A Security-Constrained Reinforcement Learning Framework for Software Defined Networks

Anand Mudgerikar
Purdue University
West Lafayette, IN, USA
amudgeri@purdue.edu

Elisa Bertino
Purdue University
West Lafayette, IN, USA
bertino@purdue.edu

Jorge Lobo
ICREA - Universitat
Pompeo Fabra, Spain
jorge.lobo@upf.edu

Dinesh Verma
IBM TJ Watson Research Center
Yorktown Heights, NY, USA
dverma@us.ibm.com

***Abstract*—Reinforcement Learning (RL) is an effective technique for building 'smart' SDN controllers because of its model-free nature and ability to learn policies online without requiring extensive training data. However, as RL agents are geared to maximize functionality and explore the environment without constraints, security can be breached. In this paper, we propose `Jarvis-SDN`, a RL framework that constrains explorations by taking security into account. In `Jarvis-SDN`, the RL agent learns 'intelligent policies' which maximize functionality but not at the cost of security. Standard network flow based attack signatures obtained from intrusion detection system (IDS) datasets cannot be used as policies because they do not conform to the state model of the RL framework and thus have poor accuracy and high false positives. To address such issue, the security policies for constraining explorations in `Jarvis-SDN` are learnt in a semi-supervised manner in the form of 'partial attack signatures' from packet captures of IDS datasets that are then encoded in the objective function of the RL based optimization framework. These signatures are learnt using Deep Q-Networks (DQN). Our analysis shows that DQN based attack signatures perform better than classical machine learning techniques, like decision trees, random forests and deep neural networks (DNN), for common network attacks. We instantiate our framework for a SDN controller with the goal of intelligent rate control to further analyze the effectiveness of the attack signatures.**

***Index Terms*—Security and Safety, Deep Reinforcement Learning, Software Defined Networks**

I. INTRODUCTION

The use of machine learning (ML) techniques in the control plane of software defined networks (SDNs) provides enhanced approaches to traffic engineering, such as maximizing quality of service (QoS). Generally, QoS is determined by the interplay within various network functionalities such as rate control, routing, load balancing, and resource management. This interplay can become very complex. The benefit of ML techniques is that they can model complexity given sufficient representative data to train upon. However, the diversity and scale of current networks together with the diversity of traffic behavior hinder the task of gathering data that captures enough sets of behaviors for training. This poses a challenge to classical ML. Reinforcement Learning (RL), on the other hand, relies on learning optimal policies online based on system state using a model-free approach. These policies are more likely to transfer over to a new environment, and these characteristics make them more suitable for network control. RL based frameworks have thus already been proposed for specific functions within networks, such as for controlling routing [1], traffic rate control [2] and load balancing [3].

Network control solutions require optimization across multiple functionalities, not just a single one. Current uses of RL for network control focus on optimizing a single functionality, which makes these existing solutions difficult to deploy in real networks. For example, learning a policy which maximizes the throughput of the network (functionality 1: optimal routing) can come at the cost of unfair bandwidth consumption by a set of users (functionality 2: per user bandwidth fairness). Perhaps even more critical is the case of security policies. For example, learning a policy which maximizes the throughput of the network (functionality 1: optimal rate control for QoS) can unknowingly facilitate the propagation of a high throughput Denial of Service (DoS) attack.

To address those issues, we propose `Jarvis-SDN`, an adaptation of our constrained RL framework for IoT [4] to SDNs. In `Jarvis-SDN`, a RL based agent using Deep Q-Learning learns optimal policies for a SDN controller to optimize across multiple network functionalities while maintaining security. Examples of such functionalities include optimal rate control (measured by per user throughput), routing optimization (measured by a metric like latency), availability of device resources, or path quality (metrics such as loss rate, or jitter). The basic idea is to define the reward to the agent as a weighted combination of individual functionality performance metrics, including a metric for security behavior of the system. A key challenge in applying our previous framework [4] to SDNs is that it is not obvious how to define performance metrics for security (see [5], [6] for proposals and discussions regarding security metrics). Our approach for quantifying security is to measure the ability to protect against known attacks. We first build offline 'attack signatures' from packet captures of previously seen attacks using different ML techniques: Decision Trees, Random Forests, Deep Neural Networks (DNN) and Deep Q-Networks (DQN). These attack signatures are then used by the RL agent to determine a quality value for the network state depending on the perceived threat a network flow has on the current and near future states of the network.

To summarize, we make the following contributions:

1) A context independent RL framework, `Jarvis-SDN`, constrained by security policies for SDN controllers.
2) A novel approach to build quantifiable security metrics (attack signatures) for network flows using DQN.
3) Extensive evaluation of different ML techniques to build

attack signatures using the CICIDS dataset [7] consisting of network attacks, such as DoS, DDoS, Brute-Force and Web based attacks. We find that DQN based signatures perform better than other ML techniques.

4) Through an instantiation of `Jarvis-SDN` for a SDN controller with the goal of optimal rate control, we show that the RL agent learns desirable behavior for malicious and benign flows.

The rest of the paper is organized as follows. In Section II, we give some background on Deep Q-Learning. Next, in Section III, we formally define the system model and discuss key challenges. In Section IV, we define and analyze our RL based DQN approach for building attack signatures. In Section V, we instantiate the `Jarvis-SDN` framework in a simulated network for optimal rate control and analyze its effectiveness. Finally, we discuss related work in Section VI and in Section VII we conclude and outline directions for future work.

II. Background on Deep Q-Learning

A Reinforcement learning (RL) framework [8] (see Figure 1) is a probabilistic state transition environment where state transitions are caused by actions executed by an agent and every state-action pair (s, a) is assigned a reward value r given by a reward function $R(s, a)$. A RL agent traverses the environment according to a policy π that selects an action to execute in a given state for policy parameters θ and receives the rewards accrued over all the state transitions that happened according to θ. In a Q learning framework, the goal is to find the optimal policy which maximizes the cumulative reward through a function $Q(s, a)$ that estimates the expected cumulative reward the agent will get at the end of an episode if the current state is s, the agent executes a and follows learnt policy π_θ.

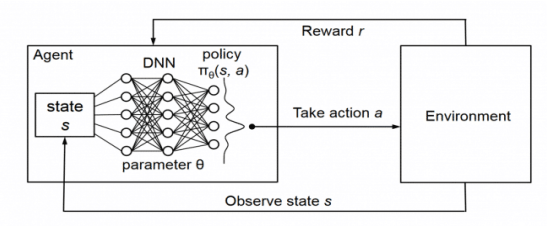

Fig. 1: Deep Q Learning Environment

In a deep Q learning system, a deep neural network, referred to as DQN, is used to determine the optimal Q function using a temporal difference equation defined as follows:

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha[R(s,a) \\ +\gamma Max_{a'}\{Q_t(s',a')\} - Q_{t-1}(s,a)]$$

where Q_t is the current estimation of the Q function, Q_{t-1} is the previous estimation of Q after $t-1$ steps of training. The estimated next state and action are denoted by s' and a', respectively. The learning rate (α) determines to what extent newly acquired information overrides old information. The discount factor (γ) determines the importance of future rewards.

III. System Model and Problem Formulation

The need for intelligent SDN controllers arises in several scenarios. One scenario is the rate control of traffic from different users within a data center or cloud network environment, where the behaviour is controlled by means of a centralized controller, as typical in SDN architectures. The controller needs to dynamically determine how to handle each traffic flow, and determine the per-flow rate limits according to the server capacity and QoS requirements. Another scenario is related to a telecommunications or Internet service provider. The service provider is responsible for accepting packets from connected clients, and route them to the appropriate egress points. The bandwidth rates per user are dynamically determined according to the user subscription service level agreement and to maximize overall network throughput. A third scenario is related to the 5G cellular networks, where network protocols are designed to implement their control and data planes using SDN technologies. In these cases, the control plane needs to determine the rate of traffic from a cellular device, depending on various factors like user channel quality, whether a mobile edge computing server is being leveraged, or packets are being forwarded to an alternative location for processing. While the protocols are different than in a data center or the Internet, the operational paradigm is very similar. In most of these environments, the SDN controller needs to take decisions based on the inspection of the headers within the IP network packets. Therefore, we focus on approaches that learn optimal flow rate-control policies based on the inspection of features extracted from network packet headers.

In general terms, the problem we want to address in this paper using ML techniques is the design of an intelligent security-aware SDN controller. The controller is responsible for dealing with network flows from several users. Depending on the amount of active flows, anticipated traffic in the near future, and the properties of the current flows (such as potential attacks), the controller configures the optimal rate limits for each flow. In what follows, we first define the state model underlying our approach. We then formulate our problem and discuss key challenges.

A. System Model

State Model: We define the state of the SDN environment as a tuple $S_t = {s_1}^t, {s_2}^t, ..., {s_n}^t$ of network flows at time t. The SDN controller manages n network flows from η users. In the current implementation, we assume that $n = \eta$ at all times for ease of execution in the simulation. The flow state ${s_i}^t$ of user i at time t is defined in terms of k network flow parameters ${s_i}^t = {P_{1_i}}^t, {P_{2_i}}^t, ...{P_{k_i}}^t$. These parameters can take numeric values, such as packet counts, packet length, packet rate, inter arrival times, or categorical values, such as whether SYN/FIN/ACK flags are set or not, and protocol type.

State Transition Model: We monitor state transitions in the network in terms of "episodes". We define two configuration

parameters for an episode: time period T and interval I. The state transitions occur every I time-units until the timestamp reaches T time units, after which the state is reset to the initial state and marks the end of an episode. An episode basically consists of T/I *time instances* at which the state transitions of the environment are recorded. For example, in our current implementation, for $\{T, I\} = \{60, 1\}$, the episodes are one minute long with state transitions every second.

Action Space: After each interval, the SDN controller can take various actions corresponding to different rules used to manipulate the flow table as specified in the action set of the Openflow protocol [9]. For example, in our current prototype, the actions correspond to allow/drop a percentage of the packets for each flow. More specifically, an action A_t at time t, is defined as $A_t = {a_1}^t, {a_2}^t, ..., {a_n}^t$. Here, ${a_i}^t$ is the action taken on user flow i at time t. Each action $a_i \in [0, 1]$ corresponds to the percentage of packets to drop while allowing the rest. 0 corresponds to allowing all traffic to pass and 1 corresponds to dropping all traffic. It is important to note that dropping the packets is a crude mechanism of rate control and is a commonly implemented method by network switches. However, new techniques which enforce a rate cap by smoothly delaying packets rather than discarding them can be easily incorporated in our system model. So, in the rest of the paper, we use the terms dropping and delaying packets interchangeably.

B. Problem Definition

We formulate the functionality optimization goal of `Jarvis-SDN` as a Markovian decision process (MDP). A MDP is a sequential decision making problem where outcomes are under the control of an agent. The agent's goal in this case is to maximize the functionality as specified by the user/application by choosing a sequence of actions for the upcoming episode of the environment. In our model, functionality requirements defined by the user/application are measured through a reward function. The specified functionality requirement determines the utility ($F()$) gained by the user/application in the environment, which is one part of the reward function. The other part, derives from the security metric of the environment ($D()$). The general structure of the reward function is defined as follows:

$R(S_t, A_t) = (1 - \delta)F(S_t, A_t) + \delta D(S_t, A_t)$

where S_t and A_t are the current state and action at time instance t in the environment. We analyze the effect that manipulating δ has on the overall functionality goals and security of the environment in Section V. The goal of `Jarvis-SDN` is to maximize the cumulative reward at the end of the episode which is a MDP problem defined formally as follows.

Definition 1. *A MDP consists of a tuple (R, T, I, S_0). R is the reward function; T is the time period; I is the interval; and S_0 is the initial state of the environment. The goal of the agent is to find an execution strategy of actions, which maximizes the total value of R of the next upcoming episode, for the environment in state S_i where $0 \leq i \leq \lceil T/I \rceil$.*

C. Challenge

key challenges when applying the system model to a SDN environment that we discuss in what follows together with the envisioned approach. *1. Unknown Security Metric $D()$:* Quantifying the security metric accurately in a complex and dynamic environment is not trivial. We address this issue in Section IV by developing an efficient security metric using RL based attack signatures. In particular we will use the following reward function: Notice that other security metrics can also be incorporated into our framework.

2. Unknown Security Ratio δ: The value of the ratio to optimally balance functionality vs security depends on the environment and user/application requirements. Finding this value accurately is not trivial. In Section V, we analyse tuning the security ratio δ as a hyper parameter for `Jarvis-SDN` instantiation with the functionality goal of optimal rate control.

IV. Building Attack Signatures

In this section, we address the first challenge in building accurate security metrics, referred to as 'attack signatures'. Deep packet inspection at wire speed seems impractical with the growing amount of data and size of networks. Instead, flow based features, which can be maintained using counters and meters, and are easily available through standard SDN protocols such as Openflow [9], are less expensive to monitor. Most supervised attack signatures are built from flow parameters similar to the network flow parameters in our states $P_1, P_2, ..P_k$. These parameters are collected from IDS datasets like NSL-KDD [10] and CICIDS[7]. However, these datasets contain features collected for the entirety of the network flows and not for intervals of the flows as in our framework. This inherently puts the IDS trained on those datasets at a disadvantage since it delays detection time because the flows can only be classified as malicious or benign after the attack ends or enough packets of the malicious flow have reached the controller. In contrast, we build and analyze 'partial attack signatures' using features collected after every interval I in the episode of length T rather than the entire flow. In this respect, our attack signatures are only a partial representation of the more comprehensive traditional attack signatures.

A. Design of an IDS Based on Partial Attack Signatures

We compare four different ML techniques for building attack signatures in order to determine the most suitable one for our framework: Decision Trees (DT), Random Forests (RF), Deep Neural Networks (DNN), and Deep Q-Networks (DQN). The first three are feature-based classification models that predict benign or malicious flows. The DQN approach, on the other hand, learns optimal quality values using deep Q-Learning on replayed episodes. We model this DQN framework in the same state-action model of our optimization problem. However, the reward function is modified such that the agent receives negative rewards for allowing malicious traffic and positive rewards for allowing benign traffic depending on the hyper parameters $\beta, \mu \in [0, 1]$. Here, μ and β represent the importance of reducing false positives (Type 2 errors) and

false negatives (Type 1 errors), respectively. These rewards are proportional to the amount of traffic let through. The state-action-reward model is defined as follows:

State Model: The state of a user i at time instance t, ${s_i}^t = {P_{1_i}}^t, {P_{2_i}}^t, {P_{3_i}}^t, {P_{4_i}}^t$. Here, P_{1-4} represent the following features, respectively: P_1: #packets sent from user to server, P_2: #bytes sent from user to server, P_3: #packets sent from server to user, and P_4: #bytes sent from server to user.

Actions: After every interval, the controller can take 11 possible actions $a_i \in \{0, 0.1, 0.2, .., 1\}$ corresponding to percentage of packets to drop while allowing the rest. 0 corresponds to allowing all traffic to pass and 1 corresponds to dropping all traffic. We discretize the action space to allow better convergence of the DQN.

Rewards: The reward function is defined as follows.

$$R({s_i}^t, {a_i}^t) = \begin{cases} (1 - {a_i}^t) * ({P_{2_i}}^t) * \beta & \text{if benign episode} \\ -(1 - {a_i}^t) * ({P_{2_i}}^t) * \mu & \text{if malicious episode} \end{cases}$$

We build a simulation environment using the OpenAI gym [11]. For training the DQN, we replay 20177 malicious and 126714 benign episodes from the CICIDS dataset [7]. During replay of the episodes, we simulate packet drops by using a state transition function Δ, manually configured according to the network flow features used in the model. The exploration factor of $\epsilon = 1$ with a decay rate 0.995 is used to balance exploration (random actions) and exploitation (using learnt policy) while training. A discount factor $\gamma = 0.95$ and learning rate $\alpha = 0.001$ are used. The deep neural network used has 2 fully connected hidden layers comprising of 64 neurons each.

Attack Type	Total Packet Capture Size (Gb)	Categories
Brute Force	11	FTP-Patator, SSH-Patator
DoS	13.4	SlowLoris, Hulk, GoldenEye, SlowHTTPtest
DDoS	8.8	DDoS LOIT
Web	8.3	XSS, SQL Injection, Brute Force

TABLE I: Attack Taxonomy

B. Evaluation Metrics

For our analysis, we consider four common attacks, Brute Force, DoS, DDoS (Distributed Denial of Service) and Web-based (see Table I), taken from the CICIDS dataset. This dataset has full packet captures of attacks and benign behavior recorded over five days. We define four key performance metrics as follows.

Naive Accuracy: It measures the accuracy of detecting known attacks similar to the ones in the training dataset.

Robustness: Features collected from networks flows in real time tend to be noisy because of network errors, packet drops, jitter, throughput throttling, user behavior etc. Introduction of noise or perturbations in the immutable features is a typical obfuscation technique employed in adversarial ML attacks. The attack signatures must be robust enough to deal with noisy data. So, white Gaussian noise is introduced into the testing dataset. The resulting accuracy is measured as a function of the noise introduced.

Adaptability: It measures the ability of the system to detect new types of attacks. We consider two forms of adaptability: (1) known attacks with minor modifications, e.g. payload differences (string changes, varying malware bytecode), and packet fragmentation variations; (2) new attacks employing similar concepts. Specifically, we exclude FTP-Patator (brute force), SlowLoris (DoS) and SlowHTTPTest (DoS) attacks during training and use them for testing. The resulting accuracy in these scenarios is used to represent the adaptability metric for the attack signatures.

Episode metrics: Instead of detecting malicious traffic on a per interval basis, we attempt to determine whether any malicious interval exists over the range of the episode for unknown attacks under noisy conditions. We define three *episode metrics*: (1) *Episode Accuracy*: Overall accuracy; (2) *Episode True Positive Rate (TPR)*: Rate of correctly identifying malicious episodes; and (3) *Episode False Positive Rate (FPR)*: Rate of incorrectly identifying benign episodes as malicious.

TABLE II: Attack Signature Analysis

Signature Type	Naive Accuracy	Robustness Noise(low-high)	Adaptability	Episode Metrics (Accuracy, TPR, FPR)			Quality Value
DT	99.92	70.58-26.60	53.22	44.88	100	99.2	No
RF	99.93	80.14-24.83	53.40	44.88	100	99.2	No
DNN	97.22	95.56-80.24	70.28	59.55	100	72.8	Yes
DQN	88.27	79.53-71.21	59.48	73.77	94	42.4	Yes

C. Comparison to other ML Techniques

Results from our evaluation are shown in Table II. We make the following observations on those results.

Naive Accuracy: All signatures work well in terms of naive accuracy for known attacks. The tree based signatures (DT/RF) perform slightly better than neural network based signatures (DNN/DQN).

Robustness: Performance of all signatures degrades with the amount of noise added. Comparatively, neural network based signatures are less affected.

Adaptability: All algorithms have decreased accuracy when trying to identify unknown and modified attacks. The highest accuracy, 70.28%, is achieved by DNN based signatures.

As we observe, all the attack signatures perform poorly when dealing with unknown or zero-day attacks in terms of per-interval accuracy. We can infer that these unknown malicious flows resemble behavior of both known malicious and benign flows in many intervals of an episode. So, building attack signatures based on episodes rather than individual intervals is a more suitable approach as we see when analyzing the *episode metrics*. It is important to note here that unsupervised learning techniques, like anomaly detection on benign behavior (clustering, SVMs), could be a better approach for identifying unknown attacks. Although these techniques could be incorporated into our framework, in this work, we only focus on supervised learning and leave that as future work.

Episode metrics: DQN attack signatures outperform the other signatures in terms of *Episode* metrics even though they perform poorly in terms of per interval accuracy. One reason is that the DQN signatures are learnt from a cumulative reward by replaying attack ranges rather than individual intervals. The

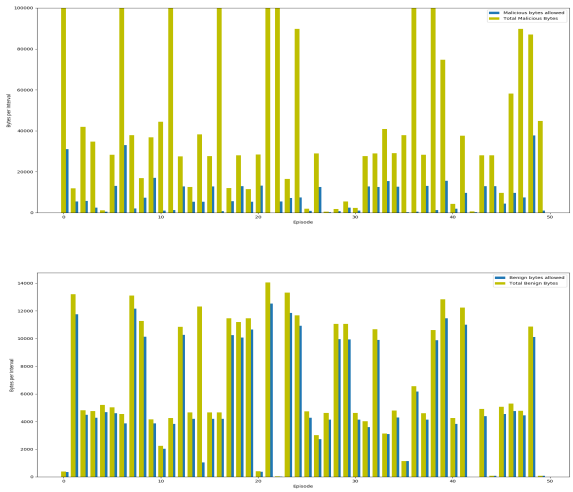

Fig. 2: Bytes allowed: Malicious (top) and Benign (bottom)

other reason is the structure of the reward function, which gives a higher negative reward for false positives than false negatives over the episode range.

The classification based signatures (DT/RF) perform badly for the episode metrics because they learn optimal discrete values (benign/malicious) rather than smooth quality values. We can see that, although these signatures are able to correctly identify all the malicious episodes (*episode TPR* = 100%), they also result in a high *FPR* = *99.2%*. Such overly strict signatures are unacceptable as a high percentage of benign flows end up being dropped. On the other hand, the DQN agent learns quality values $Q(S_t, A_t)$ for each interval within the episode. We can then build an intelligent policy based on quality values that minimizes the *episode FPR* and simultaneously maintains a good *episode TPR* by tuning the hyper-parameters β and μ. We infer that DQN based agents perform better for new attacks in terms of false positives. Note that DNN signatures can also generate quality values using state values $V(S_t)$ from a final softmax layer. However, these quality values do not incorporate the state and actions dynamics of the model. Therefore, using these quality values as metrics in a RL framework for functionality optimization results in random quality values being assigned to new actions. This results in poor performance of the RL agent (see Section V-F).

D. Evaluation and Analysis

To analyse the effectiveness of RL based (DQN) attack signatures, we randomly choose 1000 malicious and benign episodes, and apply the policy learnt to them. We monitor the amount of malicious bytes let through in case of malicious episodes and the amount of benign bytes let through for benign episodes as shown in Figure 2. We see that the attack signatures show the desired behavior in all the episodes. It is important to note that these attack signatures have been built using a minimal set of observable states at the SDN controller. With the recent advances in programmable switches and data plane programming languages, like P4 [12], detailed information about packets and their headers is accessible to the controller to make even better security decisions.

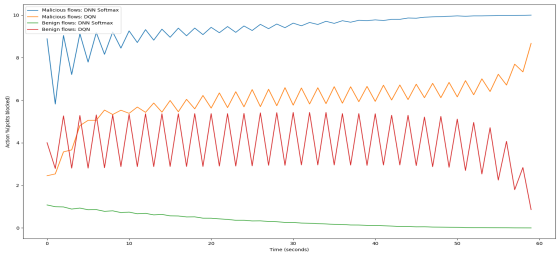

Fig. 3: Intra-Episode Actions: DQN and DNN Softmax

To further investigate the behavior of the RL based attack signatures, we analyze the actions taken by the RL agent in the episode itself. Figure 3 shows the behavior averaged over 1000 malicious and benign episodes. This is desirable behavior as malicious flows get throttled while benign flows are not. Also, we see that at the beginning of the episodes, the RL agent is not sure whether the flow is malicious or not. In such situations, we see that the RL agent exhibits the following desirable behavior. Malicious looking flows are throttled until the agent is sure they are an attack and then are blocked completely. Benign looking flows are throttled until the agent is sure they are benign and then are allowed. We compare the behavior with the policy based on state values obtained from the softmax layer of the DNN classifier. Surprisingly, the softmax layer approach also shows these same desirable properties and performs better than the RL based signatures. This could be attributed to the fact that since no other functionality optimizations ($F() = 0$) are required during this analysis, the agent does not fully explore the action space and thus the state-action dynamics of the model have little impact on the optimal policy. However, as we see later in Section V-F, the softmax based signatures do not perform well when other functionality optimizations are needed.

V. Instantiation of Jarvis-SDN

In this section, we instantiate our framework with the goal of optimal rate control in our simulated environment.

A. Implementation Details

We have built a prototype of Jarvis-SDN and integrated it with a network emulated in Mininet [13]. The emulated network includes benign and malicious users accessing a web server. Jarvis-SDN sits in the control plane of the SDN controller. It is able to access counters maintained for each flow in the flow table of the SDN controller similar to architectures like [14]. Information from these counters is then used to monitor the state of each flow in terms of the flow parameters $P_1 - P_4$. Similar to approaches discussed in [15], the actions specified by the RL agent are converted into per flow bandwidth limits.

B. Simulation Environment

The simulated environment consists of n users accessing a web server in a SDN enabled environment. Jarvis-SDN running at the SDN controller is responsible for taking optimal rate control actions on each flow so as to maximize two

functionalities: ❶ Rate control for the server according to a maximum allowed server load threshold: $SLT \sim \mathcal{N}(\mu_1, {\sigma_1}^2)$; ❷ User throughput fairness according to user service level agreements: $SLA \sim \mathcal{N}(\mu_2, {\sigma_2}^2)$. We use standard Gaussian distributions to model the server load threshold and user SLA per time interval. The mean μ_1, μ_2 and standard deviation σ_1, σ_2 are chosen manually by observing the server threshold and benign flows in the simulation. Along with this, we maintain δ (or d), f_1 and f_2 as hyper parameters to incorporate security metrics, rate control and user fairness weights in the objective function, respectively.

C. System Model

The state and action model of `Jarvis-SDN` are the same as the ones defined before for building attack signatures. However, the reward function is modified to include optimal rate control functionality metrics and security metrics from RL based attack signatures. The reward function is defined as follows.

$$R(S_t, A_t) = (1-\delta)[f_1 F_1(S_t, A_t) + f_2 F_2(S_t, A_t)] + \delta D(S_t, A_t)$$

The normalized functionality and security metrics used in the reward function are defined as follows. We observe that, unlike the case when building attack signatures, here the rewards are expressed as expectations over all the users flows in the network.

Rate Control: It gives a positive reward proportional to the amount of traffic allowed when the current load is less than the threshold and a large negative reward C_1 for overloading the server. The positive rewards are proportional to the amount of traffic let through, which encourages higher throughput.

$$F_1(S_t, A_t) = \mathop{\mathbb{E}}_{i \in \eta}[F_1({s_i}^t, {a_i}^t)]$$

$$= \begin{cases} \mathbb{E}_{i\in\eta}[(1-{a_i}^t) * {P_2}_i{}^t] & \{SLT(t) > (1-{a_i}^t) * {P_2}_i{}^t\} \forall i \in \eta \\ -C_1 & \textit{otherwise.} \end{cases}$$

User Fairness: It gives a positive reward proportional to the amount of traffic allowed when the service level agreement is upheld and a large negative C_2 otherwise. The values of parameters C_1 and C_2 are chosen to be greater than the maximum throughput values possible in the environment.

$$F_2(S_t, A_t) = \mathop{\mathbb{E}}_{i \in \eta}[F_2({s_i}^t, {a_i}^t)]$$

$$= \begin{cases} \mathbb{E}_{i\in\eta}[(1-{a_i}^t) * {P_2}_i{}^t] & \{SLA(t) \leq (1-{a_i}^t) * {P_2}_i^t\} \forall i \in \eta \\ -C_2 & \text{otherwise.} \end{cases}$$

Security: It is the quality value generated by RL based attack signatures built before. Here Q is the learnt Q function using the DQN approach. It gives a positive reward when the flow matches a benign flow and a negative reward when it matches a malicious flow.

$$D(S_t, A_t) = \mathop{\mathbb{E}}_{i \in \eta}[D({s_i}^t, {a_i}^t)] \approx \mathop{\mathbb{E}}_{i \in \eta}[Q({s_i}^t, {a_i}^t)]$$

D. Training

We use a DQN approach to learn the optimal policy similar to the attack signature generation model. We conduct experiments with δ values ranging from 0 (no security guarantees) to 0.99 (high security guarantees). For our experiments, we configure the functionality weights $f_1, f_2 = 0.5$ as constants. We refer the reader to our previous work [4] for more details about functionality weights.

E. Evaluation and Analysis

Figure 4 shows the actions (averaged over 1000 episodes) taken in a single malicious and benign episode. We observe that higher values of δ result in better security guarantees for malicious episodes but at the cost of more false positives per interval during benign episodes. We further analyze the action space of the environment in terms of range of δ. We represent the action space (red region) below $\delta = 0$ as the *unsafe action space* of the model. Similarly, the blue region above $\delta = 1$ represents the *safe action space*. The green region represents the *region of interest* with a range of $\delta = [0, 1]$. The optimal action policy of the environment lies in this region.

Fig. 4: Action Space: Malicious (top) and Benign (bottom)

The analysis of the *region of interest* in the action space allows one to build ethics or logical hypotheticals for exploration of the environment by the RL agent. For example, the RL agent can explore the action space with $\delta > 0.8$, for highly security conscious or risk-averse ethics. Similarly, with $\delta < 0.2$, the RL agent can explore actions with more adventurous high risk high reward ethics. Learning hypothetical human objectives or ethics to model the reward function or the exploration process is a hard problem because of the intractability of value of information (especially security information) to the user for a given environment. This is the focus of various active learning schemes [16], [17]. The exploration strategy can be dynamically altered based on user/application feedback on these hypothetical behaviors. `Jarvis-SDN` provides an effective way of altering the exploration process by using such hypotheticals along the *region of interest*.

F. Comparison to Traditional IDS Metrics

Figure 4 also shows the actions taken by the RL agent using DNN softmax based security metrics. We see that for both malicious and benign episodes, the RL agent converges to similar actions. This means that these security metrics perform poorly in the detection of malicious flows. This confirms our intuition that when the optimization of multiple

functionalities is required, RL agents using softmax based DNNs or traditional IDS security metrics do not converge to secure policies.

VI. Related Work

As part of previous work we designed and evaluated a framework to constrain RL agents by security and safety policies for IoT-based smart homes [4]. However in a smart home scenario, 'attack signatures' are well defined in terms of device states and can be learnt efficiently by observing anomalies against naturally occurring behavior of users. Such an approach does not work for detecting attacks in networks based on network flows. The reason is that in networks, there are numerous applications for which it is difficult to build an accurate baseline of benign behavior. So, an anomaly based detection is inherently prone to false positives. To address this issue, in this paper we have used a semi-supervised learning approach to build attack signatures which provide quantifiable security metrics. The other reason is that the state space of a network environment is not limited to discrete device states like in smart homes and thus suffers from the state explosion problem.

Attempts have been made to define quantifiable security metrics [18], [5], [6]. Proposed approaches include building metrics from user or host vulnerabilities, defense strengths, attack (or threat) severity and situation understanding of the environment. To the best of our knowledge, security metrics or attack signatures based on network flow parameters have not been explored. In terms of encoding security metrics in RL frameworks, there are two main approaches: (1) transformation of the optimization criterion, and (2) modification of the exploration process [19]. Our approach falls under the first category as we modify the objective/reward function using the security metrics. But additionally, we also analyze the action space according to the hyper parameter δ, which can be used to inject external knowledge or advice to guide the exploration process of the RL framework. Due to space limitations, we leave the integration of such an approach into Jarvis-SDN as our future work.

VII. Conclusion and Future Work

In this paper we have designed and evaluated `Jarvis-SDN`, a RL framework for optimizing network functionalities constrained using RL-based network flow attack signatures. While our initial results show that our framework represents significant progress, we plan to carry out additional experiments to assess its performance for different network conditions, network functionalities like routing and RL algorithms like policy gradient approaches.

ACKNOWLEDGMENTS

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001, and the U.S. Army Research Office under Agreement Number W911NF1910432. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] G. Stampa, M. Arias, D. Sánchez-Charles, V. Muntés-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," *arXiv preprint arXiv:1709.07080*, 2017.

[2] X. Huang, T. Yuan, G. Qiao, and Y. Ren, "Deep reinforcement learning for multimedia traffic control in software defined networking," *IEEE Network*, vol. 32, no. 6, pp. 35–41, 2018.

[3] P. Sun, Z. Guo, G. Wang, J. Lan, and Y. Hu, "Marvel: Enabling controller load balancing in software-defined networks with multi-agent reinforcement learning," *Computer Networks*, p. 107230, 2020.

[4] A. Mudgerikar and E. Bertino, "Jarvis: Moving towards a smarter internet of things," in *40th IEEE International Conference on Distributed Computing Systems*. IEEE, 2020.

[5] M. Pendleton, R. Garcia-Lebron, J.-H. Cho, and S. Xu, "A survey on systems security metrics," *ACM Computing Surveys*, vol. 49, no. 4, 2017.

[6] W. H. Sanders, "Quantitative security metrics: Unattainable holy grail or a vital breakthrough within our reach?" *IEEE Security & Privacy*, vol. 12, no. 2, pp. 67–69, 2014.

[7] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[10] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*.

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[12] P. Bosshart and et al., "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[13] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 1–6.

[14] D. J. Hamad, K. G. Yalda, and I. T. Okumus, "Getting traffic statistics from network devices in an sdn environment using openflow," *Information Technology and Systems*, pp. 951–956, 2015.

[15] M. Karakus and A. Durresi, "Quality of service (qos) in software defined networking (sdn): A survey," *Journal of Network and Computer Applications*, vol. 80, pp. 200–218, 2017.

[16] S. Reddy, A. D. Dragan, S. Levine, S. Legg, and J. Leike, "Learning human objectives by evaluating hypothetical behavior," *arXiv preprint arXiv:1912.05652*, 2019.

[17] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[18] A. Ramos, M. Lazar, R. Holanda Filho, and J. J. Rodrigues, "Model-based quantitative network security metrics: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2704–2734, 2017.

[19] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.