A Security-Constrained Reinforcement Learning Framework for Software Defined Networks

Anand Mudgerikar*, Elisa Bertino*, Jorge Lobo‡, Dinesh Verma†
* *Purdue University, West Lafayette, IN, USA*
† *IBM TJ Watson Research Center, Yorktown Heights, NY, USA*
‡ *ICREA - Universitat Pompeo Fabra Spain*

***Abstract*—Reinforcement Learning (RL) is an effective machine learning (ML) techniques for building 'smart' SDN controllers because of its model-free nature and ability to learn policies online without requiring extensive training data. However, as these RL agents are geared to maximize functionality and explore the environment without constraints, security policies can be breached. In this paper, we propose 'Jarvis-SDN', a RL framework that constrains explorations with security policies. In Jarvis-SDN, the RL agent learns 'intelligent policies' which maximize functionality but not at the cost of security. Standard 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 the form of 'partial attack signatures' from IDS datasets that are then encoded in the reward function.**

I. INTRODUCTION

The use of machine learning (ML) techniques in the control plane in software defined networks (SDNs) provides enhanced approaches to optimize goals of interest to applications, such as maximizing quality of service (QoS). Generally, QoS is determined by the interplay within various network functionalities such as routing, load balancing, and resource management. This interplay can become very complex, but the benefit of ML techniques is that they can model complexity given sufficient representative data to train upon. However, such environments also have a very dynamic behavior, which poses a challenge to ML. The quality of the models captured by classical ML depends on the training data. However, due to the complexity and scale of current networks, training data that can capture a diverse enough set of behaviors is difficult to gather. Due to this limitation, classical supervised ML techniques may not be suitable for control management in modern networks. Reinforcement Learning (RL) on the other hand, relies on learning optimal policies online based on system state, and these policies are more likely to transfer over to a new environment. As a result, RL techniques may be more suitable for network control.

For specific functions within the network, RL based frameworks have been proposed, such as for controlling routing [1], and load balancing [2], etc.. One weakness of the approach is that they ignore security, which is an indispensable network functionality. Furthermore, a practical network control solution requires optimization across multiple functionalities, not just a single one. As a result, current approaches looking at applying RL to network control face the following key limitations:

1) They end up selecting a policy optimal in the context of one functionality and sub-optimal in terms of other functionalities. For example, learning a policy which maximizes the throughput of the network (functionality 1: optimal routing) but at the cost of unfair bandwidth consumption by one user (functionality 2: per user bandwidth fairness).
2) They end up selecting a policy which violates security. For example, learning a policy which maximizes the throughput of the network (functionality 1: optimal routing) but unknowingly facilitates the propagation of a high throughput Denial of Service (DoS) attack.

To address those issues, we propose `Jarvis-SDN`, a constrained RL framework for SDNs, based on our prior work [3] on applying RL to the Internet of Things (IoT). In `Jarvis-SDN`, a RL based agent learns optimal policies for a SDN controller which optimizes across multiple network functionalities while maintaining security. Examples of such functionalities include 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 [3] to SDNs is that defining performance metrics for security is non-trivial. We refer the reader to [4], [5], [6] for discussions regarding security metrics. Our approach for quantifying security is to measure the ability to protect against known attacks. We build 'attack signatures' from packet captures of previous attacks using different ML techniques. These attack signatures are then used to determine a quality value for the network state depending on the perceived threat a SDN controller has on the current and near future states of the network.

II. PROBLEM DESCRIPTION

The problem we want to address using ML techniques is that of the design of an intelligent SDN controller. The controller is responsible for dealing with requests that are coming from several users. The requests are going to two or more devices. Depending on the amount of active requests at different devices, anticipated requests in the near future, and the properties of the current request, the controller forwards the request to one or more of the devices.

This problem arises in many networking contexts. One context in which the problem would arise is that of routing of requests within a data center or cloud network environment, where the routing behaviour is controlled by means of a centralized controller, as is typical in Software Defined Network (SDN) architectures. The controller needs to determine how to handle each of the requests, and determine the right outbound router to dispatch the request to. Another context would be in a telecommunications or Internet Service Provider. The service provider is responsible for accepting packets from connected clients, and route them to the appropriate egress point. As SDN based control is used to influence and determine the routes in these networks, the SDN controller would need to determine how best to balance the network traffic. Finally, another context would be in 5G based cellular network protocols, which are designed to implement their control and data planes based on SDN. In these cases, the control plane needs to determine how to route packets from a cellular device, incorporate decisions about leveraging a mobile edge computing server if configured, or forward packets to an alternate location for processing. While the protocols are different than in a data center or Internet, the operational paradigm is very similar. In most of these environments, the SDN controller needs to make the decision based on the inspection of the headers within the IP network packet. Therefore, we focus on approaches that learn optimal routing policies based on the inspection of features extracted from network packet headers.

III. The Jarvis-SDN Framework

`Jarvis-SDN` works in the control plane of the SDN controller. The SDN controller would typically interact with the data plane using protocols like OpenFlow (for legacy switches) or P4 [7] (for programmable switches). The communication from the user is characterized on the basis of traffic flow from the users by means of four parameters: P_1:#packets from user to server, P_2:#bytes sent from server to user, P_3: #packets from user to server and P_4:#bytes from server to user. Note that the framework can be extended easily to consider additional parameters, but these four are the ones we have done our evaluation upon. These parameters are maintained for each flow and are readily accessible through counters maintained on the SDN network switches. The RL agent learns optimal policies to control packet forwarding (that is, to modify the flow table on the network switch) in the data plane.

The state and action model of Jarvis-SDN are as follows:

State Model: The state of a user i is defined as, $S_i = P_1, P_2, P_3, P_4$. The overall state is the combination of all the states of every user within the network.

Actions: After every interval, the controller can take two possible actions $A = \{0$: Allow traffic forwarding from user to server, or 1: Drop/Flag traffic from user to server$\}$.

Rewards: The reward function for the framework is a weighted composition of three functionalities defined as $R(S,A) = \sum_{i=1}^{3} \omega_i F_i(S,A)$ where ω_i are the functionality weights. The performance metrics $F_i()$ for each functionality i are defined as follows.

Load balancing F_1: $\{$Server load threshold - current server load $\}$. It gives positive reward when current load is less than threshold and negative for overloading the server.

User Fairness F_2: Mean of $\{$current user service - User Service level Agreement (SLA) threshold$\}$ for each user. It gives positive reward when service level agreement is upheld and negative otherwise.

Security F_3: Quality value generated by attack signatures. It gives a positive reward when the flow matches a benign flow and a negative reward when it matches a malicious flow.

A key challenge of the security metrics is the accurate classification of benign versus malicious flows. The approach we use is based on the concept of 'attack signatures' borrowed from the framework described in [3]. An attack signature is built to differentiate between malicious and benign traffic based purely on the set of features that are used within the SDN RL framework, namely the parameters P_{1-4}. In this respect, our attack signatures are only a partial representation of the more comprehensive attack signatures built from flow parameters collected by intrusion detection systems (IDS) datasets like NSL-KDD[8] and CICIDS[9]. However, the attack signatures we propose can be determined efficiently, and nevertheless be highly accurate.

Within this framework, we work with four different techniques to define attack signatures: decision trees (DT), random forests (RF), deep neural networks (DNN), and deep reinforcement learning (DRL). The first three are feature-based classification models to predict benign or malicious flows. The DRL approach on the other hand learns optimal quality values using a Q-Learning (DQN) approach by replaying a range of the attack. Such a range is defined by a fixed number of time intervals of the flow sufficient to identify it. The current implementation uses an attack range of 60 time intervals of 2 seconds each.

IV. Implementation and Performance Evaluation

We have built a prototype of `Jarvis-SDN` and integrated it with a network emulated in Mininet [10]. The emulated network includes five benign users and one malicious user accessing a web server. The benign and malicious flows are replayed from [9] in different network conditions simulated by congestion control and throughput throttling.

TABLE I: Attack Taxonomy

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

To compare the performance of the four different algorithms, we consider four common attacks, Brute Force, DoS, Web-based and DDoS (Distributed Denial of Service) (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.

Accuracy: It measures the accuracy of detecting signatures of malicious traffic over intervals of fixed duration.

Robustness: 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 evaluate two adaptability levels: (1) a *low* level where accuracy is measured when the testing dataset has known attacks but with minor modifications, e.g. payload differences (string changes, varying malware bytecode), and packet fragmentation variations. And (2) a *high* level where accuracy is measured when the testing dataset has 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.

OVR metrics: Instead of detecting malicious traffic on a per interval basis, we attempt to determine whether any malicious interval exists over a range of several intervals. We define two *Over Attack Range metrics: OVR accuracy* and *OVR false positive rate*. A range of the attack uses 60 rather than just one interval.

TABLE II: Attack Signature Analysis

Signature Type	Accuracy (%)	Robustness (%)	Adaptability (High) (%)	OVR Accuracy	OVR False Positive Rate	Quality Value
DT	98.98	78.76	70.09	41.96	0.98	No
RF	98.65	81.93	76.41	41.96	0.98	No
DNN	89.60	88.47	74.09	41.96	0.98	Yes
DRL	77.66	76.37	62.38	71.42	0.02	Yes

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

Accuracy: In terms of naive accuracy for known attacks, tree based signatures (DT/RF) perform the best. Neural network based signatures (DNN/DRL) perform worse.

Robustness: Tree based signatures (DT/RF) performance degrades with noise. RF works slightly better because of its ensembling nature. 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, 76.5%, is achieved by random forests.

OVR metrics: DRL attack signatures, which are trained, outperform the other signatures in terms of *OVR* metrics but perform poorly for overall accuracy and adaptability. One reason is that the DRL signatures are learnt from a cumulative reward by replaying attack ranges rather than individual intervals. The other reason is the structure of the reward function which gives a higher negative reward for false positives than true negatives over the range of the attack.

The classification based signatures (DT/RF/DNN) perform badly for the attack range metrics because they learn optimal discrete values (benign/malicious) rather than quality values for the environment. When using DRL signatures, the RL agent replays attack ranges to assign a quality value for each instance according to rewards accumulated during the whole attack range, i.e., it learns a quality value for each instance within an attack range. We can then build an efficient policy based on quality values that minimizes the *OVR false positive rate* and simultaneously maintains a good *OVR accuracy*. We infer that DRL based agents perform better for new attacks in terms of false positives. Note that DNN signatures can also generate quality values from the softmax layer values.

We performed preliminary experiments with varying functionality weights. Our experiments show that the RL agent is able to converge to an optimal policy in around 20 episodes. In terms of performance of the attack signatures, a flow from the attacker (user 5) gets so throttled that is effectively dropped after the first 4 episodes for known attacks.

V. Future Work

While our initial results show that the `Jarvis-SDN` holds significant progress, we need more experimental data to assess its performance under other conditions. We also plan to analyze temporal properties of state transitions to build better security signatures using recurrent neural networks and other RL techniques.

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] 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.

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

[4] S. M. Bellovin, "On the brittleness of software and the infeasibility of security metrics," *IEEE Security & Privacy*, vol. 4, no. 4, 2006.

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

[8] 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*.

[9] 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.

[10] 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.