (1-RI) \times (1-RA)] \times v}{V_N}$

, where $V_N \in V$ is the set of vulnerabilities on Host n and $v \in \mathbb{R}$ is a normalization factor based on the defender's choice of quantitative value allocation for Confidentiality Requirements, Integrity Requirements and Availability Requirements on target system.

5.4 A *Revised Simplex* Method Solver

To solve the zero-sum game, the method mentioned in Chapter 3 is implemented into code and to efficiently solve the minmax: the Revised Simplex solver for Linear Programming from the **scipy.optimize** Python package [24][25]. Below the implemented function is expressed in pseudo-code. The input is the payoff matrix as created from Section 3.7, regardless of the dimensionality feature. The solver used has no restrictions to the dimension of the array, but it is logical to assume that due to the inherent complexity of the linear solver, while the size of columns and rows is getting bigger, the algorithm is getting considerably slow.

Table 10: Pseudocode for the Linear Solver process

Step	*Action*
	Input: • Payoff matrix **Output:** • Optimal mixed strategy for **Player 1** • Optimal mixed strategy for **Player 2** • Value of the zero-sum game
1	Determine the number of strategies for each player by getting the number of rows in the payoff matrix
2	Define the objective function `**c**` to minimize the negative of the artificial variable **(-v)** appended to an array of zeros
3	Create inequality constraints `**A_ub**`:
3.1	For each strategy of Player 2, add constraint: $\boldsymbol{sum(Player\ 1\ strategy * payoff) - v \leq 0}$
3.2	Define **A_ub** by concatenating the negative of the payoff matrix and an additional column of ones representing **-v**
4	Define the right-hand side of the inequality constraints `**b_ub**` as a zero vector
5	Define the equality constraints `**A_eq`:**
5.1	Ensure that the sum of probabilities is equal to 1
5.2	Create `**A_eq**` by appending a row of ones so as to represent the sum of probabilities and a zero-representing **v**
6	Define the right-hand side of the equality constraint `**b_eq**` as in **Step 1**
7	Define bounds for each variable
7.1	Each strategy probability is **[0,1]**
7.2	**v** is unbounded

```
8     Solve the linear program using the Revised Simplex solver of scipy.optimize
9     Extract optimal strategies for Player 1 from the result
10    Compute value of the game as the negative of the last variable in the result
      (Zero-sum)
11    Solve the dual problem to find the optimal strategy for Player 2
11.1          Negate the coefficients and transpose the payoff matrix
11.2          Use the Revised Simplex solver as in Step 8
12    Exrtact optimal strategies for player 2 from the result
13    Return the optimal mixed strategies for both players and the value of the game
```

The linear solver **linprog** takes as arguments the following variables (the names of the of the class constructor are written as specified from the documentation of the library [25]:

- **c**: 1-D Array representing the co-efficient of the linear objective function that is going to be minimized.
- **A_ub:** 2-D Array representing the inequality constraint matrix
- **b_ub**: 1-D Array representing the constraint vector
- **A_eq**: 2-D Array representing the equaility constraint matrix
- **b_eq**: 1-D Array representing the constraint vector
- **bounds**: A Sequence of (min, max) tuples for each element, defining the minimum and the maximum values of that decision variable.
- **Revised Simplex (method)**: A string that is used for selecting the algorithm to solve the standard form problem.

5.5 Inherent Complexity

The complexity of solving the linear programs for both players increases linearly with the number of strategies they can choose at an individual node, which typically remains small compared to the overall network size. This is mostly achieved, due to the modelling and the rules defined in the attack graph model. This allows for efficient computation of optimal defense and attack strategies, ensuring Nash equilibrium. Nevertheless, it must also be noted that when there is a significant number of vulnerabilities residing on host, the complexity is not affected due to the formulation of global variables that capture all the exploitations. When the defender simultaneously allocates multiple honeypots to cover a set of edges, denoted by 'l', the complexity of the linear program for such scenarios grows exponentially with 'l'.

Chapter 6: An Example Demonstration

In this chapter, an example attack graph is created, in which the methodology presented in Chapter 5 is tested out. The attack graph assets can be seen in the Appendix of the Thesis. The attack graph defined is based on the 3-host example attack graph of Chapter 4 and is as lightly changed version of it with changes made by hand in the previous .json file. The topology is as follows:

```
{
  "configOrigins":"custom_example",
  "totalNumberOfHosts": 5,
  "hosts":[
    {
      "id":1,
      "hostname": "Host 1",
      "vulnerabilities": [
      {
        "cve":"CVE-2024-21851",
        "cvss":"CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"
      },
      {
        "cve":"CVE-2024-23109",
        "cvss":"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H"
      },
      {
        "cve":"CVE-2021-34527",
        "cvss":"CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"
      }
      ],
      "connections":[{
        "source" : 1,
        "destination": 3
      },
      {
        "source" : 1,
        "destination": 2
      }]
    },
   {
      "id":2,
      "hostname": "Host 2",
      "vulnerabilities": [
      {
         "cve":"CVE-2021-44228",
         "cvss":"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H"
      },
      {
         "cve":"CVE-2021-27190",
         "cvss":"CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"
      },
      {
         "cve":"CVE-2021-26084",
         "cvss":"CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H"
      }
      ],
      "connections":[
      {
         "source" : 2,
         "destination": 3
      }
      ]
   },
   {
      "id":3,
      "hostname": "Host 3",
      "vulnerabilities": [
      {
         "cve":"CVE-2024-21851",
         "cvss":"CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:H"
      },
      {
         "cve":"CVE-2024-23109",
         "cvss":"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H"
      },
      {
         "cve":"CVE-2021-34527",
         "cvss":"CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"
      }
      ],
      "connections":[
      {
         "source" : 3,
         "destination" : 4
      }
      ]
   },
   {
      "id":4,
      "hostname": "Host 4",
      "vulnerabilities": [
      {
         "cve":"CVE-2024-21851",
         "cvss":"CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"
      },
      {
         "cve":"CVE-2024-23109",
         "cvss":"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H"
      },
      {
            "cve":"CVE-2021-34527",
            "cvss":"CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H"
        }
        ],
        "connections":[
            {
                "source" : 4,
                "destination" : 1
            },
            {
                "source" : 4,
                "destination" : 5
            }
        ]
    },
    {
        "id":5,
        "hostname": "Host 5",
        "vulnerabilities": [
        {
            "cve":"CVE-2024-21851",
            "cvss":"CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"
        },
        {
            "cve":"CVE-2024-23109",
            "cvss":"CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H"
        },
        {
            "cve":"CVE-2021-34527",
            "cvss":"CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H"
        }
        ],
        "connections":[

        ]
    }

  ]
}
```

As it may noticed, some vulnerabilities appear more than one time, but this has been implemented this way for practical reasons. However, the corresponding CVSS string of each vulnerability has minor adjustments to capture the dynamic nature of the attack graph and showcase more attack examples. Indicatively, exploits in Host 1 can be exploited both on network level, local level and adjacent level. According to the integrity violation to the system exploits can lead to user or root privilege rights accordingly. However, in Host 3 an exploit can lead to user privileges and the attacker can exploit his way via other local exploit in order to get root privileges on the target system. This is practically represented by node 27 which represents a network exploitation that leads to user privileges. Upon having user privileges in Host 3, the attacker is able to exploit the target system with vulnerability on node 23 which does a high integrity violation to the system, thus leading to root rights. In this example, the attacker's goal is defined as the compromise of Host 5.

The attack graph model is shown below in 2 parts due to the lengthy representation of the graph-viz library. The red nodes indicate vulnerabilities while the blue nodes indicate privileges acquired.

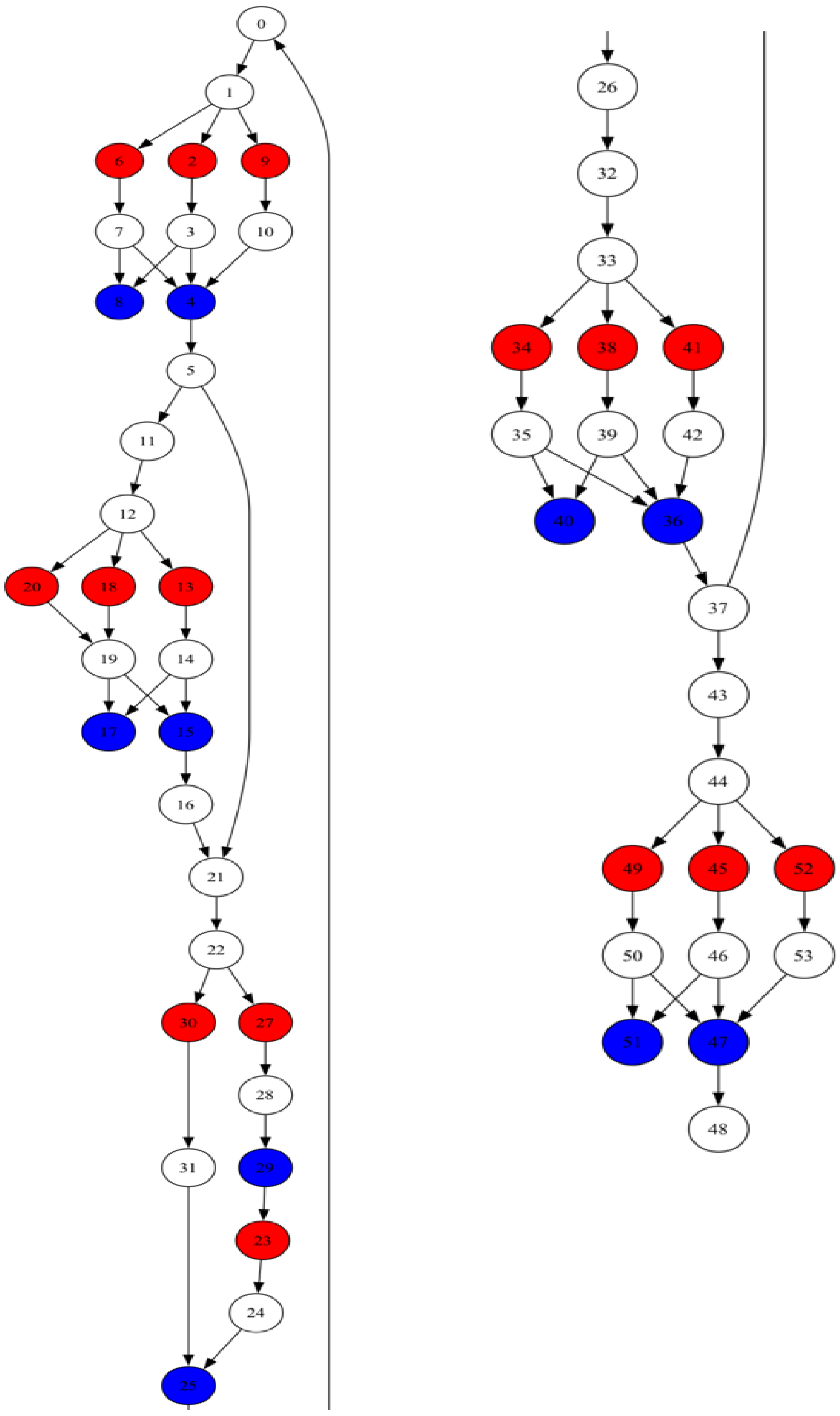


Figure 11: 5-Host Topology in Graph-viz Representation

In the demonstration in this example, the required modified confidentiality, modified integrity, and modified availability for the defender are defined as follows:

- $\boldsymbol{None = 0.20}$
- $\boldsymbol{Low = 0.50}$
- $\boldsymbol{High = 0.80}$

The placement cost for the attacker is statically set to $\boldsymbol{P_c = 5.}$

These values, as mentioned in Chapter 4, are a subject that can be changed, considering how the defender values the metrics and the systems itself. For example, in a scenario in which we consider the modified Confidentiality to matter the most, the values for High and Low can change to be greater than the originals or to those defined in this example.

Following, the dimension of the matrices of the zero-sum game that is played in this example, are equivalent to the Hosts connections in the attack graph, plus 1 for the action of doing no actions. For example, upon compromising Host 1, the attacker can choose between Host 2 and Host 5. In this simple scenario, the matrix is as follows:

Importance of Host 2 & Host 5 is $\boldsymbol{w_2 = 0.4 \; and \; w_5 = 0.6}$, respectively.

D \ A	Attack H1 -> H2	Attack H1 -> H5	Do Nothing
Defend H2	7.6076	-4.207	-5.00
Defend H5	-1.6115	6.0818	-5.00
Do Nothing	3.3884	2.6226	0

The logs from the python application are as follows:

```
Honeypot placement cost: 4
{'AV': 0.85, 'AC': 0.77, 'PR': 0.85, 'UI': 0.85, 'C': 0.56, 'I': 0.56, 'A': 0.56, 'RC': 0.5, 'RI': 0.5, 'RA': 0.5}
{'AV': 0.55, 'AC': 0.77, 'PR': 0.62, 'UI': 0.85, 'C': 0.56, 'I': 0.56, 'A': 0.56, 'RC': 0.5, 'RI': 0.5, 'RA': 0.5}
{'AV': 0.55, 'AC': 0.77, 'PR': 0.85, 'UI': 0.85, 'C': 0.56, 'I': 0.56, 'A': 0.56, 'RC': 0.8, 'RI': 0.2, 'RA': 0.5}

AC =  7.047677816666667
esc =  9.14816
cap =  8.9

5.607677816666667 3.3884138166666666 -1.6115861833333334



{'AV': 0.55, 'AC': 0.77, 'PR': 0.62, 'UI': 0.85, 'C': 0.56, 'I': 0.56, 'A': 0.56, 'RC': 0.2, 'RI': 0.2, 'RA': 0.2}
{'AV': 0.85, 'AC': 0.77, 'PR': 0.85, 'UI': 0.85, 'C': 0.56, 'I': 0.56, 'A': 0.56, 'RC': 0.8, 'RI': 0.8, 'RA': 0.8}
```

```
{'AV': 0.85, 'AC': 0.44, 'PR': 0.62, 'UI': 0.85, 'C': 0.56, 'I': 0.56, 'A': 0.56, 'RC': 0.8, 'RI': 0.2, 'RA': 0.5}

AC =  6.281883208333333
esc =  9.14816
cap =  8.0

6.081883208333334 -4.207012791666667 0.7929872083333329



Value of the game: -2.024222042352073
Optimal mixed strategy for player A: [0.5274171 0.4725829 0.        ]
```

The provided mixed strategy for the Defender is:

$$[\mathbf{0.5274171} \quad ,\mathbf{0.4725829}, \qquad \mathbf{0}]$$

The value of each element represents the probability with which the defender should choose **Defending Host 2**, **Defending Host 5 or Do Nothing**.

Precisely:

- For the first action: Defender should choose it with a probability of approximately $\mathbf{52.74}\%$
- For the second action: Defender should choose it with a probability of approximately $\mathbf{47.26}\%$
- For the third action: Defender should choose it with a probability of $\mathbf{0}\%$

With the probabilities summing up to 1, the mixed strategy from the solver is a valid mixed strategy.

The **value of the game being equal to -2.024222** for the Defender is generated as the multiplication of the mixed strategy with the payoff matrix of the game. In the context of zero-sum games, where defender's gain is directly offset by the attacker's loss, the sign of the value of the game doesn't inherently indicate whether it's good or bad decision rather than reflecting the perspective of the player for whom the value is calculated. That means that in our case, where the Defender has a negative value, on average, the defender can expect to lose that amount when playing the game optimally which in our case is directly affected from the high placement cost ($P_c = 5$) allocated in the simulation parameters.

In zero-sum games, the overall value is neither increased nor decreased; it simply transfers between the players. Therefore, the sign of the value of the game is relative to the perspective of the player being analyzed.

Conclusion

This thesis studied the dynamic landscape of cyber threats and game theoretic approaches for strengthening cybersecurity defenses from the scope of cyber deception, focusing particularly on the strategic deployment of honeypots as a pivotal defensive measure. Leveraging the concepts of honeypots and that of game theory, this work took an already existing framework for optimizing honeypot deployment and enhanced it in many ways aiming to establish a more concrete and detailed model and define solid cybersecurity quantitative metrics that validate the previous work.

A detailed introduction to honeypots and game theory essentials was given in order to cover the require knowledge for properly understanding the studied game and to also answer questions like *"Why is this framework useful?"and "How does it work?"*.

The foundation of this work lies in the cybersecurity modelling described in Chapter 4. The original work presented a simple attack graph model that represents a set of interconnect hosts. However, an attack graph according to the bibliography is not represented as a topological graph. Nevertheless, this work enriched the idea of using attack graphs and created a custom implementation of an attack graph called MAL attack graph. The created attack graph was based on the MAL DSL, which incorporates a set of specific rules for assets and associations, so as to develop dynamic models with strict graph forms that capture specific attributes. This work contributed to the previous with the development of violenceLang, a MAL which encapsulates the notion of exploit dependency graphs on MAL defined assets. MAL associations consider the CVSS string in order to derive exploit related information for generating the final attack graph model. In this case, we don't not only develop attack graphs that include more than one vulnerability in each host but further allow the representation of corresponding interdependencies.

The second contribution of this work was to automate all the previous and apply the game from https://www.microsoft.com/en-us/msrc/security-update-severity-rating-system[17]. The initial framework did not provide the formulas for calculating the quantitative values of the defender-attacker reward matrix. Not only, this thesis provides formulas for playing the game with a more realistic approach but also gives a degree of freedom to the defender to parametrize his/her preferences and requirements in the proposed framework. In addition, formulas consider possible discovered vulnerability and exploitation interdependencies, with respect to the modelling mentioned in Chapter 4. To address the coding of the simulation, a detailed pseudocode is provided that explains how the theory of Chapter 2, can be easily computed with Python for any $n \times n$ matrix, allowed by given complexity restrictions. An example of the optimal honeypot allocation game is played for a decision-making instance on a 5-host custom MAL attack graph. Furthermore, the simulation is also used for explaining how each of the parameters of the defender affect the mixed strategy outcome of the game.

While this work, does not provide a particularly sophisticated framework, it serves as the basis for establishing a feasible formal modeling, showcasing an indicative example which can be enriched in many ways. Having finished this thesis, the next steps include to turn the zero-sum game into a Bayesian repeated game and capture the uncertainty of players and the imperfect information about both the defender and the attacker preferences. This uncertainty will be captured using probability distributions over possible types or strategies for both players and aims to assume that the game is played over multiple rounds. The already defined attack graph language can be easily adjusted to transform the graph into a Markov Chain, so as tto study how each player's belief is affected by previous actions.

References

[1] N. Kambow and L. K. Passi, “Honeypots: The Need of Network Security,” vol. 5, 2014.

[2] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, “A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems,” *IEEE Commun. Surv. Tutor.*, vol. 23, no. 4, pp. 2351–2383, 2021, doi: 10.1109/COMST.2021.3106669.

[3] R. M. Campbell, K. Padayachee, and T. Masombuka, “A survey of honeypot research: Trends and opportunities,” in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec. 2015, pp. 208–212. doi: 10.1109/ICITST.2015.7412090.

[4] K. Sadasivam, B. Samudrala, and T. A. Yang, “Design of network security projects using honeypots,” *J. Comput. Sci. Coll.*, vol. 20, no. 4, pp. 282–293, Apr. 2005.

[5] I. Mokube and M. Adams, “Honeypots: concepts, approaches, and challenges,” in *Proceedings of the 45th annual southeast regional conference*, in ACM-SE 45. New York, NY, USA: Association for Computing Machinery, Mar. 2007, pp. 321–326. doi: 10.1145/1233341.1233399.

[6] “Honeypot Background | Honeyd,” TailBliss. Accessed: Mar. 18, 2024. [Online]. Available: https://www.honeyd.org/background/

[7] “How to build and use a Honeypot. Ralph Edward Sutton, Jr. DTEC 6873 Section 01 - PDF Free Download.” Accessed: Mar. 18, 2024. [Online]. Available: https://docplayer.net/2171531-How-to-build-and-use-a-honeypot-ralph-edward-sutton-jr-dtec-6873-section-01.html

[8] J. Von Neumann and O. Morgenstern, *Theory of games and economic behavior*. in Theory of games and economic behavior. Princeton, NJ, US: Princeton University Press, 1944, pp. xviii, 625.

[9] “Game Theory Intro”.

[10] J. Nash, “Non-Cooperative Games,” *Ann. Math.*, vol. 54, no. 2, pp. 286–295, 1951, doi: 10.2307/1969529.

[11] M. Jackson, “A Brief Introduction to the Basics of Game Theory,” *SSRN Electron. J.*, Dec. 2011, doi: 10.2139/ssrn.1968579.

[12] “Pareto Optimality,” Standford Web Educational Programme. [Online]. Available: https://web.stanford.edu/group/sisl/k12/optimization/MO-unit5-pdfs/5.8Pareto.pdf

[13] S. Hart, “Chapter 2 Games in extensive and strategic forms,” in *Handbook of Game Theory with Economic Applications*, vol. 1, Elsevier, 1992, pp. 19–40. doi: 10.1016/S1574-0005(05)80005-0.

[14] R. A. Briggs, “Normative Theories of Rational Choice: Expected Utility,” in *The Stanford Encyclopedia of Philosophy*, Winter 2023., E. N. Zalta and U. Nodelman, Eds., Metaphysics Research Lab, Stanford University, 2023. Accessed: Mar. 18, 2024. [Online]. Available: https://plato.stanford.edu/archives/win2023/entries/rationality-normative-utility/

[15] C. H. Papadimitriou, “On the complexity of the parity argument and other inefficient proofs of existence,” *J. Comput. Syst. Sci.*, vol. 48, no. 3, pp. 498–532, Jun. 1994, doi: 10.1016/S0022-0000(05)80063-7.

[16] T. Roughgarden, “CS261: A Second Course in Algorithms Lecture #10: The Minimax Theorem and Algorithms for Linear Programming∗,” [Online]. Available: https://theory.stanford.edu/~tim/w16/l/l10.pdf

[17] A. H. Anwar, C. A. Kamhoua, N. Leslie, and C. D. Kiekintveld, “Honeypot Allocation Games over Attack Graphs for Cyber Deception,” in *Game Theory and Machine Learning for Cyber Security*, IEEE, 2021, pp. 62–76. doi: 10.1002/9781119723950.ch4.

[18] “Security Update Severity Rating System.” Accessed: Mar. 18, 2024. [Online]. Available: https://www.microsoft.com/en-us/msrc/security-update-severity-rating-system

[19] "CVSS v3.1 Specification Document," FIRST — Forum of Incident Response and Security Teams. Accessed: Mar. 18, 2024. [Online]. Available: https://www.first.org/cvss/v3.1/specification-document

[20] H. S. Lallie, K. Debattista, and J. Bal, "A review of attack graph and attack tree visual syntax in cyber security," *Comput. Sci. Rev.*, vol. 35, p. 100219, Feb. 2020, doi: 10.1016/j.cosrev.2019.100219.

[21] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph-based network vulnerability analysis," in *Proceedings of the 9th ACM conference on Computer and communications security*, in CCS '02. New York, NY, USA: Association for Computing Machinery, Nov. 2002, pp. 217–224. doi: 10.1145/586110.586140.

[22] S. Noel and S. Jajodia, "Managing attack graph complexity through visual hierarchical aggregation," in *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, in VizSEC/DMSEC '04. New York, NY, USA: Association for Computing Machinery, Oct. 2004, pp. 109–118. doi: 10.1145/1029208.1029225.

[23] W. Wideł, S. Hacks, M. Ekstedt, P. Johnson, and R. Lagerström, "The meta attack language - a formal description," *Comput. Secur.*, vol. 130, p. 103284, Jul. 2023, doi: 10.1016/j.cose.2023.103284.

[24] P. Virtanen *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nat. Methods*, vol. 17, no. 3, pp. 261–272, Mar. 2020, doi: 10.1038/s41592-019-0686-2.

[25] Q. Huangfu and J. A. J. Hall, "Parallelizing the dual revised simplex method," *Math. Program. Comput.*, vol. 10, no. 1, pp. 119–142, Mar. 2018, doi: 10.1007/s12532-017-0130-5.

Appendix

Assets of the attack graph in JSON format as created from the attack graphs engine:

```
"assets": {
    "0": {
      "name": "Has Access to Host 1",
      "metaconcept": "Access",
      "eid": "0",
      "defenses": {}
    },
    "1": {
      "name": "Reconnaisance Host 1",
      "metaconcept": "Scan",
      "eid": "1",
      "defenses": {}
    },
    "2": {
      "name": "CVE-2024-21851",
      "metaconcept": "Vulnerability",
      "eid": "2",
      "defenses": {}
    },
    "3": {
      "name": "Local Exploit",
      "metaconcept": "Local",
      "eid": "3",
      "defenses": {}
    },
    "4": {
      "name": "Root on Host:Host 1",
      "metaconcept": "Privileges",
      "eid": "4",
      "defenses": {}
    },
    "5": {
      "name": "Host Compromise :Host 1",
      "metaconcept": "Host",
      "eid": "5",
      "defenses": {}
    },
    "6": {
```

```
    "name": "CVE-2024-23109",
    "metaconcept": "Vulnerability",
    "eid": "6",
    "defenses": {}
   },
   "7": {
    "name": "Network Exploit",
    "metaconcept": "Network",
    "eid": "7",
    "defenses": {}
   },
   "8": {
    "name": "User Privilege Escalation",
    "metaconcept": "Privileges",
    "eid": "8",
    "defenses": {}
   },
   "9": {
    "name": "CVE-2021-34527",
    "metaconcept": "Vulnerability",
    "eid": "9",
    "defenses": {}
   },
   "10": {
    "name": "Network Exploit",
    "metaconcept": "Network",
    "eid": "10",
    "defenses": {}
   },
   "11": {
    "name": "Has Access to Host 2",
    "metaconcept": "Access",
    "eid": "11",
    "defenses": {}
   },
   "12": {
    "name": "Reconnaisance Host 2",
    "metaconcept": "Scan",
    "eid": "12",
    "defenses": {}
   },
   "13": {
    "name": "CVE-2021-44228",
    "metaconcept": "Vulnerability",
    "eid": "13",
    "defenses": {}
   },
   "14": {
    "name": "Network Exploit",
    "metaconcept": "Network",
    "eid": "14",
    "defenses": {}
   },
   "15": {
    "name": "Root on Host:Host 2",
    "metaconcept": "Privileges",
    "eid": "15",
    "defenses": {}
   },
   "16": {
    "name": "Host Compromise :Host 2",
    "metaconcept": "Host",
    "eid": "16",
    "defenses": {}
   },
   "17": {
    "name": "User Privilege Escalation",
    "metaconcept": "Privileges",
    "eid": "17",
    "defenses": {}
   },
   "18": {
    "name": "CVE-2021-27190",
    "metaconcept": "Vulnerability",
    "eid": "18",
    "defenses": {}
   },
   "19": {
    "name": "Local Exploit",
    "metaconcept": "Local",
    "eid": "19",
    "defenses": {}
   },
   "20": {
    "name": "CVE-2021-26084",
    "metaconcept": "Vulnerability",
    "eid": "20",
    "defenses": {}
   },
   "21": {
    "name": "Has Access to Host 3",
    "metaconcept": "Access",
    "eid": "21",
    "defenses": {}
   },
   "22": {
    "name": "Reconnaisance Host 3",
    "metaconcept": "Scan",
    "eid": "22",
    "defenses": {}
   },
   "23": {
    "name": "CVE-2024-21851",
    "metaconcept": "Vulnerability",
    "eid": "23",
    "defenses": {}
   },
   "24": {
      "name": "Local Exploit",
      "metaconcept": "Local",
      "eid": "24",
      "defenses": {}
    },
    "25": {
      "name": "Root on Host:Host 3",
      "metaconcept": "Privileges",
      "eid": "25",
      "defenses": {}
    },
    "26": {
      "name": "Host Compromise :Host 3",
      "metaconcept": "Host",
      "eid": "26",
      "defenses": {}
    },
    "27": {
      "name": "CVE-2024-23109",
      "metaconcept": "Vulnerability",
      "eid": "27",
      "defenses": {}
    },
    "28": {
      "name": "Network Exploit",
      "metaconcept": "Network",
      "eid": "28",
      "defenses": {}
    },
    "29": {
      "name": "User Privilege Escalation",
      "metaconcept": "Privileges",
      "eid": "29",
      "defenses": {}
    },
    "30": {
      "name": "CVE-2021-34527",
      "metaconcept": "Vulnerability",
      "eid": "30",
      "defenses": {}
    },
    "31": {
      "name": "Network Exploit",
      "metaconcept": "Network",
      "eid": "31",
      "defenses": {}
    },
    "32": {
      "name": "Has Access to Host 4",
      "metaconcept": "Access",
      "eid": "32",
      "defenses": {}
    },
    "33": {
```

```
   "name": "Reconnaisance Host 4",
   "metaconcept": "Scan",
   "eid": "33",
   "defenses": {}
  },
  "34": {
   "name": "CVE-2024-21851",
   "metaconcept": "Vulnerability",
   "eid": "34",
   "defenses": {}
  },
  "35": {
   "name": "Local Exploit",
   "metaconcept": "Local",
   "eid": "35",
   "defenses": {}
  },
  "36": {
   "name": "Root on Host:Host 4",
   "metaconcept": "Privileges",
   "eid": "36",
   "defenses": {}
  },
  "37": {
   "name": "Host Compromise :Host 4",
   "metaconcept": "Host",
   "eid": "37",
   "defenses": {}
  },
  "38": {
   "name": "CVE-2024-23109",
   "metaconcept": "Vulnerability",
   "eid": "38",
   "defenses": {}
  },
  "39": {
   "name": "Network Exploit",
   "metaconcept": "Network",
   "eid": "39",
   "defenses": {}
  },
  "40": {
   "name": "User Privilege Escalation",
   "metaconcept": "Privileges",
   "eid": "40",
   "defenses": {}
  },
  "41": {
   "name": "CVE-2021-34527",
   "metaconcept": "Vulnerability",
   "eid": "41",
   "defenses": {}
  },
  "42": {
    "name": "Network Exploit",
    "metaconcept": "Network",
    "eid": "42",
    "defenses": {}
  },
  "43": {
    "name": "Has Access to Host 5",
    "metaconcept": "Access",
    "eid": "43",
    "defenses": {}
  },
  "44": {
    "name": "Reconnaisance Host 5",
    "metaconcept": "Scan",
    "eid": "44",
    "defenses": {}
  },
  "45": {
    "name": "CVE-2024-21851",
    "metaconcept": "Vulnerability",
    "eid": "45",
    "defenses": {}
  },
  "46": {
    "name": "Local Exploit",
    "metaconcept": "Local",
    "eid": "46",
    "defenses": {}
  },
  "47": {
    "name": "Root on Host:Host 5",
    "metaconcept": "Privileges",
    "eid": "47",
    "defenses": {}
  },
  "48": {
    "name": "Host Compromise :Host 5",
    "metaconcept": "Host",
    "eid": "48",
    "defenses": {}
  },
  "49": {
    "name": "CVE-2024-23109",
    "metaconcept": "Vulnerability",
    "eid": "49",
    "defenses": {}
  },
  "50": {
    "name": "Network Exploit",
    "metaconcept": "Network",
    "eid": "50",
    "defenses": {}
  },
  "51": {
    "name": "User Privilege Escalation",
    "metaconcept": "Privileges",
    "eid": "51",
    "defenses": {}
  },
  "52": {
    "name": "CVE-2021-34527",
    "metaconcept": "Vulnerability",
    "eid": "52",
    "defenses": {}
  },
  "53": {
    "name": "Network Exploit",
    "metaconcept": "Network",
    "eid": "53",
    "defenses": {}
  }
}
```