International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:12, No:4, 2018

Off-Policy Q-learning Technique for Intrusion Response in Network Security

Zheni S. Stefanova, Kandethody M. Ramachandran

***Abstract*—**With the increasing dependency on our computer devices, we face the necessity of adequate, efficient and effective mechanisms, for protecting our network. There are two main problems that Intrusion Detection Systems (IDS) attempt to solve. 1) To detect the attack, by analyzing the incoming traffic and inspect the network (intrusion detection). 2) To produce a prompt response when the attack occurs (intrusion prevention). It is critical creating an Intrusion detection model that will detect a breach in the system on time and also challenging making it provide an automatic and with an acceptable delay response at every single stage of the monitoring process. We cannot afford to adopt security measures with a high exploiting computational power, and we are not able to accept a mechanism that will react with a delay. In this paper, we will propose an intrusion response mechanism that is based on artificial intelligence, 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. The goal is to find an optimal policy, which will represent the intrusion response, therefore, to solve 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 IDS.

***Keywords*—**Intrusion prevention, network security, optimal policy, Q-learning.

I. Introduction

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 [16], 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 Cyber security are progressing rapidly, it is nearly impossible to have a completely secure system. The IDS observe the network traffic, analyze it and identifies 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 paper 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 there is no attack occurring. On the other hand, although the flowing traffic is examined continuously, once the attack is detected, there is a significant amount of time needed for the IDS to respond. 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 problem is related to the reliability of the protecting 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.

Z. S. Stefanova is with the Department of Mathematics and Statistics, University of South Florida, Tampa, Fl 33620 USA (e-mail: stefanova@mail.usf.edu).

K. M. Ramachandran is with the Department of Mathematics and Statistics, University of South Florida, Tampa, Fl 33620 USA (e-mail: ram@usf.edu).

II. Related Work

Many of the IDS research publications can be summarized as machine learning classification problems, which are solved with supervised or semi-supervised learning models [15]. Although some authors attempted to implement unsupervised learning, they achieved low accuracy [12]. RL has been widely employed in computer network disciplines for research purposes, 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, admission control and quality of service mechanisms. This attentiveness may be due to the fact that RL is reasonable for control situations, where a response from the environment exists. In all the occurrences mentioned above, we detect feedback, which is represented as a reward. Xu et al. [17] 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. A linear function approximation and a temporal difference algorithm were applied to update the value function. The results that they obtained, using the same training and testing sets, were superlative compared to other machine learning methods. Two years later, Xu and Luo [18] modeled the network behavior with a temporal-difference approach. In this work, they achieved even higher detection accuracy, compared to a prior implementation using HMM methods. To approximate the value function and to perform feature selection, they used a sparse kernel least-squares temporal-difference algorithm (LS-TD) [19]. Xu and Luo provided empirical results on host-based intrusion detection to prove the quality of the proposed method; they used system

calls traces from the send mail application. As illustrated by the researchers, the kernel-based LS-TD algorithm is a non-linear function that estimates a high-dimensional feature space. Miller and Inoue [13] used a model called Perceptual Intrusion Detection System with Reinforcement (SPIDeR) which consists of heterogeneous agents. A single agent can employ a self-organizing map to detect malicious activities, and there is a blackboard mechanism for the aggregation of results generated from all agents. Once a signal is detected within the system, it is distributed to all agents for a collective group analysis. They send votes to the central blackboard system, which computes weights, and it rewards the agents depending on their performance. Cannady, in two of his works [5] and [6], applied neural networks approach to obtain a feasible solution. However, this methodology was incapable of adapting to streaming data. The data were necessary to be taken off-line and retrained with a new set of representative data. Cannady applied a Cerebellar Model Articulation Controller Neural Network, to resolve this problem, which has the ability for an online-learning. He suggested a three-layer feed-forward mechanism, intended to generate a series of input-output mappings. In this research, the single IDS-agent learns how to detect flood-based Denial of Service attack based on Internet Control Message Protocol (ICMP) and user datagram protocol (UDP). The system initially learns how to identify ICMP breaches and using prior experience and reoccurring training it learns how to recognize new attacks based on the UDP protocol. One approach employed to find intrusions on host-based IDS is based on analyzing sequences of system calls. The states are defined by a short sequence of system calls in a single trace.

III. Network Environment Represented as Reinforcement Learning Process

A. Reinforcement Learning

Let us assume that there is one decision maker in the IDS and he regularly interacts with his environment. 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.

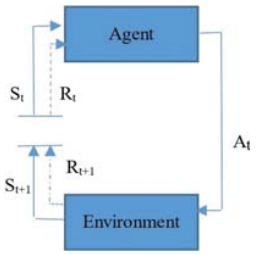

Fig. 1 Reinforcement Learning Process

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:

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

Thus, the current state captures all information from the history, and once the current state is known, it may be considered as a sufficient statistic to decide for the future. The MDP is characterized by the tuple $\langle S, A, R, T, \gamma \rangle$, where:

- 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 number of states will depend on the number of 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 transition 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 [14], 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.

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 immediate rewards at time t, discounted back to the present moment.

$$G_t = R^a_{t+1} + \gamma R^a_{t+2} + \gamma^2 R^a_{t+3} \dots = \sum_{k=0}^{\infty} \gamma^k R^a_{t+k+1} \qquad (1)$$

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 defined as the aggregate value of the expected total return in (1) beginning from state s.

$$v^a(s) = E[G_t^a | S_t = s_t]$$

The value function may be decomposed into two parts: immediate reward and discounted successor state $\gamma v^a(s_{t+1})$. This way we can obtain the Bellman equation:

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

or this equation can be represented as:

$$v^a(s_i) = R_{s_i}^a + \gamma \sum_{s_j \in S} T_{ij}^a v(s_j) \quad (2)$$

B. 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 [14].

$$\pi(a|s) = P(A_t = a | S_t = s) \quad (3)$$

The policy describes the behavior of the agent and it's like his model. MDP policy depends on the current state and not on the past information, i.e. it's stationary and not dependent on time. Given an MDP $\langle S, A, T, R, \gamma \rangle$ and $\pi(a|s)$, the state sequence is a Markov process $\langle S, ^{\pi} \rangle$, the state and reward sequence $S_1, R_2, S_2 \ldots$ is a Markov Reward Process $\langle S, T^{\pi}, R^{\pi}, \gamma \rangle$. In the MDP besides the state value function, we can also define 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 optimal state-value function $v(s)$ will be the maximum value of the function across all policies:

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

It specifies the best possible performance in the MDP. The solution of the MDP is the optimal value function. The optimal action-value function $q(s, a)$ will be the maximum value of the function across all policies:

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

The 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:

$$\pi^*(a|s) = \begin{cases} 1 & , if \arg\max_{a \in A} q^*(s, a) \\ 0 & , otherwise \end{cases} \quad (4)$$

There is always a deterministic optimal policy for any MDP and if we know $q(s, a)$, we can always find the optimal policy. Optimal Bellman Equation for the value function is:

$$v^*(s_i) = \max_a R_{s_i}^a + \gamma \sum_{s_j \in S} T_{ij}^a v^*(s_j) \quad (5)$$

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

$$q^*(s_i, a) = R_{s_i}^a + \gamma \sum_{s_j \in S} T_{ij}^a \max_{a_j} q(s_j, a_j) \quad (6)$$

The Bellman Optimality Equation doesn't have any closed solution form. In general, the following methods are usually used: Value iteration, Policy iteration, Q-learning, and Sarsa. In this paper, we will use Q-learning technique. The action-value function estimates the expected utility of taking action a in state s. It is the best expected sum of future rewards.

Reinforcement learning can be employed to discover an optimal action-selection policy [14] for a finite MDP. Moreover, it learns an action-value function that finally provides the expected value of following an optimal policy and taking a specific action in a given state. A history of an agent is a sequence of <state, action, reward>.The optimal policy can be obtained by preferring the action that provides at each state a maximum value. This learning method is also capable to evaluate the expected value, calculated by all possible actions, without any environment model.

C. 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.

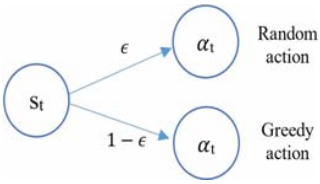

Fig. 2 ε-greedy action selection

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

The idea is to ensure continuous exploration. All actions m are considered with non-zero probability.

D. Q-Learning Algorithm

The advantage of Q-learning is the notion that it is a model-free procedure and an off-policy learning concept. Moreover, the agent contemplates his succeeding move, based on the expected utility of selecting each action in a particular state. Subsequently, he updates towards a bootstrap estimate of the actual return. At every stage, the succeeding 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 of the corresponding action in the current state. A learning rate - $\alpha, (0 < \alpha \leq 1)$, which is associated with that change will assist us to formulate an updating rule.

Let us consider an off-policy learning of action-values $q(s, a)$ and next action is chosen 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 both behavior and target policies to improve, the target policy π is greedy w.r.t. $q(s, a)$ and $\pi(s_j) = argmax_{a_j} q(s_j, a_j)$. However the behavior policy μ is ε-greedy w.r.t. $q(s, a)$. Therefore the Q-learning control equation is:

$$q(s_j, a) \leftarrow q(s_i, a_i) + \alpha[R_{s_j}^{a} + \gamma \max_{a_j} q(s_j, a_j) - q(s_i, a_i)] \quad (7)$$

Q-learning control converges to the optimal action-value function $q(s, a) \rightarrow q^*(s, a)$. Proof is provided by Watkins and Dayan [11] and additionally by Tsitsiklis [8]. An interesting problem is also it's convergence properties. A convergence result is provided by Melo et al. [2] who proved convergence under some restrictions on the sample distribution. Maei et al. [3] 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 [1] 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 $1/(1-\gamma)$.

IV. Results

A. Data Description

The Transmission Control Protocol (TCP) packets in a network allow establishing a connection, where data streams are exchanged between two IP address sources at a specified time and following determined rules [20]. The data set employed in this paper is ISCX NSL-KDD Data Set [4], which is a revised version of KDD CUP 99, DARPA [9], administered by MIT Lincoln Labs. Lincoln Labs generated a typical United States Air Force network in an experimental setup for nine weeks so that they can extract raw TCP data. They administered a local-area network imitating a real Air Force environment and simulated various types of attacks. The significance of the DARPA and KDD dataset is notable; however, many authors questioned the extent to which the data reflect the reality [10]. In our 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. There are 42 variables and one of them represents the condition of the network, labeled as either normal or as one of the 24 different variations of attacks.

B. Problem Setup

In order 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 to 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 paper 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 $R : S \times A \rightarrow R$, 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:

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

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 alteration from one state to another. For the Q-learning technique, we don't 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. We will bootstrap the data sequences following the conditional distributions of states estimated from the original one and apply first maximum likelihood estimation (MLE) on bootstrapped data sequences. 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 [7]. We can also calculate the 95% confidence intervals and to report an error rate, based on the data set. 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 $\pi : S \to A$, 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.

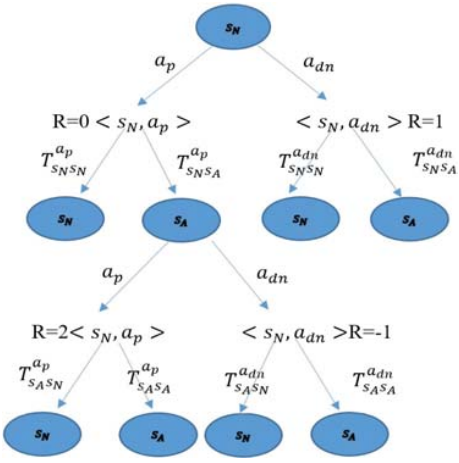

Fig. 3 Example for a Decision Path of the Agent

C. Calculation of the Q-Value and Optimal Policy Selection, Using Q-Learning

The action-value function gives the expected utility of taking a given action in a given state and following an optimal policy thereafter. Q is a $[S, A]$ matrix, in our case it's 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.

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

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

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.

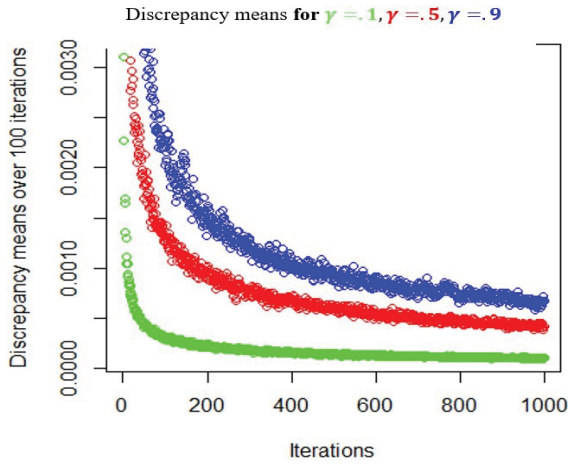

Fig. 4 Discrepancy Means

D. 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 (5) 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 (5).

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 (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 I, 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 I
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.

V. Conclusion and Future Work

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 is able to perform the analysis without any model or knowledge on 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 useful utilization of Q-learning is helpful not only because of the effective results obtained with the model but also because of its potential combination with other models, that could benefit and improve with the assistance of Q-learning.

References

[1] E. Even-Dar and Y. Mansour, Learning Rates for Q-Learning, *Lecture Notes in Computer Science Computational Learning Theory*, pp. 589-604, 2001.
[2] 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.
[3] 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.
[4] ISCX NSL - KDD Data Set, University of New Brunswick est.1785. (Online). Available: http://www.unb.ca/cic/datasets/index.html.
[5] 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, Jul. 2000.
[6] 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.
[7] J. Fu and U. Topcu, Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints, *Robotics: Science and Systems X*, 2014.
[8] J. N. Tsitsiklis, Asynchronous stochastic approximation and Q-learning, *Machine Learning*, vol. 16, no. 3, pp. 185-202, 1994.
[9] KDD Cup 1999 Data. (Online). Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[10] 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.
[11] P. Dayan and C. Watkins, Q-learning, *Machine Learning*, vol. 8, no. 3-4, pp. 279-292, 1992.
[12] 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.
[13] P. Miller and A. Inoue, Collaborative intrusion detection system, *22nd International Conference of the North American Fuzzy Information Processing Society*, NAFIPS 2003, pp. 519-524.
[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* s.l.: MIT Press, 1998.
[15] V. Chandola, A. Banerjee, and V. Kumar, Anomaly detection, *ACM Computing Surveys*, vol. 41, no. 3, pp. 1-58, 2009.
[16] 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.
[17] 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.

[18] 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.
[19] 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.
[20] 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.

Zheni Stefanova Zheni S Stefanova is a PhD candidate in the Mathematics and Statistics Department at the University of South Florida (USF). Her research interest is in cyber security and precisely statistical machine learning techniques, applied to data mining in network security and software reliability problems with an advisor Professor Kandethody Ramachandran. She is a founder of the American Statistical Association Student Chapter at USF, two times recipient of Tharp Endowed Award 2015 and 2017 and MV Johns Jr. Scholarship 2016.

Kandethody Ramachandran Kandethody M Ramachandran is a Professor of Mathematics and Statistics at the University of South Florida (USF). His research interests are concentrated in the areas of applied probability and statistics. His research publications span a variety of areas such as control of heavy traffic queues, stochastic delay systems, machine learning methods applied to game theory, finance, cyber security, and other areas, software reliability problems, applications of statistical methods to microarray data analysis, and streaming data analysis. He is also, co-author of three books. He is the founding director of the Interdisciplinary Data Sciences Consortium (https://idscbigdata.com/). He is also extensively involved in activities to improve statistics and mathematics education. He is a recipient of the Teaching Incentive Program award at the University of South Florida. He is also the PI of 2 million dollar grant from NSF, and a co-PI of 1.4 million grant from HHMI to improve STEM education at USF.