KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Applied and Computational Mathematics

Second cycle, 30 credits

Self-Play Reinforcement Learning for Finding Intrusion Prevention Strategies

JAKOB STYMNE

Stockholm, Sweden 2022

Abstract

This Master thesis studies automated intrusion prevention using self-play reinforcement learning. We extend a decision-theoretic model of the intrusion prevention use case based on optimal stopping theory proposed in previous work to a game-theoretic setting. We model the use case as a zero-sum onesided partially observed stochastic game where the defender's stop actions determine the times to take defensive actions and the attacker's stop actions determine when to attack. To find optimal defender strategies, we use multi-agent reinforcement learning. In a novel approach, we extend the Neural Fictitious Self-Play algorithm to partially observed stochastic games. With this approach, we narrow the gap between the theoretical framework of partially observed stochastic games and the framework of model-free reinforcement learning. We show that the learned strategies converge near a Nash equilibrium. Inspection of the converged strategies shows that they imitate human strategies but are heavily dependent on hyperparameters and the reward function.

Keywords

Network security, automation, reinforcement learning, Markov Security Games, Neural Fictitious Self Play

Sammanfattning

Svensk titel: Självspelsförstärkningsinlärning för att hitta intrångsförebyggande strategier

Detta examensarbete handlar om att automatisera intrångsförebyggande strategier genom att använda självspelsförstärkningsinlärning. Vi bygger vidare på en beslutsteoretisk modell av det intrångsförebyggande användningsfallet baserad på optimal stoppteori föreslagit i tidigare arbeten, till en spelteoretisk situation. Närmare bestämt så väljer vi att modellera användningsfallet som ett stokastiskt nollsummespel med ensidig partiell observabilitet, där försvararens stoppaktioner motsvarar tidpunkterna för att ta försvarande aktioner och anfallarens stoppaktioner motsvarar tiden för att starta anfallet. För att hitta optimala försvarsstrategier så använder vi oss av multi-agent förstärkningsinlärning. Med ett nytt tillvägagångssätt använder vi algoritmen Neural Fictitious Self Play för partiellt observerbara spel. Med detta tillvägagångssätt så minskar vi gapet mellan det teoretiska ramverket för partiellt observerbara stokastiska spel och modellfri förstärkningsinlärning. Vi visar att de inlärda strategierna konvergerar mot en Nashjämvikt. Närmre inspektion av de konvergerade strategierna visar att dem imiterar mänskligt beteende, men är mycket beroende av hyperparametrar och belöningsfunktionen.

Nyckelord

Nätverkssäkerhet, automatisering, förstärkningsinlärning, Markovianska säkerhetsspel, Neural Fictitious Self Play

Acknowledgments

First and foremost I would like to thank my supervisors Kim Hammar and Rolf Stadler for their help during the work with this thesis. I specifically want to thank Kim for constantly being available to answer questions and for interesting discussions. Kim helped provide parts of the text for the mathematical background (Section 3.2), helped me formalize the game model (Chapter 4) and helped with coding issues. For the theoretical Section 4.7, Kim and I collaborated to derive the proofs. Furthermore I would like to thank my examiner Jimmy Olsson for constructive feedback during the course of the project.

Stockholm, June 2022
Jakob Stymne

Contents

Chapter 1

Introduction

In the last few years, cyber attacks against organizations, companies, and even governments have become commonplace. Recent examples are the "Kayesa" hacking attack which forced Swedish supermarket Coop to close all its stores, as well as the hacking attack against the Ukrainian government websites in January 2022, warning Ukrainians to "prepare for the worst".

Some of the key findings of the report “Global Cyber Security Outlook 2022” from the World Economic Forum [1], in which 120 global cyber leaders were interviewed, include:

- In 2021, the ransomware attacks have increased by 151%.
- Each organization faced 270 cyber-attacks on an average.
- 45% of organizations have been affected by a third-party cyber attack in the past 2 years
- 48% of leaders say that "Automation and machine learning" is expected to have the greatest influence on transforming cyber security in the next two years

Hence, cyber security is becoming increasingly important in the modern and digitalized era. Current cyber security solutions, however, are largely based on heuristics and manual configurations. As cyber attacks are getting increasingly automated and sophisticated, there is a need to complement current security solutions with automated security. A promising framework for automating cyber security tasks is reinforcement learning, where security strategies are learned through interaction with an environment and analyzed theoretically using game theory and decision theory. This master thesis project makes a contribution in this context.

In this master thesis, we consider an *intrusion prevention use case* involving an IT infrastructure. It involves two actors: an IT operator, who defends the infrastructure, while maintaining service to the client population, and an attacker, who is trying to intrude the infrastructure and avoid getting detected by the defender. The optimal strategy of the IT operator is to maximize the provided service to the client population and to prevent a possible hacking attack at minimum cost.

Previous work on the intrusion prevention use case shows that automated defense strategies for simplified environments can be learned effectively using reinforcement learning and self play [2]. In this line of work, the use case is modeled as a Markov game or Markov decision process and simulations of the game are used to learn optimal defense and attack strategies simultaneously based on experience from played games. One model that has been proposed is [3], where the intrusion prevention use case is formulated as a multiple optimal stopping problem and optimal defender strategies against static attackers are learned through reinforcement learning. In this formulation, each defensive action is associated with a stopping action and insight into the structure of optimal strategies is obtained through the theory of optimal stopping. In this project, we extend [3] to a game-theoretic setting by considering a dynamic attacker.

Our goal is to analyze and find optimal defense strategies for the intrusion prevention use case. With an optimal defender strategy we mean a strategy that guarantees some minimum level of reward for the defender agent, even in the case of a worst-case attacker who plays optimally. In game-theoretical terms, this means that we are looking to find a *Nash equilibrium*, where both the attacking and defender player are playing the *best-response strategies* against their opponent. To obtain the Nash equilibrium strategies, two main methods can be used: computational game theory and reinforcement learning. In our use case, most of the algorithms from computational game theory are either intractable to apply or make strong assumptions. Therefore, we use a reinforcement learning approach where Nash equilibrium strategies of the attacker and defender are learned in an iterative fashion. Ultimately, this project lies at the intersection of reinforcement learning and game theory, where reinforcement learning is used to solve a game theoretical problem.

1.1 Related Work

The goal of finding automated security strategies has gained increased attention in recent years and has been studied by many different fields and points

of views, for example using reinforcement learning, game theory, dynamic programming, control theory, attack graphs, statistical tests, and evolutionary computation.

The main references for this thesis project are the previous work done by Kim Hammar and Rolf Stadler, [2, 3, 4, 5]. These works study the same intrusion prevention use case as in this thesis and formulates the use case as an intrusion prevention problem. These works are the first to study self-play in the context of networks and systems security, and the first to formulate the use case as an optimal stopping problem. The stochastic game model studied in this project is a direct extension of the models formulated in these works. In this project, we extend the referenced works by considering the case of a dynamic attacker who adapts their strategy to the defender. Other work that use reinforcement learning for finding security strategies include [6], [7], and [8], all of which focus on single-agent reinforcement learning and different use cases than ours.

In another line of work, game theory is used to find and analyze automated security strategies [9, 10, 11]. In one survey from 2013 [12], roughly 20 different security or privacy problems which are formulated as games are listed, with different outcomes such as optimal defense strategies, equilibrium analysis and performance limits. When looking at available security games in the literature, there are many different classes of games. This includes for example games related to Distributed Denial-of-Service (DDoS) [13, 14], resource allocation games [15], and intrusion prevention games [16, 2, 17, 18, 19, 20]. However, few of the game-theoretical approaches to intrusion prevention use reinforcement learning, and to the best of our knowledge, there are none which uses neural fictitious self play, which is the algorithm studied in this thesis.

Lastly, in this thesis, we implement the *Neural Fictitious Self Play* (NFSP) [21] algorithm, which in turn is an extension of fictious self play [22] and fictitious play [23]. The NFSP algorithm implemented in this thesis extends the original algorithm by including the use of beliefs. Other algorithms for multi-agent reinforcement learning are Policy-Space Response Oracle algorithm [24] and Alpha-Zero [25]. For an overview of available algorithms and challenges of multi-agent reinforcement learning, we refer to [26] and [27].

1.2 Problem and Research Questions

We study the use case of intrusion prevention and the problem of obtaining automated security strategies. Traditional approaches to intrusion prevention

use packet inspection and static rules for detection of intrusions and selection of response actions. Their main drawback lies in the need for domain experts to configure the rule sets. Therefore, a promising research direction is to develop methods for automatically obtaining security strategies, which would reduce the need for domain experts. We investigate this research direction using reinforcement learning and analyze the optimal security strategies using game theory. Our main research questions are:

1. How can the intrusion prevention use case be modeled as a game? Ideally, the model should have the following properties:
    - Captures relevant properties of practical IT infrastructures.
    - Analytically tractable when scaled down.
    - Can be scaled up to realistic use cases.
2. What structural insights can be derived for the model?
    - Does there exist a Nash equilibrium? Pure or mixed?
    - Does an optimal intrusion prevention strategy with specific structure exist?
3. Which reinforcement learning algorithms and/or game-theoretic methods are suitable for computing automated intrusion prevention strategies for the given model?

1.3 Methodology for Analysis

The project follows a quantitative research methodology. Theories are developed and tested empirically. Specifically, game theory and Markov decision theory are used to model the intrusion prevention use case and analyze optimal strategies. Simulations are used to evaluate computational algorithms, e.g. rates of convergence and solution quality. Finally, a testbed at KTH is used to run experiments and collect empirical results that complement the numerical simulations.

1.4 Scope and Limitations

We limit the scope of this thesis to studying a specific use case in network security, namely intrusion prevention. The observations generated by the

attacker actions are based on intuitive attacker characteristics or data traces from simulated hacking attacks against the KTH servers. One central concept is the notion of *belief*, which is a measure of how likely the defender think there is an ongoing intrusion. We limit the analysis of the game to the case when both the defender and the worst-case attacker take this probabilistic belief into consideration, which could be considered an oversimplification of the behavior of the agents. Furthermore, we focus on one specific implementation of a multi-agent reinforcement learning algorithm, which is the Neural Fictitious Self Play.

1.5 Contributions

We make two main contributions. Firstly, we combine the approaches of multi-agent self play using reinforcement learning and the optimal stopping formulation of the use case. Specifically, we extend the optimal stopping intrusion prevention model proposed in [3] to the case with two intelligent agents (see Chapter 4).

Secondly, we apply the Neural Fictitious Self Play (NFSP) algorithm to learn security strategies. NFSP has previously mainly been used for card games, so this is a new application of the algorithm. To get a NFSP implementation that works with the implemented partially-observed game model, we had to make some modifications to the original implementation, using insights from [28]. Specifically, the belief distributions of the defender had to be calculated as each time step. This proposed modification, which is described in Chapter 5, is a novel approach how to apply beliefs in NFSP. Our results (found in Chapter 6) show that NFSP successfully can be used to find optimal stopping defense strategies.

1.6 Report Outline

The thesis is structured as follows. In Chapter 2 we describe the intrusion prevention use case. In the next chapter, Chapter 3, we define the necessary mathematical framework of game theory. The next Chapter, 4, formalizes the model using game theory. Chapter 5 describes reinforcement learning and our implementation of Neural Fictitious Self Play as well as the evaluation procedures. Chapter 6 presents the general experiment setup and the results of the experiments. The results are discussed and the conclusion of the thesis is presented in Chapter 7.

Chapter 2

The Intrusion Prevention Use Case

In this chapter, we describe the intrusion prevention use case. First, we give a basic introduction to network security. Since the focus of this report is mathematical, the background on network security is rather limited. We describe the necessary components to define the intrusion prevention use case as a stochastic game, such as the anatomy of a cyber attack as well as the description of Intrusion Detection Systems and Intrusion Prevention Systems. Using this basic domain knowledge, we elaborate on how we define the intrusion prevention use case.

2.1 Basic Concepts of Computer Security

NIST defines computer security as "The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/ data, and telecommunications)" [29]. This definition outlines the three main objectives of a computer security system: **integrity, availability and confidentiality**. Integrity means a system works in the way they are intended, and that information and programs in the systems are changed only in a specified and authorized manner. Availability means service is not denied to authorized users. Confidentiality means that confidential information and data should be kept private to unauthorized individuals. When constructing optimal automated intrusion defense strategies, we want to find these strategies in a manner so that all three of these objectives are taken into account.

In a review conducted by FOI, five different stages of an AI-supported cyberattack was identified in the literature; **Reconnaissance; Access and**

penetration; Internal reconnaissance and lateral movements; Command, control, and actions on objectives; and **Exfiltration and sanitation** [30].

Reconnaissance is the process of research before the intrusion starts, when the hacker collects intelligence on how the system can be hacked. Access is the means by which a hacking attacker uses the knowledge to penetrate the target infrastructure. Internal reconnaissance and lateral movement is collection of intelligence once the agent is within the system. Command, control and actions on objectives occur once the attacker agent has established itself within the IT network. These objective could include stealing data or manipulating data, or damaging system functions. Exfiltration and sanitation is the last step of the hacking attack, when the attacker removes itself from the infrastructure. As we are trying to simulate an AI-supported hacking attack in our intrusion prevention use case game model, these different stages of the attack should be included in some sense.

An **Intrusion Detection System (IDS)** is a device or software that monitors for malicious activities and generates alerts when they are detected. The intrusion detection works by monitoring for irregularities. An **Intrusion Prevention System (IPS)** is a further development of the IDS. The IPS can be defined as an system that detects and blocks malicious network activity in real time.

2.2 Description of the Use Case

We consider an intrusion prevention use case that involves the IT infrastructure of an organization. The use case is almost identical to the one defined in [3]. Therefore, we refer to this article for a more elaborate description. Here, we will summarize it briefly.

The operator (defender) of the IT infrastructure has two main objectives: providing service to the client population, while defending it against hacker attacks. The defender can monitor the system using an IDS that logs events in real time. For a schematic visualization, we refer to Figure 2.1.

The attacker's goal is to intrude on the infrastructure and compromise a set of its servers. To achieve this, the attacker must explore the infrastructure through reconnaissance and exploit vulnerabilities, while, at the same time, avoid getting detected by the defender. The attacker decides the time to start the intrusion and may also decide to abort an ongoing intrusion at any moment. During the intrusion, it is assumed that the attacker follows a fixed sequence of commands. When deciding on the time to start and stop the intrusion, the attacker must consider both the gain of compromising additional servers in the

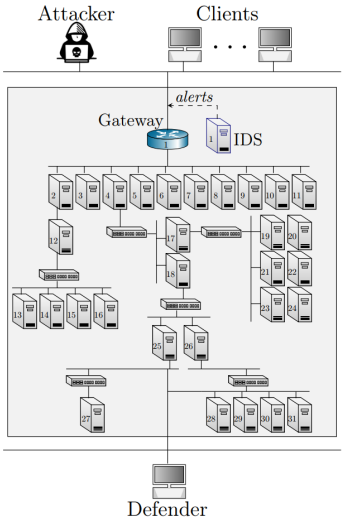

Figure 2.1: The IT infrastructure and the actors in the use case. Courtesy of Kim Hammar.

infrastructure and the risk of getting detected by the defender.

The defender continuously monitors the infrastructure through accessing and analyzing IDS statistics and login attempts at the servers. The defender can then take different defensive actions. These could be for example resetting user accounts, updating firewall configurations or ultimately blocking all external access. These actions comes with a cost, and the defender starts by taking the actions which has the lowest cost. The defender is trying to both maintain service as long as possible, but then shut out the attacker as fast as possible with minimal cost when the intrusion starts. Therefore, the objective for the defender is to find the optimal time to take the defensive actions.

We want to study the use case from a game-theoretic perspective and model the use case as a zero-sum game. The reward function of this game encodes the defender's benefit of maintaining service and its loss of being intruded. Hence, the defender seeks to maximize reward and the attacker wants to minimize it.

Chapter 3

Game Theoretical Framework

In this project, we model the intrusion prevention use case as a game. This section covers the necessary mathematical background and notation on noncooperative game theory, stochastic games, and partially observed stochastic games.

3.1 Introduction to Noncooperative Game Theory

Game theory contains analytical tools and concepts to study the optimal behavior when several decision-taking actors are involved in strategic interactions. In this context, an actor is an *agent* that is situated in an environment and acts autonomously to reach a defined goal. It is able to perceive its environment and able to react to observations and observed changes in the environment. It can also evaluate the current state of the environment (for example using some kind of utility function), and choose an action so that its utility is maximized.

3.1.1 Game Representations: Normal Form and Extensive Form

There are many different possible representations for games. The normal form game, sometimes called a matrix representation, could be considered the most basic one.

Definition 3.1.1 (Normal form game)**.** *We define a normal form game as a triplet*

$$\mathcal{G} = \langle \mathcal{N}, (\mathcal{A}_i)_{i \in \mathcal{N}}, (u_i)_{i \in \mathcal{N}} \rangle$$

where $\mathcal{N}$ is a finite set of players, $(\mathcal{A}_i)_{i\in\mathcal{N}}$ denotes the sets of actions of players $1, \ldots, |\mathcal{N}|$, and $(u_i)_{i\in\mathcal{N}}$ denotes the utility functions for players $i \in \mathcal{N}$, where $u_i : \prod_{i\in\mathcal{N}} \mathcal{A}_i \to \mathbb{R}$.

Normal form games can be classified as *zero-sum games* and *nonzero-sum games*. A two-player zero-sum game is a game where the total utility of any combination of actions is zero, meaning that the gain of one player results in the loss of one player.

		Player 2		
		Rock	Paper	Scissors
Player 1	Rock	$(0, 0)$	$(-1, 1)$	$(1, -1)$
	Paper	$(1, -1)$	$(0, 0)$	$(-1, 1)$
	Scissors	$(-1, 1)$	$(1, -1)$	$(0, 0)$

Figure 3.1: Example of rock-paper-scissors, a two player zero-sum normal form game. The first value in each cell corresponds to the utility of the row player; the second that of the column player.

A normal form game representation is limited due to the fact that it only shows "one-shot" of the game. Oftentimes, we study sequential games, where agents dynamically play several rounds of the game against each other. In this type of dynamic game, we want to have a representation which reflects all possible situations that can be encountered in the game, including stochastic events and moves by the opponents. One representation of dynamic games is the *extensive form* representation. See Figure 3.2 for an example of such a representation.

Extensive form games can be classified as *perfect information games* and *imperfect information games* are used. In an imperfect information game, a player might not be able to observe the state of the game perfectly; the states that are not distinguishable belong to the same *information set*. A special case of the game is the *repeated game*, where the same normal form game, called the stage game, is played repeatedly.

Any extensive form game can be converted to a normal form game, although the resulting number of utility payoffs is typically exponential in the number of game states. This means that the normal form representation of a large extensive game becomes intractably large [31].

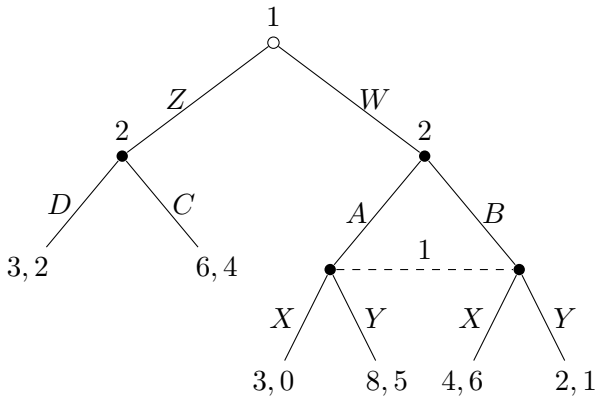

Figure 3.2: Example of a game with two players (1, 2). After player 1 makes decision W, player 2 can either make decision A or B. These two decisions are indistinguishable for player 1 (marked by the dashed line), and thus belong to the same information set.

3.1.2 Solution Concepts: Strategies, Best Response and Nash Equilibrium

The actions that an agent takes in a game is defined by its strategy. A strategy is a complete plan for playing the game, describing how the agent should act in every possible situation in the game. A strategy profile is a vector of strategies for all agents and can be defined as $\pi = \langle \pi_1, \ldots, \pi_n \rangle \in \Pi$. The strategy profile for all agents except agent i can be defined as $\pi_{-i} = \langle \pi_1, \pi_2, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n \rangle$.

Take for example the game in Figure 3.2. In this case, player 1 has two choices, Z/W and X/Y (if he chooses W as a first choice). Player 2 also 2 choices: D/C, if player 1 chooses Z; A/B, if player 1 chooses W. The possible strategies and an example of a strategy profile in this game are:

$$\pi_1 = \{(Z, X), (Z, Y), (W, X), (W, Y)\}$$
$$\pi_2 = \{(D, A), (D, B), (C, A), (C, B)\}$$
$$\pi = \langle \pi_1, \pi_2 \rangle, \text{ for example } \pi = \langle (Z, X), (D, A) \rangle.$$

Notice that some strategies are redundant and impossible to reach, such as $\pi_1 = (Z, X)$. However these are still part of the strategy set.

The objective of each agent is to maximize its utility. We say that a strategy

is a best response for player i against the opponents' strategies if it maximizes the utility against those strategies.

Definition 3.1.2 (Best response strategy)**.** *Let $\mathcal{G} = \langle \mathcal{N}, (\mathcal{A}_i)_{i \in \mathcal{N}}, (u_i)_{i \in \mathcal{N}} \rangle$ be a normal form game. Then the best response strategy for player i π_i^* is the strategy so that $\forall \pi_i \in \Pi_i, u_i(\pi_i^*, \pi_{-i}) \geq u_i(\pi_i, \pi_{-i})$, i.e. the strategy that gives the highest utility to player i holding the other players' strategies π_{-i} fixed. When π_i^* is a best response strategy against π_{-i} we say that $\pi_i^* \in BR_i(\pi_{-i})$.*

For example, in Figure 3.2, if $\pi_2 = \pi_{-1} = (D, A)$ then the best-response strategy for player 1 π_1^* is (W, Y) as this yields the highest utility (8) for player 1.

The definition of best response strategies leads us to a solution concept for noncooperative games; the *Nash equilibrium*. The Nash equilibrium is the strategy profile for all agents, where each agent plays the best response against the other agents' strategies. In this solution point, no agent can improve his or her payoff by changing their strategy, meaning that no agent has any incentive to deviate from their current strategy. We can define the concept formally as:

Definition 3.1.3 (Nash equilibrium)**.** *Let $\mathcal{G} = \langle \mathcal{N}, (\mathcal{A}_i)_{i \in \mathcal{N}}, (u_i)_{i \in \mathcal{N}} \rangle$ be a normal form game. Strategy profile $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ is a Nash equilibrium iff $\forall i \in \mathcal{N}, \pi_i \in BR_i(\pi_{-i})$.*

To illustrate the Nash equilibrium concept, consider the example of the prisoner's dilemma. In this example there are two prisoners, locked up in two separate interrogation rooms. The two prisoners (or agents) can choose to either betray the other prisoner, or stay silent. If both prisoners betray each other, both of them serve two years on prison. If one of them betrays while the other stay silent, the person who betrays is set free but the person who stays silent gets three tears in prison. If both of them stay silent, they both only serve one year in prison.

This game can be represented as a two player normal form (matrix) game as in Figure 3.3.

		Prisoner Y	
		Stay quiet	Betray
Prisoner X	Stay quiet	$(-1, -1)$	$(-3, 0)$
	Betray	$(0, -3)$	$(-2, -2)$

Figure 3.3: The prisoner's dilemma

In this game, the Nash equilibrium is given by (Betray, Betray) as no agents has any incentive to deviate. However, both agents would receive less

sentences if they both stayed quiet. This example illustrates how game theory can explain seemingly unintuitive behavior in decision making situations.

We also differentiate between *pure* and *mixed* strategies. A pure strategy is defined as in the last subsection, i.e. a complete description how a player will play a game for each situation they may face. A mixed strategy is an assignment of a probability to each pure strategy. One can see it as a way to randomly choose among different available strategies to avoid being predictable.

Definition 3.1.4 (Mixed strategies)**.** *Let $\mathcal{G} = \langle \mathcal{N}, (\mathcal{A}_i)_{i\in\mathcal{N}}, (u_i)_{i\in\mathcal{N}} \rangle$ be a normal form game. A mixed strategy is a sequence $(\pi_j)_{j=1}^k \in \Pi_i$ for player i and a probability distribution $\sigma = (\sigma_j)_{j=1}^k$ where the player i chooses strategy π_j with probability σ_j.*

We can extend the concepts for pure strategies such as best response and Nash equilibrium for mixed strategies as well. Furthermore, Nash [32] famously proved the following theorem regarding the existence of equilibrium points:

Theorem 3.1.1 (Nash [32])**.** *Every game with a finite number of players and action profiles has at least one Nash equilibrium in mixed strategies.*

Consider for example the game of rock-paper-scissor, see the payoff table in Figure 3.1. Here it is obvious that there exists no pure Nash equilibrium, since each player always can improve their strategy by changing their next action to the one that beat the other player's last action. However, because of the symmetrical nature of the game, there exists a mixed Nash equilibrium $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, i.e. the strategy for each player is to randomly choose one of the three possible actions rock-paper-scissors.

3.1.3 Maxmin, Minmax, and Value of a Zero-sum Game

Playing a Nash strategy does not give any guarantees for the expected payoff. If we want guarantees, we need to introduce the concept of *maxmin*-strategy. The maxmin-strategy is the strategy that guarantees, and maximizes, some minimum payoff. We can see it as a conservative strategy against a worst-case opponent. On the other hand, the *minmax*-strategy is the strategy that minimizes the opponents' maximum reward. In the zero-sum game case, von Neumann [33] proved the following game-theoretical theorem:

Theorem 3.1.2 (von Neumann [33])**.** *In any finite, two-player zero-sum game, in any Nash equilibrium each player receives a payoff that is equal to both their maxmin value and the minmax value of the opponent.*

This theorem has two important implications for this project. Firstly, it means that we can safely play Nash strategies in zero-sum games, and thus guarantee some minimum reward. Secondly, it means that all Nash equilibria have the have the same payoff. By convention, we call the maxmin value, which in the zero-sum case is the Nash equilibrium value, the *value of the game*.

3.2 Stochastic games and Partial Observability

In this project, we build on the formal model for the intrusion prevention use case developed in [3]. In the referenced work, the use case is modeled using Markov Decision Theory. Here, we extend the Markov model to the multi-agent case. In particular, we extend the Markov model to a *stochastic game*.

A stochastic game is a form of repeated game with several players, and where the game is set in some *environment*. In the use case of this project, the environment constitutes an IT infrastructure. The environment can be in different *states* (for example, the IT environment can be in the states of intrusion or no intrusion). In each time step, each player in the stochastic game simultaneously choose one action. The environment then *transition* to the same or a different state according to some defined *transition probabilities*, whereas the agents receive *rewards* depending on what actions they chose. The game evolves in discrete time steps from $t = 1$ to $t = T$, which constitute one *episode* of the game.

The transition probabilities to a new state depend on the current state as well as the actions, meaning that we can define it as

$$\mathcal{T}^{a_t}_{s_t,s_{t+1}} = \mathcal{T}\left[s_{t+1}|s_t, a_t\right].$$

This equation has the Markov property, meaning that the probabilities of the next state is independent of past states given the current state: $\mathcal{T}\left[s_{t+1}|s_t\right] = \mathcal{T}\left[s_{t+1}|s_1, \ldots, s_t\right]$.

We can denote the rewards for each player as $(\mathcal{R}_i)_{i\in\mathcal{N}}$. The reward of player i at time t is a scalar $r^{(i)}_t$ and $\mathcal{R}_i(s_t, s_{t+1}, a_t)$ denotes the expected reward of player i when transitioning from state s_t to state s_{t+1} after action profile a_t is played (Eq. 3.1)

$$\mathcal{R}_i(s_t, s_{t+1}, a_t) = \mathbb{E}\left[r^{(i)}_{t+1}|s_t, s_{t+1}, a_t\right]. \tag{3.1}$$

The rewards of the game are discounted, meaning that each reward $r_t^{(i)}$ is multiplied by a factor $\gamma^t, \gamma \in (0, 1]$. The discount factor γ can be thought of a measure how much the agent values future returns compared to immediate returns.

3.2.1 Partially Observed Stochastic Games (POSGs)

In the intrusion prevention use case, the defender does not know exactly what state of intrusion or no intrusion the IT infrastructure is in. Instead, the defender can analyze IDS signals from the infrastructure and from these try to deduct if there is currently an ongoing hacker attack. Mathematically, this corresponds to saying that the use case is *partially observable*. A partially observed stochastic game is a stochastic game where the states are not directly observable. Instead the players can observe *observations* of the game, which are generated by some *observation function* which has a probabilistic correlation with the current states and the actions taken by the players [34, 35, 36, 37]. In the intrusion prevention use case, these observations correspond to the IDS signals.

We can now formally define the Partially Observed Stochastic Game (POSG):

Definition 3.2.1 (Partially Observed Stochastic Game)**.** *A POSG is defined by a ten-tuple*

$$\Gamma^P = \langle \mathcal{N}, \mathcal{S}, (\mathcal{A}_i)_{i\in\mathcal{N}}, \mathcal{T}, (\mathcal{R}_i)_{i\in\mathcal{N}}, (\mathcal{O}_i)_{i\in\mathcal{N}}, \mathcal{Z}, \gamma, \rho_0, T \rangle$$

where $\mathcal{N}$ is a finite set of players, S refers to the set of states, and $(\mathcal{A}_i)_{i\in\mathcal{N}}$ denotes the sets of actions of players $1, \ldots, |\mathcal{N}|$. The transition function $\mathcal{T}$ is an operator on the set of states and the combined action space of all players, $\mathcal{T} : \mathcal{S} \times (\times_{i\in\mathcal{N}} \mathcal{A}_i) \to \Delta(\mathcal{S})$, where $(\times_{i\in\mathcal{N}} \mathcal{A}_i) = \mathcal{A}_1 \times \ldots \times \mathcal{A}_{|\mathcal{N}|}$. $(\mathcal{R}_i)_{i\in\mathcal{N}}$ are the reward functions for players $1, \ldots, |\mathcal{N}|$, where $\mathcal{R}_i : \mathcal{S} \times \mathcal{S} \times (\times_{i\in\mathcal{N}} \mathcal{A}_i) \to \mathbb{R}$.

$(\mathcal{O}_i)_{i\in\mathcal{N}}$ denote the set of observations of players $1, \ldots, |\mathcal{N}|$ and $\mathcal{Z}(o_{t+1}, s_{t+1}, t) = \mathbb{P}[o_{t+1}|s_{t+1}, a_t]$ is the observation function, where $o_{t+1} \in \times_{i\in\mathcal{N}} \mathcal{O}_i$, $s_{t+1} \in \mathcal{S}$, and $a_t \in \times_{i\in\mathcal{N}} \mathcal{A}_i$. Lastly, $\gamma \in (0, 1]$ is the discount factor, $\rho_0 : \mathcal{S} \to [0, 1]$ is the initial state distribution, and T is the time-horizon.

The game theoretical solution concepts introduced for normal form and games can also be extended to partially observed stochastic games. This include the concepts of strategies, strategy profiles, best-response strategies,

and Nash equilibrium. Furthermore, every finite-horizon POSG and zero-sum POSG has a Nash equilibrium [38, 32, 34, 39, 40].

3.2.2 Belief States and One-sided POSGs

Since the players in a POSG cannot observe the exact state the game is in, the players can instead use the observations o_t to form a *belief* of what current state the partially observed stochastic game is currently in [41].

A belief state $b_t \in \mathcal{B}$ is a probability distribution over the states, so we can define the belief of player i at time t, $b_t^i \in \mathcal{B}_i$ as

$$b_t^{(i)}(s_t, \Pi_{-i}) = \mathbb{P}[s_t, \Pi_{-i} | h_t^i]$$

and $\mathcal{B}_i$ is the unit $(|\mathcal{S} \times \Pi_{-i}| - 1)$-simplex [42]. For example, if the defender believes that there is equal probability of the state of intrusion and no intrusion in some timestep, then the belief is $b_t = [0.5, 0.5]$. The belief a sufficient statistic of the state s_t and the strategies of the other players based on player i's history h_t^i of the initial state distribution, the actions, and its observations: $h_t^i = (\rho_1, a_1^{(i)}, o_1^{(i)}, \ldots, a_{t-1}^{(i)}, o_t^{(i)}) \in \mathcal{H}_i$.

In practice, computing the belief state b_t^i is computationally intractable due to the problem of *nested beliefs* [40]. It refers to the problem that, to compute the belief states, each player must compute a probability distribution over the state space and over the other players' strategies, which depends on their beliefs, which depend on the other player's belief, and so on in an infinite recursion.

To circumvent the nested beliefs problem, additional structure can be imposed on the POSG. In particular, a POSG is called *one-sided* if $\mathcal{N} = \{1, 2\}$ and one player has perfect information. That is, only one player has partial information [36, 40].

In one-sided POSGs, the player with perfect information observes both the state of the game and the history of actions and observations of the other player. Hence, it does not require a belief state. Therefore, the player with partial information can compute the belief b_t^i without having to consider the belief of the opponent. Specifically, let $\pi_2 : \mathcal{S} \to \Delta(A)$ denote the one-stage strategy of the player with perfect information. Then, the belief state of the

player with partial information can be computed recursively as follows:

$$b_{t+1,\pi_2}(s_{t+1}) = C \sum_{s_t \in \mathcal{S}} \sum_{a_t^{(2)} \in \mathcal{A}_2} \Bigg(\mathcal{Z}(o_{t+1}, s_{t+1}, (a_t^{(1)}, a_t^{(2)})) \\ \mathcal{T}(s_{t+1}, s_t, (a_t^{(1)}, a_t^{(2)})) b(s_t) \pi_2(a_t^{(2)} | s_t) \Bigg) \tag{3.2}$$

where $C = 1/\mathbb{P}[o_{t+1}|a_1^{(1)}, \pi_2, b_t]$ is a normalizing factor independent of s_{t+1} to make b_{t+1} sum to 1 [28].

3.2.3 Solving for Best Response Strategies in POSGs

We now want a mathematical formulation of the POSG so that we can find each players' best response strategies, as defined in Section 3.1.2. Since a partially observed stochastic game is an extension of a partially observed Markov decision process, we can use Markov decision theory to formulate this objective.

Let π_i be the current strategy player by player i, π_{-i} the fixed strategies played by the other players, and $\mathbb{E}_{(\pi_i, \pi_{-i})}$ denotes the expectation under the strategy profile (π_i, π_{-i}). Then the best response strategy would be the strategy π_i^* which satisfy the following equation:

$$\pi_i^* \in BR_i(\pi_{-i}) = \arg\max_{\pi_i \in \Pi_i} \mathbb{E}_{(\pi_i, \pi_{-i})} \left[\sum_{t=1}^{T} \gamma^{t-1} r_t^{(i)} \right]. \tag{3.3}$$

Specifically, this is the strategy which maximizes the sum of discounted rewards of player i. Now we can define the *state-value* of a certain strategy in a POSG, i.e. the expected value of the strategy with current belief b_t at time t, as

$$V_\pi(b_t) = \mathbb{E}_\pi \big[\sum_{k=0}^{T} \gamma^k r_{t+k+1} \big].$$

and furthermore the *action-value* of a strategy, meaning the value of action a when in belief b in time t, as

$$Q_\pi(b_t, a_t) = \mathbb{E}_\pi \big[\sum_{k=0}^{T} \gamma^k r_{t+k+1} \big]$$

In the case of a one-sided POSG, the equations for the player with full observability contain s_t instead of b_t.

A *stationary* strategy is a strategy which does not depend on time t, i.e. $\pi = \pi_1 = \pi_2 = ...$ Using Markov theory, we can derive the so called ***Bellman equation*** [43] for the stationary strategy, stated as follows:

$$V_\pi(b_t) = \mathbb{E}\big[r_{t+1} + \gamma V(b_{t+1}) | b_t\big].$$

This means that the state-value of a strategy in time t is the reward we can obtain at that time plus the discounted state-value of the next state. This is a fixed-point equation since both the left and right-hand side of the equation depend on the state value V_π.

Each players' goal in an POSG is to maximize the expected discounted return of rewards. Thus, the optimal state-value is $V^*(b) = \sup V_\pi(b)$. Using similar derivations as for the Bellman equation, we can derive the Bellman optimality equations for the POSG. To solve specifically for the best response strategy, we take into consideration the opponents strategies π_{-i} :

$$V^*_{i,\pi_{-i}}(b_t) = \max_{a_t \in \mathcal{A}_i} \mathbb{E}_{\pi_{-i}, b_t}\big[r_{t+1} + \gamma V^*_{\pi_{-i}}(b_{t+1}) | s_t, a_t\big] \tag{3.4}$$

$$Q^*_{i,\pi_{-i}}(b_t, a_t) = \mathbb{E}_{\pi_{-i}, b_t}\big[r_{t+1} + \gamma V^*_{\pi_{-i}}(b_{t+1}) | s_t, a_t\big] \tag{3.5}$$

$$BR_i(\pi_{-i}) = \arg\max_{a_t \in \mathcal{A}_i} Q^*(b_t, a_t). \tag{3.6}$$

Here, $V^*(s_t)$ and $Q^*(s_t, a_t)$ denote the expected cumulative discounted reward under π^* for each state and state-action pair, respectively. Solving the Bellman equations (Eqs. 3.4-3.5) means computing the value functions from which an optimal strategy can be obtained (Eq. 3.6). Hence, BR_i is obtained from the Bellman equations given a fixed strategy from the opponents. Thus, solving for the best response strategy is equivalent to solving a partially observed Markov decision process. Going in the other direction, one can model a POMDP as a one-sided partially observed game by adding a perfectly-informed but completely ineffective player, i.e. by preventing this player from affecting the transitions and rewards of the model [44].

Finally, if the POSG is zero-sum, it has a unique value function, $V^*(b) = V^*_{1,\pi^*_2}(b) = V^*_{2,\pi^*_1}(b)$ [33, 34, 40]. Further, if the POSG is one-sided, V^* is characterized by the following Bellman-style equation:

$$V^*(b) = \max_{\pi_1 \in \Delta(\mathcal{A}_1)} \min_{\pi_2 \in \Delta(\mathcal{A}_2)} \mathbb{E}_{\pi_1, \pi_2, b}\big[r_{t+1} + \gamma V^*(b_{t+1})\big]$$

where b_{t+1} is computed using Eq. 3.2.

3.2.4 Computing Nash Equilibria in POSGs

The methods to compute or approximate Nash equilibria in POSGs can be divided in two categories: the finite horizon case and the infinite horizon case. A finite-horizon POSG can be viewed as a type of extensive form game with imperfect information [45]. In this case, it's therefore possible to use computational game theory methods to find Nash equilibrium, such as linear or quadratic optimization [46, 47] or search algorithms [48, 49].

Computing equilibria in general infinite-horizon POSGs is computationally intractable due to the problem of nested beliefs (see Section 3.2.2) [40, 42]. Therefore, the research has focused on sub-classes of infinite-horizon POSGs that are tractable to solve. Two such sub-classes are one-sided POSGs and POSGs with public observations, which can be solved using dynamic programming and search-based algorithms [40, 40, 44, 50].

Lastly, *approximate* Nash equilibria in POSGs can be found by constructing a dynamic process that converges to an equilibrium in the limit, such as fictitious play processes [51, 46], regret minimization processes [52, 53], evolutionary processes [54, 55], and self-play reinforcement learning processes [37, 56, 57, 25, 21, 24]. In this project we focus on self-play reinforcement learning to find approximate Nash equilibria in an infinite-horizon POSG. This framework will be described further in the next chapter.

Chapter 4

Modeling the Use Case with Game Theory

In this section we address the first two research questions, i.e. we show how the intrusion prevention use case can be modeled as a game and what theoretical insights we can derived for this model. The game model is largely inspired from the intrusion prevention use case set up in [2], [3] and [4].

4.1 Intrusion Prevention through Optimal Stopping

Our game model builds on *optimal stopping* theory. Optimal stopping is the problem of choosing the time to take a specific action in order to maximize the expected reward. For a more elaborate theoretical background, we refer to [3]. In short, the game consists of two dependent stopping problems, one for the attacker and one for the defender.

In an optimal stopping problem, at each time step t of the decision process, two actions are available: "stop" (S) and "continue" (C). A *stop* action yields a reward $\mathcal{R}^{S}_{s_t,s_{t+1}}$ and if it is a terminal stop action, the process terminates. In the case of the *continue* action or a non-final stop action a_t, the decision process transitions to the next state according to some transition probabilities and yields a reward $\mathcal{R}^{a_t}_{s_t,s_{t+1}}$.

The defender has L stop actions. We associate each stop action of the defender with a defensive action. For example, this could be resetting user accounts, updating firewall configurations and ultimately blocking all incoming traffic.

The attacker has two stop actions. The first stop action of the attacker starts the intrusion and the second one ends it.

4.2 Game Structure

We model the intrusion prevention use case with a zero-sum Partially Observed Stochastic Game (POSG)

$$\Gamma^P = \langle \mathcal{N}, \mathcal{S}, (\mathcal{A}_i)_{i \in \mathcal{N}}, \mathcal{T}, (\mathcal{R}_i)_{i \in \mathcal{N}}, \gamma, \rho_0, T, (\mathcal{O}_i)_{i \in \mathcal{N}}, \mathcal{Z} \rangle.$$

We assume a one-sided POSG where the attacker is fully informed whereas the defender has partial observability. This assumption serves two purposes. First, it makes solving the game tractable [40]. Second, it reflects our goal of finding defender strategies that are robust against any attacker, including attackers with perfect information.

The game has two players:

$$\mathcal{N} = \{1, 2\}.$$

Player 1 is the defender and player 2 is the attacker.

The game state $s_t \in \{0, 1\}$ is zero if no intrusion is occurring and $s_t = 1$ if an intrusion is ongoing. In the initial state, no intrusion is occurring and $s_1 = 0$. Hence, the initial state distribution is the degenerate distribution

$$\rho_1(0) = 1.$$

Further, we introduce a terminal state $\emptyset \in \mathcal{S}$, which is reached when the game ends. Thus, the set of states is

$$\mathcal{S} = \{0, 1, \emptyset\}.$$

4.3 Actions $\mathcal{A}$ and Reward Function $\mathcal{R}(s_t, (a_t^{(1)}, a_t^{(2)}))$

The defender has two actions: "stop" (S) and "continue" (C), where each stop action is associated with a defensive action. Hence,

$$\mathcal{A}_1 = \{S, C\}.$$

The defender has to perform L stop actions during an episode to prevent an intrusion where $L \geq 1$ is a public parameter of our use case that is known to both the attacker and the defender.

Let $\mathcal{A}_2(s)$ denote the set of actions of the attacker in state s. When $s_t = 0$

the attacker has two options: "attack" (A) and "wait" (W). Hence,

$$\mathcal{A}_2(0) = \{A, W\}.$$

When $s = 1$, it can either choose to continue with the intrusion (C) or to stop and abort the intrusion (S):

$$\mathcal{A}_2(1) = \{S, C\}.$$

The objective of the defender in the intrusion prevention use case is to maintain service on the infrastructure while, at the same time, preventing a possible intrusion at minimal cost. Therefore, we define the reward function of the defender to give the maximal reward if the defender maintains service until the intrusion starts and then prevents the intrusion by taking L stop actions. Further, we assume that the attacker's goal is the direct opposite of the defender's goal. That is, they play a zero-sum game where $\mathcal{R}_2(s_t, a_t) = -\mathcal{R}_1(s_t, a_t)$. Hence, it is sufficient to define the reward of the defender: $\mathcal{R}(s_t, a_t) = \mathcal{R}_1(s_t, a_t)$.

The defender's reward per time step $\mathcal{R}(s_t, a_t)$ is parameterized by the reward that the defender receives for stopping an intrusion, R_{st}, the reward that the defender receives for maintaining service R_{sla}, the cost of stopping R_{cost}, and the loss of being intruded R_{int}:

$$\mathcal{R}(\emptyset, \cdot) = 0 \tag{4.1}$$
$$\mathcal{R}(0, (C, W)) = \mathcal{R}(1, (C, S)) = R_{sla} \tag{4.2}$$
$$\mathcal{R}(0, (C, A)) = R_{sla} + pR_{st}/L \tag{4.3}$$
$$\mathcal{R}(0, (S, W)) = \mathcal{R}(1, (S, S)) = R_{cost}/L \tag{4.4}$$
$$\mathcal{R}(0, (S, A)) = \mathcal{R}(1, (S, C)) = R_{cost}/L + R_{st}/L \tag{4.5}$$
$$\mathcal{R}(1, (C, C)) = R_{sla} + R_{int}. \tag{4.6}$$

Eq. 4.1 states that the reward in the terminal state is zero (probability p is defined in the next subsection). Eqs. 4.2-4.3 indicate that the defender receives a positive reward for maintaining service. Eqs. 4.4-4.5 state that each stop incurs a cost and possibly a reward if it affects an ongoing intrusion. Lastly, Eq. 4.6 indicates that the defender receives a loss for each time step that it is under intrusion.

4.4 Transition Operator $\mathcal{T}$ and Time Horizon $T_\emptyset$

The state transitions are stochastic and depend on the actions of both the attacker and the defender. In any time step, there is a small probability that the attacker is detected by the defender and the game ends, regardless of the actions taken. Specifically, the probability that the attacker is detected and the game ends is defined by a Bernoulli random variable $Q \sim Ber(p = 0.05)$.

We define the time-homogeneous transition operator $\mathcal{T}_{l_t}(s_{t+1}, s_t, a_t) = \mathbb{P}_{l_t}[s_{t+1}|s_t, a_t]$ as follows:

$$\mathcal{T}_{l_t>1}(0, 0, (S, W)) = \mathcal{T}_{l_t}(0, 0, (C, W)) = 1 - p \tag{4.7}$$
$$\mathcal{T}_{l_t>1}(1, 1, (\cdot, C)) = \mathcal{T}_{l_t}(1, 1, (C, C)) = 1 - p \tag{4.8}$$
$$\mathcal{T}_{l_t>1}(1, 0, (\cdot, A)) = \mathcal{T}_{l_t}(1, 0, (C, A)) = 1 - p \tag{4.9}$$
$$\mathcal{T}_{l_t>1}(\emptyset, 0, (\cdot, W)) = \mathcal{T}_{l_t>1}(\emptyset, 1, (\cdot, C)) = p \tag{4.10}$$
$$\mathcal{T}_{l_t>1}(\emptyset, 0, (\cdot, A)) = \mathcal{T}_{l_t}(\emptyset, 0, (C, A)) = p \tag{4.11}$$
$$\mathcal{T}_{l_t}(\emptyset, 1, (C, C)) = p \tag{4.12}$$
$$\mathcal{T}_1(\emptyset, 0, (S, W)) = \mathcal{T}_1(\emptyset, 0, (S, A)) = 1 \tag{4.13}$$
$$\mathcal{T}_{l_t}(\emptyset, 1, (\cdot, S)) = \mathcal{T}_1(\emptyset, 1, (S, C)) = \mathcal{T}_{l_t>1}(\emptyset, \emptyset, \cdot) = 1. \tag{4.14}$$

All other state transitions have probability 0.

Eqs. 4.7-4.8 define the probabilities of the recurrent state transitions: $0 \to 0$ and $1 \to 1$. Specifically, the game stays in state 0 or with probability $1 - p$ if the attacker selects the wait action W and $l_t - a_1 > 0$. Similarly, the game stays state 1 with probability $1 - p$ if the attacker chooses the continue action C and $l_t - a_1 > 0$.

Eq. 4.9 captures the transition $0 \to 1$. Specifically, the game transitions to the intrusion state with probability $1-p$ if the attacker chooses the attack action A and $l_t - a_1 > 0$. Finally, Eqs. 4.10-4.14 define the transition probabilities to the terminal state $\emptyset$. The terminal state is reached in three cases: when $l_t = 1$ and the defender takes the final stop action S (i.e. $l_t - a_t = 0$), when the attacker takes the stop action S and aborts an ongoing intrusion, and when the attacker is detected by the defender by chance with probability p.

With this definition of the transition probabilities, the evolution of the system can be understood using the state transition diagram in Fig. 4.1.

The time horizon $T_\emptyset$ is a random variable that indicates the time t when the terminal state $\emptyset$ is reached and the game ends. Since the probability that the

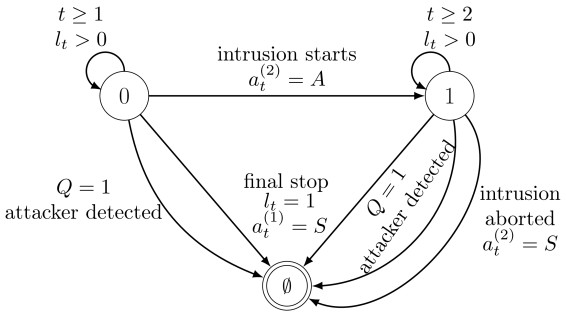

Figure 4.1: State transition diagram of the POSG: each circle represents a state; an arrow represents a state transition; a label indicates the event that triggers the state transition; a game episode starts in state $s_1 = 0$ with $l_1 = L$. Image courtesy of Kim Hammar.

game ends in any state except the initial state is non-zero ($p > 0$), it follows that $\mathbb{E}_\pi [T_\emptyset] < \infty$ for any strategy profile $\pi \in \Pi$. (Remark: it is also possible to define $T = \infty$ and let $\emptyset$ be an infinitely absorbing state.)

4.5 Observations $\mathcal{O}$ and Observation Function $\mathcal{Z}(o_{t+1}, s_{t+1}, (a_t^{(1)}, a_t^{(2)}))$

The defender has a partial view of the game and if $s_t \neq \emptyset$ it observes $o_t = (l_t, \Delta x_t)$. Here, $l_t \in \{1, 2, \ldots, L\}$ is the number of stops remaining and Δx_t denote the number of severe IDS alerts generated during time step t, respectively. If the game is in the terminal state, the defender observes $o_T = \emptyset$.

In this project, we consider two cases of the observation function. Firstly, we limit the observation space, and base the observations and the observation function on a simple intuitive heuristic, as described in Section 6.1.1. This is to create a benchmark, and to test if the algorithm is working in this simple case.

In the second part of the analysis, we use an observation function closer to the real world application. Here, we assume that the number of IDS alerts generated during one time step is a random variable $X \sim f_X$ that depends on the state. These observations are generated using a real emulation of the IT infrastructure. Consequently, the probability that Δx severe alerts occur during time step t can be expressed as $f_X(\Delta x | s_t)$.

We define the time-homogeneous observation function $\mathcal{Z}(o_{t+1}, s_{t+1}, a_t) =$

$\mathbb{P}[o_{t+1}|s_{t+1}, a_t]$ as follows:

$$\mathcal{Z}\big((l_t, \Delta x), 0, \cdot\big) = f_X(\Delta x|0) \tag{4.15}$$
$$\mathcal{Z}\big((l_t, \Delta x), 1, \cdot\big) = f_X(\Delta x|1) \tag{4.16}$$
$$\mathcal{Z}\big(\emptyset, \emptyset, \cdot\big) = 1 \tag{4.17}$$

4.6 Strategy Space Π and Objective

Since the POSG is stationary and the time horizon $T_\emptyset$ is not pre-determined, it is sufficient to consider stationary strategies. We consider the space of stationary strategy profiles $\Pi = \Pi_1 \times \Pi_2$ where Π_1 and Π_2 denote the strategy space of the defender and the attacker, respectively.

The goal of the defender and the attacker is to maximize, respectively minimize, the expected discounted cumulative reward over the horizon $T_\emptyset$. Hence, the best response functions BR_1 and BR_2 are defined as follows:

$$BR_1(\pi_2) = \arg\max_{\pi_1 \in \Pi_1} \mathbb{E}_{(\pi_1,\pi_2)} \left[\sum_{t=1}^{T_\emptyset} \gamma^{t-1} \mathcal{R}_1(s_t, a_t) \right] \tag{4.18}$$

$$BR_2(\pi_1) = \arg\min_{\pi_2 \in \Pi_2} \mathbb{E}_{(\pi_1,\pi_2)} \left[\sum_{t=1}^{T_\emptyset} \gamma^{t-1} \mathcal{R}_1(s_t, a_t) \right]. \tag{4.19}$$

We set the discount factor to be $\gamma = 1$. (Eqs. 4.18-4.19 are bounded when $\gamma = 1$ since $\mathbb{E}_{\pi_\theta}[T_\emptyset]$ is finite for any strategy profile $\pi \in \Pi$.)

The objective of our intrusion prevention use case is to find a robust defense strategy which provides the optimal defense against the worst-case attacker and to find the equilibrium outcome of the game. The problem of finding such strategies corresponds to finding a strategy pair (π_1^*, π_2^*) that satisfy:

$$\pi_1^* \in BR_1(\pi_2^*) \text{ and } \pi_2^* \in BR_2(\pi_1^*) \tag{4.20}$$

where π_1^* is the optimal defender strategy against the worst-case attacker and (π_1^*, π_2^*) is a Nash equilibrium. Further, the value v of the game is:

$$v = \mathbb{E}_{(\pi_1^*,\pi_2^*)} \left[\sum_{t=1}^{T_\emptyset} \gamma^{t-1} \mathcal{R}_1(s_t, a_t) \right]. \tag{4.21}$$

Eqs. 4.18-4.21 define an optimization problem which reflects the objective of our use case. In the following sections, we analyze the structure of the solution

to this problem using game theory and present our approach for computing the solution using reinforcement learning.

4.7 Game-Theoretic Analysis of the Intrusion Prevention Game Model

In this section we address the second research question. Thus, we derive structural insights for the game model, which include showing that a Nash equilibrium exists, and describe the properties of the pure Nash equilibrium of the game. Furthermore, we show that there exists an optimal intrusion prevention strategy of a thresh-holding structure.

Theorem 4.7.1 (Structural Insights for the Intrusion Prevention Game)**.** *Given the partially observed stochastic game defined in Sections 4.2-4.6 and L stops for the defender, the following holds:*

A. *A Nash equilibrium exists for the game.*

   Proof of theorem A. The proof that every one-sided partially observed stochastic game with finite horizon or infinite horizon with $\gamma \in (0, 1]$ has a Nash equilibrium is available in [44]. The idea behind the proof is that a POSG with finite horizon can be modeled as an extensive form game. For an extensive form representation of the intrusion prevention game, see Appendix A. □

B. *In any pure Nash equilibrium the attacker takes stopping action S in both state $s_t = 0$ and $s_t = 1$ for any $b \in B$. Any equilibrium where the attacker follows a different strategy is mixed.*

   Proof. See Appendix B.1. The idea behind the proof builds on the fact that the defender reciever R_{sla}, meaning that the attacker has an incentive to attack faster, and that a pure strategy is simpler to defend against (from the defender perspective). □

C. *For any attacker strategy π_2, if f_x is TP2, there exist L values of $\alpha_1^* \dots \alpha_L^*$ that are decreasing and a best response strategy π_1^* for the defender that satisfies:*

$$\pi_{1,l}(b) = S \iff b \geq \alpha_l^*, l \in (1 \dots L).$$

Proof. See Appendix B.2. The idea behind the proof is that the intrusion prevention POSG builds on the POMDP model defined in [4], where a similar structural property was proven for the POMDP model. □

Theorem 4.7.1 A. states that there exists a Nash equilibrium, meaning that there exist best response strategies for both the defender and the attacker in the defined game model. This means that we can use methods to find Nash equilibrium strategies to find automated defense strategies.

Theorem 4.7.1 B. states that the only pure Nash equilibrium that exists is when the attacker starts and stops the game immediately. This means that when learning equilibrium strategies through reinforcement learning, we can expect most strategies to be mixed.

Theorem 4.7.1 C states that there exist best response strategies of the defender with a specific structure. The interpretation is as follows. There exists a defender best response stopping strategy that is a threshold strategy where if there are l stops remaining, then the defender will stop for any belief $b \geq \alpha_l^*$. Since these α_l are decreasing it means that the expected stopping threshold is lower the more stops the defender has left. As pointed out in [4], knowing that there exists optimal strategies with a special structure has two benefits. Firstly, this insight can lead to a more efficient implementation of the strategies. Secondly, the complexity of learning an optimal strategy can be reduced.

Chapter 5

Finding Automated Strategies using Reinforcement Learning

In this chapter, we describe our reinforcement learning approach. We address the third research question and rationalize the choice of Neural Fictitious Self Play as the reinforcement learning algorithm to compute automated intrusion prevention strategies. We also describe the implementation of the algorithm and our evaluation method.

5.1 Reinforcement Learning

Reinforcement learning is a branch of machine learning where agents learn optimal behavior based on a trial-and-error procedure by interacting with the environment. Within the environment, the agent receives a reward (reinforcement) depending on which action is taken [58, 43]. The reinforcement learning procedure can be summarized in the following steps:

- The agent observes the environment
- The agent decides what to do according to some strategy
- The agent receives some reward for this action
- The agent learns from this experience and improves their strategy
- The process is repeated until an optimal strategy is found

The process can be visualized as in Figure 5.1.

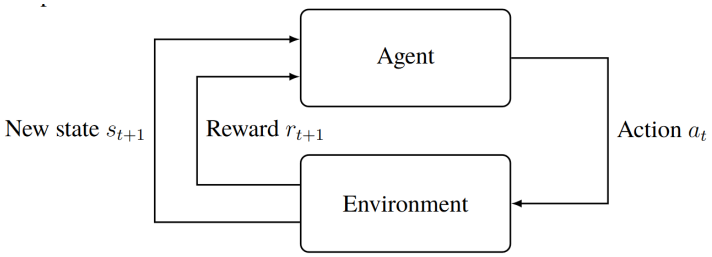

Figure 5.1: The reinforcement learning process visualized.

The reinforcement learning process can be modeled with a Markov Decision Process (MDP) if the state space is observable, or with a Partially Observed Markov Decision Process (POMDP) if the state space is not fully observable [58, 43]. The objective of reinforcement learning is to find the optimal strategy, or *policy*, in the Markov decision process.* In this project, the intrusion prevention use case will be modeled as a stochastic game; however, as described in Section 3.2.3, solving for the best response strategy is equivalent to solving a POMDP. Therefore, reinforcement learning becomes a relevant methodology even in the multi-agent case. Reinforcement learning algorithms generally have no guarantees to converge to an optimal policy except for special cases [59, 60].

We separate between some different classes of RL algorithms. Firstly, we can separate between *model-free* versus *model-based* algorithms. In a model-free algorithm, we are not trying to model the environment, while a model-based algorithm tries to learn the transition or observation probabilities of the model [43]. Among the model-free algorithms, we can separate between *value-based algorithms* and *policy-based algorithms*. Value-based algorithms tries to approximate solutions to the Bellman equations, while policy-based algorithms tries to search through the policy space using gradient-based methods. The algorithms can also be combined, e.g. through *actor-critic* algorithms.

We also separate between *tabular reinforcement learning* and *deep reinforcement learning* algorithms. Tabular reinforcement learning refer to methods in which the approximate value functions can be stored in arrays and tables. Deep reinforcement learning incorporates deep learning, which allows agents to make decisions by approximating value functions using neural networks.

* By convention the term *policies* is used in Markov decision theory and *strategies* is used in game theory. These two can be used interchangeably, although we try to use the game-theoretical term as much as possible in this thesis.

5.1.1 Q-learning

Q-learning [61] is a form of tabular, model-free reinforcement learning algorithm, presented first in 1989 in a PhD thesis by Christopher Watkins. It was a major breakthrough in reinforcement learning as it was the first algorithm with guaranteed convergence to the optimal policy [59]. By trying all states repeatedly, and thereby training the algorithm, the agent can learn which strategy yields the highest discounted reward. The Bellman optimality equation assures there exists an optimal state-value $V^*(s)$ for the optimal policy π^*.

For some policy π, we define the action values (Q-values), meaning the value of action a when in belief b in time t, as

$$\begin{aligned} Q_\pi(s_t, a_t) =& \mathbb{E}_\pi\big[\sum_{k=0}^{T} \gamma^k r_{t+k+1}\big] \\ &= \mathbb{E}_\pi\big[r_{t+1} + \gamma V(s_{t+1}) | s_t, a_t\big] \end{aligned}$$

In other words, the Q-value is the expected discounted reward for executing action a at state s given that we follow policy π thereafter. By showing that $V^*(s_t) = \sup_{a_t} Q^*(s_t, a_t)$ we have that the optimal policy can be found by picking the action corresponding to the largest Q-value (that we call a^*), so $\pi^*(s) = a^*$. For the Q-learning algorithm to find the optimal Q^* values, Watkins suggests the following learning rule:

$$Q_t(s, a) = (1 - \alpha_t) Q_{t-1}(s, a) + \alpha_t [r_t + \gamma V_{t-1}(s_t)]$$

where $\gamma V_{t-1}(s_t)$ is the best action the agent think it can commit in the next state s_t using the old q-table from Q_{t-1}. By training the algorithm long enough, the values of $Q_t(s, a)$ are guaranteed to converge to Q^* [59]. However, because this is a tabular algorithm, the required memory space is correlated to the agent's observation space and action space. Therefore Q-learning quickly becomes intractable for larger problems.

In Q-learning, the agent always chooses the best action based on the highest reward. However, the agent initially has zero knowledge of the environment. Therefore there is a risk that the agent will quickly find and get stuck with a sub-optimal policy if it just commits to the action choices based on exploitation of what it already knows. This is a *trade-off between exploitation and exploration* to find the globally optimal strategies of the game. A common exploration strategy is the *epsilon-greedy* exploration strategy. With this strategy, the agent will most of the time select the action with the highest estimated reward.

However, with some small probability $p = \epsilon$, the agent will pick a random action in the agent's action space. The ϵ parameter is often decreased over training episodes as exploration becomes less important.

5.1.2 Deep Q-networks

The Deep Q-network (DQN) combines the logic of basic tabular Q-learning with function approximation using neural networks [62]. The goal for the neural network, just like for Q-learning, is to find the optimal action-value function $Q^*(s,a) = \sup_\pi \mathbb{E}_\pi\big[\sum_{k=0}^{T} \gamma^k r_{t+k+1}\big]$.

The authors proposed the use of a deep convolutional neural network, in combination with several problem specific techniques described below, to create the deep Q-network.

As the focus of this report is reinforcement learning, the basics of neural networks and convolutional neural networks will only be described briefly. A deep neural network is a sequential model structure in which mathematical nodes are connected in several layers, which enables non-linear function approximation, given that each node has a non-linear activation function. A convolutional neural network is a neural network architecture which has been particularly successful, especially for tasks such as image processing, object classification and other two dimensional signal processing. A convolutional neural network uses convolutional layers, which are layers based on convolution and local receptive fields, which means that each node of the network takes more data into consideration. Essentially, the task for the deep Q-network is to use the information from the agent and the environment (a, r, s, s') to improve the agent's strategy and maximize its rewards.

The agent is trying to find the optimal Q values

$$Q^*(s_t, a_t) = \mathbb{E}_\pi\big[r_{t+1} + \gamma \sup_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t, a_t\big].$$

and we approximate this $Q^*(s,a) \approx Q(s,a;\theta)$ with a convolutional neural network where θ is the network weights.

At iteration i, if the optimal target values at last iteration is given by $r + \gamma \sup_a Q^*(s,a;\theta_i^-)$ using parameters θ_i^- from the last iteration, then the squared loss function can be calculated as

$$L_i = (Q^*(s,a)_i - Q(s,a)_i)^2 = (r + \gamma \sup_a Q^*(s,a;\theta_i^-) - Q(s,a;\theta_i))^2.$$

This means that the target function depends on the network weights, in contrast to supervised learning where the targets are fixed.

Because of the online updating of the target function used in the loss, the DQN also uses a *target network* to improve the stability of the training. The target network is a clone of the training network Q, and it is used to generate the Q^* values used in the loss function. Every fixed amount of iterations, the target network is updated with the newest weights θ from Q. Generating the loss target values using an older set of parameters creates a delay between the update of the Q network and the update of the target values, meaning that the loss function becomes less likely to diverge.

Another method to improve the effectiveness of the DQN is a tecnique called *experience replay*. What it means is that the agent's experience at each time step (the state, action, reward and next state) i.e. $e_t = (s_t, a_t, r_t, s_{t+1})$ is stored in a dataset D. Training in the DQN is then done by drawing a sample uniformly from the data set D, instead of using the newest experience. This breaks the correlation of the data points used for training and also improves data efficiency, as the same data point can be used several times to train the network.

5.1.3 Single-agent versus Multi-agent Reinforcement Learning Algorithms

A *Multi-Agent Reinforcement Learning* (MARL) scenario is a context where several agents have to improve their strategies at the same time, such as in the intrusion prevention use case in this report. Researchers have attempted to simply extend the existing reinforcement learning methods (such as the DQN) with limited success. The main reason for this is that the environment, from each agent's perspective, is no longer stationary [26]. For example, in a stationary environment, an agent making a certain action in a certain state will learn the probabilities of the different possible outcomes and reward when trained over many episodes. But in a multi-agent environment, these probabilities will change, as the other agents adapts their strategy in relation to the first agent.

For this reason, we explored different multi-agent reinforcement learning algorithms, as a simple single-agent reinforcement learning algorithm would not effectively be able to handle the multi-agent case. One state of the art multi-agent reinforcement learning algorithm, that builds on a game theoretical approach, is the *Neural Fictitious Self Play* (NFSP) algorithm. Neural fictitious self play is based on *fictitious play* and *fictitious self play*, so we

will first introduce these algorithms.

5.2 Fictitious Play, Fictitious Self Play and Neural Fictitious Self Play

Fictitious play [51] is an iterative method for learning Nash equilibria in normal form games. In fictitious play, the agents choose the best response to the opponents' average behavior. After updating their strategy, the players play each other another time, until the average strategy converges. It has been shown that in for example two-player zeros-sum games, the average strategy profile of the fictitious players converges to a Nash equilibrium. In [63], a general version of the fictitious play is defined, called the Generalized Weakened Fictitious Play. It shares convergence guarantees with normal self play, but allows for approximate best responses and average strategy updates.

Fictitious Self Play (FSP) [22] is a machine learning framework for fictitious play, to make it possible for use in extensive form games. Because it builds on Generalized Weakened Self Play, it is possible to use approximate best response and average response calculations.

There are two machine learning components of the fictitious self play. For the approximate best responses, the authors let each agent learn the best response through reinforcement learning. The average strategy can be approximated using some supervised learning model. Each episode of the general machine learning framework and algorithm can be summarized in three steps.

- In the first step, we simulate episodes by playing the agents' strategies against each other. The resulting experience or data is stored in two types of agent memory: $\mathcal{M}_{RL}$ and $\mathcal{M}_{SL}$. $\mathcal{M}_{RL}$ stores experience of the opponents' action. $\mathcal{M}_{SL}$ stores the agents' own behavior.
- In the second step, each agent uses some reinforcement learning method to approximate a best response by using the experiences stored in $\mathcal{M}_{RL}$
- In the last step of each iteration, the agent uses supervised learning to update its own average strategy, using experience stored in $\mathcal{M}_{SL}$.

Neural Fictitious Self Play (NFSP) is a deep reinforcement learning algorithm introduced by Heinrich et al. in 2017 [21]. NFSP uses the logic of the Fictitious Self Play framework in combination with neural networks to learn approximate Nash Equilibrium strategies. Each agent in NFSP

uses two separate deep learning neural networks; one deep Q-network using reinforcement learning to learn the approximate best responses, and one network using supervised learning to learn the average strategy. As in the FSP, the agent will behave according to a mixture of the average strategy and the best response strategy, and the average strategy will approximate the Nash Equilibrium strategy. In this subsection some components of the NFSP algorithm will be described. The complete pseudocode and implementation of NFSP adapted to our use case is described in Section 5.3.

NFSP uses Deep Q-learning as reinforcement learning algorithm. Deep Q-learning is a relatively simple reinforcement learning algorithm, and is deemed suitable in the NFSP framework. This means that the NFSP automatically incorporates several of the components described in Section 5.1.2, as one of the networks trained in the NFSP will be the deep Q-network. This includes components such as the use of a target network, the experience replay methodology, the ϵ-greedy exploration strategy as well as the same loss function used by the DQN.

Two other components used in NFSP are *reservoir sampling* and *average network dynamics*. Reservoir sampling [64] is an algorithm to choose a random sample, without replacement, from a population of unknown size. The supervised learning in the NFSP uses reservoir sampling to store the experience (a_t, s_t) when following the approximate best response strategy. At each iteration, the average network is fed a data sample from the reservoir to train the average response. This is done by optimizing the log-probability of past actions taken.

Furthermore, the authors suggest a discrete-time approximation of a method called *anticipatory dynamics*, to properly sample best response strategies and the average strategies. In theory, the agents could only play the average strategies against each other to learn an approximate equilibrium. However, the agents need to play the approximate best response strategy to fill the reservoir memory $\mathcal{M}_{SL}$, which is needed to train the average network. Thus, the authors suggest the use of an anticipatory parameter. The anticipatory parameter is the probability that the best response strategy is played. This means that with some strategy η the agent chooses the best response strategy and samples from the DQN, and with some probability $1-\eta$ the agent samples from the average network.

5.3 Choice and Implementation of Main Algorithm

In this project, we choose to implement the NFSP algorithm to learn approximate Nash Equilibrium strategies in the Intrusion Prevention Game. There are four main reasons for this. Firstly, the NFSP algorithm shows state-of-the art results in the original paper, when evaluated on imperfection-information games such as Leduc Poker and Limit Texas Hold'em poker. NFSP vastly outperformed standard reinforcement learning algorithms such as the simple DQN [21]. Secondly, NFSP is designed for games with imperfect information, which is a requirement for the intrusion prevention game [21]. Thirdly, because the NFSP algorithm builds on deep reinforcement learning and neural network approximation, it can more effectively handle large state and observation spaces compared to for example certain computational game theoretic approaches, such as counterfactual regret minimization [65] which does not scale to our use case. Fourthly, because of the game-theoretical foundation of the Fictitious Self Play and Neural Fictitious Self Play algorithms, the results can still be discussed and interpreted from a game-theoretic perspective, although we are using reinforcement learning instead of computational game theory.

The general algorithm of FSP and NFSP is described in Section 5.2. Below we describe certain adjustments to the algorithm, to take the partial observability of the intrusion game into account. Furthermore, we present the pseudocode for the implementation of the main algorithm.

To generate the defender observation, we have to implement the observation function $\mathcal{Z}(o_{t+1}, s_{t+1}, a_t) = \mathbb{P}[o_{t+1}|s_{t+1}, a_t]$. For the first benchmark evaluations, we use a simple discrete probability distribution to describe this stochastic correlation. Thereafter, we implement an observation function using traces from real hacking attacks to generate a continuous observation value. In both cases, the value from the observation function is used to calculate the belief of the defender player.

As described in the mathematical background, the belief of the defending player can be calculated recursively [28]:

$$b_{t+1,\pi_2}(s_{t+1}) = C \sum_{s_t \in \mathcal{S}} \sum_{a_t^{(2)} \in \mathcal{A}_2} \Big(\mathcal{Z}(o_{t+1}, s_{t+1}, (a_t^{(1)}, a_t^{(2)})) \\ \mathcal{T}(s_{t+1}, s_t, (a_t^{(1)}, a_t^{(2)})) b(s_t) \pi_2(a_t^{(2)}|s_t) \Big)$$

where $C = 1/\mathbb{P}[o_{t+1}|a_1^{(1)}, \pi_2, b_t]$ is a normalizing factor independent of s_{t+1} to make b_{t+1} sum to 1.

This recursive equation can be implemented in Python, by taking the specified elements defined in the equation multiplied with each other, an operation we can call a Bayesian filter. The belief must be updated every timestep, as it is dependent on the current observation and state as well as the most recent actions.

Furthermore, the belief update is dependent on the attacker stage strategy $\pi_2(a_t^{(2)}|s_t)$. The stage strategy is the probabilities of actions of the attacker at a given time step. The attacker stage strategy will be a 2x2-matrix with the current action probabilities for action (Continue, Stop) for the 2 states (No Intrusion, Intrusion). However, while the observation function $\mathcal{Z}$ and transition tensor $\mathcal{T}$ will be fixed over time, the stage strategy will be dependent both on which strategy the attacker is using (average or best response), the current state and also the current belief. The worst-case attacker will in practice know exactly what the current belief of the defender is and adapt their stage strategy depending on this. The stage strategy given a certain belief and state will also change with the training of the agent. Thus, to update the belief and the stage strategy, we do the following:

- Start with initial belief of intrusion 0 for the defender, $b_{0,\pi_2}(0) = 1, b_{0,\pi_2}(1) = 0$
- Evaluate the agents' actions $a_t^{(1)}, a_t^{(2)}$
- After each timestep, update the current attacker stage strategy $\pi_2(a_t^{(2)}|s_t, b_t)$ by evaluating the current attacker strategy network σ_2 (i.e., the strategy network used in this specific episode) for states s_t and belief b_t
- Apply a Bayesian filter to calculate $b_{t+1,\pi_2}(s_{t+1}|o_{t+1}, s_{t+1}, s_t, a_t^{(1)}, a_t^{(2)}, b(s_t), \pi_2(a_t^{(2)}|s_t))$ using equation 3.2

The complete pseudocode for our NFSP implementation, along with the belief update and stage strategy calculation, follows in algorithm 1.

Algorithm 1 Neural Fictitious Self Play implemented for the Intrusion Prevention use case

Initialize security game Γ and execute the agents through RunSecurityGame
procedure RunSecurityGame(Γ)
  for each agent $i = 1$ (defender), 2 (attacker) **do**
    Initialize replay memories $\mathcal{M}_{RL}$ and $\mathcal{M}_{SL}$
    Initialize average-strategy network $\Pi(s, a|\theta^{\Pi})$ with random weights θ^{Π}
    Initialize action-value network $Q(s, a|\theta^{Q})$ with random weights θ^{Q}
    Initialize DQN target network parameters $\theta^{Q'} \leftarrow \theta^{Q}$
    Initialize anticipatory parameter η
  end for
  while within computational budget, for many episodes **do**
    Set agent strategies $\sigma^{(i)} \leftarrow \begin{cases} \epsilon\text{-greedy(DQN) with probability } \eta \\ \Pi \text{ with probability } 1-\eta \end{cases}$
    Initialize state $s_0 = 0$, belief $b_0 = 0$ and reward $r_0 = 0$
    for timestep $t = 1...T$ **do**
      Sample actions $a_t^{(i)}$ from strategy $\sigma^{(i)}$
      Execute actions $a_t = (a^{(1)}, a^{(2)})_t$ in game at the same time
      Observe rewards r_{t+1}, next information state s_{t+1}
      Sample observation o_{t+1} from observation function $\mathcal{Z}(o_{t+1}, s_{t+1}, a_t)$
      for each agent **do**
        Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in reinforcement learning memory $\mathcal{M}_{RL}$
        if agent follows best response strategy $\sigma = \epsilon$-greedy (Q) **then**
          Store behaviour tuple (s_t, a_t) in supervised learning memory $\mathcal{M}_{SL}$
        end if
        Update weights θ^{Π} with stochastic gradient descent on loss:
        $\mathcal{L}(\theta^{\Pi}) = \mathbb{E}_{(s,a)\sim\mathcal{M}_{SL}}[-\log \Pi(s, a|\theta^{\Pi})]$
        Update weights θ^{Q} with stochastic gradient descent on loss:
        $\mathcal{L}(\theta^{Q}) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{M}_{RL}}[(r + \gamma \sup_{a'} Q(s', a'|\theta^{Q'}) - Q(s, a|\theta^{Q}))^2]$
        Periodically update DQN target network weights $\theta^{Q'} = \theta^{Q}$
      end for
      When both agents played:
      Update stage strategy $\pi_2(a_{t+1}^{(2)}|s_{t+1})$ by evaluating episode strategy σ_2
      Recursively update belief $b_{t+1,\pi_2}(s_{t+1}|o_{t+1}, s_{t+1}, s_t, a_t^{(1)}, a_t^{(2)}, b(s_t), \pi_2(a_t^{(2)}|s_t))$
      Evaluate stochastic game transition $\mathcal{T}$
      if terminal state reached **then**
        break current episode
      end if
    end for

  end while
end procedure

5.4 Evaluation of Multi-Agent RL Algorithms

5.4.1 Exploitability

One way to measure the closeness of a learned strategy to a Nash Equilibrium is to measure the *Nash Convergence* of the strategy [66]. If the game value for player i is gv_i, we define the worst case value for player i playing strategy π_i as $\delta_i = gv_i - v_i(\pi_i, BR(\pi_i))$. Intuitively, this is the difference of reward for player i when playing current strategy π_i, while opponent is playing the best response to that strategy, $BR(\pi_i)$. If player i is playing the Nash Equilbrium strategy, then this is $\delta_i = 0$. We can thus define the Nash convergence (NashConv) of a stategy profile as

$$\text{NashConv}(\pi) = \sum_i \delta_i(\pi_i).$$

We can also define a ϵ-Nash Equilibrium as one where $\max_i \delta_i(\pi) \leq \epsilon$. In a two player zero-sum game, we have that $gv_1 = -gv_2$. Thus we can define the *exploitability* of a player profile $\pi = (\pi_1, \pi_2)$ of a zero-sum game as

$$\text{Exploitability}(\pi) = \frac{\sum_i \delta_i(\pi_i)}{2} = \frac{v_1(BR(\pi_2), \pi_2) + v_2(\pi_1, BR(\pi_1))}{2}.$$

In other words, the exploitability of a strategy profile (π_1, π_2) can be calculated by fixing one of the strategies, calculating the average reward of a best response strategy to the fixed strategy, and thereafter following the same procedure for the other agent's strategy. A small exploitability or an exploitability close to 0 will indicate closeness to the Nash Equilibrium. A simple algorithm to calculate exploitability in algorithm 2.

Algorithm 2 Exploitabilty calculation in a two-player zero-sum game

input Strategy profile to $\pi = (\pi_1, \pi_2)$ to evaluate
output The exploitability value

procedure EXPLOITABILITY(π)
  for player i = 1, 2 **do**
    Calculate best response strategy $BR(\pi_i)$
    Calculate v_i = $v_i(\pi_i, BR(\pi_{-i}))$
  end for
  return $(v_1 + v_2)/2$

Evidently, the algorithm is dependent on two subroutines; one to calculate the best response of a strategy, and one to calculate the average reward when two agents with fixed strategies play against each other. In the next two subsections we will discuss these two subroutines briefly.

5.4.2 Approximate Best Response and Approximate Exploitability

The problem with the exploitability calculation is that it requires a calculation of the exact best response of a strategy. The calculation of the exact best response, in turn, requires a full tree traversal of the game tree. This is an intractable calculation for a game with large state space or action space, as the tree increases exponentially in size. Furthermore, in the context of this thesis, finding the best response for the defender is equivalent to solving a POMDP, which is PSPACE-complete [67].

To circumvent this problem, one can use standard reinforcement learning methods for learning the approximate best response. By fixing the strategy of one of the agents, the environment becomes a single-agent environment, since one of the players will not change their strategy in response to the other. The best response is the same as the optimal strategy in that single-player environment [66]. Thus, we can implement the following subroutine to find the (approximate) best response strategy:

Algorithm 3 Approximate Best Response calculation to strategy π_i

input Strategy profile π_i to exploit
output The approximate best response strategy $ABR(\pi_i)$
procedure APPROXIMATE BEST RESPONSE CALCULATION(π)
  Initialize $ABR(\pi_i)$ uniformely
  while within computational budget, for many episodes **do**
    initialize state s
    for each step of the episode, state s is not terminal **do**
      $a \leftarrow$ action given by $BR(\pi_i)$ for s
      take action a, observe r, s' given by fixed strategy π_i
      Use RL algorithm to improve $BR(\pi_i)$ given a, r, s, s'
      $s \leftarrow s'$
    end for
  end while
  return $ABR(\pi_i)$

We can run algorithm 2 to find the *approximate exploitability* using

the approximate best response, found by using algorithm 3. There are two problems with this approach. The first one is that the calculation of the approximate best response carries a high computational cost, as it requires training a reinforcement learning algorithm from scratch. The better approximation of the exploitability we want to find, the more episodes we need to run the learning algorithm. Thus, if we want to evaluate the convergence of an MARL algorithm as we are training it, it will increase the total training time of the MARL algorithm.

The second problem is that the approximate best response only will count as a lower bound of the exact exploitability. The exact exploitability is the exploitability calculated by a worst-case player. It is possible to reach this exact exploitability by training a reinforcement learning algorithm, depending on how long we train the approximate best responses, but in practice the exploitability will always be a approximation. Another way to phrase this is that the exact best response is going to be at least as good as the approximate best response learned in algorithm 3. Thus, if the approximate best response value is high, then it is an indication that the agent is weak, but if the approximate best response value is low, then we cannot draw any definitive conclusions. To evaluate the approximate best response, one can look at the average reward of the agent at each episode to see if that value converges within the computational budget.

5.4.3 Approximation of Average Episode Reward and Value of the Game

The second subroutine needed for the exploitability calculation is the calculation of the average reward a player receives when playing a certain strategy against its best response strategy, $v_i(\pi_i, BR(\pi_i))$. A simple way to implement a general evaluation of the average episode reward is by using a Monte Carlo-simulation, and then taking the average of the episode reward for many episodes. To evaluate some strategy profile $\pi = (\pi_1, \pi_2)$, do the following:

- Let attacker and defender play against each other using strategies $\pi = (\pi_1, \pi_2)$ (without training the networks), and register the total reward each episode
- After some amount of Monte Carlo episodes, stop the evaluation, and take the average reward to get the Monte Carlo approximation of expected reward per episode

This general algorithm can be used to calculate $v_i(\pi_i, BR(\pi_i))$, that is, the expected episode reward of the best response strategy to π_i. But it can also be used for other types of evaluation. For example, in the case that both players play their Nash Equilibrium strategies, the expected cumulative reward will be the *value of the game*. To approximate the value of the game we can use the Monte Carlo procedure with the average strategies from NFSP as input, as the average NFSP strategies are expected to converge to the Nash strategies. Furthermore, the Monte-Carlo approximation can also be used to evaluate a strategy against a fixed agent, for example an agent playing a completely random strategy or heuristic strategy.

Chapter 6

Experiment Setup and Results

The intrusion game follows the setup presented in Chapter 4. All the necessary transition, reward and observation tensors are implemented in Python, as they are defined in Chapter 4. In each time step, the input to each agent's neural network is either (l, b, s) or (l, b, b), depending on if the attacker or defender agent was playing. Here l is the number of stops remaining of the defender, b is the defender's belief state, and s is the true state of the game. The reason for the extra input to the defender is because we want identical networks training the defender and the attacker.

The implementation of the Neural Fictitious Self Play is based on the implementation in OpenSpiel [68], which is a public repository of state-of-the-art algorithms for multi-agent reinforcement learning. We have extended the implementation of NFSP in OpenSpiel to fit our game. The main difference between the original OpenSpiel implementation and ours is the belief calculation.

Moreover, the intrusion prevention game environment is implemented in Python following the OpenSpiel conventions, to make the game compatible with the OpenSpiel implementation of NFSP. Furthermore, the algorithms to compute exploitability as well as the subroutines to compute approximate best response and average episode reward, as described in Section 5.4, are implemented.

NFSP was manually calibrated using the parameters in table 6.1. Many of the parameters were based on the original NFSP paper [21]. Although some variations were tested in an attempt to find optimal hyperparameters, no extensive parameter search was conducted, due to the high computational cost of training and evaluating the networks. After tuning the hyperparameters, training was repeated several times with different random seeds. The results

presented in the next section are the averages and standard deviations from the three best random seeds.

Hyperparameter	Value
$\mathcal{M}_{SL}$	30 000 000
$\mathcal{M}_{RL}$	600 000
LR_{SL}	0.1
LR_{RL}	0.01
Total training time	1 000 000 episodes
Anticipatory parameter η	0.1
Layer sizes	[128, 128, 128]
Optimizer	Stochastic Gradient Descent
Batch size	256
Target network refit of DQN	Every 1000 episodes
ϵ of DQN	0.08

Table 6.1: Hyperparameters used in the base case for the limited game

The game parameters specified in table 6.2 are used.

Game parameter	Value
R_{sla}	1
R_{st}	2
R_{cost}	-3
R_{int}	-3

Table 6.2: Game parameters used in the intrusion prevention game

6.1 Analysis of the Intrusion Game with Limited Observation Space

In the first part of the analysis, the game is limited in observation space. Due to the limitation of game size, we lose some of the comparative advantages of the NFSP in comparison to other algorithms. However, we can use the limited game to evaluate benchmark strategies.

6.1.1 Observation Function

In the first part of the experiment, the observation space for the defender was defined as a discrete grid of 10 observations from 0-9, with each observation having a probabilistic correlation with the state 0 and 1. In the base case, the distributions of the correlations were defined in a way so that lower observations were more probable when there was no intrusion (state 0) and in the same way higher observations were more probable during intrusion (state 1). This small observation space was inspired from domain knowledge, where a small observation could be interpreted as for example a low amount of IDS signals and therefore low, but not 0, probability of intrusion.

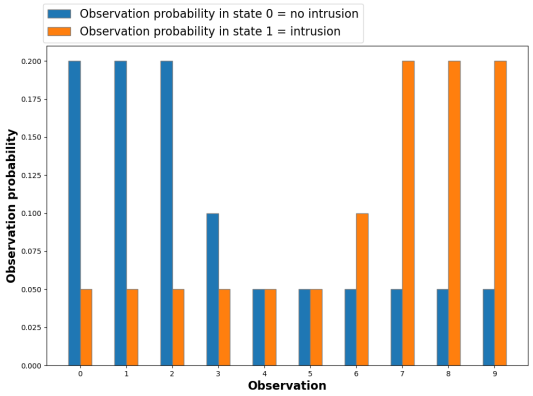

Figure 6.1: The probability distribution of the discrete observation space for states 0 and 1

6.1.2 Learning Nash Equilibria in the Intrusion Prevention Game

In this subsection we show that the NFSP algorithm learns an approximate Nash Equilibrium, by looking at evaluation metrics such as the approximate exploitability and the average game value.

Exploitability Analysis

The approximate exploitability, as defined in Section 5.4, was calculated every 20,000 time steps. For the approximate best response calculation, a reinforcement learning algorithm had to be used. For this single-agent reinforcement learning task (holding one of the agent's strategies fixed) we

chose to evaluate using the Proximal Policy Optimization [69] as implemented in the standard reinforcement learning library Stable Baselines [70]. PPO has been shown to have good results on a collection of benchmark RL tasks and is relatively simple to implement. The PPO was trained for 50000 episodes, with a learning rate of $3 * 10^{-4}$, a network architecture of [128, 128, 128], and a batch size of 2048. The result of this approximate exploitability calculation over time can be found in Figure 6.2.

When experimenting, it was found that the exploitability often reached a global minimum after a certain number of episodes. After this, continuing training the agents resulted in diverging (increasing) exploitability. This behavior could happen in deep reinforcement learning for reasons such as overfitting. Therefore, using empirical evaluation, we find a cutoff episode, after which we stop training the agents. We found that a cutoff after 1,000,000 episodes gives the best results.

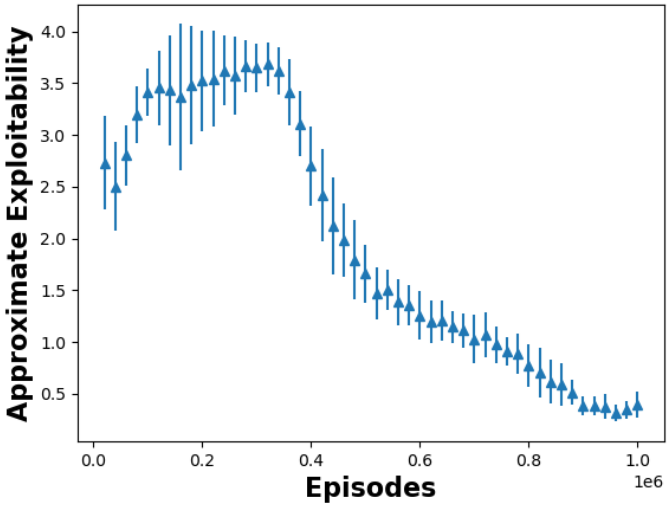

Figure 6.2: Approximate exploitability

We observe that the approximate exploitability decreases in both value and variance over time. Starting from a value of around 2.5, the approximate exploitability converges to a value of around 0.3 in the last 100,000 episodes, with some fluctuations. As noted in Section 5.4, this approximate exploitability should only be considered a lower bound for the true exploitability. The fact that the approximate exploitability converges is an indication that the game reaches an equilibrium state. Further analysis of the game value and average episode reward can help to analyze if this is truly a Nash Equilibrium.

Game Value and Average Episode Reward Analysis

As described in Section 5.4, the approximate exploitability was calculated by calculating the approximate best response for the attacker and the defender against the NFSP strategy. The average best response episode rewards gives an approximate bound for the game value, that we can calculate by letting the average NFSP attacker and average NFSP defender play against each other. The results of these evaluations can be seen in Figure 6.3 and 6.4.

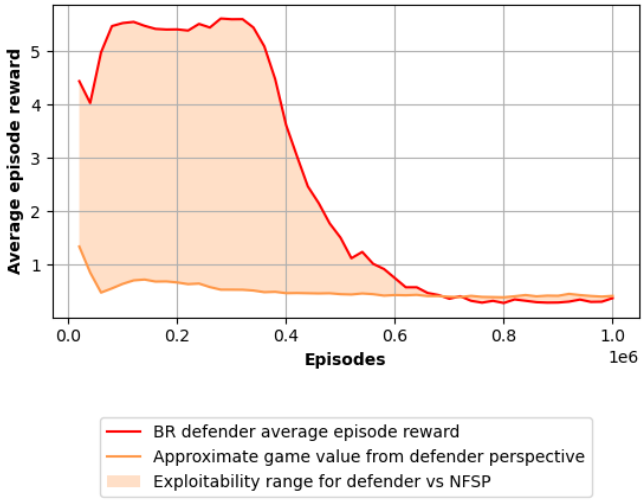

Figure 6.3: Graph of the approximate game value as well as the defender best response average episode rewards when playing against the NFSP attacker agent

The approximate game value converges to roughly 0.4. The best response average episode rewards take longer to converge. For the best response defender, the average reward decreases from 4 to roughly 0.4, as we can see in Figure 6.3. The orange exploitability bound between the best response defender and the average NFSP defender corresponds to how much a BR defender could improve their value by exploiting the NFSP attacker agent. In the last 350,000 episodes, the trained PPO best response defender receives even less reward than the NFSP defender. Theoretically this should not be possible (as described in 5.4) but could happen if the best-response is not trained long enough, and if the NFSP agent closely approximate the best response strategy. Thus this is an indication that there is Nash convergence.

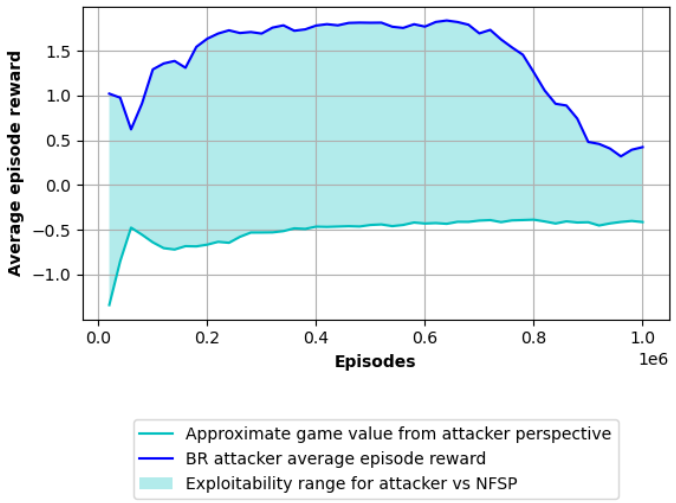

Figure 6.4: Graph of the approximate game value from the attacker perspective, as well as the attacker best response average episode rewards when playing against the NFSP defender agent

In Figure 6.4, we observe a similar visualization for the attacker. The attacker receives the negative approximate game value, -0.4, in the calculated Nash Equilibrium. The best response attacker can improve this to roughly 0.5 average reward. The blue exploitability bound between the best response attacker and the average NFSP attacker corresponds how much a BR attacker could improve their value by exploiting the NFSP defender agent.

In the analysis of the game value as well as the average episode reward, two evaluation attacker agents are implemented; one random agent, and one heuristic agent. The trained NFSP defender agent can be tested against these evaluation agents over time, to see how the average reward develops.

The random attacker simply chooses a completely random action, independent of state and defender belief. This means that the random attacker agent will play strategy

$$\pi_2(0) = \pi_2(1) = (0.5, 0.5).$$

For the heuristic attacker the strategy is manually chosen to

$$\pi_2(0) = (0.8, 0.2), b \leq 0.5, (0.2, 0.8), b > 0.5$$
$$\pi_2(1) = (0.2, 0.8), b \leq 0.5, (0.8, 0.2), b > 0.5.$$

The intuition for the heuristic agent strategy is that attacker is more likely to begin the attack (stop) if the belief of the defender is low and there currently is no intrusion. On the other hand, the attacker is more likely to stop the current attack if the belief of the defender is high. The strategy profiles for these agents are depicted in Figure 6.5.

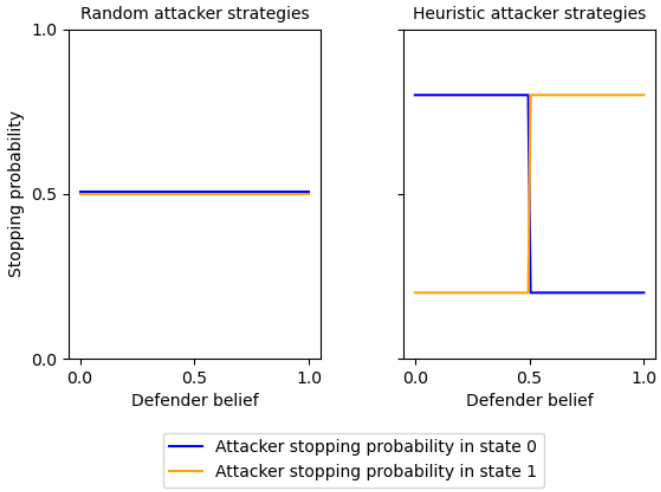

Figure 6.5: The strategy profiles of the evaluation agents

When evaluating the NFSP defender against the evaluation attackers as well as the best response agent, we observe the results in Figure 6.6.

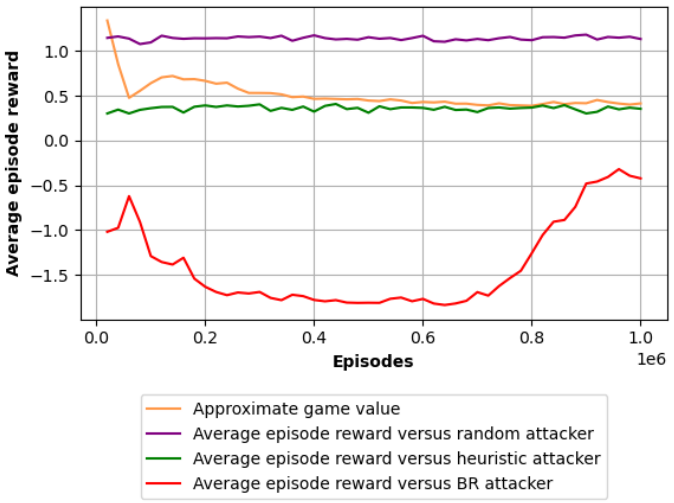

Figure 6.6: The average episode reward when playing against different attackers

All values in Figure 6.6 correspond to the average episode reward for the defender agent. As already visualized in Figure 6.4, the best response attacker outperforms the other attackers and acts as a lower bound for the average episode reward for the NFSP defender. The heuristic attacker receives roughly the same average episode reward as the NFSP attacker against the NFSP defender. The NFSP attacker does better on average than the random attacker.

6.1.3 Characterization of the Learned Nash Equilibrium

When evaluating the optimal strategy as calculated by the NFSP, we have to take into account that the worst-case attacker considers three factors: number of defender stops left l, the current state s, as well as the current belief of the defender b. The defender only takes into consideration their own belief as well as the number of stopping actions it can still perform. We can visualize these strategies in 6 graphs, three for each agent and one for each $l = 1, 2, 3$.

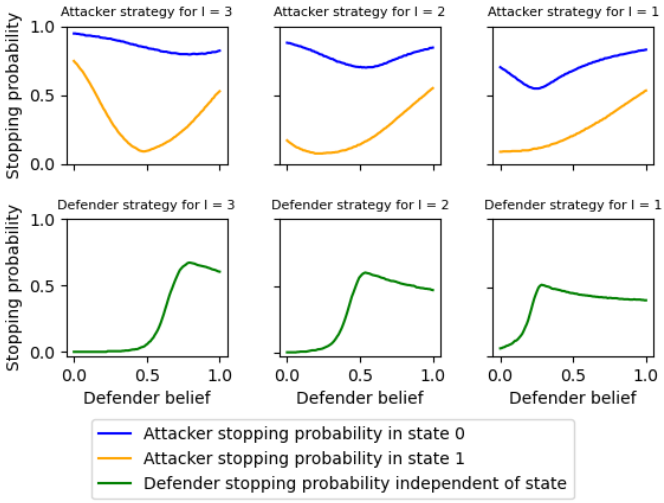

Figure 6.7: The average strategies from the NFSP algorithm

For each agent, we evaluate the average strategies for 100 belief steps between 0 and 1. We can observe the results of this evaluation in Figure 6.7. The top row of figures demonstrates the attacker strategies while the second row demonstrates the defender strategies.

The blue line, which represents the attacker stopping probability in state 0, starts at a value close to 1 for $b = 0$ and decreases slightly as belief goes up for l= 3 and 2. For l=1, the attacker seems to be more careful when choosing to start its attacks.

In state 1 (intrusion), the attacker stopping probability increases as b increases. However, for $l = 3$, the attacker has a high stopping probability even for lower belief values. This could perhaps correspond to a wary attacker, that performs a short intrusion and thereafter quickly tries to escape before the defender notices the intrusion.

For the defender strategies, the NFSP strategy converges to a high stopping probability for $b > 0.5$, and a lower stopping probability for smaller beliefs. The fewer stops the defender agent has remaining, the more likely the agent becomes to stop even for low beliefs.

Furthermore, we also analyze the Nash Equilibrium stopping strategies in the case a game parameter is changed. This is to demonstrate how much the learned strategies depend on how the subjective defender value for example server uptime, or how costly they consider a stopping action or an intrusion.

We consider the case when $R_{int} = -5$ instead of $R_{int} = -3$. This means

that the cost of intrusion increases for the defender, and the reward for intrusion increases for the attacker. All other parameters are held constant in comparison with the base case. The result of this experiment can be seen in Figure 6.8.

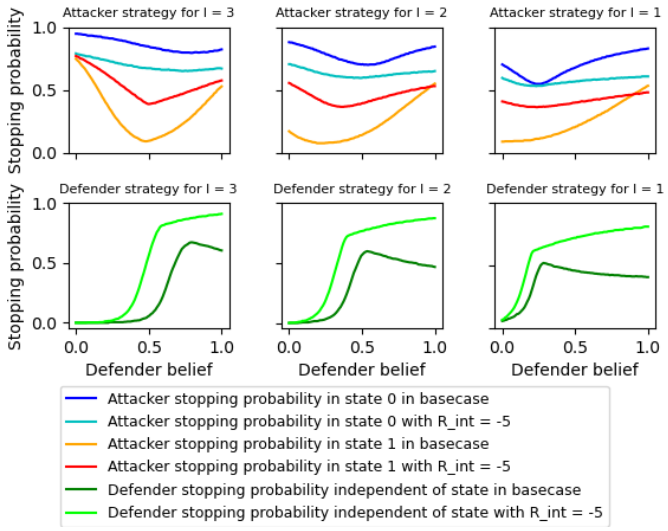

Figure 6.8: The average strategies from the NFSP algorithm in the basecase and when $R_{int} = -5$

In this case, for the defender, we can observe that the converged strategies in the increased intrusion cost case have a higher stopping probability for all $l = 1, 2, 3$. This corresponds to the intrusion cost being relatively more expensive than the stopping cost compared to the base case. For the attacker, we see that the defender's increased probability of stopping is reflected in that the probability of starting the attack is decreased. Furthermore, the attacker is more likely to stop in state 1, maybe due to the defender being more probable to stop.

6.2 Analysis of Game with Data Observations from Emulation

In the second part of the experiment, the observation space for the defender is defined using real data traces from simulations of hacking attacks on the KTH testbed.

6.2.1 Observation Function

The number of severe IDS alerts during the states of no intrusion and intrusion is measured on an emulated IT infrastructure as specified in Section 5. A of [4]. The emulation evolves in time-steps, where the attacking agent takes a specific sequence of actions. In this project, we use the data traces generated by the *expert attacker* profile described in [4]. This means that the attacker uses sophisticated exploits such as remote execution vulnerabilities rather than brute-force methods, meaning that the agent is harder to discover.

In total, there were 271 possible observations of severe IDS alerts, labeled from 0-270. The probability distribution of the observations in state 0 and 1 is seen in Figure 6.9. In state 0, the probability of all observations are fairly evenly distributed, with the lower observation 0-50 being slightly more probable. In state 1, we observe probability spikes for some higher number of IDS signals; for example for the highest observation (270) the probability of observation is roughly 28% in state 1.

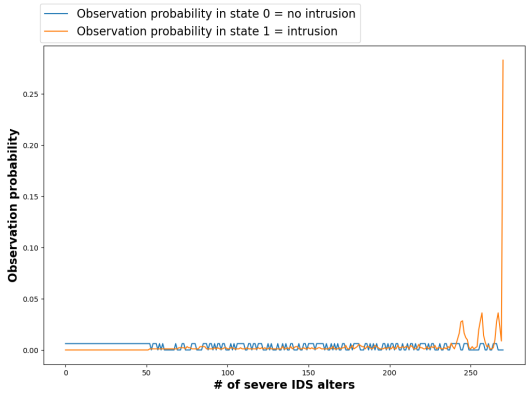

Figure 6.9: The probability distribution of severe IDS signals for state 0 and state 1 generated by the expert attacker agent

6.2.2 Learning Nash Equilibria

As in Section 6.1, we evaluate the NFSP algorithm using approximate exploitability. The exploitability is calculated in the same manner as in the first part of the analysis, i.e. by using a PPO algorithm to find the approximate best response every 20,000 time steps. We use the same hyperparameters and game parameters for this part of the analysis. The approximate exploitability results are found in Figure 6.10.

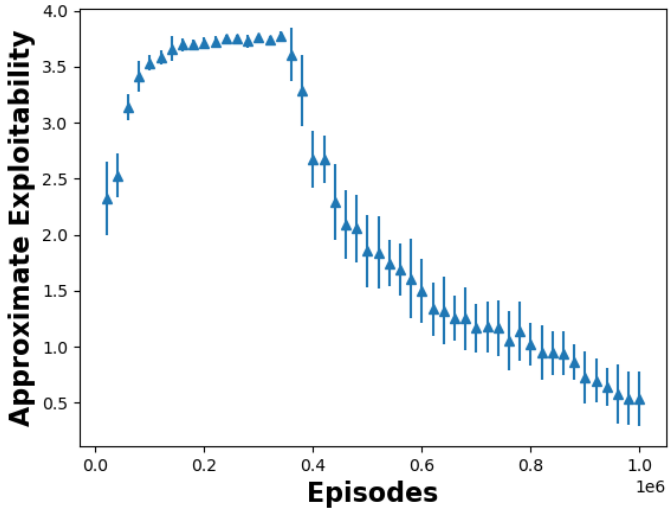

Figure 6.10: Approximate exploitability

We see that like for the game with small observation space, the exploitability converges, although to a slightly higher value (0.5) and with slightly higher variance than for the benchmark game. This is an indication that the game reaches an equilibrium state. When looking at the game value plots in this case, we see similar results as in Figures 6.3, 6.4, and 6.6; therefore, these plots are omitted in this section. In the next section, we compare the learned strategies with the base case case.

6.2.3 Comparison of Learned Strategies with Base Case

In Figure 6.11 we compare the average strategies learned in the game with observations from real data compared to the one from a small observation space. We observe that the learned strategies are similar both for the defender and the attacker. The defender in the second case has slightly higher stopping probabilities for higher beliefs, and similarly on the other hand the attacker has slightly lower attacking probabilities for higher beliefs.

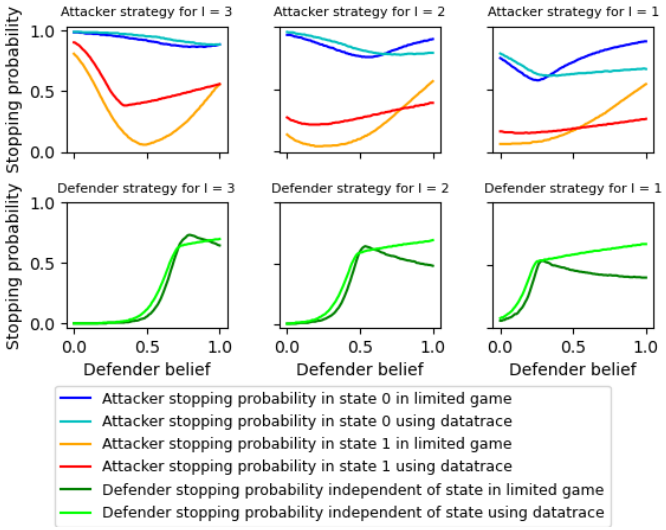

Figure 6.11: Comparing the average strategies from the NFSP algorithm in the base case and learning using data traces

Chapter 7

Discussion and Conclusion

In this chapter, we discuss and analyze the obtained results. Furthermore, we present some suggestions for further research as well as the conclusion of the thesis.

7.1 Discussion of Results

The main research questions this thesis aims to answer is 1) how can the intrusion prevention use case be modeled as a game 2) what structural insights can be derived for the model and 3) which reinforcement learning algorithms and/or game-theoretic methods are suitable for computing automated intrusion prevention strategies for the given model.

By extending previous models of the intrusion prevention use case, we present a possible answer to question 1), by modeling the game as a POSG. For question 2), we refer to the structural and theoretical results in Section 4.7. Here, we find that a Nash equilibrium exists for the game. Furthermore, we find that the pure Nash equilibrium will be uncommon and of a specific structure. This is confirmed in the results we see in Chapter 6, where all the equilibria found are mixed. Lastly, we derive that a best response strategy for the defender will be of a thresh-holding structure, with lower thresh-holds for stopping the more stops the defender has left. This theoretical result is in contrast to what was actually found in Figure 6.7. Here, there seems to be an indication of a thresh-holding strategy, where there is some constant belief α_l for which the defender uses the stopping action with high probability. However, in the strategies found empirically, we have $\alpha_1 < \alpha_2 < \alpha_3$, which could indicate that there are some instability in the experimental results, as will be discussed further.

For question 3), we choose to implement an adapted NFSP algorithm to try to find an approximate Nash equilibrium in the defined game. The initial results presented in Chapter 6 indicate that a Nash equilibrium is possible to find using this methodology. The approximate exploitability converges to a value of around 0.3 in the case of small observation space and 0.5 using data from emulations, compared to a benchmark exploitability of roughly 0.1 achieved in the original NFSP paper [21]. Thus the adapted NFSP algorithm achieves state-of-the art results for the POSG-model in terms of exploitability.

When inspecting the converged strategies, we find that they imitate human strategies and intuitive behavior. For example, in Figure 6.7, we observe that the the defender is more likely to perform a defensive stopping action when it has a higher belief, and in the same manner the attacker is more likely to stop the intrusion when the belief of the defender is high. However, we also see a high dependency on hyperparameters and game parameters. When changing one of the game parameters, for example the cost of intrusion, the learned Nash strategies change substantially, as can be seen in Figure 6.8. Hence, to be able to use the game model and the NFSP algorithm practically in finding automated security strategies, one would have to think carefully about how to define the rewards in a way so that they most closely reflect the goals of the organization owning the IT infrastructure.

The results of Section 6.2 indicate that the same NFSP setup applied for an intrusion game with a small observation space also could be used for a game where the observations are from data traces generated using simulations. In Figure 6.11 we observe that the learned strategies in both cases are similar, except that the defender has slightly higher stopping probabilities when using data traces. One interpretation of this is as follows. The better data signal we have from the IDS, the higher belief should be required for the defender to perform a stopping action. If we have a noisy signal, the defender would require lower belief to perform a stop since there would be more uncertainty associated with the signal. Since we have quite similar stopping probabilities in both the case when a synthetic probability distribution of observations is used, as described in Figure 6.1, as we have when using the severe IDS signals (Figure 6.9) then it would mean these two distributions work almost equally well for the defender to form beliefs.

We can further discuss the instability of the solutions. As already noted, the solutions are highly dependent on the reward function. One other instability we found when running experiments is the dependency on the random seed. When running the experiments, oftentimes up to 10-15 different random seeds had to be tested to find 3-4 converging results. In the other cases,

the reinforcement learning algorithm got stuck in a local optimum, and the exploitability could be stuck on a value such as 3.5 indefinitely. This instability could cast doubt on the efficiency of using NFSP for finding automated security strategies, and should be researched further.

Although we show that the NFSP algorithm could be used to learn automated defense strategies in the intrusion prevention use case, there still remains the question of relevancy of the learned strategies in practical applications. When finding the Nash equilibrium defense strategy, we find a strategy which guarantees some minimum reward. In real life applications when defending against automated hacking attacks, we could perhaps do better than just the "minimum reward" by using a specific defense heuristic.

7.2 Future Work

There are many different aspects that could be considered in the future when extending this line of work.

Firstly, a natural extension of this master thesis is to further research how different IDS signals could be used to form the beliefs of the defender. In this research only a 1-D data signal is used, and one could explore how for example a 3-D IDS signal would affect the learned strategies. In for example [4] a 3-dimensional observation space is used. One problem with this is that an increase of dimension in observation space would increase the size of the observation space exponentially. For example, if three different IDS signals were used to form an observation, where each signal could take a value between 0-199, then the size of total observation space would be $200^3 = 8,000,000$ observations. A tensor of this size would require roughly 32 GB of memory, meaning that the belief calculation would be extremely slow, and would require some clever feature engineering to handle. Furthermore, it would be interesting to research how well the automated intrusion strategies could be learned depending on the noisiness of the signal. As described in the previous section, a more noisy signal, where it is harder for the defender to differentiate between intrusion and no intrusion, should intuitively mean that the defender is more likely to perform a stopping action.

Secondly, one line of research could try implementing other types of multi-agent reinforcement learning algorithms than NFSP to see if they converge better. This could be for example the Policy-Space Response Oracle algorithm [24] or Alpha-Zero [25]. Furthermore, the fictitious play framework should be possible to implement with other reinforcement learning algorithms than the DQN when finding the best response. One possible candidate could be

the PPO algorithm, which in this thesis is used to find the approximate best response when finding the approximate exploitability.

Thirdly, one could further develop the POSG model to make it more complex, and explore whether multi-agent reinforcement learning still is a viable option for the more complex model. This includes for example adding a component which not only decides when to take a stopping action, but also decides what action to take. Another example would be to use different types of attackers or increase the complexity of attack. In future research we want to find a game model which more closely reflect how an actual IT system operates, and these sorts of adjustment could make the game model more precise.

Fourthly, in this thesis, we had to adapt the NFSP algorithm by adding the calculation of the defender belief, which is central in the theoretical framework of partially observed games. To the best of our knowledge, this is the first time player beliefs have been used in a model-free reinforcement learning or multi-agent reinforcement learning algorithm. This novel implementation, described in 5.3, should be possible to extend for other multi-agent reinforcement learning algorithms as well.

7.3 Conclusion

In this thesis, our main research goals were to model the intrusion prevention use case as a game and to find a suitable reinforcement learning algorithm for computing automated intrusion prevention strategies for the given model. We have shown that the use case can be modeled with a zero-sum onesided partially observed stochastic game and we have explored the theoretical properties of this game. We have shown that a Nash equilibrium exists for the game and that the equilibrium strategies have specific properties. Furthermore, we have demonstrated that the Neural Fictitious Self Play algorithm is an effective way to find automated strategies in the intrusion prevention game that we have defined. By adapting the NFSP specifically to the game, we narrow the gap between the theoretical framework of partially observed stochastic games and model-free reinforcement learning. Although the converged strategies imitate human behavior, they are heavily dependent on the hyperparameter setup and the reward function. Thus there remains more research to be done before NFSP or some other multi-agent algorithm could be used practically to prevent hacking attacks.

References

[1] W. E. Forum, "Global cybersecurity outlook," 2022. [Online]. Available: https://www.weforum.org/reports/global-cybersecurity-outlook-2022

[2] K. Hammar and R. Stadler, "Finding effective security strategies through reinforcement learning and Self-Play," in *International Conference on Network and Service Management (CNSM 2020)*, Izmir, Turkey, 2020.

[3] ——, "Learning intrusion prevention policies through optimal stopping," in *International Conference on Network and Service Management (CNSM 2021)*, Izmir, Turkey, 2021, https://arxiv.org/pdf/2106.07160.pdf.

[4] ——, "Intrusion prevention through optimal stopping," *IEEE Transactions on Network and Service Management*, 2022. doi: 10.1109/TNSM.2022.3176781

[5] ——, "A system for interactive examination of learned security policies," 2022. [Online]. Available: https://arxiv.org/abs/2204.01126

[6] K. Malialis and D. Kudenko, "Multiagent router throttling: Decentralized coordinated response against ddos attacks," in *IAAI*, 2013.

[7] B. Ning and L. Xiao, "Defense against advanced persistent threats in smart grids: A reinforcement learning approach," in *2021 40th Chinese Control Conference (CCC)*, 2021. doi: 10.23919/CCC52363.2021.9549271 pp. 8598–8603.

[8] M. Alauthman, N. Aslam, M. Alkasassbeh, S. Khan, A. al Qerem, and K. Choo, "An efficient reinforcement learning-based botnet detection approach," *J. Netw. Comput. Appl.*, vol. 150, 2020.

[9] T. Alpcan and T. Basar, *Network Security: A Decision and Game-Theoretic Approach*, 1st ed. USA: Cambridge University Press, 2010. ISBN 0521119324

[10] M. Tambe, *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*, 1st ed. USA: Cambridge University Press, 2011. ISBN 1107096421

[11] C. Kamhoua, C. Kiekintveld, F. Fang, and Q. Zhu, *Game Theory and Machine Learning for Cyber Security*. Wiley, 2021. ISBN 9781119723929. [Online]. Available: https://books.google.se/books?id=EBxszQEACAAJ

[12] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Basar, and J.-P. Hubaux, "Game theory meets network security and privacy," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 25:1–25:39, Jul. 2013. doi: 10.1145/2480741.2480742. [Online]. Available: http://doi.acm.org/10.1145/2480741.2480742

[13] A. Aydeger, M. H. Manshaei, M. A. Rahman, and K. Akkaya, "Strategic defense against stealthy link flooding attacks: A signaling game approach," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 751–764, 2021. doi: 10.1109/TNSE.2021.3052090

[14] O. Tsemogne, Y. Hayel, C. Kamhoua, and G. Deugoue, *Partially Observable Stochastic Games for Cyber Deception Against Network Epidemic*, 12 2020, pp. 312–325. ISBN 978-3-030-64792-6

[15] O. Vaněk, Z. Yin, M. Jain, B. Bošanský, M. Tambe, and M. Pěchouček, "Game-theoretic resource allocation for malicious packet detection in computer networks," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, ser. AAMAS '12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2012. ISBN 0981738125 p. 905–912.

[16] K. C. Nguyen, T. Alpcan, and T. Basar, "Stochastic games for security in networks with interdependent nodes," in *2009 International Conference on Game Theory for Networks*, 2009. doi: 10.1109/GAMENETS.2009.5137463 pp. 697–703.

[17] J. Gabirondo-López, J. Egaña, J. Miguel-Alonso, and R. Orduna Urrutia, "Towards autonomous defense of sdn networks using muzero based

intelligent agents," *IEEE Access*, vol. 9, pp. 107 184–107 199, 2021. doi: 10.1109/ACCESS.2021.3100706

[18] A. Laszka, W. Abbas, S. S. Sastry, Y. Vorobeychik, and X. Koutsoukos, "Optimal thresholds for intrusion detection systems," in *Proceedings of the Symposium and Bootcamp on the Science of Security*, ser. HotSos '16. New York, NY, USA: Association for Computing Machinery, 2016. doi: 10.1145/2898375.2898399. ISBN 9781450342773 p. 72–81. [Online]. Available: https://doi.org/10.1145/2898375.2898399

[19] T. Alpcan and T. Basar, "A game theoretic analysis of intrusion detection in access control systems," in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, vol. 2, 2004. doi: 10.1109/CDC.2004.1430267 pp. 1568–1573 Vol.2.

[20] Q. Zhu and T. Başar, "Dynamic policy-based ids configuration," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, 2009. doi: 10.1109/CDC.2009.5399894 pp. 8600–8605.

[21] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," *CoRR*, vol. abs/1603.01121, 2016. [Online]. Available: http://arxiv.org/abs/1603.01121

[22] J. Heinrich, M. Lanctot, and D. Silver, "Fictitious self-play in extensive-form games," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 805–813. [Online]. Available: http://proceedings.mlr.press/v37/heinrich15.html

[23] G. W. Brown, "Iterative solution of games by fictitious play," 1951, activity analysis of production and allocation.

[24] M. Lanctot, V. F. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel, "A unified game-theoretic approach to multiagent reinforcement learning," *CoRR*, vol. abs/1711.00832, 2017. [Online]. Available: http://arxiv.org/abs/1711.00832

[25] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go

through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018. [Online]. Available: http://science.sciencemag.org/content/362/6419/1140/tab-pdf

[26] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spanò, "Multi-agent reinforcement learning: A review of challenges and applications," *Applied Sciences*, vol. 11, no. 11, 2021. doi: 10.3390/app11114948. [Online]. Available: https://www.mdpi.com/2076-3417/11/11/4948

[27] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," 2019. [Online]. Available: https://arxiv.org/abs/1911.10635

[28] K. Horák, B. Bošanský, and M. Pěchouček, "Heuristic search value iteration for one-sided partially observable stochastic games," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 2017. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/10597

[29] W. Stallings and L. Brown, *Computer Security: Principles and Practice*, 3rd ed. USA: Prentice Hall Press, 2014. ISBN 0133773922

[30] E. Zouave, M. Bruce, K. Colde, M. Jaitner, I. Rodhe, and T. Gustafsson, "Artificially intelligent cyberattacks," 2020, ,Swedish Defence Research Agency.

[31] H. W. Kuhn, *Extensive games and the problem of information*, H. W. Kuhn and A. W. Tucker, Eds. Princeton, NJ: Princeton University Press, 1953.

[32] J. F. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, pp. 286–295, 1951.

[33] J. von Neumann, "Zur Theorie der Gesellschaftsspiele. (German) [On the theory of games of strategy]," vol. 100, pp. 295–320, 1928.

[34] L. S. Shapley, "Stochastic games," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953. doi: 10.1073/pnas.39.10.1095. [Online]. Available: https://www.pnas.org/content/39/10/1095

[35] J. Filar and K. Vrieze, *Competitive Markov Decision Processes*. Springer, 1997.

[36] A. Neyman and S. Sorin, *Stochastic Games and Applications*, ser. Classics in Applied Mathematics. Springer, 2003. [Online]. Available: https://link.springer.com/book/10.1007/978-94-010-0189-2

[37] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ser. ICML'94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994. ISBN 1558603352 p. 157–163.

[38] J. F. Nash, "Equilibrium points in n-person games," *Proc. of the National Academy of Sciences*, vol. 36, pp. 48–49, 1950.

[39] J. Hespanha and M. Prandini, "Nash equilibria in partial-information games on markov chains," in *Proceedings of the 40th IEEE Conference on Decision and Control*, vol. 3, 2001. doi: 10.1109/CDC.2001.980562 pp. 2102–2107 vol.3.

[40] K. Horak, B. Bosanský, V. Kovarík, and C. Kiekintveld, "Solving zero-sum one-sided partially observable stochastic games," *CoRR*, vol. abs/2010.11243, 2020. [Online]. Available: https://arxiv.org/abs/2010.11243

[41] K. Åström, "Optimal control of markov processes with incomplete state information," *Journal of Mathematical Analysis and Applications*, vol. 10, no. 1, pp. 174–205, 1965. doi: https://doi.org/10.1016/0022-247X(65)90154-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022247X6590154X

[42] E. A. Hansen, D. S. Bernstein, and S. Zilberstein, "Dynamic programming for partially observable stochastic games," in *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, San Jose, California, 2004, pp. 709–715. [Online]. Available: http://rbr.cs.umass.edu/shlomo/papers/HBZaaai04.html

[43] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998. ISBN 0262193981

[44] K. Horák, "Scalable algorithms for solving stochastic games with limited partial observability," Ph.D. dissertation, 2019.

[45] H. W. Kuhn, *11. Extensive Games and the Problem of Information*. Princeton University Press, 2016, pp. 193–216. [Online]. Available: https://doi.org/10.1515/9781400881970-012

[46] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge, UK: Cambridge University Press, 2009. ISBN 978-0-521-89943-7

[47] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. USA: Cambridge University Press, 2007. ISBN 0521872820

[48] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artificial Intelligence*, vol. 6, pp. 293–326, 1975.

[49] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," vol. 4630, 05 2006. doi: 10.1007/978-3-540-75538-8$_7$

[50] K. Horák and B. Bošanský, "Solving partially observable stochastic games with public observations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 2029–2036, Jul. 2019. doi: 10.1609/aaai.v33i01.33012029. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4032

[51] G. W. Brown, "Iterative solution of games by fictitious play," *Activity analysis of production and allocation*, vol. 13, no. 1, pp. 374–376, 1951.

[52] D. Blackwell, "An analog of the minimax theorem for vector payoffs." *Pacific Journal of Mathematics*, vol. 6, pp. 1–8, 1956.

[53] H. Young, *Strategic Learning and its Limits*. Oxford University Press, 2004. [Online]. Available: https://EconPapers.repec.org/RePEc:oxp:obooks:9780199269181

[54] L. Samuelson, *Evolutionary Games and Equilibrium Selection*, ser. MIT Press Books. The MIT Press, September 1998, vol. 1, no. 0262692198. ISBN ARRAY(0x484e5310). [Online]. Available: https://ideas.repec.org/b/mtp/titles/0262692198.html

[55] D. Fudenberg and D. Levine, *The Theory of Learning in Games*, 1st ed. The MIT Press, 1998, vol. 1. [Online]. Available: https://EconPapers.repec.org/RePEc:mtp:titles:0262061945

[56] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, no. null, p. 1039–1069, Dec. 2003.

[57] M. Bowling, "Multiagent learning in the presence of agents with limitations," Ph.D. dissertation, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, May 2003, available as technical report CMU-CS-03-118.

[58] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*. Belmont, MA: Athena Scientific, 1996.

[59] T. Jaakkola, M. Jordan, and S. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Advances in Neural Information Processing Systems*, vol. 6, 1994. [Online]. Available: https://proceedings.neurips.cc/paper/1993/file/5807a685d1a9ab3b599035bc566ce2b9-Paper.pdf

[60] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400 – 407, 1951. doi: 10.1214/aoms/1177729586. [Online]. Available: https://doi.org/10.1214/aoms/1177729586

[61] C. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 05 1992. doi: 10.1007/BF00992698

[62] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[63] D. Leslie and E. Collins, "Generalised weakened fictitious play," *Games and Economic Behavior*, vol. 56, pp. 285–298, 08 2006. doi: 10.1016/j.geb.2005.08.005

[64] J. S. Vitter, "Random sampling with a reservoir," *ACM Trans. Math. Softw.*, vol. 11, no. 1, p. 37–57, mar 1985. doi: 10.1145/3147.3165. [Online]. Available: https://doi.org/10.1145/3147.3165

[65] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Advances in Neural Information Processing Systems*, J. Platt, D. Koller,

Y. Singer, and S. Roweis, Eds., vol. 20. Curran Associates, Inc., 2008. [Online]. Available: https://proceedings.neurips.cc/paper/2007/file/08d98638c6fcd194a4b1e6992063e944-Paper.pdf

[66] F. Timbers, E. Lockhart, M. Schmid, M. Lanctot, and M. Bowling, "Approximate exploitability: Learning a best response in large games," *CoRR*, vol. abs/2004.09677, 2020. [Online]. Available: https://arxiv.org/abs/2004.09677

[67] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Math. Oper. Res.*, vol. 12, p. 441–450, Aug. 1987.

[68] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei, D. Hennes, D. Morrill, P. Muller, T. Ewalds, R. Faulkner, J. Kramár, B. D. Vylder, B. Saeta, J. Bradbury, D. Ding, S. Borgeaud, M. Lai, J. Schrittwieser, T. Anthony, E. Hughes, I. Danihelka, and J. Ryan-Davis, "OpenSpiel: A framework for reinforcement learning in games," *CoRR*, vol. abs/1908.09453, 2019. [Online]. Available: http://arxiv.org/abs/1908.09453

[69] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, 2017, http://arxiv.org/abs/1707.06347. [Online]. Available: http://arxiv.org/abs/1707.06347

[70] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[71] V. Krishnamurthy, *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge University Press, 2016.

Appendix A

Extensive Form Representation of the Game Model

Using a fictitious player "nature" (N) with a fixed strategy that represents the stochastic transitions and observations in the game, the partially observed stochastic game model can be visualized in extensive form as shown in Fig. A.1. First, player N decides the public parameter L. Then, the defender (player 1) stops (action S) or continues (action C). If $L = 1$ and the defender chose S, the game ends. Otherwise, the attacker starts the intrusion (action A) or waits (action W). Next, the attacker is detected with probability p. If the attacker was not detected, player N generates an observation $o \in \mathcal{O}$ sampled from the observation function $\mathcal{Z}$. Then, if the intrusion has started ($s = 1$), the attacker either continues the intrusion (action C), or aborts (action S), in which case the game ends. This procedure continues until either the attacker aborts the intrusion, the attacker is detected, or the defender performs the final stop, after which the episode ends.

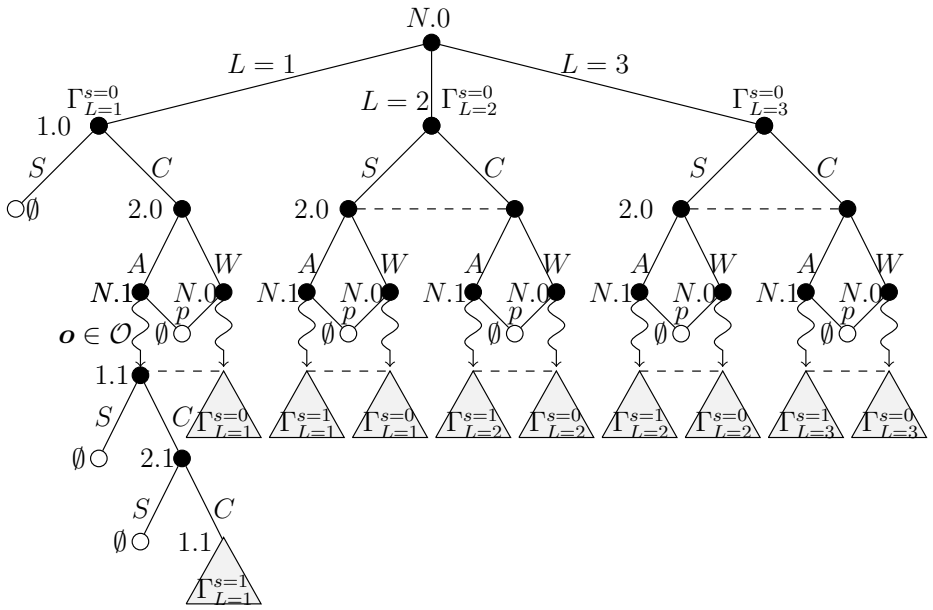

Figure A.1: A partial view of the POSG in extensive form; a filled circle denote a decision node; a unfilled circle denote a terminal node; the first label next to each node indicate the player who makes the decision; the second label next to each node and the dashed lines indicate the states and the information sets; a zig-zag branch is a short-hand for many branches; a triangle indicates a subgame; Γ^s_L denotes a subgame that starts in state s with L stops remaining of the defender. Image courtesy of Kim Hammar.

Appendix B

Proof of Properties of the Intrusion Prevention Game

In this appendix we present the proofs of Theorem 4.7.1 presented in Section 4.7.

B.1 Proof of Theorem 4.7.1 B.

The proof regarding the properties of the pure Nash equilibrium strategies for the attacker builds on two lemmas. In the first lemma, we show the attacker will attack in state $s_t = 0$ at some point. In the second lemma, we show that there exists a pure Nash equilibrium strategy where the attacker stops in both states intrusion and no intrusion. Then, we combine the lemmas and show that this is the only pure Nash equilibrium that exists.

Lemma B.1.1. *In any Nash equilibrium* $(\pi^*_{1,l}, \pi^*_{2,l})$ *there exists a combination of* $l \in \{1 \dots L\}$ *and* $b \in B$ *such that* $\pi^*_{2,l}(S|b, 0) > 0$*, i.e. the probability of attacker to start the attack is non-zero.*

Proof of Lemma 1. Assume by contradiction that there exists a best response strategy

$$\pi^*_{2,l}(S|b, 0) = 0, \forall l \in \{1 \dots L\},$$

i.e. the attacker never starts the attack. It then follows from the definition of the reward function (Eqs. 4.1-4.6) that the best response strategy for the defender is to never stop, $\pi^*_1(b) = C, \forall b \in B$. But then π^*_2 is not a best response to π^*_1. This follows because $R_{sla} > 0$, and waiting forever would yield an infinite reward for the defender. Since π^*_2 is not a best response, this strategy profile is not a Nash equilibrium. □

Lemma B.1.2. *If $f_x(.|0)$ and $f_x(.|1)$ are disjoint, there exists a pure Nash equilibrium where the attacker takes the first stop action at t=1 to start the intrusion and takes the second stop action at t=2 to stop the intrusion.*

Proof of Lemma 2. If $f_x(.|0)$ and $f_x(.|1)$ are disjoint, it follows from Equation 3.2 that the defender knows the state at any time-step i.e. $b_t(s_t) = 1$. Then it follows that the best response strategy of the defender against any attacker strategy is a pure strategy that takes the continue action in state $s_t = 0$ and stop action in state $s_t = 1$. Since $R_{sla} > 0$ it then follows that the best response strategy of the attacker is a pure strategy that stops in both $s_t = 0$ and $s_t = 1$. Thus the strategies $(\pi^*_{1,l}, \pi^*_{2,l})$ form a pure Nash equilibrium. □

Now we combine the two Lemmas to prove Theorem 4.7.1 B.

Proof of Theorem 4.7.1 B. It follows from Lemma 2 that if $f_x(.|0)$ and $f_x(.|1)$ are disjoint, a pure Nash equilbrium where the attacker takes action S in both $s_t = 0$ and $s_t = 1$ for any $b \in B$ exists. From Lemma 1 we know that in any pure Nash equilibrium, the attacker strategy takes action S in state s_t for some $\bar{b} \in B$. Assume by contradiction that there exists a pure Nash equilibrium $(\pi^*_{1,l}, \pi^*_{2,l})$ where there exists a $\bar{b} \in B$ where $\pi^*_{2,l}(\bar{b}, 0) = S$ and $\pi^*_{2,l}(\bar{b}, 1) = C$ where $\bar{b}$ is any belief obtained from Eq. 3.2 after taking the first stop action. Since $\pi^*_{2,l}$ is pure and $\rho_1(0) = 1$, the defender can infer the state s_t at any timestep using Eq. 3.2, i.e. $b_t(s_t) = 1$. It follows from the definition of the reward function that $\pi^*_{1,l}(\bar{b}) = S$. If $\pi^*_{1,l}(\bar{b}) = S$ then it follows from the reward function that $\pi^*_{2,l}(\bar{b}, 1) = C$ is not a best response and thus $(\pi^*_{1,l}, \pi^*_{2,l})$ is not a Nash equilibrium.

□

B.2 Proof of Theorem 4.7.1 C.

The last theorem relates to the structural properties of the optimal stopping strategy. This theorem states that there exists an optimal strategy with thresholding properties. The theorem build on the fact that the partially observed stochastic game is an extension of the partially observed Markov decision process defined in the previous work [4].

If $L \geq 1$, an additional condition applies: the probability matrix of f_x must be TP2 (all second order minors must be non negative), as of definition 10.2.1 of [71]. This is true for the probabilities generated by the probability function $\mathcal{Z}$ since f_x is stochastically montone in s.

Proof of Theorem 4.7.1 C. Given any attacker strategy $\pi_{2,l}$ the best response strategy $\pi^*_{1,l}$ of the defender is an optimal policy in the POMDP that results from keeping $\pi_{2,l}$ fixed in the POSG, and vice versa. Thus, it is sufficient to show that there exists L decreasing values $\alpha^*_1 \dots \alpha_L$ and an optimal policy $\pi^*_{1,b}$ in the POMDP that satisfies:

$$\pi^*_{1,b}(b) = s \iff b \geq \alpha^*_l, l \in \{1 \dots L\}.$$

Since f_x is TP2 by assumption and the POMDP satisfies the conditions of Theorem 1 in [4], the result follows. □