July 2018

Machine Learning Methods for Network Intrusion Detection and Intrusion Prevention Systems

Zheni Svetoslavova Stefanova
University of South Florida, stefanova@mail.usf.edu

Machine Learning Methods for Network Intrusion Detection and Intrusion Prevention Systems

by

Zheni Svetoslavova Stefanova

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosopy
Department of Mathematics & Statistics
College of Arts and Sciences
University of South Florida

Major Professor: Kandethody Ramachandran, Ph.D.
Christos Tsokos, Ph.D.
Lu Lu, Ph.D.
Yicheng Tu, Ph.D

Date of Approval:
June 19, 2018

Keywords: Network Security, Intrusion Detection, Q-learning, Machine Learning, Streaming data

Dedication

I would like to dedicate this dissertation first to God, to my parents: Zlatka and Svetoslav, to my sister Dilyana, to my husband Dan Utic, who made my graduation possible with his love and support, to all my friends who helped me during the years of study: Ivo, Zhanet, Tharu and Maryam.

Acknowledgments

I would like to sincerely thank my advisor Professor Kandethody Ramachandran for his never-ending support since my admission to the Ph.D. in Statistics program, for his advices, guidance, and reasonable trust in my intuition. His inspirational personality is one of the main reasons for me to be able to complete my dissertation, to try to reach new horizons by comparing myself to nobody else but myself. His responsible and reliable attitude towards our research made the progress of our work smooth. Furthermore, his openness to new ideas and flexible working approach allowed me to discover not merely advisor but also a mentor and friend for a lifetime.

Next, I would like to express my gratitude to Professor Chris Tsokos, who taught me how to always stay positive and to believe in myself, who wished me always best for every challenge on my way, for his desire to support and guide me while I was a student and for his endless inspirational and friendly attitude. I am deeply grateful to Dr. Lu Lu, her incredible organized lectures and her brilliant mind made my study process manageable. Her ambitious and devoted personality inspired me to be like her someday. I would also like to thank Professor Gangaram Ladde, who was continuously interested in my development as a student and who provided me with support and knowledge, with encouragement and wisdom until the end of my study. I sincerely feel thankful to Dr. Yicheng Tu for his willingness to help me when I needed, for his time that he spent in listening to my ideas and for agreeing to serve as a committee member of this dissertation. Last but not least, I would like to express my gratitude to Professor Nasir Ghani for agreeing to serve as a committee chair, his valuable positive suggestions and edits improved the quality of my work.

Table of Contents

List of Tables

List of Figures

Abstract

Given the continuing advancement of networking applications and our increased dependence upon software-based systems, there is a pressing need to develop improved security techniques for defending modern information technology (IT) systems from malicious cyber-attacks. Indeed, anyone can be impacted by such activities, including individuals, corporations, and governments. Furthermore, the sustained expansion of the network user base and its associated set of applications is also introducing additional vulnerabilities which can lead to criminal breaches and loss of critical data. As a result, the broader cybersecurity problem area has emerged as a significant concern, with many solution strategies being proposed for both intrusion detection and prevention. Now in general, the cybersecurity dilemma can be treated as a conflict-resolution setup entailing a security system and minimum of two decision agents with competing goals (e.g., the attacker and the defender). Namely, on the one hand, the defender is focused on guaranteeing that the system operates at or above an adequate (specified) level. Conversely, the attacker is focused on trying to interrupt or corrupt the systems operation.

In light of the above, this dissertation introduces novel methodologies to build appropriate strategies for system administrators (defenders). In particular, detailed mathematical models of security systems are developed to analyze overall performance and predict the likely behavior of the key decision makers influencing the protection structure. The initial objective here is to create a reliable intrusion detection mechanism to help identify malicious attacks at a very early stage, i.e., in order to minimize potentially critical consequences and damage to system privacy and stability. Furthermore, another key objective is also to develop effective intrusion prevention (response) mechanisms. Along these lines, a machine learning based solution framework is developed consisting of two modules. Specifically, the first module prepares the system for analysis and detects

whether or not there is a cyber-attack. Meanwhile, the second module analyzes the type of the breach and formulates an adequate response. Namely, a decision agent is used in the latter module to investigate the environment and make appropriate decisions in the case of uncertainty. This agent starts by conducting its analysis in a completely unknown milieu but continually learns to adjust its decision making based upon the provided feedback. The overall system is designed to operate in an automated manner without any intervention from administrators or other cybersecurity personnel. Human input is essentially only required to modify some key model (system) parameters and settings. Overall, the framework developed in this dissertation provides a solid foundation from which to develop improved threat detection and protection mechanisms for static setups, with further extensibility for handling streaming data.

Chapter 1

Importance of the Intrusion Detection Domain in Network Security

Nowadays, the significant development of our computer systems transformed our daily life entirely and made our existence reliant on them. According to Cisco Visual Networking Index 2017 [77], there are expected 3.5 computer devices per capita worldwide in 2021 and almost 106 Terabytes per second of global Internet traffic. With the rapid progress of the Internet, our computer structures are exposed to an increased number of threats. Although the research and technological innovations in are progressing rapidly, an absolute cybersecurity remains a challenge.

The IDS observe the network traffic, analyze it and identify possible anomalies or unauthorized access to the network behavior. Some of the IDS also respond to the intrusion, which is a necessary measure in protecting our computer network. There are several limitations and problems of the existing methods that we will address in this chapter and attempt to solve with the proposed off-policy Q-learning intrusion response model. On the one hand, exploitation and misuse of resources happen, because the IDS is designed to observe the network all of the time; consequently, resources are utilized even if no attack is occurring. On the other hand, although the flowing traffic is examined continuously, once an attack is detected, there is a considerable time necessary for a response to be provided. The network traffic often travels a certain distance in the form of packets; moreover, the intruder can alternate or even terminate it before reaching the IDS. Another challenge is to provide a reliable way of protecting our system or to what extent we can trust the IDS. The administrators should regularly update their protection mechanisms; otherwise, once the intruder recognizes specific weaknesses and limitations, he will send even more attacks, therefore challenging the detection system.

The common classification procedures are based on familiar machine learning models such as Naive Bayes [51] or discriminant models such as Support Vector Machines [50] and [60].Those machine learning approaches study primarily a set of network characteristics (where there is no

order or sequence in the specifics of the network), and their objective is typically to calculate a category score. Furthermore, the question that arises is how properly they will perform in network classification, especially when we observe more noise and the information structure is not homogenous [50]. We need to test how accurate are these methods for predicting categories for large networks, where the information is concentrated in only a few characteristics. Besides, the focus will be on assisting in an approachable way the potential readers of this work to discover an efficient defense mechanism that will protect their network system. As an example of the crucial role of the cybersecurity, the progressively-increasing use of wireless technology is making networks even more vulnerable to attacks. Nevertheless, advanced and sophisticated challenges make the classical security measures, such as a firewall, inadequate. If a hacker wants to steal data, he will not try to penetrate the firewall, but he will search for the least secure access to take over control of the system. The cyber-criminals are continually inventing new techniques. Therefore, we need to find effective solutions that can dynamically and adaptively defend our systems.

In the current dissertation, we will introduce an automated machine learning algorithm for Network Intrusion Detection and Prevention Systems (IDPS) that will serve as a successful tool for defending our network. The proposed IDPS incorporates two layers: the primary layer will provide us with the opportunity to monitor the network and to detect a breach in our system, based on machine learning dimensions reduction and classification techniques. The second layer will generate an automated response to an eventual attack, given the classification outcome, generated by the detection module and will employ Q-learning mechanism to prevent and in some cases to response back to the attack. This network IDPS is entirely adapted to streaming data and can be implemented to any firewall, where online data analysis is required.

Most of the IDS systems operate on their own in conjunction with a firewall. However, they are restricted to their detection and monitoring functions. Therefore introducing an intrusion response mechanism that will work together with the IDS will allow comprehensive protection, that will not only secure our systems but also create a unique agent that will make an automated decision in an unknown environment. In the end, we will represent the whole interaction process among the attacker and the and the protecting agent as a Game theory approach that could serve as a balancing and strategizing tool for predicting the behavior of the attacker and based on the solution to that game, to find an equilibrium point for optimal results.

The proposed schema is an advanced technique of machine learning systems for features reduction and classification in network security set up. It will be capable of outperforming some of the current methods, used in cybersecurity and will be effective against not just one but many types of attacks. As an addition the suggested representation of the computer network as a probability structure is an innovative approach in which the decision agent will operate in an unknown environment, obtain feedback and based on the received reward, he will learn to select an optimal policy, so the network to be continuously protected.

Chapter 2

Literature Review of Machine Learning Methods in Intrusion Detection Systems[1]

2.1 Related Work Machine Learning in Intrusion Detection and Prevention Domains

2.1.1 Classification Related Machine Learning Methods

Cannady [44] and [56] made some of the earlier works on network classification. He indicates that neural networks are reasonable solutions when they are trained for a specific problem domain with representative sets of training data. The model was not able to handle streaming data, and therefore, it is necessary for the individual protecting our system, to take off-line the data whenever he needs to train the model and to run it to the updated set of representative data. Furthermore, the authors employed a three-layer control feedforward mechanism intended to yield a series of input-output mappings. The particular Intrusion Detection System (IDS) agent, consequently, learns how to spot flood-based Denial of Service attacks based on Internet Control Message Protocol (ICMP) together with the User Datagram Protocol (UDP). The method initially studies how to detect the ICMP attacks and as a result, updates and retrains the model frequently. Therefore, it is capable of learning about how to recognize new attacks based on the UDP protocol. Cannady applied a Cerebellar Model Articulation Controller Neural Network, to propose an online-learning method. The authors proposed a multiple layer mechanism, intended to generate a sequence of input-output mappings. The system initially is trained how to identify ICMP breaches and to use prior experience and training it determines how to recognize new attacks based on the UDP protocol. However, the method is concentrated only on the flood-based Denial of Service attacks. One method employed to detect anomalies in IDS is built upon the analysis on the sequences of system calls.

[1]Portions of this chapter have been previously published in Proceedings of Dynamic Systems and Applications , vol. 7, 2016, pp. 303-310

In [57] the researchers used Random Forest and Regression Trees to recognize breaches in the system . Nguyen et al. [55] employed Principal Component Analysis (PCA) for outliers and anomaly detection in IDS. Shilpa et al. [40] compared different methods for attribute selection and examination of abnormality detection. It is an important task to select a proper model, which will assist the further analysis and outline an optimal solution in dealing with the tradeoff between accuracy and complexity. One approach used to find breaches in host-based intrusion detection systems is established by observing sequences of system calls. A process running on the host initiates these calls, and they are grouped in sets of traces. Each trace includes a file of system calls produced from the start to the end.

Many of the IDS research publications can be summarized as machine learning classification problems, which are solved with supervised or semi-supervised learning models [76]. Although some authors attempted to implement unsupervised learning, they achieved low accuracy [73]. RL has been widely employed in computer network disciplines; however, the utilization in the intrusion detection or intrusion preventions area has not been substantially explored. Scientists perceive considerably intriguing the domain of routing protocols, validation processes, entrance control and service quality mechanisms. The attentiveness is because RL is reasonable for control situations, where a response from the environment exists [75]. In all the occurrences mentioned above, we detect feedback, which is represented as a reward.

2.1.2 Reinforcement Learning in Network Security

Xu et al. [78] implemented reinforcement learning in an association with Hidden Markov Models (HMM) to identify breaches by learning the state transition probabilities. The authors claimed that HMM could offer a suitable estimation of the state transitions on IDS. For the value function to be modified, they applied temporal difference methods and obtained results, using the same training and testing sets. Two years later, Xu and Luo [79] modeled the network behavior with a temporal-difference approach. In this work, they achieved better detection accuracy, matched to the previous implementation of HMM approaches. To estimate the value function and to implement parameter reduction, they used a kernel least-squares temporal-difference algorithm (LS-TD) [80]. Xu and Luo provided a practical solution for IDS to affirm the quality of the outlined model; they used system calls traces from the send mail application. Miller and Inoue [74] used a model called

Perceptual Intrusion Detection System with Reinforcement, which operates with multiple different agents. A single agent can employ a self-organizing map to detect malicious activities, and there is a blackboard technique for assembling the results provided by all agents. Once a signal is detected within the system, it is distributed to all agents for collective group analysis. They send votes to the central blackboard system, which computes weights, and it rewards the agents depending on their performance.

2.1.3 Machine Learning Techniques for Streaming Data

Some of the early literature related to data streaming manipulation originates with the concept of employing partial and not full memory of data storage. The sliding-window notion represents the idea that at each time a window will contain the most up-to-date information that is associated with the learning process. It is challenging to decide the windows size W, so the most straightforward approach for solving this problem is to let the user select of how large to set the window and keep it fixed while the algorithm is executed [84]. The detection of the change is achieved by using a score, that can recognize the indication of the change between current and referred window. Other proposals denote the concept that we should keep in the memory only aggregate statistics with a "decay function," that signifies the importance of those aggregates over time [85].

In other approaches, the drift detecting concept is established by monitoring the rates of three operational thresholds [86] such as precision, accuracy, and recall. Their values are analyzed with a moving average, concerning the standard sample errors' confidence intervals (using the latest set of examples) of each particular indicator. The fundamental purpose is to decide upon a window size of data so that the predicted error on new instances is minimized. The approach uses unlabeled dataset. Therefore the model does not require complex calculations, and it is easy to be applied. Another approach [87] for recognizing shifts in the distribution is by observing a real-time error-rate of the algorithm. In this method, learning is performed in a series of tests. If a new testing instance is available, it is labeled utilizing the present model. The authors determine a warning k_w and drift levels k_d that are activated once the error rate reaches predetermined levels. Those levels serve as a suggestion for a distribution shift of the instances.

The drifting idea can be summarized in two classes [88], [89], [90] and [91]. The first class outlines procedures that apply the learner periodically without analyzing the occurrence of changes.

The second category summarizes the research where we detect a change first, and the learner adapts to it. Representatives of the strategies as mentioned above are the time windows of fixed size and weighted instances. The weighted instances are characterized by the notion that the value of an instance diminishes over time [88], [89], [92], [93] and [91]. If a time window is utilized, the learner is affected only by instances that are included in the window. A small window can produce bias results. There exist feasibility to utilize a time window with an adaptive size, or modifying the window according to the concept of drift [88]. FLORA2 developed by Widmer and Kubat [91] introduces an idea for window change for classification purposes that is based on specific rules. To recognize distribution changes, the authors proposed a window size, based on the accuracy and the applied learner. Klinkenberg and Lanquillon [90] and [92] assessed the accuracy and the precision as monitoring performance indicators. They estimated the standard error and also its confidence intervals and applied a moving average. There are a couple of challenges in this methodology. The first one is related to the limited information that we possess about the real class most of the time. The second one is associated with the substantial amount of parameters that has to be tuned. Klinkenberg and Joachims proposed a robust approach to recognize changes using support vector machines [90]. The central concept is to select an appropriate size of the window so that the expected error of the new examples to be as small as possible.

A shift detection process can be two varieties: sequential change detection or batch change detection. Given a series of observations, the assignment of the batch design is to recognize a modification at a specific point in the series by using all possible instances. However, the shortcoming of is that its running time is considerable when detecting changes in a significant amount of data. In contrast, the sequential change detection problem is based on the observations until the current time. If no change is detected, the next instance proceeds. Whenever a difference is noticed, then the detector is reset. Let us assume a series of pairs $\{x_i; y_i\}$, for any instance, we can estimate $\hat{y}_i$, however this estimate can be either correct if $y_i = \hat{y}_i$ or a mistake if $y_i \neq \hat{y}_i$. Therefore each error rate for a specific pair will follow a Bernoulli distribution, and If we observe a sample of n instances, the error rate will follow a Binomial distribution. Where p_i is the probability of the model of making an incorrect prediction, and i is the instance in the sequence. Probably Approximately Correct [94] learning supports the idea that if there are not any changes in the distribution, then the learning model's error rate will diminish as the number of instances grows.

The large growth in the error indicates a change in the class of the underlying distribution. Most of the authors assume that for an adequate large quantity of instances, the Binomial distribution, they can approximate the Binomial distribution with Normal. If the distribution is stable while the model is stationary and if $n > 30$, the p_i has a $1 - \frac{\alpha}{2}$ confidence interval approximately equal to $p_i + \alpha * s_i$. The distribution shift model maintains two estimators as the model is being trained the minimum values of p and s. A warning level is established, when for every distinct currently analyzed instance i, the sum of $p_i + s_i$ is less than the benchmark sum of $p_{min} + s_{min}$. The new window includes some of the former instances which are in the current setup and a small number of instances from the old one. If the warning confidence level is 95%, a warning level is attained if $k_w : p_i + s_i \geq p_{min} + 2s_{min}$, however if a drift level is required with a confidence of 99%, then it will be $k_d : p_i + s_i \geq p_{min} + 3s_{min}$. If the drift level is reached, this will be an indication for distribution change of the instances. Once the drift level is detected, the model starts automatically rerunning the algorithm for retraining the data.

2.2 Literature Review in Stochastic Game Theories [1]

This literature section intends to outline a brief survey of stochastic game methods in cybersecurity [81]. The rapid development of this area explains the abundance of literature and models. Due to our specific emphasis and unintentional overlook, deserving references may not appear in this work. The primary emphasis when describing the variety of theories will be on how to apply these models in a cyber security set-up.

John Von Neumann (mathematician) introduced game theory in 1928 for the first time as a mathematical instrument, used to define and solve games [37]. It is a useful, analytical and quantitative approach for characterizing interactive decision situations and also handles dilemmas between rivals with competitive goals. Game theory has the potential to analyze various potential outcomes at a time, and therefore to provide each player with an optimal, feasible and strategic set of actions. A stochastic game is a system, which consists of two or more players that follow some rules with a probabilistic transition, strategically interact with each other, make decisions based on their current or past information, search for the best resolution for their current (or future) outcome and act

[1] Portion of this section was previously published in the Dynamic Systems and Applications, vol. 7, 2016

accordingly. They can continue to play the game forever or win or lose. A defense game illustrates the interplay between foes and intrusion detection systems (IDS) that utilizes resources for monitoring the network and some cases to respond. A mathematical decision structure is needed so we can analyze problems related to modeling the attacker's behavior, limited resources allocation, and selection of an optimal response, [1], [3], [8], [13], [15], [19], [23], [24] and [36]. In this chapter, we will adopt the idea that the game is played under the assumption of rationality. In this section, we will concentrate our effort on describing stochastic games and evaluate the different types of games, according to a specific informational structure. Since the players make their actions and strategies based on the available past or current knowledge, it is crucial to recognize the informational structure first, so that we can think of an appropriate solution to the game.

Information performs a critical function in the game theory. The primary importance level is because it provides us with an outline of different possible strategies that the players might undertake.

- Dynamic Games of Complete Information

  Complete information implies that every player possesses knowledge of the strategies and returns of the other players participating in the game, but they may not be aware of the particular actions of the others in the game.

- Dynamic Games of Incomplete Information

  These games exist when one or more of the players are not aware of the potential payoffs and strategies of all other or at least one other player. We can evaluate the situation when the attacker has a higher information level and exploits the resources of the defender.

In the stochastic games with complete and perfect information, every agent possesses an awareness of the actions of all other agents that were in the past. They identify the policies and the other agent's payoffs. In these games, they are aware of the complete history of the game. Usually, there is one leader, and then the rest of the players are followers.

Many authors use a two-player zero-sum game for modeling a robust protective mechanism. For example, in the model of Nguyen et al. [28], the attacking and the protecting agents play a zero-sum game. Again, they used nodes to model the system, but this time they made the nodes

correlated to each other, depending on some weighted factors related to the security assets involved in the process and the vulnerability dependency of the nodes. The same idea was developed in [26], where the authors considered a practical example to explain and test their idea of finding an optimal strategy.

Players move at different, sequential moments and their return functions are shared information. Every time, they move simultaneously, and the agents are in general not informed for the past actions of the other agents. A solution of that type of game is provided by Selten (1965) Sub-game-perfect Nash Equilibrium (SGPNE). A Nash Equilibrium is sub-game perfect if the strategies of the players establish a Nash Equilibrium in each sub-game. SGPNE includes not only the optimal feedback to the unique action, played in the first stage, but also provides a full plan of action (strategy) with a suggestion of what would be the most optimal approach to reply to any possible action in the unknown portion of the game (sub-game).

The main problem discussed in these types of games is to define the optimal policies for the defender to diversify the risk when he builds his strategy against the attacker and to find an optimal defense strategy. Since we have a sequence of actions and there is a transition process involved, some of the authors describe the system as a Markov process. Several approaches to finding a solution of the game were described in the past. Some of the most popular methods are Q-learning [7], NPL1 [9], which is an algorithm that helps us to find an optimal solution with Nash Equilibrium and Shapleys method [35].

Sallhammar et al. [34] presents the game as a two-player game in which there is not any interaction between the efforts of the opponents, involved in the system. The status of the network, may or may not be subject to change. For example, in one standard set up where there is a defense mechanism, it is possible for the system to reboot because of different reasons. We can model the game as a stochastic Markov process. The relationship of the players in the game will affect this transition matrix in a way that depends on the different strategies that they use. The whole process is represented as a Markov decision process (MDP) by Filar and Vrieze [9], where the transition probabilities are subject to change, conditional on the agents' actions and the next state can be obtained based on the current and previous states. The MDP turns out to be one of the most helpful approaches, which can provide us with a tool that is convenient to be calculated. It also takes into account changes that might occur in the system. In [5] we can find different optimizing

mechanisms and tools for dynamic programming.

The interaction between attacker and the IDS (intrusion detection system) was presented by Alpcan et al. [2], [3] and [4] as a Markov game. They considered three possibilities for information availability, if: (a) the attacker and the defender have full information about the system, (b) the attacker has no information (c) nobody has any information for his or her opponent, but only about their own costs, actions in the past and the previous states. Main mechanisms for finding optimal strategies of the agents were minimax-Q [20] as well as naive Q-learning, defined in [5] and [7].

Xiaolin et al. [38] stressed the importance of risk assessment in cybersecurity, the authors proposed an automatic generate reinforcement Markov model that will assist the administrator in protecting the system. They considered the potential and the current security status and assessed the risk as a combination of vulnerabilities and threats. They also created a function to measure the harm caused by the attacker, and it represented the level of risk involved in the process. According to this function, the administrator will select a strategy that will minimize the maximum possible damage to the system. To assess their model, they considered four different sub-systems, which are united together, so the best decision process to be made. Fault Tree Analysis (FTA) were presented in [31] and [18]. They were based on Chain-of-Events Model, together with COBIT 15 [25], and were described as the basic methods for analyzing the reasons for hazards in our system. Tree construction demands an in-depth evaluation, emphasizing system problems and facilitating improvements by an analyst.

In the dynamic games with incomplete information, the agents possess limited knowledge about the return functions of their opponent. However, they are aware of the past actions of everybody else. An example of these games is the two-player zero-sum game that also serves as a base for the Intrusion detection mechanism. It illustrates the connection between criminal attackers and IDS which allocates resources for detection and response [3] and [6]. IDSs observe various events in the cybersecurity and examines them for signs of a safety problem in the protection process. It is becoming more and more apparent that the traditional protective measures such as firewalls or virus and malware prevention systems are not adequate to deal with sophisticated attackers. The majority of the research on IDS is concentrated on an empirical application. A mathematical decision framework may be vital for predicting the behavior of the attacker.

Patcha et al. [30] incorporates a signaling game to present the intrusion detection mechanism.

The defender has incomplete information because he does not know what type his the other player is. A Perfect Bayesian Equilibrium is such a combination of strategies and beliefs, which offers an optimal approach, conditional on the opinions of the agents in every state. Perfect Bayesian Equilibrium is always Nash Equilibrium, but not the other way around. Given the players' beliefs, the strategies must be sequential, i.e., at each information set the actions taken must be optimal [30].

Nguyen et al. [27] described the system safety dilemma as many consecutive games that are played by both agents. Nguyen et al. observed this type of play as a fictitious play because the participants did not know the previous actions of their opponents. The authors observed the influence of the so-called error probabilities in the process, and they considered the implementation of a sensor system based on two main scenarios: (a) each player knows the error probabilities, and (b) none of them knows it. Other authors also considered the fictitious play in their analysis. For example, in [22], Luo et al. suggested design to outline the uncertainty of the attacked object. In a similar way Liu et al. [21] developed a Bayesian game in a wireless network. Each node was assigned a transition probability, and two situations were discussed: a fictitious and a gradient play, as the players revise their expectations at the completion of each state.

Incomplete and imperfect information implies that one of the agents is not aware of the actions and payoff functions of the other players that were in the past. One possible representation of these types of games is the Two-player hybrid Bayesian type of a game; the players amend their beliefs for the nature of their rival. The solution of the game is a sequence of updated strategies, based on the beliefs of the agent. For example, in [21] the authors suggest several possible solutions with the Bayesian approach to solving a game with incomplete and imperfect information.

The authors in [39] designed a model of interaction between the hacker and the defender. They suggested that this model could be represented as a repeated game. Linear algorithms were employed for finding a solution to their game, and they provided a deep insight into the Bayesian and Nash Equilibriums. The authors proposed two methods for finding the solution the Min-Max method [29] and a linear programming [22]. Additional research for these type of games is made in [3], [27] and [30].

2.3 Other Related Work

Sallhammar, et al. [33] proposed an approach of integrating reliability and cybersecurity. The authors implemented a stochastic game to predict the hackers' behavior. The basic idea is to evaluate the relationship between hacker and administrator as a two-player zero-sum game. The researchers calculated the transition probabilities and created and solved a Markov model.

Hansman and Hunt [12] introduced a classification including four distinct dimensions. As a whole, their classification system includes specific types of breaches, assisting with the security by developing the consistency in the language which is describing the different types of attacks. They suggested that a robust style with a detailed description of the distinct attack types can enhance the system. The first dimension is assisting the administrator to categorize the breach, the second dimension emphasizes the classification of the breach victim, and the third one outlines the process of representing the separate levels of vulnerability. The final aspect describes the potential effects that will be obtained before the final act.

Hausken [11] also used a strategic reliability approach to describe the game model. His work recommends different techniques, depending on the type of the network. He uses Markov analysis to repeated games and studies the strategic defense of a system, which has been targeted by multiple attackers. Hausken takes into account the essential dissimilarities of the network elements and considers several parallels and complicated series of defensive techniques. In [10], the authors describe the optimality process of a possible interaction between the hackers, depending on the information structure and availability. They consider a process of building a shared information platform among the hackers so that the hackers can be aware of the level of vulnerabilities among the different protected systems. That implies that the weakest systems will be targeted and there are better chances of a successful attack on the protected system. Kjaerland [16] proposed a taxonomy of cyber-intrusions to profile cyber-criminals and victims. His primary insight is an emphasis on the examination of both the attacker and the defender. He also focuses on reported cyber intrusions by Computer Emergency Response Team. These breaches were investigated implementing a multiple dimension scale, represented by the following categories: a Method of Operation, Target, Source, and Impact. He concluded the chapter by distinguishing the commercially related and the associated government breaches; he also stressed the importance of understanding intrusions.

2.4 Concluding Remarks

This chapter summarizes the research made in the IDPS domains with regards to the different machine learning classification and Reinforcement Learning processes. It also demonstrates a promising future application and efficiency of the game theory in cybersecurity. Additional research needs to be done in analyzing the strategies and the solutions of the players, according to the informational structure in cybersecurity. Furthermore, there is the existence of different challenges associated with the theoretical framework and the practical application of the possible variety of games. It can be stated that there are some problems in quantifying the diverse factors that define the game. So far, the applied research on game theories has been limited to the computation of the Nash Equilibrium and the use of other related classical approaches. However, new and advanced methods need to be implemented to account for the fast developing cyber environment and the innovative strategies, invented by the attackers nowadays. Firewalls and other intrusion detection mechanisms may be useful for our basic protection, but new and high-tech software and hardware applications are necessary so the administrator can create a quick and adequate response to each possible attack.

Chapter 3

Feature selection and Classification process for predicting the attacker's behavior[2]

3.1 Introduction

The Network Attribute Selection, Classification, and Accuracy (NASCA) procedure is a four-step procedure for intrusion detection systems. The model first begins by extracting the information from the Transmission Control Protocol (TCP) and then ranking the relevant information that characterizes the network. During the second stage, it classifies the network as being under attack or not. Moreover, if the model detects that the system is under attack, it executes another level of a classification approach to identify the type of attack that is occurring. The third stage provides us with results about the accuracy of the performed analytical estimate. The attribute selection method begins with choosing a subset of appropriate information by removing redundant, unrelated, and noisy data from the original dataset. The classification method starts initially with an existing set of labeled networks; consequently, it learns the dependency between the content of the network and its corresponding label and then predicts the label of a set of unlabeled networks as accurately as possible using a decision tree type of analysis. The goals of this chapter will be the following; (1) To propose a NASCA procedure for intrusion detection systems; (2) To test it on a real data set; (3) To report the obtained results; and (4) To provide evidence of why this procedure can outperform the existing ranking and classification techniques.

3.2 Used Methodology

3.2.1 Network Intrusion Detection - Outline of the Procedure.

The classification agent is capable of making decisions in a constantly changing environment and therefore testing the model while evaluating the network. As an addition, an essential advantage

[2]Portions of this chapter have been previously published in IEEE Xplore Digital Library, 2017

is the fact that only relevant network information will be used before the particular classification decision is accomplished. Under those circumstances, the agent can learn how to classify the network promptly, accurately and efficiently.

The steps of the procedure are presented in Fig. 3.2.1:

1. Network Data Collection: Collect the network data from the TCP/IP, using for any raw socket communication instrument for reading the data;

2. Attribute Selection: Apply an Information Gain Attribute Evaluation Approach, which is a tool for feature reduction and attribute ranking for classification purposes (Mitchell, 1997);

3. Classification: Provide a classification procedure, which will learn how to label the network precisely and promptly. In this work, we will use a two-stage procedure:

   (a) Classify the network as being under attack or not. In the first step of the classification, we will use a Random Forest (RF) to label the network as being under attack or not; if Step (a) labels the network as being is under attack, then use step (b).

   (b) Classify the network further only if an attack is occurring. We use a partial decision tree (PART) method to classify further possible estimate for the type of the attack that is undergoing.

4. Accuracy: Here, we will demonstrate the accuracy of the estimate for the specific dataset used. Once the data is trained, and a model is built, accuracy factors are reported.

3.2.2 Attribute Selection, using the Information Gain Attribute Evaluation Approach.

Attribute selection is the method of recognizing and eliminating the irrelevant and redundant information from an evaluated system. If we can eliminate some of the unrelated features, we can moderate the complexity, by removing irrelevant dimensions and enhancing the performance of the prospective classification procedure. The decision agent is capable of operating not only quicker and with less information, but also to improve the classification accuracy process, [48]. There exist a variety of different proposed attribute selection methods. For example, Hall and Holmes et al.

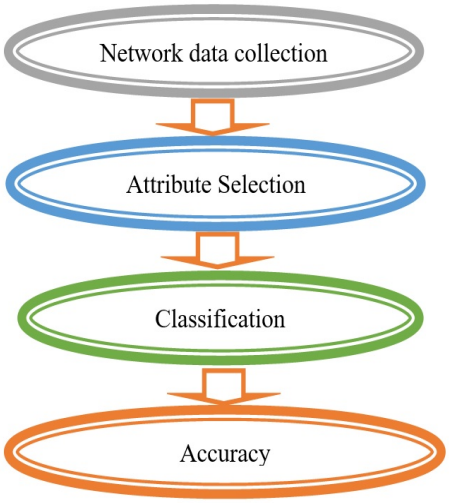


Figure 1.: NASCA Procedure

analyzed number of these attributes selection techniques and outlined the ones that achieved noticeable results, namely [49] Correlation-Based Feature Selection, Information Gain Correlation, Wrapper Subset Evaluation [53], Recursive Elimination of Features [52], and Consistency-Based Subset Evaluation [54]. The main idea of the attribute selection procedure is to rank the relevant variables and henceforth to use only the appropriate information to perform the classification of the network. Attribute reduction is the process of mapping the existing high-dimensional data onto a lower-dimensional space. For example, for a given dataset points of n variables $\{x_1, x_2, , x_n\}$, we need to compute their dimensional representation $x_i \in R_d \rightarrow y_i \, (p << d)$. The criterion for feature reduction can be different based on diverse problem settings. In this chapter, we will test different ranking algorithms. Consequently, we will provide results, so that we can demonstrate the outperformance of the Information Gain Ranking filter in comparison to other algorithms for attribute selection. Information gain (IG) quantifies the volume of information in bits about the class estimate. It measures the expected decrease in entropy of the class variable after the value of the feature is observed (uncertainty associated with a random feature) (Mitchell, 1997). It is an entropy-based filter that categorizes the gain of the attributes. For example, an Entropy for i classes can be defined as:

$$H(X) = -sum_i P(x_i) log_2 P(x_i) \tag{3.1}$$

Entropy signifies the level of insecurity in the system. In the equation (1), $P(x_i)$ is the marginal probability density function for the random variable X, which is obtained by integrating the joint probability density function. First, we observe the values of X in the training dataset S and separate them according to the values of a second feature Y. Correspondingly, we measure the entropy of X with respect to the partitions induced by Y. In the case that the measure of the entropy is smaller than the entropy of X before partitioning, we say that there is a relation between features X and Y.

$$H(X|Y) = -\sum_{j} P(y_j) \sum_{i} P(x_i|y_j) log_2 P(x_i) \tag{3.2}$$

$P(x_i|y_i)$ is the conditional probability of X given Y. Having in mind the fact that the entropy is a condition of impurity in training set S, we can describe a measure reflecting additional information about X provided by Y, which represents the amount by which the entropy of X decreases. This measure is known as information gain and it is given by:

$$IG(X|Y) = H(X) - H(X|Y) \tag{3.3}$$

The larger the value of the informational gain IG, the further the attribute contributes to the data set. Although, a disadvantage of the IG criterion is that it will favor attributes with more values because it is biased towards choosing attributes with a more significant number of values that produce higher IG. However, given the characteristics of our particular data set, IG is a preferred instrument for attribute selection.

3.2.3 Classification Process

The classification method is an essential step of the proposed procedure. It is a two-stage process with objectives - accuracy, precision and faster classification. Therefore, to accelerate the procedure, further analyses will be completed, only whenever the network is classified as being under attack. The proposed procedure combines two major classification techniques, namely Random Forest and consequently partial decision tree (PART). In Fig. 3.2.3 we can observe the main idea of this stage and how the classification process is analyzed. Breiman [41] introduced Random forest, it is a cooperative learning method that produces various classifiers and summarizes the outcomes. As an addition, it can be executed if needed with two major procedures to perform the

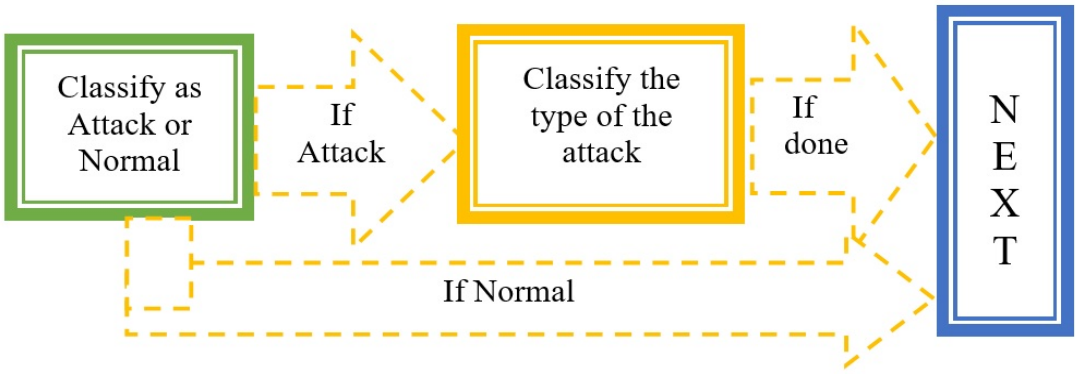


Figure 2.: Classification Process

classification or prediction analysis, namely boosting and bagging. On the one hand, in boosting, the succeeding trees assign additional weight to instances that were incorrectly classified by earlier trials, and at the end, a weighted score is calculated for the classification purposes. On the other hand, in bagging, the succeeding trees are independent of the previous trees. Moreover, every tree is grown utilizing a bootstrap sample.

The classification is accomplished based on the so-called majority score split (Liaw and Wiener 2002). The random forest grows multiple trees, and each of them produces a classification with an assigned score for the specific class. As a result, the forest indicates the classification with the highest score. The term originated from random decision forests that was first proposed by Tin Kam Ho by Bell Labs in 1995. Random forest (RF) is a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution of all trees in the forest. The error of a forest of tree classifiers depends on the score of the distinct trees in the forest and the correlation among them. The Random Forest process in our analysis starts with the creation of many trees. It introduces randomness into trees such that each tree has a minimum correlation with the other trees. Each tree in the collection is formed by first selecting at random, at each node, a small group of the input characteristics to split and, secondly, by calculating the best split based on these features in the training set. The method that we will use in the splitting process is a two-step randomization technique. Initially, the tree is grown by using a bootstrap sample, and then we introduce another stage of randomization, using random feature selection approach. In a summary, instead of splitting the node of the tree using all k features, we

will randomly select at each node of each tree a subset of m-tries, where $m \in [1, k]$ for splitting the node. Brieman et al., 1984 outlines some of the splitting suggestions and development of the tree. The method suggests a subspace randomization structure that is combined with bagging, Buehlmann, and Yu, 2002 [42]. The idea is to re-sample, with replacement, the training data set whenever a new individual tree is built. Biau and Devroye (2010), as well as Meinshausen (2006), studied the consistency of random forests in the setting of conditional quantile prediction [45].

The detailed procedure of RF starts with the creation of a new random vector n for each n-th tree that is independent of the previous random vectors θ_1, θ_2.... Moreover, it is generated from the same distribution and based on a training set θ_n; a tree is grown. Consequently, the tree organization of RF is based on classifiers $\{h(X, \theta_n, n = 1, 2, 3\}$, where $\{\theta_n\}$ are i.i.d. random vectors. Respectively, each tree is assigned a score as described above and an input vector X.

If the network is under attack, then we will employ the partial decision tree PART [46] and [47] method for classification purposes has several advantages compared to the other methods. PART will label the network with the type of the 37 types of attack, and therefore it will perform the process faster rather than the RF. The reason of why we will employ PART algorithm instead of RF again will be because it is a rule-based method, which does not need to achieve a global optimization to produce accurate results, which will speed up the classification process. It knows how to label new occurrences quickly and possesses an outstanding accuracy and precision. Moreover, it adopts the separate-and-conquer approach, and consequently, once it constructs a rule, it eliminates the covered alternatives. Hence, it keeps repeatedly creating rules for the residual instances until it executes all possible outcomes. In essence, to create a single rule, a pruned decision tree is built for the current set of instances. As a result, the leaf with the highest coverage is converted into a rule, and afterward, the tree is discarded.

The idea of recurrently building decision trees only to reject the majority of them in PART is not as unusual as it seems. A pruned tree can be employed to obtain a rule, instead of constructing it incrementally by adding combinations one by one. PART is capable of avoiding the over-pruning problem of the rule learner - separate-and-conquer [43]. The model performs with improved speed, although the above advantages are still achieved. The fundamental idea is to build a partial" decision tree instead of a fully explored one. A partial decision tree is a regular decision tree which builds divisions to unknown subtrees. Once a partial tree has been built, a single rule is produced

based on it [43]. The aim is to find the supreme general rule by choosing the leaf that covers the highest number of instances or the leaf with the lowest error rate.

3.2.4 Accuracy

The Accuracy step is related to testing the performance of the model based on two main criteria: accuracy and complexity. A too complex model will take a longer time to classify the network, as an addition, there is a trade-off between complexity and accuracy. Different classification and attribute selection models are performed, and a ranking comparison is made based on those two criteria. The results are reported with regards to the cost analysis of the complexity, Kappa statistic and the error rate for accuracy. The data set is tested on the basis of 80-20% split, and the results are compared and reported, depending on the applied classification methodology.

3.3 Data Description

A network is structured by Transmission Control Protocol (TCP) packets starting and ending at some well-defined time between which data streams to and from one source IP address to another target IP address under a determined protocol. Each network is labeled as either normal or as an attack with exactly one specific attack type. The data that are used in this paper are ISCX NSL-KDD Data Set that is an improved version of the KDD CUP 99, DARPA, conducted by MIT Lincoln Labs. Lincoln Labs simulated an environment to obtain nine weeks of raw TCP dump data for a local-area network (LAN), pretending a typical United States Air Force network. Moreover, they operated the LAN as if it were a true Air Force atmosphere, and simulated numerous attacks. Even though the contribution of DARPA and KDD (University of California) dataset is remarkable [59], their ability to reflect real-world situations has been widely questioned, McHugh (2000) and Brown et al. (2009). Therefore, to conduct meaningful research, in the current paper, we will use the ISCX NSL-KDD dataset, provided by The Information Security Centre of Excellence (ISCX) within the Faculty of Computer Science, University of New Brunswick, Canada. The data in the NSL-KDD dataset is either labeled as normal or as one of the 24 different kinds of attack. Additionally, these 24 attacks are clustered into four groups: Denial of Service (DoS), Probing (Probe), Remote to Local (R2L), and User to Root (U2R). The attack types according to the dataset

Table 1: Types of Labels of the Network

Types of network			Number of instances
Normal			67,343
Attacks (37)	DoS (11)	Land, Pod, Smurf, Teardrop, Mailbomb, Processtable, Neptune Udpstorm, Apache2, Worm, Back	45,927
	Probe (6)	Satan, IPsweep, Nmap, Portsweep, Mscan, Saint	11,656
	R2L (13)	Guess_password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named	52
	U2R (7)	Buffer_overflow, Loadmodule, Rootkit,Perl, Sqlattack,Xterm,Ps	2,756
Total			127,734

are summarized in Table 1.

There are 42 variables, one of them represents the condition of the network, labeled as being under a specific type of attack or normal. The dataset is balanced in a sense that the number of attacks and the number of normal instances is balanced. They are summarized in three categories. (1) Essential features: this group contains all the characteristics that are collected from a TCP/IP. (2) Traffic features: this class outlines the features which are measured with regards to a period of time, and they are divided into two clusters: same host and same service features. (3) Content features: to recognize the suspicious behavior, we may evaluate features like the number of failed login attempts.

3.4 Results

The primary objective of the initial step is to eliminate the redundant variables and the features which do not contribute to the classification process. The dataset contains 127,734 observations, and the test is done based on the 80-20% split. A variety of attribute selection methods are tested before selecting an approach. This model is chosen concerning some eliminated variables and the time to perform the elimination. The nominated method for attribute selection is the Information Gain Ranking Filter; wherein total eleven variables are excluded from the further analysis since they do not contribute to the classification process, Table 2. Additionally, in the next step of the procedure, we will analyze how the model increases its performance, based on the attribute selection step. The top three variables are likewise essential to appropriate evaluation, especially

Table 2: Attribute Selection Results

Select attribute method	Eliminated variables number	Total	Top 3 variables	
			Name	Number
Information Gain Ranking filter	20,22,18,17, 19,7,9,15,11,16 ,21	11	src bytes, service, flag	5,3,4
Correlation Attribute evaluation	18,20,7,21	4	same serv rate, srv serror rate, serror rate	29,26,25
Gain Ratio Feature evaluator	15,20,22,11, 19,18,7,16,9,17, 21	11	wrong fragment, root shell, dst host serror rate	8,14, 40
Relief Attribute Evaluation	18,20,7,21,5	5	flag,serror rate, srv serror rate	4,25, 26
Symmetrical Uncertainty Ranking Filter	20,22,18,17,19, 7,9,15,11,16,21	11	src bytes, flag, diff srv rate	5,4,30

whenever the administrator is concerned with the vulnerability of the network. However, the objective of this article is to present how to effectively classify the network based on the information structure that we can obtain from the TCP packets. Therefore, we intend to reduce the dimensions of the data set, consequently abandon the features that do not contribute to the classification process. The Information Gain Ranking filter is an appropriate method for attribute selection, established with the assistance of our data set. It completes a prompt and accurate analysis and eliminates a significant number of variables, which are not influencing the ranking process and the performance of the classification step. The classification process is the next phase of our proposed procedure as described above. After we eliminate the variables that do not contribute to the model, the objective here is to classify the network as being under attack or not. Several discrimination models are tested and compared, and the best concerning cost structure and accuracy is selected. Based on the Random Forrest (RF) classification method, subsequently, the suggested first stage is accomplished. The splitting criteria used at this stage is mainly based on a reduction of the Mean Squared Error as the method is described above in part IV. Moreover, our investigation suggests that RF outperforms the other classification approaches. The results for the top six models on the first step are presented in Table 3 and the corresponding Receiver operating Characteristics Curves (ROC) are illustrated in Fig. 3.4. The ROC curves assist us to recognize the tradeoff between sensitivity and specificity. The slope of the tangent line at the threshold points represents the likelihood ratio for that value of the test, whereas the area under the curve is a measure of accuracy. Out of the 127,734 networks, there are 23 attacks that the model was not able to recognize and classified

Table 3: Accuracy Classification Process Step 1

Model	RF	J-48	PART	SVM	Logistic Regression	Naïve Bayes
Correctly classified	99.88%	99.86%	99.85%	97.42%	96.76%	90.63%
Kappa Statistic	0.998	0.9971	0.9970	0.948	0.9348	0.8111
Mean absolute Error	0.0029	0.0019	0.0016	0.0258	0.0433	0.0939
Root mean squared Error	0.0325	0.0372	0.0384	0.1606	0.1553	0.3024
Relative absolute error	0.59%	0.39%	0.33%	5.18%	8.70%	18.87%
Root relative squared error	6.52%	7.46%	7.69%	32.19%	31.12%	60.61%

them as normal and eight regular connections that the model classifies as an attack, but they are normal, in total 31 out of 127,734 instances were wrongly classified. We can observe that Random Forrest outperforms the other methods and the area under the ROC curve is almost 1, which signifies an excellent classification. The other two approaches, PART, and J-48 also give exceptional results, and we can employ them in the analysis. Support Vector Machines, with an area under the ROC curve of .9746 and Logistic regression, decreases the capacity of the classification process, although they exceed 95% accuracy. Only the Nave Bayes is around 90%, which categorizes it as the least desired approach from the top six, but also a reasonable one.

J-48 and PART provide worthy results and take a reasonable amount of time Fig. 3.4, However, the results suggest that it is better to select RF at this stage as a classification method because the accuracy is higher, the time to perform the classification is shorter, and there is less chance of overfitting in its algorithm. Real world data inevitably contain noise - in either the feature values, the class labels, or both. The models are compared with regards to time and complexity. The results in seconds are presented on Fig. 3.4 and Fig. 3.4 respectively. Although RF takes third place in the assessment of the time, it is still the preferred classifying technique that handles the trade-off between accuracy and complexity. The complexity is a measure that is related to the informational structure of the data set and how long does it take for the model in seconds to

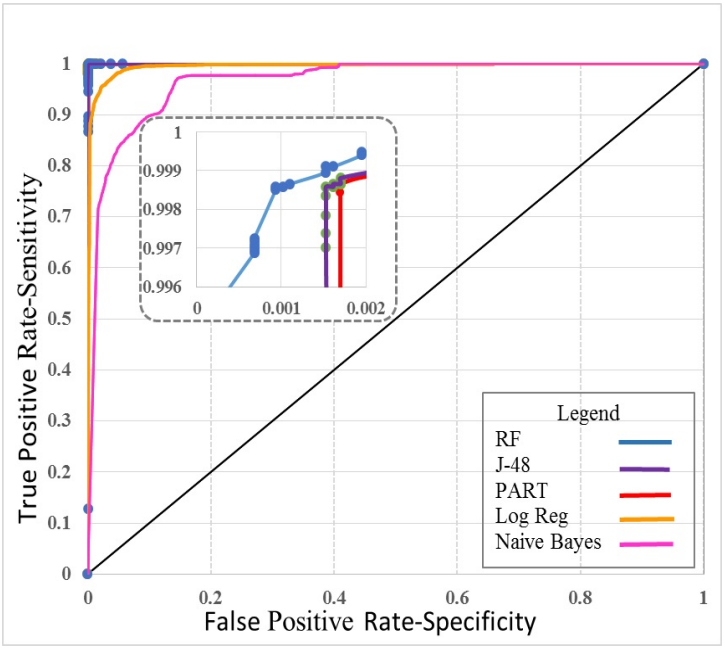


Figure 3.: ROC curve for classification ”normal”

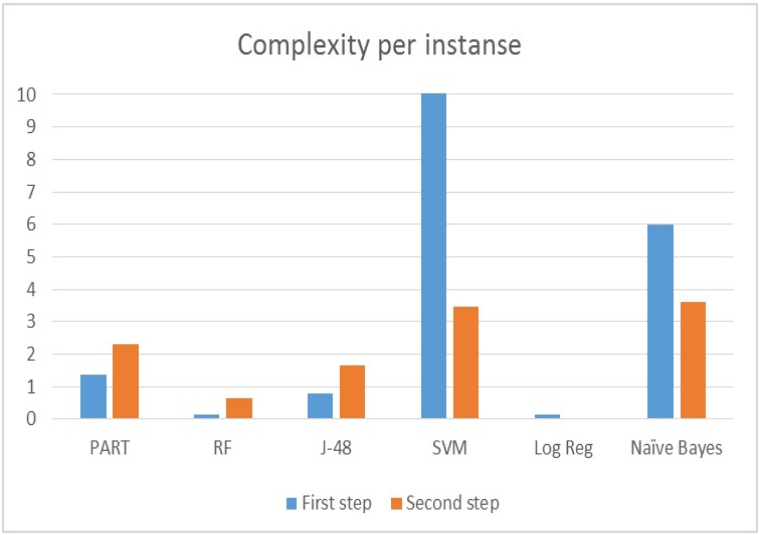


Figure 4.: Time in Seconds to Build a Model

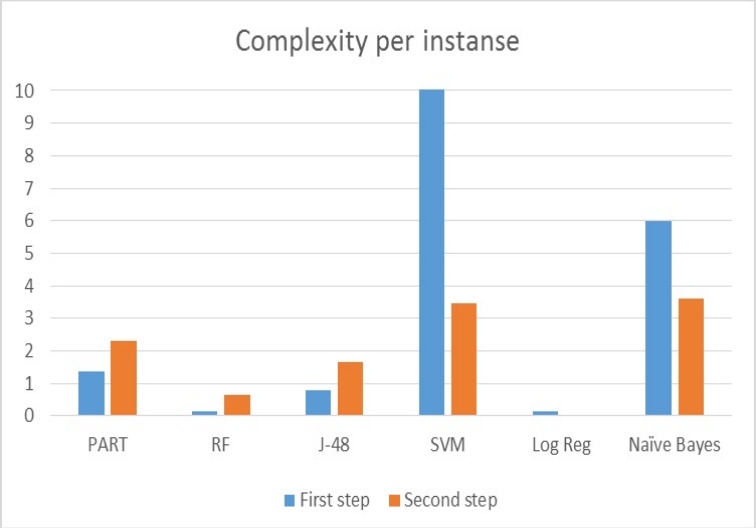


Figure 5.: Complexity per Instance

evaluate the information. The complexity is measured in bits per second and implies computational effort. We will apply the PART algorithm in the second stage of the classification process only if the network is under attack. PART model on this step achieves superior results in comparison to RF concerning the accuracy, and the model was able to perform better the classification process.

Unlike PART, which can accomplish the classification process with an incomplete tree, RF needs additional resources for creating and evaluating an entire tree. PART employs pruning in the evaluation process. Therefore, PART benefits the reduction of the complexity and decreases the error. Compared to the J-48, the risk of overfitting the data in the evaluation process is minor. The results for the second stage of the classification process are presented in Table 4. All of the methods slightly decrease their accuracy concerning the correctly classified type of attacks. However, once the normal networks are not taken into consideration, the SVM approach increases the number of the correctly classified instances. Regardless of the similar accuracy of the top six methods, we can observe that the Mean Absolute Error and especially the Relative Absolute Error are lesser for the PART model. Additionally, the accuracy of the PART model is higher and the time for the classification process of PART is reasonable.

Table 4: Accuracy Classification Process Step 2

Model	PART	RF	J-48	SVM	Log Reg	Naïve Bayes
Correctly classified	99.75%	99.72%	99.72%	99.59%	99.51%	83.17%
Kappa Statistic	0.995	0.994	0.994	0.9923	0.9906	0.70559
Mean absolute Error	0.0002	0.0006	0.0003	0.0826	0.0011	0.0151
Root mean squared Error	0.0149	0.0153	0.0153	0.1999	0.0199	0.1186
Relative absolute error	0.524%	1.421%	0.678%	183.6%	2.41%	33.48%
Root relative squared error	9.862%	10.142%	10.18%	132.7%	13.19%	78.76%

3.5 Concluding Remarks and Contribution

NASCA is a four-step Intrusion Detection procedure [82], which is created using machine learning techniques. Moreover, it is a superior method compared to the existing data mining models in network security. The time for performing the analysis is relatively short, and the accuracy is significant. The suggested classification process is capable of serving as a convenient and efficient protection tool for detecting an intrusion in our network. This chapter provides a comparison between existing supervised machine learning techniques that could be applied for feature selection and classification purposes in cybersecurity set up and the best ones concerning accuracy and error rate are selected. We propose step by step a feature selection and two-stage classification procedure - NASCA that will help the administrators to reduce the vulnerabilities in a specific system.

Chapter 4

Off-Policy Q-learning Technique for Intrusion Response [3]

In this chapter, we will propose an intrusion prevention mechanism that is based on machine learning, and more precisely, reinforcement learning techniques (RLT). The RLT will help us to create a decision agent, who will control the process of interacting with the undetermined environment. An objective is to obtain an optimal policy, which will represent the intrusion response to the attack and therefore to prevent it. It solves the Reinforcement learning problem, using a Q-learning approach. Our agent will produce an optimal immediate response, in the process of evaluating the network traffic. This Q-learning approach will establish the balance between exploration and exploitation and provide a unique, self-learning and strategic artificial intelligence response mechanism for Intrusion Detection and Intrusion Prevention Systems.

4.1 Network environment represented as Reinforcement Learning process

4.1.1 Reinforcement Learning

Let us assume that there is one decision maker in the IDS and he regularly interacts with his environment [83]. Based on the actions that he undertakes, he can modify his states and subsequently his performance is evaluated by feedback (reward). The aim is to select a set of actions which will optimize his long-term reward.

To understand how RL operates, we need to introduce the principle of Markov property and to familiarize ourselves with the concept of a Markov Decision Process (MDP).Let us define S as a countable set of states or the state space $S : \{S_1, \dots S_t\}$, where $S_t = s_t$ is a random variable with a range of $S_t \in (0, ..t]$, this set will be Markov if and only if:

$$P(S_{t+1} = s_{t+1}|S_t = s_t) = P(S_{t+1} = s_{t+1}|S_t = s_t..., S_0 = s_0)$$

[3] Portions of this chapter have been previously presented in 20th International Conference on Cyber Security of Cyber Physical Systems, and published in Special Journal Issue on Cyber Security of Cyber Physical Systems, 2018

Thus, the present state includes all past knowledge, and if the current state is observed, it will be considered as a sufficient statistic to decide for the future. The MDP is characterized by the tuple $\langle S, A, R, T, \gamma \rangle$:

- S is a countable set of states $S : \{S_0, S_1, \dots S_t\}$ in terms of the network set of states as $S : \{s_N, s_A\}$. Where s_A is the state of being under attack and s_N is the state when the network is normal. The amount of states is represented by the possible attacks that we are experiencing, or whether the network is normal or under attack;

- A is a set of actions, called the action space $A : \{A_1, A_2, \dots A_n\}$, where $A_n = a_n$ and $A_n \in (0, n]$ or in our case we have $A : \{a_p, a_{dn}\}$, where a_p is the action when the agent protects the network and a_{dn} is the action when the agent " do nothing" or doesn't protect the network.

- R defines the immediate reward that the agent can receive at each state, it is described as the reward for taking action A_n at state S_t , therefore $f : S \times A \to R$

$$R^a{}_s = \mathrm{E}[R_{t+1} | S_t = s_t, A_n = a_n]$$

- T is a state transition probability matrix. It specifies the probability of change from state i to state j, on taking action $A_n = a$, where $i \in (0, t]$ and $j \in (0, t+1]$

$$T^a{}_{ij} = P[S_{t+1} = s_j \,| S_t = s_i, A_n = a_n]$$

$T_{ij}{}^a = \begin{pmatrix} T^a{}_{11} & \cdots & T^a{}_{1t} \\ \vdots & \ddots & \vdots \\ T^a{}_{t1} & \cdots & T^a{}_{tt} \end{pmatrix}$, where the number of transition matrices will depend on the number of actions. Each transition matrix will represent the transition from state s_N to s_A for taking action a_p or a_{dn}. Therefore for our set up, there will be two transition matrices $T^{a_{dn}}_{S_N S_A}$ and $T^{a_p}_{S_N S_A}$.

- $\gamma \in [0, 1]$ is a discount factor [75], which assists us in determining the present value of a future expected immediate rewards. It is used for emphasizing the significance of the present in comparison to the future rewards. The larger the value of the discount factor, the farther the future rewards are considered over time.

Let us define the total return G_t that the decision agent (in our case the entity, protecting the computer network) will obtain as a function of the sum of all direct rewards at time t, discounted back to the present moment.

$$G_t = R^a_{t+1} + \gamma R^a_{t+2} + \gamma^2 R^a_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R^a_{t+k+1} \tag{4.1}$$

Mathematically it's convenient to use discounted reward decision process because it avoids the infinite returns in cyclic Markov processes and gives the opportunity for the decision agent to think about the long-term future.

The state-value function $v(s)$ of MDP will be determined as the cumulative measure of the predicted total return in (1) since the beginning of state s.

$$v^a(s) = E[G^a_t | S_t = s_t] \tag{4.2}$$

The value function may be divided into two components: direct reward and discounted reward of the following states $\gamma v^a(s_{t+1})$.

$$\begin{aligned}
v^a(s_t) &= E[G^a_t | S_t = s_t] \\
&= E[R^a_{t+1} + \gamma R^a_{t+2} + \gamma^2 R^a_{t+3} + \ldots | S_t = s_t] \\
&= E[R^a_{t+1} + (\gamma R^a_{t+2} + \gamma^2 R^a_{t+3} + \ldots) | S_t = s_t] \\
&= E[R^a_{t+1} + \gamma (R^a_{t+2} + \gamma R^a_{t+3} + \gamma^2 R^a_{t+4} \ldots) | S_t = s_t] \\
&= E[R^a_{t+1} + \gamma (\sum_{k=0}^{\infty} \gamma^k R^a_{t+k+2}) | S_t = s_t] \\
&= E[R^a_{t+1} + \gamma (G^a_{t+1}) | S_t = s_t] = \\
&= E[R^a_{t+1} | S_t = s_t] + \gamma E(G^a_{t+1} | S_{t+1} = s_t) | S_t = s_t] = \\
&= E[R^a_{t+1} | S_t = s_t] + \gamma [v^a(s_{t+1}) | S_t = s_t] = \\
&= E[R^a_{t+1} + \gamma [v^a(s_{t+1})] | S_t = s_t]
\end{aligned} \tag{4.3}$$

This way we can obtain the Bellman equation:

$$v^a(s) = E[R^a_{t+1} + \gamma [v^a(s_{t+1})] | S_t = s]$$

This equation can be represented as:

$$v^a(s_i) = R^a_{s_i} + \gamma \sum_{s_j \in S} T^a_{ij} v(s_j) \tag{4.4}$$

If we need to write the equation in matrix form, we will obtain:

$$\mathbf{v} = \mathbf{R} + \gamma \mathbf{T}\mathbf{v} \tag{4.5}$$

Here **v** will be a vector with a dimension equal to the number of states.

$$\begin{bmatrix} v^a(1) \\ \vdots \\ v^a(t) \end{bmatrix} = \begin{bmatrix} R_1^a \\ \vdots \\ R_t^a \end{bmatrix} + \gamma \begin{bmatrix} T_{11}^a & \cdots & T_{1t}^a \\ \vdots & \ddots & \vdots \\ T_{t1}^a & \cdots & T_{tt}^a \end{bmatrix} \begin{bmatrix} v^a(1) \\ \vdots \\ v^a(t) \end{bmatrix}$$

4.1.2 Policy and Policy Selection

Almost all reinforcement learning problems can be formalized as MDP. The agent maps the set of the states onto the probability space of taking each possible action. We can define this mapping process as a policy for the agent, which is a probability distribution formed out of possible actions, given the current states [75].

$$\pi(a|s) = P(A_t = a|S_t = s) \tag{4.6}$$

The policy describes the behavior of the agent, and it is like his model. MDP policy depends on the present state only, so it does not include information about the previous states, or it is time independent. Suppose we have an MDP $\langle S, A, T, R, \gamma \rangle$ and $\pi(a|s)$, the possible sequence of states is a Markov process $\langle S^\pi \rangle$, as well as the sequences of the states and the rewards $S_1, R_2, S_2 \ldots$ is a Markov Reward Process $\langle S, T^\pi, R^\pi, \gamma \rangle$. In the MDP besides the state-value, there exists an action-value function:

$$q^\pi(s, a) = E_\pi[G_t^a | S_t = s, A_t = a]$$

A policy is greedy with regards to a value function, as well as it is optimal according to that value function. The maximum of the function over all policies is attained where the optimal solution of the state-value function $v^*(s)$ exists.

$$v(s) = \max_\pi v(s)$$

It specifies the best possible performance in the MDP. The solution of the MDP is the optimal value function. The maximum of the action-value function over all policies is attained where the optimal solution of the action-value function $q^*(s, a)$ exists.

$$q(s,a) = \max_{\pi} q_{\pi}(s,a)$$

An optimal policy is the best policies over all policies and it is defined as follows: $\pi \geq \pi^{'}$if $v_{\pi}(s) \geq v_{\pi'}(s)$ for $\forall s$, and also:

$$\pi^*(a|s) = \begin{cases} 1 & , if \arg\max_{a \in A} q^*(s,a) \\ 0 & , otherwise \end{cases} \tag{4.7}$$

For every MDP there will exist an optimal deterministic policy and also if we find $q^*(s,a)$, there will be a corresponding optimal policy.

We can define an optimal Bellman Equation for the value function:

$$v_B^*(s_i) = \max_a R_{s_i}^a + \gamma \sum_{s_j \in S} T_{ij}^a v^*(s_j) \tag{4.8}$$

We can also define an optimal Bellman Equation for the action-value function:

$$q_B^*(s_i,a) = R_{s_i}^a + \gamma \sum_{s_j \in S} T_{ij}^a \max_{a_j} q(s_j,a_j) \tag{4.9}$$

We employ the Bellman Equation under the assumptions that its solution exists and it is unique. We assume appropriate conditions of Sutton and Barto [75] of existence and uniqueness.

In general, the following methods are usually used: Value iteration, Policy iteration, Q-learning, and Sarsa. In this work, we will use Q-learning technique with a finite set of states.The action-value function estimates the benefit of taking action a in state s. It is the best-expected sum of future rewards.

Reinforcement learning may be applied to identify an optimal action-selection policy [75] for a finite MDP. Furthermore, it learns an action-value function, which ultimately provides the anticipated value of following an optimal policy and taking specific action in a given state. A history of an agent is a sequence of <state, action, reward>. An optimal policy is achieved by favoring actions that at any state provide maximum value. This learning method is also capable of evaluating the expected value, calculated by all possible actions, without any environment model.

4.1.3 Exploration vs. Exploitation

On the one hand, the agent unavoidably should explore further opportunities and therefore deviate from the usual behavior. This divergence is called exploration or taking non-policy action. On the other hand, he should follow the procedures for estimating the value functions. Whenever he decides to obey, or follow the policy, we call the process exploitation or taking policy action. There is a trade-off between both terms, and it is challenging and necessary to find a suitable balance, so the agent to be allowed to decide appropriately.

The ε-greedy action selection provides a simple heuristic approach in justifying between exploitation and exploration. The concept is that the agent can take an arbitrary action a from a uniform distribution with probability ε, $0 \leq \varepsilon \leq 1$, and subsequently to select with probability $1 - \varepsilon$ the current best (greedy) action (Fig.2). It is a standard practice to decrease the value of epsilon over time as soon as the decision agent becomes confident and needs less exploration. Low rate implies a strong bias towards exploitation over exploration. The idea is to ensure continuous

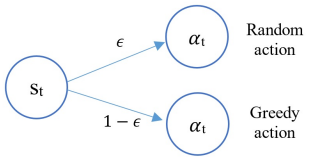


Figure 6.: ε-greedy action selection

exploration. All actions m are considered with non-zero probability.

$$\pi^*(a|s) = \begin{cases} \frac{\varepsilon}{m} + 1 - \varepsilon & , if \alpha^* = \arg\max_{a \in A} q(s,a) \\ \frac{\varepsilon}{m} & , otherwise \end{cases}$$

THEOREM 4.1 *For any $\epsilon - greedy$ policy, the policy π_j with regard to q_π is improvement, $v_{\pi_j}(s) \geq v_\pi(s)$.*

$$
\begin{aligned}
q_\pi(s, \pi_j(s)) &= \sum_{a \in A} \pi_j(a, s) q_\pi(s, a) \\
&= \frac{\epsilon}{m} \sum_{a \in A} q_\pi(s, a) + (1 - \epsilon) \max_{a \in A} q_\pi(s, a) \\
&\geq \frac{\epsilon}{m} \sum_{a \in A} q_\pi(s, a) + (1 - \epsilon) \sum_{a \in A} \frac{\pi(a|s) - \frac{\epsilon}{m}}{1 - \epsilon} q_\pi(s, a) \\
&= \sum_{a \in A} \pi_j(a|s) q_\pi(s, a) = v_\pi(s)
\end{aligned}
$$

Therefore $v_{\pi_j}(s) \geq v_\pi(s)$.

4.1.4 Q-learning Algorithm

The Q-learning has many advantages, comparing to the other methods for solving the MDP. First and most important is the idea for a model-free environment and the concept for an off-policy learning procedure. Moreover, the decision agent contemplates his succeeding move, based on the anticipated benefit of selecting each action at any particular state. Subsequently, he updates towards a bootstrap estimate of the actual return. At every stage, the following state is observed, and the maximum possible rewards, available for all actions in that state are determined. Consequently, using this information, the decision agent updates the action-value function with the related action in the current state. There is a so-called learning rate, denoted by $\alpha, (0 < \alpha \leq 1)$, which is associated with that change will assist us to formulate an updating rule.

Let us consider initial action-values $q(s, a)$ and every following action is selected based on a behavior policy $a_b \sim \mu(\cdot|s_i)$, there is also an successor action $a_j \sim \pi(\cdot|s_i)$. Then the updated $q(s, a)$ will be given by:

$$
q(s_j, a_i) \leftarrow q(s_i, a_i) + \alpha[R_{s_j}^{a_j} + \gamma q(s_j, a_j) - q(s_i, a_i)]
$$

If we allow the behavior and the target policy to evolve, then we say that the target policy π is greedy and $\pi(s_j) = argmax_{a_j} q(s_j, a_j)$. Also the behavioral policy μ is ε-greedy with respect to $q(s, a)$. Therefore the Q-learning control equation is:

$$
q(s_j, a) \leftarrow q(s_i, a_i) + \alpha[R_{s_j}^{a} + \gamma \max_{a_j} q(s_j, a_j) - q(s_i, a_i)] \tag{4.10}
$$

The Q-learning control equation eventually will converge to the optimal action-value function $q(s,a) \rightarrow q^*(s,a)$. The proof for this convergence is provided by Watkins and Dayan [72] and additionally by Tsitsiklis [69]. An interesting problem is also the convergence properties. Melo et al. [63] proved convergence, assuming some constraints on the sample distribution. Maei et al. [64] introduced a greedy gradient Q-learning approach that removes the previous conditions and proved convergence regardless of the sampling distribution.

The described algorithm can be summarized in the subsequent lines:

Algorithm 1 Q-learning approach

- Initialize $q(s,a)$, for each $s \in S$, $a \in A(s)$, randomly and $q(terminal - state) = 0$

  Repeat **for** each episode:

  - Initialize $s_i \in S$

    Repeat for each episode:

    * Chose a from s, using ε-greedy policy derived from Q.
    * Take action a_i
      · observe $R_{s_i}^{a_i}$
      · observe the new state s_j
    * $q(s_i,a_i) \leftarrow q(s_i,a_i) + \alpha[R_{s_j}^{a_j} + \gamma \max_{a_j} q_\pi(s_j,a_j) - q(s_i,a_i)]$
    * move to next state $s_i \leftarrow s_j$

  - until s is terminal

- end **for**

It is interesting to mention that there is a connection between the learning rate that we select α and the convergence rate. Even and Mansour [62] proved that for a polynomial learning rate of the type $1/t^\omega$ at time t, the convergence rate is polynomial in $1/(1-\gamma)$, where γ is the discount factor. However, for a linear learning rate of the type $1/t$ at time t, the convergence rate has an exponential dependence on $1/(1-\gamma)$.

4.2 Results

4.2.1 Problem Setup

To start the analysis, we first need to consider how we will set up the problem, so we can define it as a reinforcement learning problem and then attempt solving it, using Q-learning approach. The environment of the agent is completely unknown and non-stationary; therefore it is useful to use a model-free procedure. The Q-learning approach will allow us to calculate the Q-values, without estimating the transition probability, just by setting a reward matrix, based on the actions and states set up of MDP. The main purpose of our work is to find an optimal policy for the administrator at any given step of his decision-making problem. As we have mentioned above, the MDP is characterized by the following components $\langle S, A, R, T, \gamma \rangle$. We already provided information about the possible states and actions. The immediate reward that the agent will receive at each state is defined by $R : S \times A \to R$, or this is what he will obtain for taking action a in state s. In our case, we will outline the reward based on the initial behavior policy:

$$R : \begin{array}{c} \begin{matrix} s_N & s_A \end{matrix} \\ \begin{bmatrix} 0 & 2 \\ 1 & -1 \end{bmatrix}^T \end{array} \begin{array}{l} a_p \\ a_{dn} \end{array}$$

If the network is under attack s_A, then the agent has two options of actions to select from: either to "protect" a_p or to "do nothing" a_{dn}. On the one hand, if the current state is for example "attack" and the agent decides to "protect", then the reward that he will be rewarded with is 2, however, if he selects to "do nothing", then he will be penalized with a value of -1. On the other hand, if the state is "normal" and the agent decides to "protect," he will receive 0 reward and if he selects to "do nothing," he will get 1. This matrix is selected in a way that the agent to be interested in protecting the network only if there is an attack occurring. There are transition probability functions associated with the change from one state to another. For the Q-learning technique, we do not need to possess knowledge on them; however, we can provide an estimate, using the data set, so that we can test our results in subsection D. An approach that we will apply in this paper is a bootstrap estimation. Bootstrap samples will be obtained from the dataset with a conditional estimated distribution, and then a maximum likelihood estimation is performed. Then we will

take the average of the estimates across all samples, row normalized. We use 500 samples for the purpose of the estimation. MLE for MDP is described in details in [68]. We can also calculate the 95% confidence intervals and to report an error rate, based on the dataset. The formula used in the calculations is the following:

$$T_{S_N S_A}^{a_{dn}}{}^{MLE} = \frac{n_{S_N S_A}}{\sum_{u=1}^{k} n_{S_{Nu}}} \text{ with } SE_{S_N S_A} = \frac{T_{S_N S_A}^{a_{dn}}}{\sqrt{n_{S_N S_A}}}$$

Another second method is using Laplace smoothing approach, which is very similar to the MLE, but uses an arbitrary positive stabilizing parameter ϵ:

$$T_{S_N S_A}^{a_{dn}} = \frac{n_{S_N S_A} + \epsilon}{\sum_{u=1}^{k} (n_{S_{Nu}} + \epsilon)}$$

Both methods give similar results for the transition probability matrices:

$$T_{S_N S_A}^{a_p} : \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{matrix} s_N \\ s_A \end{matrix}; T_{S_N S_A}^{a_{dn}} : \begin{bmatrix} .53 & .47 \\ 0 & 1 \end{bmatrix} \begin{matrix} s_N \\ s_A \end{matrix}$$

$$SE_{S_N S_A} : \begin{bmatrix} .0026 & .0028 \\ 0 & 0 \end{bmatrix} \begin{matrix} s_N \\ s_A \end{matrix}$$

The discount factor $\gamma \in [0, 1]$ will depend on whether we would like to create our agent narrow-minded, who is more concerned about the present, or we would like to create him more strategic oriented, who will first consider the future and then he will make decisions about the present. In our analysis, we will set $\gamma = .9$, but we will provide a sensitivity analysis for three different levels of the $\gamma = .1, \gamma = .5$ and $\gamma = .9$, Fig.4. We can represent the decision path that the agent will follow in Fig.3.

Before starting the analysis, we need to check whether the MDP will hold for this specific problem and we will do that in R. The goal is to find an optimal policy if we start with the initial behavior policy $\pi : S \rightarrow A$, or that is $\pi(a|s)$: if there is an attack s_A, the agent will select to protect the network a_p. We do not possess any information about the environment. The only thing that we need is to set the reward matrix in a way that the agent is more likely to choose to protect the network if there is an attack occurring, but not necessarily, only if by deciding to defend the network, the Q-value function is maximized.

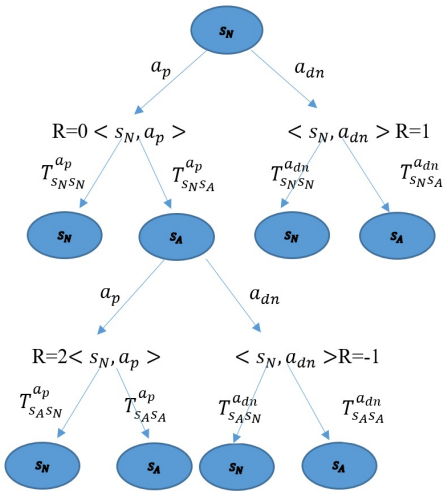


Figure 7.: Example for a Decision Path of the Agent

4.2.2 Calculation of the Q-value and Optimal Policy Selection, using Q-learning

As we have mentioned before, the action-value function represents the anticipated benefit of performing a specific action in a particular state and adopting an optimal policy after that. Q is a matrix that can be denoted with $[S, A]$, in our case, it is calculated with 100,000 number of iterations. We will use the provided algorithm with a decaying learning rate of $\alpha = 1/\sqrt{n+2}$, where n is the number of transitions.

$$\begin{matrix} & s_N & s_A \end{matrix}$$

$$Q^{a*}_s : \begin{bmatrix} 15.35415 & 17.08029 \\ 16.34766 & 13.71229 \end{bmatrix} \begin{matrix} s_N \\ s_A \end{matrix}$$

The Value function is an S length vector, with the same number of iterations, we obtain:

$$V^*_s : \begin{bmatrix} 17.08029 \\ 16.34766 \end{bmatrix} \begin{matrix} s_N \\ s_A \end{matrix}$$

The policy is also an S length vector. Each element of it is a value that corresponds to an action which maximizes the value function. The agent follows a specific policy π when selecting actions in a given state as we defined in (3). Once the action-value function is determined, the optimal policy can be recreated by choosing the action with the most substantial value in each state. We

obtained the following result:

$$(\pi_s^a)^* : \begin{bmatrix} 2 \\ \\ 1 \end{bmatrix}$$

That implies that the policy, in this case, will be the following: if there is a state 2 or if there is an "attack," the agent should choose action 1, which we assigned earlier as "to protect." The agent observes the current state, selects an action randomly, notes the resulting reward, then the new state occurs.

The sensitivity analysis of the different levels of γ, as well as the levels of iterations of the model, can be observed in Fig.4.

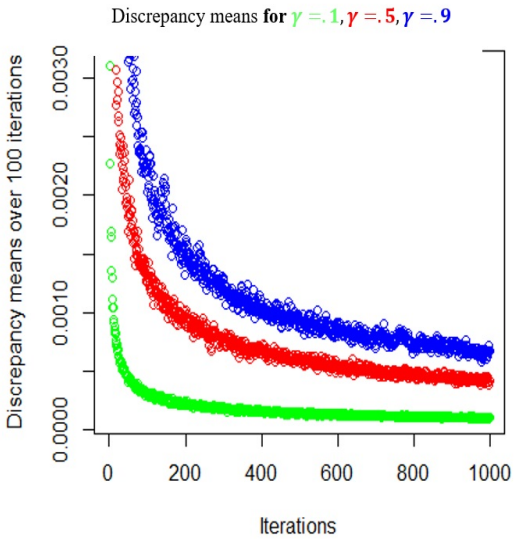


Figure 8.: Discrepancy Means

4.2.3 Evaluation of the Model

We can use the estimated transition probabilities, to determine how effective is our Q-learning agent performing. For that purpose, we need to find a solution to Equation **??** and to solve the MDP with the knowledge that we possess for the environment. In that situation, we will be able to evaluate our model, by comparing the value $V(s)$ that we calculated with the Q-learning approach and the value that we will obtain by solving the MDP, knowing the transition probabilities in **??**.

The aim is to create a Root Mean Square Error (RMSE), so we can test the effectiveness of the Q-learning approach. There are several methods to solve equation (5), that is Linear Programming (LP), Policy Iteration (PI) and Value Iteration (VI). All these solutions require knowledge of the environment, represented by the estimated transition probabilities. As shown in Table 1, we can observe the RMSE errors, using the results for $V(s)$ from Q-learning.The RMSE is calculated as:

$$RMSE = \frac{1}{N}\sqrt{\sum_{s}\left[V(s) - V^{*}(s)\right]^{2}}$$

where:

$$V_s : \begin{bmatrix} 17.08029 \\ 16.34766 \end{bmatrix} \begin{matrix} s_N \\ s_A \end{matrix}$$

Table 5: RMSE for Q-learning

$Method$	$V^{*}(s) for s_N$	$V^{*}(s) for s_A$	$RMSE$
LP	17.02741	16.32467	0.02883
PI	17.02741	16.32467	0.02883
VI	16.89542	16.19268	0.12062

All models calculate the optimal policy as:

$$(\pi_s^a)^* : \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Which is that if there is an ”attack” occurring, the decision agent should protect the network.

4.3 Concluding Remarks and Contribution

Q-learning, as a model-free control approach, is remarkably promising, especially when employed in challenging decision processes, where the traditional optimization techniques and supervised learning methods are not applicable. An essential advantage is the fact that the decision agent does not need any information about the environment and it can perform the analysis without any model or knowledge of the distribution. Q-learning has an auspicious future in the intrusion detection- and

prevention domains, it is a useful tool that needs to be further developed and explored. Despite the research work that has been published on the convergence properties, there are still few challenges that need to be analyzed. The beneficial utilization of Q-learning is helpful not only because of the useful results obtained with the model but also because of its potential combination with other models, that could improve with the assistance of Q-learning. For example in the last chapter, we provide a Game theory approach that is linked successfully to Q-learning. It provides a complete automated response mechanism that could be implemented in the Intrusion Detection and Intrusion Prevention systems.

Chapter 5

Streaming data adaptation of NASCA

In this section, we will suggest a streaming data learning approach of applying the NASCA procedure. In general, the majority of the research is dedicated to the analysis of a machine learning methods, which consider that random samples from a stationary distribution are derived. In fact, that is not always the case and nearly all of the time the distribution changes. Here, we will adopt the idea that the distribution is possible to vary and will provide an adaptation of the classification process. We will introduce a change detecting model so that we can modify the procedures in the previous chapters and adapt it to streaming data. We will aim to build a drift detection method, which will serve as an alarm for a distribution change and will re-sample from the stream of the data. Data streams are obtained at a high rate, and they flow continuously without any interruption. Therefore, we can not save all of them in the memory; moreover, we can not lose substantial time analyzing the data. Every time we download a limited training set which is presented as a sequence. The plan is to manage the value of the accuracy rate of the model. If the distribution of the underlying data is unchanged, then the classification accuracy rate will remain unchanged or will increase, however, if the distribution shifts, the accuracy rate will decrease. We will define a warning level, which serves as a signal that in case that the data set follows the same pattern of change, shortly the distribution will shift. Additionally, we will provide a drift level, which alarms that the distribution has already shifted and the entire classification procedure has to be repeated in the new set of data with the latest distribution. Most of the analysis in this chapter will be dedicated on the drift detection as a tool for detecting distribution shift and outlining the window's size. We will start the analysis based on the NASCA procedure, so we will first describe the Network Data Collection, followed by the Attribute Selection, Classification, and Accuracy. Finally, we will discuss the proposed Drift detection approach so that we can establish an updated NASCA procedure for streaming data: NASCA Drift.

5.1 Network Data Collection

In data streaming, it is frequently used in the so-called interleaved test-then-train evaluation. The central concept is that it first starts testing each instance, and then it trains the model. The evaluation work is to assess the classifier, based on several criteria. We can apply three approaches to determine the chosen classification technique. The primary method estimates the accuracy of the classifier since the beginning of the evaluation. The second method is called a window based procedure, and it measures the accuracy on the current sliding window of recent instances, and the third method is the fading factors approach, which weight observations accuracy, using a decay factor α. The advantage of using this type of evaluation is the fact that we do not need to employ any testing set in the classification process. Every individual case is utilized to test the model before it is trained; therefore, the accuracy is incrementally updated. When performed this method, we should keep in mind that the model is always applied to instances that have not been examined. It also guarantees steady progress of accuracy over time, as each case will become less critical to the total average [95]. There is an accuracy and an error rate that can be calculated, where the error rate represents the aggregated sum of a loss function between predicted and observed values: $S_n = \sum_{t=1}^{n} Loss(\hat{y}_t, y_t)$. The benefit of the test-then-train assessment structure is that it can be applied in the circumstances with limited feedback. Furthermore, it is not necessary for us to possess knowledge about the actual measure of y_i, for every object in the data stream. Instead, we can compute the loss function for those instances only, where y_i is observed, where i is the $i - th$ observation. The test-then-train evaluation examines the progress of the learning process by producing a learning function. The estimated total error has the risk of being influenced by the beginning portion of the error sequence. The classifier is evaluated by first testing and after that training, engaging every instance. The principal concept is to calculate an error, using a forgetting mechanism or fading factor, which can implement a decaying weight for the windows from the past [87]. Let us define n_i as the number of examples used to compute a loss function L_i and a loss function is computed for each example.

The fading factor is the following:

$$E_i = \frac{S_i}{B_i}$$

where $S_1 = L_1$ and:

$$S_i = Li + \alpha \times S_{i-1}$$
$$B_i = n_i + \alpha \times B_{i-1}$$

S_1 will be a constant defining the forgetting portion of the sum that in general is close to 1. Also B_i is a fading increment and $B_1 = 1$. The fading increment [87] has an essential characteristic that:

$$\lim_{i \to +\infty} B_i = \frac{1}{1-\alpha}$$

5.2 Attribute selection

The feature selection with Information Gain Criterion is an interesting process in an online analysis. As we have mentioned before, there may be a change in the distribution from time to time, and sometimes it happens suddenly depending on the data stream. As an addition, a significant characteristic is the fact that we are not allowed to store many observations at a time, and there may be cases when we never see the same example again. The authors in [103] propose an Information Gain adaptation for streaming data. They presented a straightforward algorithm that can be adapted to discrete and continuous domains. Also, they calculated a probable error affiliated with their method and potential theoretical limits, where the user is provided with the opportunity to select between error and performance. However, the information that the procedure demands is a reasonable estimate or an upper boundary of the size of the whole stream. The researchers also simulated sudden fluctuations in the underlying distribution so that they can examine their method. They also compared their results with other authors interested in this field.

The authors observe the maximum memory utilized in the feature selection process. Let us assume that there are f features denoted by $\mathbf{X} = X_1, X_2, ..., X_f$ and l labels defined by the set $\mathbf{Y} = Y_1, Y_2, ..., Y_l$. Then we can denote a set of pairs $S = [(x_1, y_1), (x_2, y_2), ..., (x_t, y_t), ...]$ to be a stream of pairs, in which x_t is a feature vector and y_t is a label. Then we will define S_n as a set of n instances seen till time $t = n$. Let us define S_n^L as a subset of instances in S_n and the rest of the examples are represented by the complement of S_n^L or S_n^R. As an addition let us define $S_{n,y}$ as the instances in S_n with a label y. Then we can calculate the information entropy as:$y \in \mathbf{Y}$

$$Inf(S_n) = -\sum_{y \in \mathbf{Y}} \frac{|S_{n,y}|}{|S_n|} \log \frac{|S_{n,y}|}{|S_n|} \tag{5.1}$$

Then the information gain as before is the difference in the information structures for each specific feature, but this time adapted to streaming setup of the data:

$$IG(S_n, X_i) = I(S_n) - [\frac{|S_n^L|}{|S_n|} I(S_n^L) + \frac{|S_n^R|}{|S_n|} I(S_n^R)] \tag{5.2}$$

5.3 Classification and Accuracy for Streaming Data

Using test-then-train evaluation, we can update the NASCA procedure to establish a process that will be applied in a streaming data setup. It will be challenging to use the Random forest as a classification technique, which is the third step of the NASCA algorithm; therefore we will have to make some adjustments and consider employing Adaptive Random Forest (ARF) [97]. The method uses bagging, to avoid overfitting as well as it chooses at random a set of features at every node split. The complete process involves the selection of a variety of passes of the data to train the model. Having that in mind we need to make some adaptations that will allow the process of classification to occur.

Several factors influence the ARF. First, we need to apply an online criterion for bagging. This method requires the creation of bootstrap samples with replacement, where each one of them includes every instance R times and the $P(R = r)$ follows a binomial distribution, that can be approximated by Poisson under specific circumstances. Based on that, Oza et al. [98] introduced a procedure called online bagging, which employs the idea for random sampling with replacement by giving weights to the instances, following a $Poisson(\lambda = 1)$ distribution. The authors in [97] applied $Poisson(\lambda = 6)$ so they can raise the probability of allotting larger weights to some instances, while training is in process. Second, we need to create a process that narrows the feature subspace per leaf split. To do so, we have to reconsider the primary procedure of tree growing and to limit the set of variables, reviewed for the split process. Hoeffding Tree [99] is a very appropriate approach that we can employ to consider the feature selection process. The Hoeffding Tree is an incremental decision tree learner with a reliable theoretical execution. It uses Hoeffding bound [100] which states that there is a probability of $1 - \nu$ that the true mean of a random variable

with a range R will not deviate from the estimated mean in n independent instances by more than:

$$\epsilon = \sqrt{\frac{(\ln \frac{1}{\nu})R^2}{2n}} \tag{5.3}$$

Let us say that we have a training set D with elements (x_i, y_i) and we denote a classifier $h : X \to Y$, we can specify the sample error of h to be: $E_{in}(h) = \frac{1}{n} \sum_{i=1}^{n} 1_{h(x_i) \neq y_i}$ The inequality is : $P(|E_{in} - E_{out}| \geq \epsilon) \leq 2e^{-2n\epsilon^2}$ An appealing quality of the concept of the Hoeffding Trees is that it possesses a solid theoretical performance [99]. An ARF uses the idea of Hoeffding trees with the difference that it does not involve initial pruning of the tree. Once a node is generated, a subset of m random features are elected. The splits are based on those features only. In general, the depth of the tree also plays an important role. The variance reduces from averaging various trees and even if there is a risk of overfitting each tree, this risk of the random forest to overfit is minimized by averaging across trees. The advantage of ARF, in short, is that it can incorporate warning and drift levels and can handle different data streams. Once a warning level is identified, the ARF initiates a set of background trees, and if there is a drift level detected, the algorithm replaces the current trees with the set of background trees and starts over the training process. The ARF can incorporate a variety of drift and warning detection models.

Besides the warning and the drift mechanism, we can also select different voting systems, for instance, the votes in the ARF are weighted based on the accuracy of the tree classifiers, using test-then-train approach. If we assume that a tree l has seen n_l examples since its latest set up and accurately labeled c_l examples, in a way that $c_l \leq n_l$, then its weight will be $\frac{c_l}{n_l}$. An ultimate benefit of utilizing a weighting technique is the idea that it does not need a determined in advance window.

5.4 Drift Detection

5.4.1 Theoretical Framework of Bayesian Inference for Drift Detection

The normal approximation for Binomial distribution, used by most of the researchers, is not necessarily accurate. Theoretically, based on the Central Limit Theorem, we can apply this approximation, but there is no theoretical proof of how large should be the sample size to implement it. In this chapter we will present another method, using (quasi) Bayesian Inference for establishing

a drift detection level. The benefit of using this approach over the frequentist analysis is that we can obtain a posterior distribution that includes information about the parameter. The classical methods give a point estimate and employ the asymptotic assumption for normality. We are not required to estimate uncertainty with the Bayesian structure because we establish a complete posterior distribution. As an addition, the Bayesian idea can provide us with Credible Intervals, which are easier to be understood in our streaming data context, than the classical statistics. Let us consider an independent and identically distributed instances. The estimated density, determined by the classical approaches will be the value $\hat{\theta} = \hat{\theta}(x_1, \ldots, x_n)$ which is an estimate of the parameter θ, $\theta \in \Theta$ (parameter space) into the conditional density $f_{X_{n+1}|\Theta}(x_{n+1} \mid \theta)$. The classical predictive density $f_{X_{n+1}|\Theta}(x_{n+1} \mid \hat{\theta})$ does not take into consideration the uncertainty of the estimate $\hat{\theta}$: we can obtain two equal point estimates with different confidence intervals that may provide the same predictive density. However, the Bayesian predictive distribution considers the uncertainty of the parameter, provided the information in the sample of observations.

$$f_{X_{n+1}|X_1,\ldots,X_m}(x_{n+1} \mid x_1, \ldots, x_n) = \int f_{X_{n+1}|\Theta}(x_{n+1} \mid \theta)\, \pi(\theta \mid x_1, \ldots, x_n)\, d\theta$$

The Bayesian Inference approach will further help us to capture the information for the parameters of the prior distribution in the current model. Suppose we have the following problem observations $\{X_1, X_2, ..., X_n\}$, that are independent and identically distributed. X_i represents two outcomes, our model correctly classified the instance, or it made a mistake. If we decide to apply the prequential evaluation, then the overall number of successes for each set of instances will be defined by X, and the probability of successfully identified the correct instance is θ, where $X \sim Bin(n, \theta)$.

$$p(x|\theta) = \binom{n}{x} \theta^x (1-\theta)^{n-x} \tag{5.4}$$

The underlying assumption for using Binomial distribution is that the number of observations n does not change and the instances are independent of each other. The probability of the algorithm to successfully classify the network is constant for every outcome, included in these n observations. More realistic modeling with varying parameters will be a topic of future work. We will assign a prior distribution of the parameter θ so that we can include previous beliefs for the information structure of the instances in the past. Also, we will assume a Jeffrey's uninformative prior $p(\theta)$ and

the likelihood function of $p(x|\theta)$ to obtain the posterior distribution. We could also find a conjugate prior distribution for the analysis; however, we prefer the selection process of the prior to be with the smallest amount of assumptions about the model and to have minimal effect on the posterior distribution.

We know by definition of Jeffrey's interval that:

$$p_J(\theta) = \sqrt{I(\theta)} \tag{5.5}$$

Where $I(\theta)$ is the Fisher information Criterion:

$$I(\theta) = -E(\frac{\partial^2 l}{\partial \theta^2}|\theta) \tag{5.6}$$

Therefore we need to define the likelihood function as $\mathcal{L}(\theta) \propto \theta^x(1-\theta)^{n-x}$.

$$l(\theta) = log\mathcal{L}(\theta) \propto x\log(\theta) + (n-x)\log(1-\theta) \tag{5.7}$$

Now let us obtain the first and the second partial derivative of the likelihood function with respect to θ:

$$\begin{aligned} \frac{\partial l}{\partial \theta} &\propto \frac{x}{\theta} - \frac{n-x}{1-\theta} \\ \frac{\partial^2 l}{\partial \theta^2} &\propto -\frac{x}{\theta^2} - \frac{n-x}{(1-\theta)^2} \end{aligned}$$

Therefore the Information Criterion in our case for Binomial Distribution is:

$$\begin{aligned} I(\theta) &\propto -E(\frac{\partial^2 l}{\partial \theta^2}|\theta) \\ &\propto \frac{n\theta}{\theta^2} + \frac{n-n\theta}{(1-\theta)^2} \\ &\propto \frac{n}{\theta(1-\theta)} \\ &\propto \theta^{-1}(1-\theta)^{-1} \end{aligned}$$

Easy we can substitute from 5.5:

$$p_J(\theta) = \sqrt{I(\theta)} = \theta^{-1/2}(1-\theta)^{-1/2} \tag{5.8}$$

This is a prior Beta distribution $Beta(\alpha, \beta)$ with parameters $Beta(1/2; 1/2)$.

Now that we know the prior distribution , we can easily obtain the posterior, using the Bayes theorem $P(\theta|x) \propto l(\theta|x)P(\theta)$ and 5.7 and 5.8:

$$\begin{aligned} P(\theta|x) &\propto \{\theta^x(1-\theta)^{n-x}\}\{\theta^{-1/2}(1-\theta)^{-1/2}\} \\ &\propto \{\theta^x\theta^{-1/2}\}\{(1-\theta)^{n-x}(1-\theta)^{-1/2}\} \\ &\propto \theta^{x-1/2}(1-\theta)^{n-x-1/2} \end{aligned}$$

A random variable on (0,1) interval follows a Beta distribution if the density is:

$$Beta(\alpha,\beta): \ p(x|\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \tag{5.9}$$

where B is the beta function

$$B(\alpha,\beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt$$

In order to obtain the same form as in Beta distribution,so we can find the parameters of α and β we need to represent our posterior density as:

$$P(\theta|x) \propto \theta^{(x+1/2)-1}(1-\theta)^{(n-x+1/2)-1} \tag{5.10}$$

Therefore $\alpha = (x+1/2)$ and $\beta = (n-x+1/2)$ and

$$\theta|x \sim Beta((x+1/2),(n-x+1/2))$$

That is:

$$Beta(x{+}1/2, n{-}x{+}1/2): \ p(x|(x{+}1/2),(n{-}x{+}1/2)) = \frac{x^{(x+1/2)-1}(1-x)^{(n-x+1/2)-1}}{B(x+1/2,n-x+1/2)} \tag{5.11}$$

and

$$B((x+1/2),(n-x+1/2)) = \int_0^1 t^{x+1/2-1}(1-t)^{n-x+1/2-1}dt$$

The Beta function is equal to a ratio of Gamma functions:

$$B((x+1/2),(n-x+1/2)) = \frac{\Gamma(x+1/2)\Gamma(n-x+1/2)}{\Gamma((x+1/2+n-x+1/2))} = \frac{\Gamma(x+1/2)\Gamma(n-x+1/2)}{\Gamma(n+1)}$$

We can also make inference for the variance and the mean of the posterior distribution. It makes a

summary of our belief after we analyze our data. We need to find measure of location and spread, so we can identify how and how wide is the probability. The posterior mean will serve as a location measure, and it is just the expected value of the posterior distribution: We know that the mean of a Beta distribution is :

$$\mu = \int_0^1 xp(\theta|x)d\theta \tag{5.12}$$

That means that

$$\begin{aligned}\mu = E[X] &= \frac{\int_0^1 x^{\alpha}(1-x)^{\beta-1}\,dx}{B(\alpha,\beta)} \\ &= \frac{B(\alpha+1,\beta)}{B(\alpha,\beta)} \\ &= \frac{\Gamma(\alpha+1)\Gamma(\beta)}{\Gamma(\alpha+\beta+1)}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \\ &= \frac{\alpha}{\alpha+\beta}\end{aligned}$$

Therefore we can compute the Bayes posterior point estimate as:

$$\hat{\theta} = \frac{x+1/2}{n+1} \tag{5.13}$$

Another important statistic will be the posterior spread or the variance of the posterior distribution. We can make inferences about the distribution based on it. The variance is:

$$\sigma^2 = \int_0^1 (\theta-\mu)^2 p(\theta|x)d\theta \tag{5.14}$$

We also know that the standard deviation ca be obtained by:

$$\begin{aligned}\sigma^2 + \mu^2 = E[X^2] &= \frac{B(\alpha+2,\beta)}{B(\alpha,\beta)} \\ &= \frac{\alpha(\alpha+1)}{(\alpha+\beta)(\alpha+\beta+1)}\end{aligned}$$

S, using the identity: $\Gamma(t+1) = t\Gamma(t)$, the standard deviation is:

$$\begin{aligned}\sigma^2 &= \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \\ &= \frac{(x+1/2)(n-x+1/2)}{(x+1/2+n-x+1/2)^2((x+1/2+n-x+1/2)+1)} \\ &= \frac{(x+1/2)(n-x+1/2)}{(n+1)^2((n+2)} \\ &= \frac{(2x+1)(2n-2x+1)}{4(n+1)^2((n+2)}.\end{aligned}$$

The posterior $1-\alpha$ Bayes credible interval can be found if we can find values of a and b that satisfy the following interval:

$$\int_a^b p(\theta|x)d\theta = 1-\alpha \tag{5.15}$$

Here $1-\alpha$ is the confidence level. We can also find the upper or the lower bound as:

$$\int_{-\infty}^a p(\theta|x)d\theta = \int_b^{\infty} p(\theta|x)d\theta = \alpha/2 \tag{5.16}$$

5.4.2 Warning and Drift Levels

Now we will propose a method for selecting a window size for streaming data analysis. In the beginning, we need to define a window W as a sequence of instances; it will be represented by the current data set that is analyzed and classified. For the evaluation, we need to define an algorithm that will help to set a beginning and end of one window. Our target is to analyze the posterior distribution of the accuracy rate.

The posterior distribution of the accuracy will be approximated, using sequential Markov Chain Monte Carlo methods (SMCMC) [102]. SMCMC will allow us to adapt the sampling process for streaming data and online to generate samples. That type of sampling generates a (large) random sample from $p(\theta|y)$. Using that type of sampling we can generate random numbers from the distribution. SMCMC will allow us to create a sample from joint posterior distribution: $p(\mu_{current}$, $\mu_{accumulated}, \sigma_{current}, \sigma_{accumulated})$, it will also provide us with solutions for a multidimensional integration. The values in are generated in chains and many combinations are made until the sampling includes a representation of the parameters the data that we have and our selected prior distribution. We can easily estimate the parameters applying this approach and also all credible intervals that we are interested in. The SMCMC creates a random sample every time. Therefore the repeated analyses will be somewhat diverse at each repetition. The small sampling differences will rarely hinder the conclusion of the analysis, and in case of concern, a larger chain will solve the problem.

After we have generated and sampled from the posterior distribution, we can make any inferences that we would like to for those samples. Let us say, that we analyze a specific amount of instances, for example, hundred at a time. The analysis starts from instance l_1 to l_{100}, where l denotes the label of whether the network is under attack or not or the type of the attack. There is an

accuracy rate associated with the first hundred instances that have been examined. We can denote this accuracy rate as a_1, where a_1 will represent the sum of all accurately labeled instances out of the first hundred Fig. 5.4.2. The following step is to analyze another hundred instances, the next associated accuracy rate with the second hundred instances will be a_2 and so on until we reach for example a_i, where i will represent the accuracy of the i^{-th} set of hundred instances. The number of observations that we analyze at a time does not matter for the analysis; they could be ten, hundred, thousand, etc.

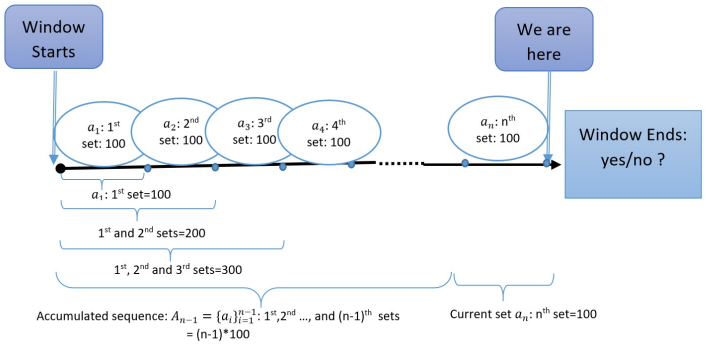


Figure 9.: Concept Window

The idea is to examine the differences or the magnitude of the means and the standard deviations of the posterior distributions of both sets of accuracy rates: the accumulated accuracy rate for the previous instances and the current accuracy rate. If there are any differences, then we need to start a new window. Therefore, we can analyze the effect size and use the Cohen's difference approach [96] as a threshold for change detection. However we will need to take into consideration the size of the sample; therefore we will apply a weighted with the sample size effect size, using a formula provided by Hedges [101].

$$\delta = \frac{\hat{\mu}_{current} - \hat{\mu}_{accumulated}}{s_{pooled}} \tag{5.17}$$

where s is a pooled standard deviation:

$$s_{pooled} = \sqrt{\frac{\hat{\sigma_{current}}^2(n_1 - 1) + \hat{\sigma_{accumulated}}^2(n_2 - 1)}{n_1 + n_2 - 2}} \tag{5.18}$$

We can define the effect size as "Small", "Medium" and "Large" as Cohen did in his work: $small$: $\delta \leq .20$, medium: $\delta \in (0.2; .5]$, large: $\delta > .50$. We can assign a warning and drifting level, using

those thresholds. For example, if the Cohen's difference is greater than $.8$ we can assign a warning level, however, if it is greater than 1.2 then there should be drift level: Sawilowsky, 2009. Another important idea that needs to be mentioned in the current work is that the Cohen's thresholds are based on the assumptions that the data follow a Normal distribution, which in our case does not have to be true. Our theoretical distribution is a Beta family distribution. It is true though that if we apply the CLT, with a large enough sample, we can still approximate beta with a normal distribution. Therefore additional work needs to be done, to find an appropriate threshold for Beta distribution and not normal if it happens that we have a small sample. The Cohen's effect size is still an appropriate measure for change in the distribution. If the first and the second moments of the Moment Generating Functions of two distributions are different enough, then there is sufficient information to claim that we can raise an alarm that will force the model to be retrained and a new window to start. The question that will remain is at what value of δ we can set the warning and the drift level. For a large enough sample though, we can use:

$$k_w : \delta \geq .8$$
$$k_d : \delta \geq 1.2$$

5.5 Results from applying NASCA Drift

5.5.1 Outline of the procedure

Now that we have outlined the used classification approach - Adaptive Random Forest and the established drift detection model for streaming data, we will present the results from the classification process for the NASCA procedure adapted for streaming data. Although the data are collected and classified at the same time and accuracy is measured right away in the interleaved test then train approach, we can apply NASCA for every single interval that we select to test and then train in the classification process.

- **Network data selection**

  We can still start at the beginning with the network data selection, but this time instead of downloading a whole lengthy data set to analyze, we will take into consideration just ten,

hundred, thousand, etc. instances, a small interval, that needs to be investigated, using the interleaved test-then-train evaluation.

- **Attribute selection** The attribute selection as a step can be applied to the interval that we have selected in the part Network selection. It will be the same idea and the same Information Gain approach for attribute selection, however, the upper mentioned adjustments for the streaming set up need to be done as it is described in [103].

- **Classification** The classification step will apply Adaptive Random Forest only. It is enough for the algorithm to identify that there is an attack occurring without to analyze the type of attack. We will analyze the type of the attack in another module that will come at a later stage as it is described in Chapter 6.

- **Accuracy** This step is performed for every single interval, described in the Network Data Selection Process.

- **Drift** The Drift step is to identify whether there is a distribution change in the procedure, so a new window of data to be selected and the NASCA procedure to start over. It analyzes separately another process that is related to the accuracy only. There is also a warning level k_w, as we have already suggested. This level manages the alert indicator; when a warning level is activated, the model automatically begins to train a parallel tree together with the current tree.

The whole updated NASCA Drift procedure on Fig. 5.5.1 is very similar to the previous NASCA procedure with the addition of the drift component and the idea that the analysis is done for a small batch of observations, accuracy is observed, and the model has to be repeated. The NASCA Drift procedure will represent the so-called Detecting Module of the intrusion detection process.

The model modification includes additional step and this is the *Drift* step. The difference is also in the idea that the NASCA procedure is applied infinity times, depending on the data stream. Once a small set of data is collected, we start a window, using the adaptation of Information Gain, we classify and test the accuracy, then we check whether there is a modification in the characteristics of the model until the current moment. If there is not, then we start over and continue the window with the next small set and so on, until a change is detected. Once that happens, we can start a new window, delete the previous data set from memory and begin the NASCA Drift procedure

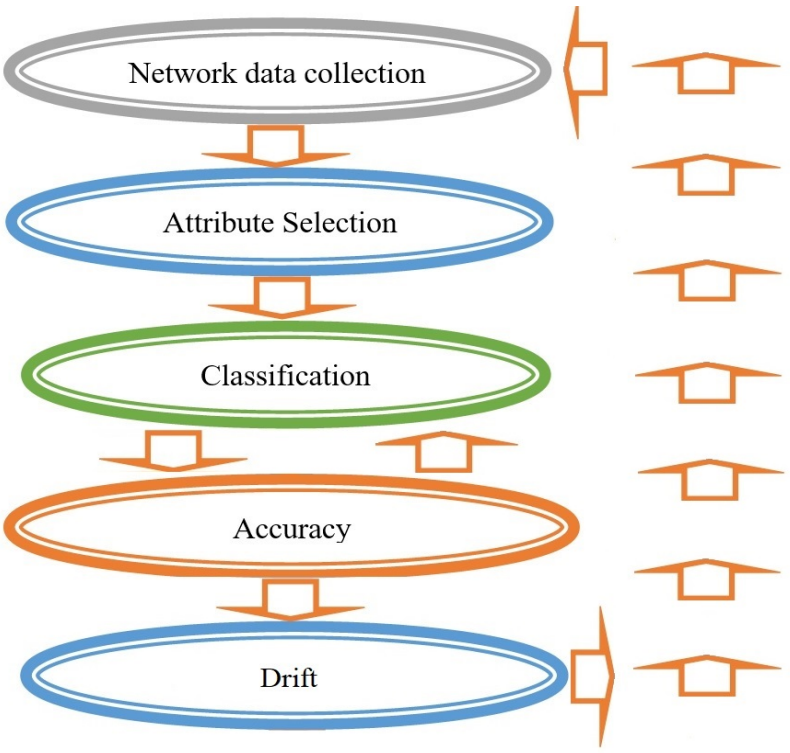


Figure 10.: NASCA Drift for streaming data

again. On Fig. 5.5.1 we can observe how the accuracy rate changes over time, with Adaptive Random Forest, using MOA (Massive Online Analysis) [104], which is a free JAVA based tool for manipulation of streaming data.

5.5.2 Drift Detection

In the following section, we will provide results for the calculation of the Cohen's distance of the posterior distribution. Let us assume that currently we are located, as it is represented in Fig. 5.4.2 right after the end of the a_n^{th} set. So the current set a_n^{th} is represented by 100 instances. It is important to mention that each accuracy rate is a random variable, so there is an associated distribution with it or $a_n^{th} \sim Bin(100, p)$, where n is known. To detect a change from the current set to the previous, we need to compare them. On Fig. 5.4.2 we can observe that we will take the accumulated accuracy for the former $n-1$ sets. In our case will be represented by the accumulated sequence $A_{n-1} = \{a_i\}_{i=1}^{n-1}$, where $A_{n-1} \sim Bin((n-1) * 100, p)$. Now that we have outlined the two sets that we will compare, we can use Bayesian inference, to identify the changes. We can

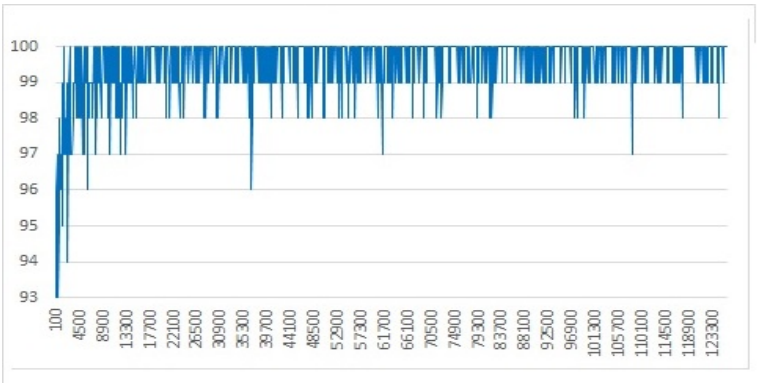


Figure 11.: Accuracy Rate with Adaptive Random Forest

find the posterior point estimate, and the spread of A_{n-1} and compare with the point estimate and the standard deviation of a_n^{th}. Therefore, we can detect whether there is a difference between both means and both deviations that are calculated from the current and the past sets.

To make sure that our analysis is correct, first, we need to consider the idea that NASCA needs to be retrained, because of possible changes in the distribution, characteristics of the attackers, network traffic limitations, etc. All features that are helping us to classify the network, of course, change over time. If a significant change occurs, we need to retrain the NASCA algorithm, to end the current window of data and to start a new one. If we do not do so, the used classification technique will be no longer capable of classifying the network correctly. As an addition to data stream set up, it is impossible to analyze all of the data at once. There is a memory insufficiency; therefore, we need to use the window concept so that we can set limitations on the analyzed data. To identify those changes, as we have mentioned before, we will employ the accuracy rate as an orientation of whether the model needs to be retrained or not. Therefore, to do the classification, parallel to the NASCA procedure, we will create another process, that will help us to identify a warning and drift level for each window. In this work, we will associate drift level as an ending the current window, deleting the data of that window and staring a new one, with a new set of data and further training of NASCA. Once the examination starts, two accuracy rates are recorded Fig. 5.4.2. The first one is the accuracy rate for the current set of hundred observations, the second one is the accumulated accuracy rate so far, excluding the current hundred examples. In case that there are no changes detected, the NASCA model can continue working in the same manner without the need of being retrained and the window will keep expanding. However, if a change is

detected, then the current window has to be ended, the model to be retrained and a new window to start so that we can take into account the new changes. We will use the theoretical distribution of the accuracy rate as a Binomial for both accuracy sets, find the posterior, using Jeffrey's prior: in our case Beta distribution. We will estimate the mean and the variance of the posterior distribution of both accuracy rates. In the case of not detected changes in both means and standard deviations, there is no need to retrain the NASCA again, and we can remain in the current window. However, if there are significant changes, we have to retrain the NASCA approach. We will provide a measure of the difference between both means and standard deviations. In summary we will observe four parameters: the two means of the current and the accumulated posterior accuracy distributions: $\mu_{current}$ and $\mu_{accumulated}$ and the two standard deviations: $\sigma_{current}$ and $\sigma_{accumulated}$. We will employ Bayesian inference for the estimation problem. The selection of the prior distribution has a minimal impact on the estimation process, and the observed data will play a significant part in the Bayesian inference process. All of the calculations and coding are performed in R.

We will measure the Cohen's effect size and will provide results for the difference in the means Fig. 5.5.2, the difference in the standard deviations Fig. 5.5.2, and the effect size Fig. 5.5.2. First, we will start the assessment from the beginning of the accuracy rate, and we will observe the changes in its distribution as it is shown in Fig. 5.4.2. Since we use test-then-train evaluation for streaming data, the first several observations will always give deviations in the accuracy, until the model is trained after that though, the accuracy has to remain steady and high. The model has to be able to classify the observation with a small error rate. Therefore, the differences in the means of both distributions (Fig. 5.5.2) is at a high level for the first 10 thousand observation. These can be interpreted as there are some differences in the means of the distribution of the accuracy rate from the first 9 thousand observations and the distribution built by the next set of 1000 observations. Then the difference for the first 19 thousand and next thousand gradually decreases, or there is not as much difference between both distributions and so on until the fluctuation almost terminates. By observing this graph it is not easy to make an inference about the changes, the difference although decreasing, it remains at a high level from a distance between 20 thousand and 30 thousand. In all cases, the mean is a positive number suggesting that the $\hat{\mu_{current}}$ is always greater than the past one and this is mainly because the past $\hat{\mu_{accumulated}}$ is the accuracy for not as a trained model as the current one. The differences in the standard deviations in Fig. 5.5.2 has the same pattern. In

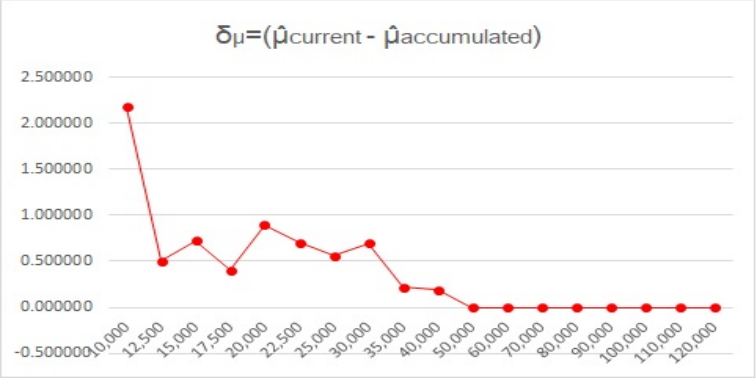


Figure 12.: Difference in means

the beginning, the graph suggests a great difference in the deviations of the accuracy rate from the current to the past evaluated set. The difference is negative, because $\hat{\sigma_{current}}$ is smaller than $\hat{\sigma_{accumulated}}$. That implies that the model is improving with the time. We can observe that the difference decreases gradually but somewhere between 20 and 30 thousand it goes back up. That suggests that something is happening between those observations around 30 thousand observation. We will calculate next the Cohen's distance, so we can observe to what extent the effect size will change due to that increase of the difference in the deviations.

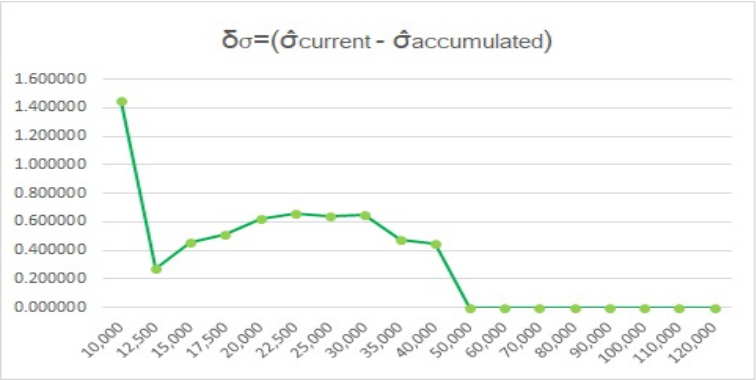


Figure 13.: Difference in standard deviations

The change in the effect size according to Equation 5.17 can be observed in Fig. 5.5.2. It is obvious now that between 20 and 30 thousand observations, there is a jump in the effect size. It is normal the level of it to be quite high for the first observations, while the model is in the process of training. It is also visible how the effect size starts decreasing at the beginning, which is because the model reduces the error, once the training is done. However change in the distribution is observed

and there are differences in the first and the second moment of the Moment generating function of the Beta distributions, which can not be denied. Therefore after 30 thousand observation, the current window needs to be stopped and a new one to start. The model will be able to adjust later to the late characteristics, because we use test-then-train evaluation, so the model learns on the go.

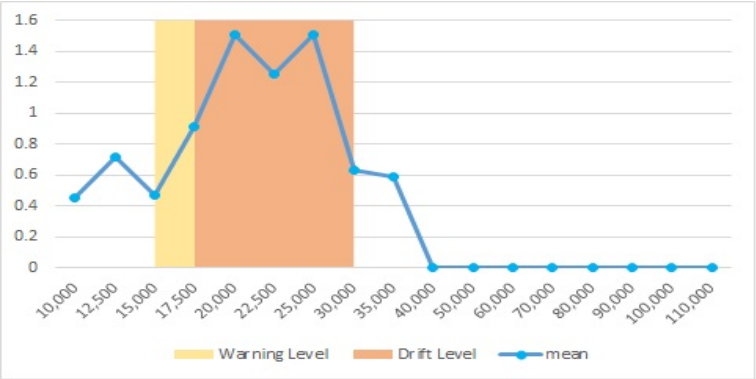


Figure 14.: Effect size

As we can observe on Fig. 5.5.2, using the proposed (quasi) Bayes Drift model, the warning level k_w starts around 15,000th example and the drift level shortly after that around 20,000th example. These results are not fixed, and since we have simulated Beta distribution with sequential MCMC algorithm, the estimated mean and variances also have their variation, as well as the resulted effect size. In Appendix A there is a table 6 of the confidence intervals of the different estimators.

5.5.3 Comparison with other drift detection models

We will try to compare with the different drift levels suggested by other models, so we can identify how well we detected a distribution change comparing to other results. It is important to mention the idea that there is a possibility that the model can make a mistake about the distribution change, the consequence will be a not accurate prediction concerning classification, and therefore there might be some significant consequences regarding intrusion detection purposes. The drift detection with other methods will be completed with MOA(Massive Online Analysis) [104], which is a free JAVA based tool for manipulation of streaming data. The results a represented in Table 6. Our model and the ADWIN model identify a drift detection at the same level at instance 20,000. The other models are not very reliable for this data set, because they either do not recognize any changes

Table 6: Comparison with other Drift Detection Models

Model	Observation with Warning Start	Observation with Drift Start
ADWIN	no detection	20,000
Dseed	no detection	9,990
DDM	no detection	no detection
HDDM	no detection	100
Moving Average	no detection	100
DRIFT Bayes	15,000	20,000

or they categorize as a change every single example. The other models identify a shift in the prior probabilities. They use multivariate statistical tests and Exponentially Moving Average [106].

5.6 Concluding Remarks and Contribution

The classification of streaming data is a necessary and commonly used procedure in network security. Most of the study is concentrated in considering a training and testing data sets, and an advantage is given to the process of downloading the data in the memory of the computer. Although this method can indeed produce accurate and precise results, the problem is that in a situation of a data stream this approach is inappropriate. The attacker will not wait for the administrator to analyze, he would attack right way, and although these techniques are helpful for investigating the attacker's behavior, they cannot serve as a protection tool. Therefore we need to find quick and prompt models, that are adaptive, according to the data stream. In this chapter, we propose an additional drift detection tool, using a (quasi) Bayesian approach that can serve as an alternative to the popular drift detection models in streaming data analysis. This method avoids the assumption of normality that is used in the current data drift detection models and can provide a better interpretation of the data stream.

Chapter 6

NASCA Drift and Q-learning Block

6.1 Outline of the Proposed IDS block

In this chapter, we will present the entire proposed procedure which could serve as an Intrusion Detection System. The schema in Fig. 6.1 consists of two main modules. The first section is a ”Detection Module” which will perform two fundamental functions: monitoring the traffic and detection of the attack if there is any. Here the traffic flows continuously, and it is monitored and analyzed, using NASCA Drift procedure.

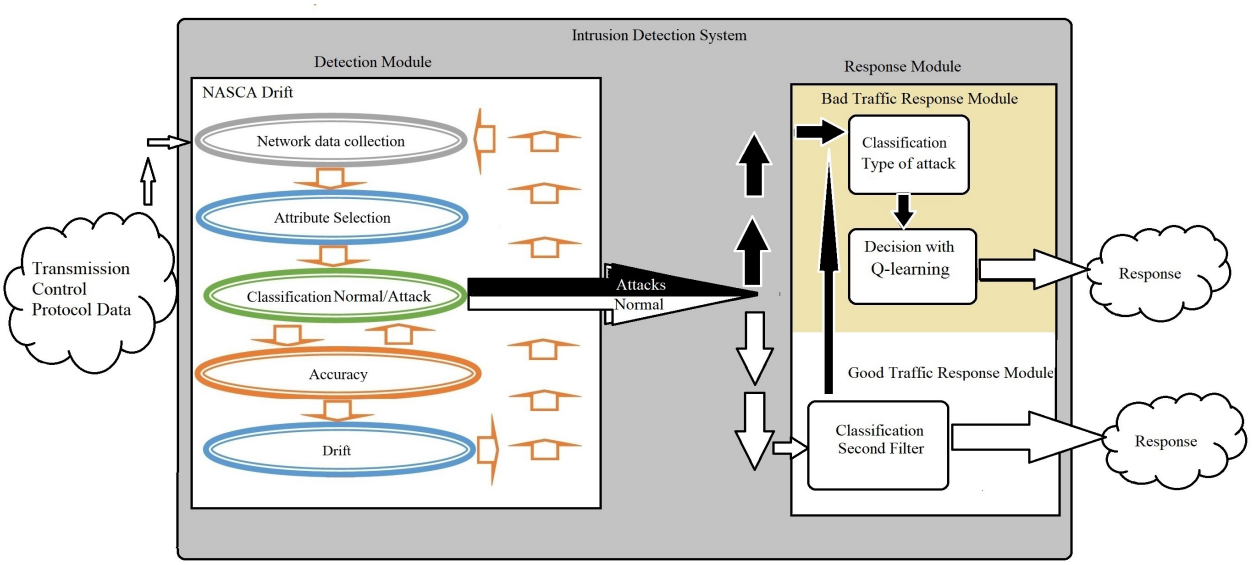


Figure 15.: Intrusion Detection Block

If we are just interested in identifying whether there is an attack occurring or not, it will be enough to employ the first model, which was outlined in Chapter 5.

It is intriguing to examine the second layer and the connection that it makes with the NASCA Drift procedure. The whole IDS is a designed as a combination of detection and response modules, where the NASCA drift is the detection module, and it serves the purpose of separating the ”good” from the ”bad” traffic. As we have discussed before most of the IDS models are limited to the monitoring process, and it is challenging to create a response, once an attack is detected. The data flow enters first the detection module and precisely the Network Data Collection step of the NASCA Drift procedure. There is a regular information exchange between the classification and the accuracy steps, because of the nature of the interleaved test then train analysis of the traffic. The traffic is divided into ”bad” and ”good” on the Classification step of the NASCA drift. Therefore all ”attack” instances are sent separately from the ”normal” labeled ones to the second layer for additional analysis. Once the ”bad” traffic is separated, it is taken offline and transmitted to the ”Bad Traffic Response” sub-layer. The ”good” traffic continues the online setup and is forwarded to the ”Good Traffic Response” sub-layer. So, both sub-layers of the response module are different in a way that the attacks are analyzed offline, they are collected in the so-called batches. However, the ”normal” instances continue the online stream and are sent to another classification process, that is adapted to streaming data. Both of the sub-layers that are part of the response layer are equipped with a second filter for the incoming traffic, and both modules respond to the system. The ”bad” traffic is gathered and collected in batches first, and additional analysis is performed. The number of batches that will be processed at a time depends on the specific bandwidth that the current system possesses and how many megabits of information per second can the system devote for the transfer for a specific dataset. The reason for using a data streaming adapted model for the regular instances, and we collect testing set for the instances that have been labeled as ”attacks” is mainly related to the idea that it is essential to find a trade-off between the need of in-depth analysis and traffic flow without interruptions.

6.2 Functions of the Classification in the Response Module

The first connection between the Detection and the Response modules starts at the classification step of the NASCA Drift procedure. This step implements Adaptive Random Forest for recognizing whether the network is under attack or not as we already suggested in the previous chapter.

Once a breach is detected, then the system isolates the ”infected” instances from the regular ones and transmits them to two separate submodules. The ”Bad” Traffic Response sub-module handles the attacked traffic, and the ”Good” Traffic Response sub-module handles the regular one. Both submodules that are part of the Response layer have additional classification filters that serve as an extra security measure for the network protection.

6.2.1 Classification in the Bad Traffic Response Sub-Module

The traffic here is taken offline for deep analysis and extracting more information for the type of the attack. The traffic is not deleted as in most of the IDS, but it is further monitored and managed. Once receives the information, the Bad Traffic Response sub-layer collects a testing set depending on the bandwidth and the megabits per second that the administrator will devote for that purpose. Here the ”bad” traffic is collected from the NASCA Drift procedure, which represents the first layer of our schema and downloaded for the type of attack classification purposes. There is a training set located in the memory of this module. We will employ as labels for the type of the attack the main four classes, as we defined in Table 1 in Chapter 3: DoS (Denial of Service), R2L (Root to Lan), U2R (User to Root) and Probe. Again as in the NASCA procedure in Chapter 3, we will use PART for classifying the type of the attack. In this sub-module, unlike the detection module, we will use training and a testing sets for the classification of the class of the attack. The training set will still include the label ”normal,” although we are just interested in the type of the attack. The idea will be to provide another ”chance” for traffic that has been classified as an ”attack” by the NASCA Drift procedure, to go further to the Q-learning for a response even if it happens that it is a ”normal” connection. Furthermore, we will reduce the possibility of the attacker to send ”normal” traffic that appears as an ”attack” with the intent to overload the protection mechanism. If we can provide two filtrations in the detection and the response modules, we increase our security and make more precise decisions. For an accurate response, the whole group of affected data is classified again, using the PART model described earlier. Unlike the NASCA Drift procedure which uses the test then train online collection of the data, this sub-module will manage a batch collection, where all data are analyzed at once. The training set has to be also updated with the newly discovered attacks; this is an essential procedure. The results from the PART analysis are presented in Fig. 6.2.1.

Based on the ROC curves, we can identify that classes of attack like Probe, DoS and R2L are

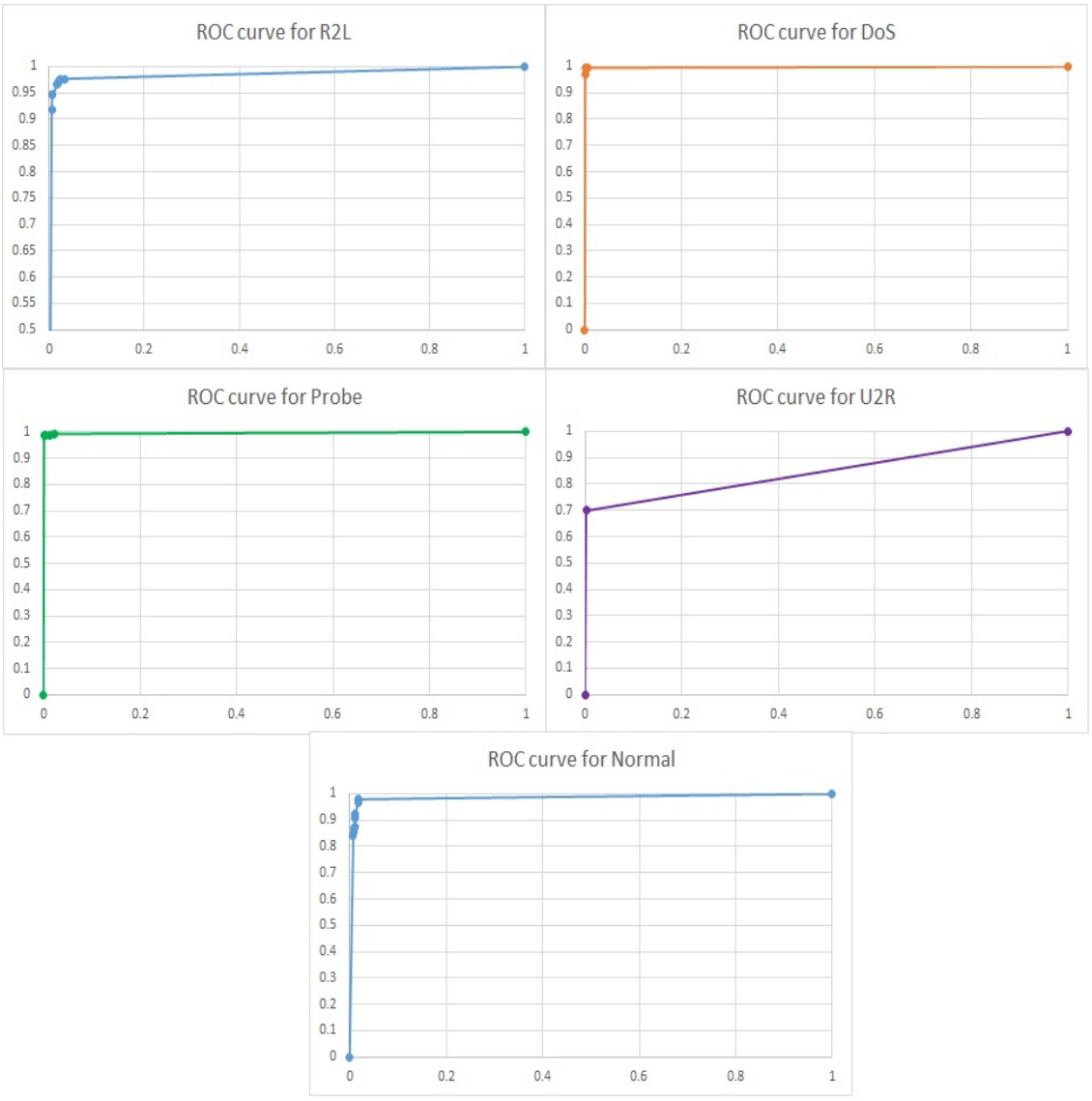


Figure 16.: Receiver Operating Curves for the Type of Attack

Table 7: Accuracy By Class for PART Classification

Accuracy By Class								
TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.966	0.017	0.946	0.966	0.956	0.942	0.985	0.962	R2L
0.993	0.001	0.998	0.993	0.995	0.993	0.997	0.996	DoS
0.925	0.012	0.945	0.925	0.934	0.92	0.984	0.94	normal
0.988	0.004	0.984	0.988	0.986	0.982	0.995	0.987	Probe
0.7	0.002	0.737	0.7	0.718	0.716	0.849	0.518	U2R
0.971	0.008	0.971	0.971	0.971	0.963	0.99	0.972	Weighted Average

Table 8: Confusion Matrix for PART Classification

Confusion Matrix					
R2L	DoS	normal	Probe	U2R	←Classified as
542	2	14	0	3	R2L
0	856	5	1	0	DoS
24	0	392	6	2	normal
4	0	2	497	0	Probe
3	0	2	1	14	U2R

manageable and the model did not find any difficulty in recognizing them. The U2R, however, is challenging for the PART algorithm and this is because we have minimal instances that can be classified as U2R, so the model is not trained well when the result is produced. We introduced few ”normal” examples in the testing set, so we can analyze how well is the algorithm doing with recognizing the regular traffic. The current response module has to investigate only the attacks; however, we would like to include the standard instances preventively in the training set, so the module to be able to fix some of the mistakes of NASCA drift if there are any.

In Table 7 we can observe the accuracy by class, including measures like precision, PRC Area and ROC area as complementary information. The ROC and PRC areas are lower for U2R and highest for DoS, which is another proof that the model can recognize best the DoS attacks, and it is logical because first DoS is more common than the other attacks so that the model can identify them quickly with good precision. Also, the DoS is caused by apparent characteristics like more volume of traffic sent from one network host for example. The reported error rate is 0.0116 for Mean Absolute Error and 0.0959 for Root Mean Squared Error for a complete PART model. We present a confusion matrix in Table 8.

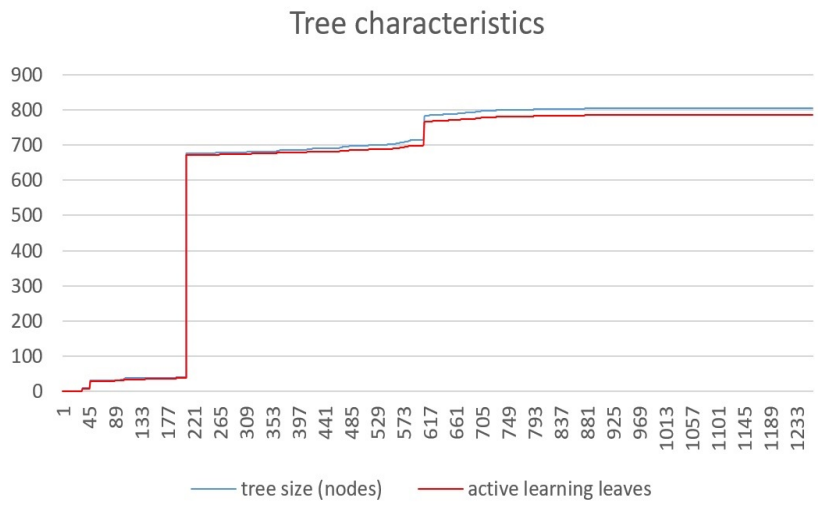


Figure 17.: Good Traffic Response Classification Characteristics

6.2.2 Classification in the Good Traffic Response Sub-Module

In the Good traffic classification sub-module, as we have mentioned in the beginning, we will make sure that there is always a constant flow of data and we will implement an on-line data stream classification technique for the data stream of the regular traffic. The primary purpose of having addition filter here is to serve as an extra protection measure that will recognize whether there is an attack or not in the current traffic. If there is malicious traffic identified at this level, the Good traffic response sub-module will send the malicious traffic to the Bad traffic response submodule to the classification step for additional analysis. The entire IDS block is designed in a way that it will provide a smooth flow for the regular cases, while if there is a doubt of an attack instance, more analysis is needed to make sure that the traffic is under attack, so the system to produce an appropriate response. For this classification section, we will apply Hoeffding trees [99], streaming data classification procedure. We described the classification process in the previous chapter because the adaptive forest uses the same learner in the structure of the analysis. The mean accuracy of the model is 99.37. The splitting criterion is Information gain, and some of the characteristics like the number of leaves and nodes are presented in the next Figure 6.2.2.

6.3 Functions of the Q-learning Process in the Response Module

The following segment of the response module which outlines the Q-learning section will assist the process of replying if the network is under attack. The beginning of the module will not only classify the type of the attack but also provide another option for returning to the network a wrongly classified as attack instances. After PART is applied, the classification will be able to contribute information about the type of the intrusion. So each intervention will be classified as one of the following labels $\langle normal, DoS, Probe, R2L, U2R \rangle$.

Those labels as mentioned above will serve as a starting point of the Q-learning algorithm and will represent the states for the new MDP that will be required. Now we will represent the second module as an MDP. As we defined in Chapter 4, the MDP is described by the tuple $\langle S, A, R, T, \gamma \rangle$. where S is the set of states, A is the set of actions, R is the reward function, T is the transition probability, and γ is the discount factor. If we determine all of the elements of the set needed for the foundation of the MDP, next, we can promptly create the Bellman equation, provided in chapter 4 (see Equation 4.9) and apply the Algorithm 2.

- S is the countable set of states $S : \{s_N, s_{DoS}, s_{Probe}, s_{R2L}, s_{U2R}\}$, where s_N is the event when the network is healthy. Another state is s_{DoS}, where the network is under Denial of Service attack, usually happens by over-flooded with data memory. s_{Probe} represents a Probing attack or data is collected about the network activity, so the vulnerabilities of the system to be identified. The s_{R2L} is Remote to local Attack, or the attacker does not possess any legitimate account, and he tries to find vulnerable points to gain access, so he can steal information or send viruses. s_{U2R} is User to Root Attack where the attacker targets the vulnerabilities of the network from a legitimate account. In these states setup, we use the class and not every single type of attack. These states of the network could also be substituted with the type of the attack that we are experiencing.

- A is the set of actions or the action space: $A : \{a_{block}, a_{allow}, a_{report}, a_{sendback}\}$. The so defined set of actions is just an example, they could be alternated, and it depends completely on the administrator what those actions could be and how many of them he could use. There is no restriction on the number of actions; however the computation time should be considered as a factor. In this particular set up a_{block} is the action when the agent blocks the infected traffic and

deletes it after that. This action is beneficial in the case of DoS attack. The action a_{allow} is the action when the agent "allows" the traffic back in the network. This action can be employed if the PART classification model classified the network as "normal," and that way can provide another chance to return regular traffic to the system. The action a_{report} can interact with the user and ask directly whether he accepts the traffic. Action $a_{sendback}$ is an advanced type of action, because it provides the decision agent besides with the protection function, the ability to act back to a potential intruder. This set up of the action space is assembled in a way so that the decision agent to be able to perform two main objectives. The goal of the agent is to protect the system. Most of the IDS systems are designed in a way to discover whether the network is under attack or not or to identify the type of the attack; this is the main object as an option of protection. The literature of the response models is limited to activities, related again to serve the main protection function. In this work, we will introduce a new option of protection for the response module, the possibility of responding back to the intruder.

– R represents the reward which the agent will receive for making a correct or wrong decision. He will take the reward for taking action A_n at state S_t, $f : S \times A \rightarrow R$. In our case, we will have twenty rewards instances, because there are five states and four actions elements.

– T is a state transition probability matrix. It specifies the probability of transition from state i to state j, on taking action $A_n = a$, where $i \in (0, t]$ and $j \in (0, t + 1]$. There will be twenty associated transition probabilities for each state that we will start with because there are four actions and five possible states for each action. Since we can start with all possible five states, there will be for each state to start with 20 transition probabilities or hundred in total, which can be represented with four matrices with dimensions five by five.

– $\gamma \in [0, 1]$: a discount factor.

The proposed possible decision tree of the agent is outlined on Fig. 6.3. The transition function

for the action ”block” for example will be:

$$T_{ij}^{block}: \begin{bmatrix} T^{block}{}_{normal,normal} & T^{block}{}_{normal,DoS} & T^{block}{}_{normal,Probe} & T^{block}{}_{normal,R2L} & T^{block}{}_{normal,U2R} \\ T^{block}{}_{DoS,normal} & T^{block}{}_{DoS,DoS} & T^{block}{}_{DoS,Probe} & T^{block}{}_{DoS,R2L} & T^{block}{}_{DoS,U2R} \\ T^{block}{}_{Probe,normal} & T^{block}{}_{Probe,DoS} & T^{block}{}_{Probe,Probe} & T^{block}{}_{Probe,R2L} & T^{block}{}_{Probe,U2R} \\ T^{block}{}_{R2L,normal} & T^{block}{}_{R2L,DoS} & T^{block}{}_{R2L,Probe} & T^{block}{}_{R2L,R2L} & T^{block}{}_{R2L,U2R} \\ T^{block}{}_{U2R,normal} & T^{block}{}_{U2R,DoS} & T^{block}{}_{U2R,Probe} & T^{block}{}_{U2R,R2L} & T^{block}{}_{U2R,U2R} \end{bmatrix}$$

Analogically we can build the other matrices for the rest of the three actions ”allow,” ”report,” ”send back.” The number of the transitions will remain the same, and the only thing that will change is the name of the action from ”block” to ”allow.” For the Q-learning process, we do not need to know these matrices, because we will find the Q-values directly using Algorithm 2 in Chapter 4. An example of reward matrix could be the following:

$$R: \begin{array}{c} \begin{matrix} a_{bl} & a_{al} & a_{rep} & a_{send} \end{matrix} \\ \begin{bmatrix} -1 & 1 & 0 & -2 \\ 3 & -3 & 0 & 2 \\ 2 & -3 & 0 & 1 \\ 2 & -3 & 0 & 1 \\ 3 & -2 & 0 & -3 \end{bmatrix} \end{array} \begin{matrix} s_N \\ s_{DoS} \\ s_{probe} \\ s_{R2L} \\ s_{U2R} \end{matrix}$$

It is important how we will set the matrix of **R**; the values will define the direction that we would like the agent to move. He can be more aggressive or more protective, depending on the values that we will set for each combination of action and state. For example, let us assume that we are currently in the state of R2L or under Remote to Lan attack. Row four of the reward matrix represents the assigned rewarding values for this state. The agent has four options based on the actions: to block and delete the traffic (to drop it), to allow it in the system, to report it to the user and to ask for a course of action, to send it back to the host or the IP address. The R2L attack is when the intruder transmits packets to our computer, with a purpose to achieve access and utilize our vulnerability, he does not hold any account as a user in our system. Suppose that we would like to make the agent more protective. In case of the scenario mentioned above, the

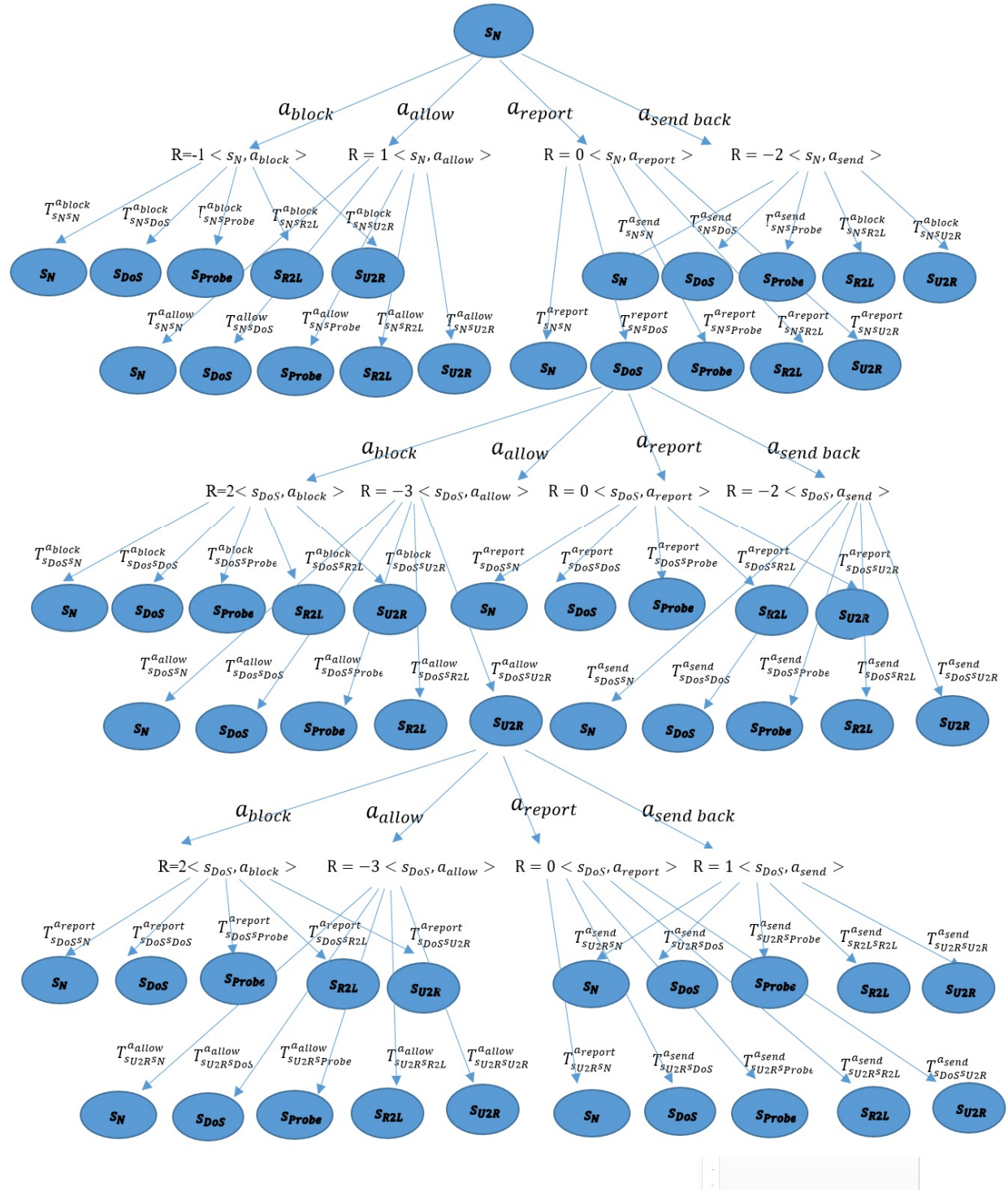


Figure 18.: Example for a Decision Tree for the Agent

worst case is if the agent decides to allow the malicious traffic in our system, therefore he will get a compensation of -3 if he does that. If he reports to the user, it will provide the user with the awareness that someone is attempting to breach in the system. The user will be responsible for determining whether she will admit the traffic or not, the reward here is zero. We will give a value of 1 for sending it back to the host. If we choose to reverse the direction of the traffic, the attacker will be informed that this system is not just protected, but also could reverse the direction of the traffic if needed. Furthermore, it will be prevented from invading our system, and consequently, the agent will collect a reward of one for this action. A different will be the situation if we witness U2R or user to root attack. This class of intervention originates from an infected account that is a current user of the system; the access could have been achieved for example with an R2L attack. The intruder intends to reach the source of the whole operation and to exploit our vulnerabilities. In this example, it will not be practical to send back the traffic to the host, because we will forward it back to our system, which will make the situation worse. Consequently, the reward for that behavior will be -3. A sound beneficial action is to block the traffic, where the agent will achieve a reward of 3, -2 for allowing the traffic and again, and zero for reporting it to the user for an action.

Running the algorithm again provides the following results for the Action-Value function (4.9) with $\gamma = .9$ the dimensions of it depend on the number of actions $A : \{a_{block}, a_{allow}, a_{report}, a_{sendback}\}$ and number of states $S : \{s_N, s_{DoS}, s_{Probe}, s_{R2L}, s_{U2R}\}$, so it is $actions \times states$:

$$Q^* : \begin{bmatrix} 17.31564 & 0.4199652 & 1.1007156 & 0.6979677 \\ 22.38493 & 0.3342630 & 1.9504433 & 1.6028717 \\ 20.14793 & 0.5375468 & 0.4697312 & 0.9841997 \\ 18.88875 & 0.2175585 & 0.7137482 & 1.1171577 \\ 30.00000 & 24.9999978 & 26.9999976 & 23.9999961 \end{bmatrix}$$

The corresponding solution for the Value function (4.8) is a vector ,that depends on the number of states with the following values:

$$V^* : \begin{bmatrix} 17.31564 & 22.38493 & 20.14793 & 18.88875 & 30.00000 \end{bmatrix}$$

The policy that was elected as an optimal (Equations 4.6 and 4.6) is a vector that has identical size as the total number of states. In a particular state, it determines one out of four actions, which

maximizes the Q and therefore V values. It is necessary to emphasize the fact that the discount factor $\gamma = .9$ makes the agent far-sided and he seeks the optimal policy and makes decisions in terms of distant future.

$$\pi_{\gamma=.9}(a|s) : \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Therefore the agent will select action a_{block} in all states. That implies that if we make our agent far sighted , he will not take risks and will always blocks the malicious network. Now we will analyze a different situation where $\gamma = .5$, the agent is not that far-sided, and he is reasonable in his expectations about the future. The policy changes according to the agents expectations, so if he finds himself in the second state which is s_{DoS}, he will prefer to exercise action a_{report}, or in other words if we experience DoS attack, the agent will report the problem to the user and inquire for directions, whether to permit the traffic in the system or not. The agent will select to "block" the traffic in all other circumstances.

$$\pi_{\gamma=.5}(a|s) : \begin{bmatrix} 1 & 3 & 1 & 1 & 1 \end{bmatrix}$$

The last case scenario that we will explore will be if we decide to create a narrow-minded agent, who will react almost based on the current situation without thinking about the future, or $\gamma = .1$. The resulting selected policy vector is as follows:

$$\pi_{\gamma=.1}(a|s) : \begin{bmatrix} 2 & 1 & 1 & 1 & 1 \end{bmatrix}$$

If the agent is short-sided, he will select action a_{allow} if the agent is in state s_N, or he will select to allow the traffic in the system if it happens to be "normal." In all other cases, he will decide to block the traffic. The decisions of the agent are the results that follow, based on the reward matrix that we defined in the beginning. Other rewards structure will produce different policies. As we have mentioned before, we can program the agent to attack back if he has been attacked if we give higher rewards for this action. The whole idea of using Q-learning is that we can set the agent and ask him to perform the tasks that we want him to do in a best possible way, based on the conditions of the environment.

6.4 Concluding Remarks and Contribution

The so proposed IDPS block guarantees the uninterrupted traffic flow in the system and still protects it without any delays. Nowadays most of the network security research is devoted to finding a sophisticated classification model that will analyze the faulty instances with high accuracy. However, once the model is created, it is not precisely described the logistics and how it will be implemented. In this chapter, we propose a clear logistic block for Intrusion Detection and Prevention Systems, that not only will be able to classify the attacks, but also will be able to operate in an entirely unknown environment, provide a response and prevent the attack. The IDS and the IPS combination, the specific schema layout and the proposed idea of combining NASCA Drift as a first layer and the response module as a second is an innovative work that could be implemented in every protection system.

Chapter 7

Game Theories in Network Security [4]

7.1 A Mathematical Definition of a Stochastic Game

Let us define two goods A and B and M as a market basket set for consumption that in general, the agent feels satisfied with. If we name a function $M \rightarrow R$, that maps each market basket to a real value, then we state that the agent favors X over Y or $(X >> Y)$, if $u(X) > u(Y)$. The Von-Neumann [37] utility theory suggests that for a particular agent to be considered as a rational agent, he has to satisfy some evident characteristics:

- Completeness: For every market basket $< A, B >$ we have: $A = B$; $A << B$; or $A >> B$;
- Transitivity: If $A >> C$ and $C >> B$, then $A >> B$;
- Continuity: If $A >> B >> C$, there exist a probability p, that $pA + (1 - p)C = B$;
- Independence: If $A >> B$, for every C and probability $p : pA + (1 - p)C >> B + (1 - p)C$.

The function of u enables us to quantify the agents' preferences in a way that helps us to combine utilities and also to estimate expected utility. We still have in general that $u(X)+u(Y) \neq u(X+Y)$. It is important to mention the moment of the choice making process. In general, all agents select actions simultaneously, all decisions are revealed at the same time, and then the utility is recorded. Each agent chooses a strategy that defines his actions.

A stochastic game can be played by one or more players; it has a probabilistic transition function that represents the probability for an agent to go from one state to another. Each one of them selects a set of strategies and corresponding actions and obtains a return that depends on the current

[4] Portions of this chapter have been previously published in Proceedings of Dynamic Systems and Applications , vol. 7, 2016, pp. 303-310

knowledge about the environment. The game can evolve to a new stage, where the player builds a strategy and acts accordingly, depending on the available information about the other players. The process can be repetitive and may be finite or infinite. A stochastic game will be characterized by $\langle P, S, A, T, R, \gamma \rangle$. Each one of the players selects a set of strategies and corresponding actions and obtains a return that is dependent on the actual knowledge about the environment. The game can evolve to a new stage, where the player builds a strategy and acts accordingly, depending on the available information about the other players. The process can be repetitive and may be finite or infinite.

A n-player stochastic game can be defined as a set, consisting of the following elements:

- $P : \{p_1, p_2, ...p_n\}$, if we have two agents in network security set up ,we will have $P : p_a, p_d$, where p_a will refer to attacker and p_d is the defender;
- $S = \{s_0, s_1 \dots s_t \dots s_N\}$ is a nonempty state set.
- The set of actions of player n at state s_t is a subset of A^n, or $A^n_s \subseteq A^n$ and $\bigcup_{t=0}^{N} A^n_{s_t} = A^n$. In each state the player makes an action. In a primary cyber security set-up the set of actions consists of two sub-sets (attack, do not attack) for the attacker and (defend, take no action) for the administrator.

  $A^a = \{\alpha^a_1, \alpha^a_2 ... \alpha^a_t ... \alpha^a_N\}$ is the set of actions of the attacker p_a at state s_N;

  $A^d = \left\{\alpha^d_1, \alpha^d_2 ... \alpha^d_t ... \alpha^d_N\right\}$ is the set of actions of defender p_d at state s_N;
- $T : S \times A^a \times A^d \times S \to [0, 1]$ is a state transition probability;
- $R^n : S \times A^a \times A^d \times S \to \mathbf{R}$ is the reward (payoff) function of the n^{th} player; After all of the actions are played, each of the agents will receive a reward. For the attacker that may be the count of penetrations in the system, the degree of trouble that he caused (the severity of the damage) and the dollar value of the resources (bandwidth, time) that he spent or saved, etc. For the defender, we can quantify the payoff as saved time, used resources and the level of importance of the information that she protected, etc.
- $0 < \gamma \leq 1$ is again a discount factor for finding a present value of the future payoffs, described in [22].

– Strategies: A strategy is the players' plan of action. They consider the information from the past and also the currently-available information and make a plan to maximize their return. This will direct to the Equilibrium point. Let $\Omega^k = \{p \in \Re^k | \sum_{i=1}^{n} p_i = 1, p_i \geq 0\}$ be a probability set of vectors with a size k, $w^n : S \rightarrow \Omega^{N^n}$ is player n's stationary strategy, $w^n(s)$ is the vector $[w^n(s, a_1) \dots w^n(s, a_N)]^T$, where $w^n(s, \alpha)$ represents the probability of agent n selecting action α at state s. A strategy w^n is considered, that does not depend on time. On the one hand, a mixed randomized stationary strategy is when $w^n(s, a_i) \geq 0$, for every $s \in S$ and every $\alpha \in A^n$. On the other hand, pure strategy is when $w^n(s, a_i) = 1$ for some $a_i \in A^n$.

– Solution:Solution to a game is a specific state called an Equilibrium, where the attackers and the defenders, following their strategies, act accordingly to reach an optimal solution. If the Equilibrium is achieved, we will be able to build an effective response mechanism in cyber security that can run for finite or an infinite period. John Nash (1928-2015) was an economist and a mathematician with fundamental contributions to the game theory because of the Equilibrium that he proposed as a solution. According to the Nash Equilibrium, named by him, the objective of each game is to attain a solution that provides the players with maximum possible returns, minimum costs for the specific game set-up and finally nobody is willing to deviate from this state, because it will lead to smaller returns.

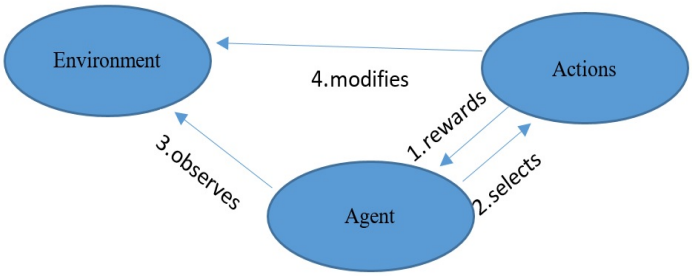
A state transition has a payoff that is equivalent to the calculated value of the reward during that state; however during the following state is equal to γ times this amount. The game begins at an initial state $s_0 \in S$, or at a discrete time t, where $s_t \in S$. The attacker for example decides to select an action ${\alpha^a}_t$ from A^a and the defender selects an action ${a^d}_t$ from A^d. The reward of player one, then is $R_t^a = R^a(s_t, a_t^a, a_t^d)$ and the corresponding one for the defender is $R_t^d = R^d(s_t, a_t^a, a_t^d)$. After that the game goes to a new state s_t or s_{t+1}, where we can define the resulting conditional probability P($s_{t+1}|s_t,\ a_t^1, a_t^2$) which can be also represented by $T(s_{t,} a_t^a, a_t^d)$.

7.2 Solution to a Game

7.2.1 Nash Equilibrium

There is a diversity of techniques that have been suggested for modeling a stochastic game. Nevertheless, solving the optimality equations and improving strategy methodology have been the

Figure 19.: Strategic Learning Model with Agents in Reinforcement Learning



most significant methods, used to define the solution to the game. Hoffman developed strategy methodology for general stochastic games [14]. In this design, the original policy of the agent develops with every iteration by changing conditions at which decisions are not locally optimal. Furthermore, the local optimality equation method is solved with a system of constraints for the anticipated payoffs. The optimal payoffs, serve as a base for creating a competitive strategy. If a single agent is observed, there are linear optimality equations, and consequently, this game has a solution obtained with linear programming methods. However, for multiple players, the conditions are not linear, so approximation procedures are needed to find an equilibrium point.

The Nash Equilibrium suggests that despite knowing the actions of their opponents, the agents do not have any urge to adjust their strategy; therefore they are better off in that state. Consequently, each deviation from the Nash Equilibrium will lead to a higher level of losses. The goal of each agent is to maximize his anticipated payoff. If s_t is a state and R_t^n is a reward, obtained by agent n at time t. Then, the expected return will be represented by the vector $v_{w^1,w^2}^n = [v_{w^1,w^2}^n(s_1) \dots v_{w,w^2}^n(s_N)]^T$, where:

$$v_{w^a,w^d}^n(s) = E_{w^a,w^d}\{R_t^n + \gamma R_{t+1}^n + \gamma^2 R_{t+2}^n + \dots \gamma^N R_{t+N}^n | s_t = s\} = E_{w^a,w^d}\{\sum_{k=0}^{N} \gamma^K R_{t+k}^n | s_t = s\} \tag{7.1}$$

$E_{W^a,W^d}\{.\}$ is the expected value when player n chooses an action, using probability $w^n(s_{t+k})$ at s_{t+k} and obtains the corresponding reward:

$$R_{t+k}^n = w^a(s_{t+k}) R^n(s_{t+k}) w^d(s_{t+k}), fork = 0. \tag{7.2}$$

$R^n(s) = [R^n(s, a^a, a^d)]_{a^a \in A^a, a^d \in A^d}$, is the reward that agent n receives in state s. On the one

hand, if $N = \infty$ as well as $\gamma < 1$, then $v^n(s)$ is the expected total discounted reward that player n will receive when he starts at state s. On th other hand, if $0 < N < \infty$ and $\gamma = 1$, then v^n is a the value that player n will obtain, represented by a vector and if the process is fixed, then the Nash Equilibrium can be determined as:

$$v^a(w_*^a, w_*^d) \geq v^a(w^a, w_*^d) \forall w^a \in \Omega^{N^a} \tag{7.3}$$

$$v^d(w_*^a, w_*^d) \geq v^d(w_*^a, w^d) \forall w^d \in \Omega^{N^d} \tag{7.4}$$

Both agents employ their fixed strategies w^a and w^d at every step simultaneously. Once equilibrium is achieved, they do not have any reason to change their optimal strategies w_*^a and w_*^d. A divergence implies that one or both will earn less expected payoffs. A set of Nash strategies produces best responses for all players. For instance, if there are two agents and the first one adopts strategy w_*^a, that can also be defined as the exercised optimal strategy for the attacker, then the second ones or the defender's optimal response strategy is w_*^d. The gain that the attacker expects to get is $v^a(w^a, w^d)$, it is achieved by exercising w^a; similarly, $v^d(w^a, w^d)$ is the payoff that the defender expects to receive when he decides to adopt strategy w^d. The best response strategies are w_*^a and w_*^d, and if the agents employ them, a Nash Equilibrium is achieved [32]. Different concepts and equilibriums can serve as a solution to a game, like min-max strategy, in Nash sense, Bayesian Equilibrium and numerous modifications of Nash Equilibrium [39]. In this chapter, another type of solution is outlined, so the interplay between the administrator and the hacker to be emphasized.

7.2.2 Bayesian Equilibrium

Let us define Θ as a set with elements θ, and this set represents the type of the attacker. The defender as a player, for example, knows his type and his actions, that are represented by α_1^d, where $\alpha_1^d \in A^d$ is the action set. Analogically let us suppose that other player, the attacker will choose $\alpha_1^a \in A^a$ and that he will experience some prior beliefs about the characteristic of the defender. In other words, let us assume the administrator believes that the probability of the attacker being a specific type is $p(\theta)$, where $\theta \in \Theta$. The return of player n will be similar as before; however, now he will also need to consider the type of the player as a part of the reward function:

$R^n(s) = \left[R^n\left(s, \alpha^a, \alpha^d, \theta\right)\right]_{\alpha^a \in A^a, \alpha^d \in A^d, \theta \in \Theta}, n = 2, \{a, d\}$

The defender will have the following strategy: $w^d\left(s|\theta\right)$ over the actions α_d, which is conditional on the type of opponent. Analogically for the attacker the strategy will be represented by the following distribution function $w^a\left(s|\alpha^a\right)$ over actions α^a conditional on α^d.

The payoff of θ with strategy $w^d\left(s|\theta\right)$, assuming that the attacker has played $w^a\left(s|a^1\right)$ is the following:

$$\left[R^d\left(s,\alpha^d,\alpha^a,\theta\right)\right]_{\alpha^d\in A^d,\alpha^a\in A^a,\theta\in\Theta}=\sum_{\alpha^d}\sum_{\alpha^a}w^d\left(s,\alpha^d|\theta\right)w^a\left(s,\alpha^a|\alpha^d\right)B^d\left(s,\alpha^d,\alpha^a,\theta\right)$$

If the defender selects strategy $w^d\left(s|\theta\right)$ then the other one's reward function to strategy $w^a\left(s|a^a\right)$ will be the following:

$$\left[R^a\left(s,\alpha^a,\alpha^d,\theta\right)\right]_{a^d\in A^d,a^a\in A^a,\theta\in\Theta}=\sum_{\theta}p(\theta)\sum_{\alpha^d}\sum_{\alpha^a}w^d\left(s,\alpha^d|\theta\right)w^a\left(s,\alpha^a|\alpha^d\right)R^d\left(s,\alpha^d,\alpha^a,\theta\right)$$

The attacker amends his beliefs about θ in order to obtain the following posterior distribution $\mu\left(s\mid\alpha^a\right)$ over Θ, where μ is a belief or a function that assigns a probability measure from the past to the possible and unobserved information structure. In this type of game, instead of Nash Equilibrium, we will be interested to obtain Bayesian Equilibrium [21]. Therefore let $w^{d*}\left(s|\theta\right)$ be the strategy used, then having information about this strategy by observing α^d,the attacker may use a Bayes rule to update $p\left(.\right)$ and $\mu\left(s\mid a^a\right)$.

A Bayes Equilibrium is obtained when the players have future beliefs about $\mu\left(s|a^a\right)$ and adopt strategy w^* so that:

$$Player^{defender}:\forall\theta,w^{d*}\left(s|\theta\right)\in\underset{\alpha^d}{argmax}\,R^d\left(s,\alpha^d,\alpha^a,\theta\right)$$

$$Player^{attacker}:\forall\alpha^d,w^{a*}\left(s|\alpha^d\right)\in\underset{\alpha^a}{argmax}\sum_{\theta}\mu(\theta|\alpha^d)R^a\left(s,\alpha^a,\alpha^d,\alpha\right)$$

$$\mu\left(\theta\mid\theta^d\right)=\frac{p\left(\theta\right)w^{d*}\left(s,\alpha^d|\theta\right)}{\sum_{\theta'\in\Theta}p(\theta')w^{d*}\left(s,\theta^d|\theta'\right)}$$

Where $\sum_{\theta'\in\Theta}p(\theta')w^{d*}\left(s,\theta^d|\theta'\right)$ is strictly positive and $\mu\left(s\mid\alpha^d\right)$ is a probability distribution on Θ.

Table 9: Games solutions according to the information structure

	Complete	**Incomplete**
Perfect	Nash Equilibrium Perfect Bayesian Equilibrium	Perfect Bayesian Equilibrium
Imperfect	Nash Equilibrium	Sequential Equilibrium
	Nash Equilibrium	PBE or SE

7.3 Q-learning in Game Theories

Let us have a set of policies $\pi_1, ..., \pi_n$, where n is the number of players, and let us define $Q_i(s_j, \pi_1, ..., \pi_n)$ as a weighted sum of all $Q_i(s_j, a_1, ..., a_n)$ then we will define the action-value function as:

$$Q_i(s, a_1, ..., a_n) = R_i(s, a_1, ..., a_n) + \gamma \sum_{s_j \in S} T(s_i, a_1, ..., a_n, s_j) Q_i(s_j, \pi_1, ..., \pi_n) \tag{7.5}$$

We can use the Q-value function as a benefit or reward that each agent will obtain during their game, then based on this benefit we can find a Nash equilibrium and define the Q-learning algorithm the same way as before but with different Q-learning updating function:

$$Q_i(s, a_1, ..., a_n) = (1 - \lambda_t) Q_i(s, a_1, ..., a_n) + \lambda_t (R_i + \gamma * Equilibrium_i(s, Q_1, ..., Q_n) \tag{7.6}$$

$Equilibrium_i(s, Q_1, ..., Q_n)$ will be the type of equilibrium that we will employ, depending on the current information structure, mentioned in Table 9 that we observe in the network security set up. The learning factor in this Game Q-learning set up will be denoted by λ. The equilibrium is calculated with the Q-functions at a state s. All of the convergences rules provided by Watkins and Dayan [72] are valid here too as long as there is a unique solution to the game and the equilibrium is a unique point. For example, Nash equilibrium does not have unique values [20], which will make Equation 7.6 to change it's valued whenever iteration is performed because Nash results with a different value, so Equation 7.6 is not able to converge. Although Watkins and Dayan proved convergence of Q-learning under some regulations, the Nash Equilibrium, which is used to calculate the Q values in Markov Games is not unique. Therefore, in order to continue the calculations we need to either set up some constraints for uniqueness, so we can apply the Q-learning method, or to create a rule of how the agent will select an equilibrium in non cooperative

games and thereafter that thereafter how the algorithm can handle many Nash equilibriums and still the Q-learning to be able to converge under the conditions provided in [72]. It will be beneficial to mention that a Nash equilibrium is unique if it exists in strictly dominant strategies. Another possibility is if we can prove that the best-reply has only one fixed point. If the game is such that the best-reply links (which maps each vector of actions into another vector of best replies to those actions) are ordinary functions and not set-valued, then we have to identify the set of fixed-points of that function and show that it is a singleton.

In [107] Moulin suggested under what conditions the Nash equilibrium in noncooperative games could exist and be unique. The mentioned assumptions in the paper are that the game is played all of the time and that each of the players uses his best strategy tomorrow to his opponent strategy today, the assumption is called Cournot-tatonnement(CT) and the game Cournot-stable (C-stable). Let us denote a game $G = (w_i, R_i), i \in N$ where N is a finite set of agents and w_i is the strategy set with R_i utility function [107].

DEFINITION 7.3.1 *The game G is C-stable if there exist an unique outcome x^* such that:*
(i) for all $x^(0) \in X$, $\lim_{t \to +\infty} x^t = x^*$, where (x^t) is a CT starting at x^0;
(ii) for all $R \in \Re(x^*)$ there exists $R' \in \Re(x^*)$ such that for all $x^0 \in Re'$, the CT starting at x^0 remains within $R : x^t \in R$ for all t. Then x^* is unique Nash equilibrium solution for G and is also Cournot-stable.

The author provided many situations under which we can obtain a unique solution to the game. The Q-learning Algorithm 2 for the games set up.

The value of $V^{equilibrium}(s_j)$ is the equilibrium value, depending on the type of the equilibrium that we decide to employ. For MinMax Equilibrium [20], we will have:

$$V(s) = \max_{\pi_j \in PD(A)} \min_{a_j^a \in A^a} \sum_{a_j \in A^d} \pi(s, a_j) q(s, \langle \alpha_j^a, \alpha_j^d \rangle) \tag{7.7}$$

where

$$\pi(s) \to \arg \max_{\pi_j \in PD(A)} \min_{a_j^a \in A^a} \sum_{a_j \in A^d} \pi(s, a_j) q(s, \langle \alpha_j^a, \alpha_j^d \rangle) \tag{7.8}$$

The convergence of the Q-learning algorithm is proved by Watkins [72], and the same rules are valid here as long as there is a unique equilibrium as we mentioned before. The Game Q-learning convergence for Friend and Foe is proved by Littman [20]. In our case, the algorithm assumes

Algorithm 2 Game Q-learning approach

– Initialize $q(s, \langle \alpha^d, \alpha^a \rangle)$, for every $s \in S$, $\alpha^a \in A^a, \alpha^d \in A^d$, randomly and $q(terminal - state) = 0$ Renew **for** every episode:

  – initiate $s_i \in S$

    Renew for every episode:

    * Select a from s, using ε-greedy policy derived from Q.
    * Exert action a_i^d
      · receive $R_{s_i}^{a_i}$
      · review the new state s_j
      · review action of the opponent α^a
    * $q(s_i, \langle \alpha_i^a, \alpha_i^d \rangle) \leftarrow q(s_i, \langle \alpha_i^a, \alpha_i^d \rangle) + \lambda[R + \gamma V^{equilibrium}(s_j) - q(s_i, \langle \alpha_i^a, \alpha_i^d \rangle)]$

  – until s is terminal

– end **for**

that the opponent will be only a Foe as that is the most often case in network security. Typically, this method will work in cases with a guiding tool where we can observe just one equilibrium point, depending on the consideration for different types of games. A guiding tool could be used to coordinate the agent, depending on the type of the game as it is provided in Table 9. The research of the convergence of Q-learning Nash provides some ideas on the challenge of designing a reliable algorithm which can operate accurately in all varieties of games. Therefore we can create a composite algorithm that first analyzes the type of the game being played, depending on the information structure and based on that to employ the Game Q-learning algorithm. The field of Game theories, especially its applicability to network security, is still extensive with several interesting problems to be clarified.

7.4 Concluding Remarks and Contribution

Game theories approach in Network security set up is an additional idea that can be implemented for Intrusion Detection and Prevention Domains. If we do not observe a zero-sum game, then additional analysis of the attackers' behavior needs to be done. The current work is more concentrated on how to protect the system with the informational structure that we possess as administrators. Therefore it would be beneficial if we can analyze different attackers behaviors for the purpose of the Game theories set up, so we can apply the Q-learning approach. As we have mentioned in the Literature Review Game Theory section, there is an abundant amount of research done on predicting the risk of being attacked and the loss function of the attacker. This work may be collaborated with the of game theories idea in the IDS domain to perform additional decision framework for the Q-learning approach. The application of the Reinforcement learning in IDS domain is something barely investigated and analyzed in the prior research, although it can give us a strategical and advanced tool for protecting our systems. The whole algorithmic structure that we provided and the idea of classifying the games according to their informational structure, so an appropriate equilibrium to be selected in the Q-learning set up is something that will significantly improve the procedures, employed by each administrator concerned with the protection of the system.

Chapter 8

Summary and Future Research

Over the years the network security reached essential to our society levels, that nobody can doubt. Each one of us is affected, and we are responsible as a society to continually develop and explore new methods and techniques to defend our systems and set up rules and laws in cyberspace. The value of collaboration among IDS and IPS is increased if both technologies work simultaneously to provide a stable protection position. The goal of this dissertation is to outline a supervised and unsupervised learning techniques for IDS, and Reinforcement learning for IPS, where there is a direct link between both systems, so the reader can overview a technology that will improve our protection structure.

In Chapter 3, we provide a NASCA procedure, that is not adapted to streaming data, but it gives valuable knowledge about the classification methodologies that could be employed and infer a possible traffic division in the classification step of the analysis. The monitoring and detecting functions of the IDS can be performed with the NASCA procedure. The prevention and response functions of IPS, however, could be achieved with the Q-learning algorithm. Chapters 3 and Chapter 4 outline the main two approaches that we will employ, so we can create a full protection schema, which could be implemented in the security system. In Chapter 5 we suggest a way of adaptation of the NASCA procedure to streaming data or the NASCA Drift procedure, we also propose a new drift detection method, based on the Bayesian inference, which can be applied in any data stream set up. Chapter 6 provides an outline of the proposed two layers schema, with a full logistic description of the traffic flow. It introduces the idea for separation of ”good” from ”bad” traffic and also explains the links between the Intrusion Detection and Intrusion Prevention domains. In Chapter 7, we demonstrate in algorithmic form how we can apply the Q-learning approach, simultaneously with the Game theory idea, where the attacker and the defender are playing a game that has a solution, and this solution is an equilibrium point. Some of the future work that

can be accomplished to improve the proposed two-layers union between IDS and IPS could be related to the online Q-learning adaption. It is an incremental learning method so that it can be adapted to real-time learning. There is also a need for an additional investigation on the convergence properties for the Q-learning algorithm in the Game theories set up as well as analysis of the attacker's behavior. It would be interesting to combine the currently proposed algorithm with the idea for vulnerabilities exploration and exploitation that could serve as a prediction tool for the attacker's response function in the Game Theories environment. It would also be beneficial to relate the current two-layer proposed schema to the idea for a Next Generation Firewall, which will accommodate hardware set up for the proposed software protection schema. Therefore, the system will be able to identify and prevent advanced breaches by executing policies in application and protocol levels.

References

[1] A. Agah, S. Das, K. Basu, and M. Asadi, "Intrusion detection in sensor networks: A non-cooperative game approach", *3rd IEEE International Symposium on Network Computing and Applications*, pp. 343-346, 2004.

[2] T. Alpcan and L. Pavel, "Nash Equilibrium design and optimization", *International Conference on Game Theory for Networks, GameNets*, 2009.

[3] T.Alpcan and T.Basar, "An intrusion detection game with limited observations", *12th Int. Symp. on Dynamic Games and Applications, Sophia Antipolis*, France, (2006).

[4] T. Alpcan, and T. Basar, **"Network security a decision and game-theoretic approach"**, *Cambridge University press*, 2011.

[5] D.Bertsekas, **"Dynamic programming and optimal control"**, 2nd ed. *Belmont, MA: Athena Scientific*, 2001.

[6] M. Bloem, T. Alpcan, and T. Basar, "Intrusion response as a resource allocation problem", *IEEE Conference on Descision and Control*, 2006.

[7] P. Bommannavar, T. Alpan and N. Bambos, "Security Risk Management via Dynamic Games with Learning", *IEEE International Conference on Communications*, pp. 1-6, 2011.

[8] A. Burke, "Towards a game theoretic model of information warfare", Masters thesis, Air Force Institute of Technology, Air University, 1999.

[9] J. Filar and K. Vrieze, **"Competitive Markov decision processes"**, *Springer, Berlin Heidelberg*, New York, 1996.

[10] A. Ghose, and K. Hausken, "A Strategic Analysis of Information Sharing Among Cyber Attackers", *Social Science Research Network*, 2007.

[11] L. Hausken, H.R. Rao, S.J. Upadhyaya "Security Investment, Resource Allocation and Information Sharing for Strategic Defenders and Attackers of Information Assets and Networks", *Annals of Emerging Research in Information Assurance, Security and Privacy Services, A Handbook in Information Systems*, Elsevier, Forthcoming, 2008.

[12] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks", *Computer and Security*, 2005.

[13] D. Han, D. Niyato, W. Saad, T. Baar, and A. Hjorungnes, **"Game Theory in Wireless and Communication Networks: Theory, Models, and Applications"**,Cambridge University Press, 2011.

[14] A.J. Hoffman,R.M. Karp, "Nonterminating Stochastic Games",*Management Sciences (Series A)*,12, pp.359-370, 1996.

[15] E.O. Ibidunmoye ,B.K. Alese , O.S. Ogundele, "A Game-theoretic Scenario for Modeling the Attacker-Defender Interaction", *Computer Engineering Information Technology*, 2013.

[16] M. Kjaerland, "A taxonomy and comparison of computer security incidents from the commercial and government sectors", *Computers and Security*, 25, pp.522-538, 2005.

[17] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe, "Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness", *Journal of Artificial Intelligence Research*, 41, pp.297-327, 2011.

[18] G. Leveson, "Fault Tree Analysis in Safeware", *Addison-Wesley*, pp.317-326, 1995.

[19] Y. Lin, Y. Wang, Y. Wang, and H. Zhu, "Stochastic Game Nets and Applications in Network Security", *Journal of Computers*, pp.461-467, 2009.

[20] M. L. Littman. Markov games as a framework for multiagent reinforcement learning, *Proc. of the 11th International Conference on Machine Learning*, pp.157-163, 1994.

[21] Y. Liu, C. Comaniciu, and H. Man, "A Bayesian game approach for intrusion detection in wireless ad hoc networks", *Proc. 2006 workshop on Game theory for communications and networks*, 2006.

[22] Y. Luo, F. Szidarovszky, Y. Al-Nashif, and S. Hariri, "Game Theory Based Network Security", *Journal of Information Security*, 1, pp.41-44, 2010.

[23] K.W. Lye and J. Wing, "Game strategies in network security", *Foundations of Computer Security Workshop in FLoC 02*, Copenhagen, Denmark, 2002.

[24] D. McMorrow, **"Science of Cyber-Security"**, *MITRE Corporation report*, 2010.

[25] R. Meadows, ACA "COBIT 5 for Information Security", *COBIT 5 for Information Security*, IL, 23, pp.55-59, 2012.

[26] A. Miura-Ko, B. Yolken, N. Bambos, and J. Mitchell, "Security investment games of interdependent organizations",*Proceedings of the 46th Allerton Conference*, 2008.

[27] C. Nguyen, T. Alpcan, and T. Basar. "Security games with incomplete information", *Proc. of IEEE Intl. Conf. on Communications (ICC)*, 2009.

[28] C. Nguyen, T. Alpcan, and T.Basar, "Stochastic Games for Security in Networks with Interdependent Nodes", *IEEE*, pp.697-703, 2009.

[29] G. Owen, **"Game Theory"**, *Academic Press*, 3rd edition, 2001.

[30] A. Patcha and J. Park, "A game theoretic approach to modeling intrusion detection in mobile ad hoc networks", *Proc. 2004 IEEE workshop on Information , Assurance and Security*, pp.280 - 284, 2004.

[31] Y. Patil, P. Zavarsky, D. Lindskog and R. Ruhl, "Fault Tree Analysis of Accidental Insider Security Events", *International Conference on Cyber Security*, Washington D.C., 2012.

[32] K. M. Ramachandran, and C. P. Tsokos, **"Stochastic differential games: Theory and Applications"**, Atlantis Studies in Probability and Statistics, Volume 2, *Atlantis/Springer Press*, 2012.

[33] K. Sallhammar, B. E. Helvik and S. J. Knapskog, "Towards a Stochastic Model for Integrated Security and Dependability Evaluation", *1st International Conference on Availability, Reliability and Security*, Washington, 2006.

[34] K. Sallhammar, S. Knapskog, and B. Helvik, "Using stochastic game theory to compute the expected behavior of attackers", *Proc. 2005, International Symposiu on Applications and the Internet Workshops*, pp.102-105, 2005.

[35] L. Shapley, "Stochastic games", *Proc. National Academy of Science USA*, Vol 39, Issue 10, pp.1095-1100, 2007.

[36] S. Shiva, R. Sankardas, H. Bedi, D. Dasgupta, Q. Wu, "A Stochastic Game Model with Imperfect Information in Cyber Security", *Computational Intelligence in Cyber Security CICS*, pp.129-136, 2011.

[37] J.von Neuman "Zur Theorie der Gesellschaftsspiele", *Math. Ann.*,100, pp.295-332, 1928.

[38] C. Xiaolin, T. Xiaobin, Z. Yong, and X. Hongsheng, "A markov game theory-based risk assessment model for network information systems", *International conference on computer science and software engineering*, pp.1057-1061, 2008.

[39] X. You and Z. Shiyong, "A kind of network security behavior model based on game theory", *Proc. Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp.950-954, 2003.

[40] Bahl, Shilpa, and Sudhir Kumar Sharma, "Improving Classification Accuracy of Intrusion Detection System Using Feature Subset Selection", *2015 Fifth International Conference on Advanced Computing; Communication Technologies*, 2015.

[41] L. Breiman, "Machine Learning", *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[42] P.Buehlmann, and B. Yu. Analyzing Bagging, *The Annals of Statistics*, vol. 30, no. 4, pp. 927-961, 2002.

[43] C. Lakshmi Devasena. Effectiveness Evaluation of Rule Based Classifiers for the Classification of Iris Data Set. *Bonfring International Journal of Man Machine Interface*, vol. 1, no. 1, pp. 5-9, 2012

[44] J. Cannady, Distributed Detection of Attacks in Mobile Ad Hoc Networks Using Learning Vector Quantization, *2009 Third International Conference on Network and System Security*, 2009.

[45] G. Foreman, Choose Your Words Carefully: An Empirical Study of Feature Selection Metrics for Text Classification., *Principles of Data Mining and Knowledge Discovery Lecture Notes in Computer Science*, pp. 150-162, 2002.

[46] E. Frank, and I. H. Witten. Generating Accurate Rule Sets Without Global Optimization *Proceeding ICML '98 Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 144-151, 1998.

[47] E. Frank, et al. Machine Learning,*Machine Learning*, vol. 32, no. 1, pp. 63-76, 1998.

[48] I.Guyon, and A. Elisseeff. An Introduction to Feature Extraction, *Feature Extraction Studies in Fuzziness and Soft Computing*, pp. 1-25, 1998.

[49] M. Hall, and G. Holmes. Benchmarking Attribute Selection Techniques for Discrete Class Data Mining, *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1437-1447, 2003.

[50] J. Thorsten. Text Classification, *Learning to Classify Text Using Support Vector Machines*, pp. 7-33, 2002.

[51] J.H. George, and P.Langley, Estimating Continuous Distributions in Bayesian Classifiers, *Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 338-345, 1995.

[52] K. Kenji, and L. A. Rendell, A Practical Approach to Feature Selection, *Machine Learning Proceedings 1992,* pp. 249-256, 1992.

[53] K. Ron, and G. H. John. Wrappers for Feature Subset Selection, *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273-324, 1997.

[54] H. Liu, and R. Setiono, Feature Selection via Discretization, *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 4, pp. 642-645, 1997.

[55] D.T. Nguyen. et al. A Reconfigurable Architecture for Network Intrusion Detection Using Principal Component Analysis,*Proceedings of the Internation Symposium on Field Programmable Gate Arrays - FPGA'06*, 2006.

[56] J. Nziga, and J. Cannady. Minimal Dataset for Network Intrusion Detection Systems via MID-PCA: A Hybrid Approach, *2012 6th IEEE INTERNATIONAL CONFERENCE INTELLIGENT SYSTEMS*, 2012.

[57] Y. Rong, and Zheng. Classification and Regression Trees, Random Forest Algorithm, *Machine Learning Approaches to Bioinformatics Science, Engineering, and Biology Informatics*, pp. 120-132, 2010.

[58] S.K. Sahu, A Study of K-Means and C-Means Clustering Algorithms for Intrusion Detection Product Development,*International Journal of Innovation, Management and Technology IJIMT*, 2014.

[59] Tavallaee, et al. A Detailed Analysis of the KDD CUP 99 Data Set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009.

[60] V. N. Vapnik, Constructing Learning Algorithms," *The Nature of Statistical Learning Theory*, pp.119-166, 1995.

[61] L. Wohlrab, and J. Frnkranz. A Review and Comparison of Strategies for Handling Missing Values in Separate-and-Conquer Rule Learning, *Journal of Intelligent Information Systems*, vol. 36, no. 1, pp. 73-98, 2010.

[62] E. Even-Dar and Y. Mansour, Learning Rates for Q-Learning, *Lecture Notes in Computer Science Computational Learning Theory*, pp. 589-604, 2001.

[63] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, An analysis of reinforcement learning with function approximation, *Proceedings of the 25th international conference on Machine learning - ICML '08*, 2008.

[64] H. Maei, C. Szepesvari, S. Bhatnagar, D. Silver, D. Precup, and R. Sutton, Convergent temporal-difference learning with arbitrary smooth function approximation, *NIPS-22*, pp. 1204-1212.

[65] ISCX NSL - KDD Data Set, University of New Brunswick est.1785. [Online]. Available: http://www.unb.ca/cic/datasets/index.html.

[66] J. Cannady, Applying CMAC-based online learning to intrusion detection, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 5, pp. 405-410, 2000.

[67] J. Cannady, Next Generation Intrusion Detection: Autonomous Reinforcement Learning of Network Attacks, *In Proceedings of the 23rd National Information Systems Secuity Conference*, pp. 1-12, 2000.

[68] J. Fu and U. Topcu, Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints, *Robotics: Science and Systems X*, 2014.

[69] J. N. Tsitsiklis, Asynchronous stochastic approximation and Q-learning, *Machine Learning*, vol. 16, no. 3, pp. 185-202, 1994.

[70] KDD Cup 1999 Data. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[71] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009.

[72] P. Dayan and C. Watkins, Q-learning, *Machine Learning*, vol. 8, no. 3-4, pp. 279-292, 1992.

[73] P. Laskov, K. Rieck, P. Dussel, and C. Schafer, Learning Intrusion Detection: Supervised or Unsupervised?, *Proceedings of the 13th ICIAP Conference*, pp. 50-57, 2005.

[74] P. Miller and A. Inoue, Collaborative intrusion detection system, *22nd International Conference of the North American Fuzzy Information Processing Society*, NAFIPS, pp. 519-524, 2003.

[75] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* s.l.: MIT Press, 1998.

[76] V. Chandola, A. Banerjee, and V. Kumar, Anomaly detection, *ACM Computing Surveys*, vol. 41, no. 3, pp. 1-58, 2009.

[77] VNI Global Fixed and Mobile Internet Traffic Forecasts, Cisco, 13-Feb-2018. [Online]. Available: http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html.

[78] X. Xu and T. Xie, A Reinforcement Learning Approach for Host-Based Intrusion Detection Using Sequences of System Calls, *Lecture Notes in Computer Science Advances in Intelligent Computing*, pp. 995-1003, 2005.

[79] X. Xu and Y. Luo, A Kernel-Based Reinforcement Learning Approach to Dynamic Behavior Modeling of Intrusion Detection, *Lecture Notes in Computer Science, Proceedings of ISNN*, pp. 455-464, 2007.

[80] X. Xu, T. Xie, D. Hu, and X. Lu, Kernel least-squares temporal difference learning, *International Journal of Information Technology*, vol. 11, no. 9, pp. 54-63, 2005.

[81] K. Ramachandran, and Z. Stefanova. Dynamic Game Theories in Cyber Security. *Proceedings of Dynamic Systems and Applications* , vol. 7, pp. 303-310, 2016.

[82] Z. Stefanova, and K. Ramachandran. Network Attribute Selection, Classification and Accuracy (NASCA) Procedure for Intrusion Detection Systems. *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, 2017.

[83] Z. Stefanova , K. Ramachandran, "Off-Policy Q-learning Technique for Intrusion Response in Network Security," *World Academy of Science, Engineering and Technology, International Science Index 136, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, pp. 262 - 268, 2018.

[84] G. Dong, J. Han, L.V.S. Lakshmanan, J. Pei, H. Wang, P.S. Yu, "Online mining of changes from data streams: research problems and preliminary results," *Proc. of the 2003 ACM SIGMOD Workshop on Management and Processing of Data Streams*, 2003.

[85] E. Cohen, M. Strauss, ”Maintaining time-decaying stream aggregates,” *Journal of Algorithms*, vol.59, Issue 1, pp. 19-36, 2006.

[86] R. Klinkenberg, T. Joachims, ”Detecting concept drift with support vector machines,” *Proc. of the 17th Int. Conf. on Machine Learning*, pp.487-494, 2000.

[87] J. Gama, P. Medas, G. Castillo, P. Rodrigues, ”Learning with drift detection,” *Lecture Notes in Computer Science*, pp.31-71, 2004.

[88] R. Klinkenberg and I. Renz.”Adaptive information filtering: Learning in the presence of concept drifts in learning for text Categorization”, *AAAIPress*, pp. 33-40, 1998.

[89] R. Klinkenberg and T. Joachims, ”Detecting concept drift with support vector machines,” *Proceedings of ICML-00, 17th International Conference on Machine Learning,* pages 487-494, 2000.

[90] R. Klinkenberg, ”Learning drifting concepts: Example selection vs. example weighting,” *Intelligent Data Analysis*, 2004.

[91] G. Widmer and M. Kubat, ”Learning in the presence of concept drift and hidden contexts,” *Machine Learning*, 23, pp.69-101, 1996.

[92] C. Lanquillon, ”Enhancing Text Classification to Improve Information Filtering,” *PhD thesis, University of Madgdeburg, Germany*, 2001.

[93] M. Maloof and R. Michalski, ”Selecting examples for partial memory learning,”, *Machine Learning,* 41, pp. 27-52, 2000.

[94] T. Mitchell, **”Machine Learning,”** *McGraw Hill*, 1997.

[95] Gama, Joao, et al. Issues in Evaluation of Stream Learning Algorithms, *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD ’09*, 2009.

[96] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, 1988.

[97] Gomes, M. Heitor, et al. Adaptive Random Forests for Evolving Data Stream Classification, *Machine Learning*, vol. 106, no. 9-10, pp. 1469-1495, 2017.

[98] N.C. Oza, Online Bagging and Boosting, *2005 IEEE International Conference on Systems, Man and Cybernetics*, 2005.

[99] P. Domingos and G. Hulten, Mining High-Speed Data Streams, *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '00*, 2000.

[100] Hoeffding, Wassily. Probability Inequalities for Sums of Bounded Random Variables. Journal of the American Statistical Association, vol. 58, no. 301, pp. 13-30, 1963.

[101] L. V. Hedges, Distribution Theory for Glass's Estimator of Effect Size and Related Estimators. *Journal of Educational Statistics*, vol. 6, no.2, p.107, 1981.

[102] Y. Yang and D. Dunson. Sequential Markov Chain Monte Carlo. ArXiv e-Prints, 2013.

[103] A. Pawling et al. Computing Information Gain in Data Streams, *IEEE ICDM Workshop on Temporal Data Mining*, 2005.

[104] A. Bifet, et al. MOA: Massive Online Analysis, *Journal of Machine Learning Research*, vol. 11, pp. 1601-1604, 2010.

[105] A. Bifet, and R. Gavalda, Learning from Time-Changing Data with Adaptive Windowing, *Proceedings of the 2007 SIAM International Conference on Data Mining*,pp. 443-448, 2007.

[106] G. J. Ross, et al. Exponentially Weighted Moving Average Charts for Detecting Concept Drift.*Pattern Recognition Letters*, vol. 33, no. 2, pp. 191-198, 2012.

[107] H. Moulin, Dominance Solvability and Cournot Stability, *Mathematical Social Sciences,* vol. 7, no. 1, pp. 83-102, 1984.

Appendix A
Copyright Notices