International Journal of Research in Engineering, Science and Management
Volume-1, Issue-10, October-2018
www.ijresm.com | ISSN (Online): 2581-5792

Intelligent Threat Response

N. Saurabh Kumar[1], Sukumar Varma[2]
[1,2]*Student, Department of Electronics and Communications, SRMIST, Chennai, India*

***Abstract*—Software-defined networks decouple the control plane from the data plane, enabling researchers to evaluate protocols and network configurations through the centralized point of control, the controller. They provide easy management and automation, scalability, and flexibility in the traditional computer network. In spite of these advantages, software-defined networks fall prey to various denial-of-service attacks specific to network protocols and applications despite their simplicity. There is a need to implement intelligence in the controller as a countermeasure for not only the various types of denial-of-service attacks but also the increasing sophistication involved in them. In this paper, an intelligent threat-aware response system is proposed for defending against any attack by using reinforcement learning. Reinforcement learning can acquire intelligence for detection and reactive actions through experience with various attacks. This experience is obtained from interactions with the computer network through the controller. With the combination of reinforcement learning and the software-defined networking controller, the goal of the autonomous threat response system can be achieved.**

***Index Terms*— Intelligent Threat Response**

I. Introduction

Software-defined networking (SDN) provides an abstraction of programmability in traditional computer networks. Computer networks can be simulated virtually and interfaced through the SDN controller from topology configurations to protocol-specific behavior. This abstraction is facilitated by OpenFlow [1], a protocol which decouples the data plane from the control plane on the switch. OpenFlow protocol also makes the computer network more flexible. In addition, since the control plane is separated from the data plane, it is possible for researchers to deploy and test various protocols in real-time. However, along with all the advancements in the field of computer networks, there have been advances for compromising the networks as well. Protocol features such as three-way handshake in TCP, HTTP GET request processing, and ICMP response can be exploited for selfish gains. The same features which provide a secure connection for communication, availability, and hassle-free data transfer lead to the misuse. A few examples of such misuse are TCP SYN flood attack, HTTP GET Request flood attack, and Ping flood attack. Recent history of network attacks [2, 3] brings to light the innovative approaches taken by adversaries in terms of denial-of-service (DoS) attacks. This calls for a measure which can surpass the same innovation and counter the attacks. The other concept which enters the picture is one of reinforcement learning [4]. It is a paradigm which closely imitates the human way of learning. The human brain learns with the help of interaction with and feedback from a corresponding environment. Given a state of the environment, the human brain evaluates the best possible action to interact with it and gain the best possible outcome. The definition of the word ‘‘best’’ is highly subjective. However, the same learning procedure helps 1 us understand concepts from natural language [5] to aeronautics [6]. This paper presents an approach in which a reinforcement learning paradigm is combined with SDN to create an intelligent threat-aware framework which is able to take responsive actions, given access to network behavior information.

II. System Architecture

system architecture for the intelligent threat-aware response system in software-defined networks. It gives an explanation of the different components of the framework in a unique combination of northbound and southbound interfaces for knowledge discovery and data mining (KDD) process, reinforcement learning, and the reactive actions. The three main components are as follows:

- Traffic Analyzer: A southbound interface for monitoring the network traffic propagating through each of the switches connected in the network. The same implementation facilitates the initial steps of the KDD process.
- Reinforcement Learning Agent: An additional component developed in the SDN controller operating system for the implementation of the Q-learning algorithm.
- Threat Response: Another south bound interface which provides APIs to communicate with OpenFlow and ovsdb to manage the bandwidth of the switches and also to update flow tables on the switch.

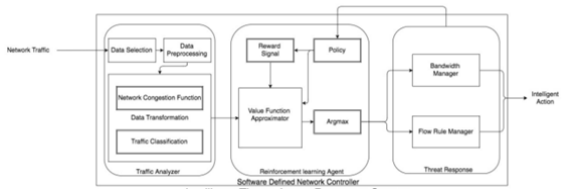

Fig. 1. Intelligent threat-aware response system

A. Traffic Analyzer

A threat-aware system needs to continuously monitor the

network to detect any unusual behavior. The traffic analyzer is the module responsible for the network monitoring in the proposed framework. It collects information about the total number of packets sent and received along with the total number of bytes sent and received through the network. However, the total number of packets and bytes alone do not provide much of an insight to discriminate between various types of network threats. To achieve that, traffic analyzer module collects information specific to different protocols such as TCP, UDP, HTTP, etc. The traffic analyzer module is a southbound interface in the SDN controller operating system that monitors network traffic. The OpenFlow protocol APIs allow the controller to check for flow-specific and port-specific statistics. This framework is implemented as a part of the RYU [15] SDN controller. RYU is an event-driven controller wherein the southbound APIs are invoked whenever there is an appropriate event. For example, one could set up an event listener for an OpenFlow packet and define an API to be invoked whenever there is an incoming OpenFlow packet. In a similar way, there are listeners for port-specific statistics on the switch. The traffic 10 analyzer sends a stats request to the switch, in Hence, network latency is also one of the features included during the data selection phase.

1) Data Selection

Just as the name suggests, the data selection process involves the determination of the respective data type and source. As discussed earlier, the selected statistics comprise the total number of packets, the total bytes of data being transferred, and protocol-specific information with respect to the overall values. For example, let us suppose that 10% of the bandwidth capacity is under use in the network. The first set of features would determine how much of that 10% is occupied by the respective switches present in the network and then calculate what percent of the overall usage belongs to a specific protocol. The features are computed with the help of the network congestion function and traffic classification while transforming the data into an appropriate representation for the reinforcement learning agent. DoS attacks mainly focus on exhausting the bandwidth of the network or the resources of the switch and/or a host. The information about the total number of packets and the total size allows one to estimate the bandwidth usage to some extent. Resource exhaustion is achieved by sending spoofed or fake requests to the victim; therefore, protocol-specific statistics are selected to address these type of attacks. Also, if a network is under attack, the network latency would be relatively high. 11 Hence, network latency is also one of the features included during the data selection phase.

2) Data Preprocessing

The data that exist in the natural state or exist in the world are not necessarily in the best possible format. First of all, it is not necessary for the data to have all the uniform values as an ideal data set. For this research, an ideal data set is one where there are no missing or invalid values for any considered parameter. The challenge to solve in this phase is generating the features in a synchronized manner. It is important to measure the number of packets, the total number of bytes exchanged, and the network latency at the same time in order to represent the network state for that time. Even a slight shift in the measurement would lead to confusion in the network state. For example, if the number of packets and the number of bytes are recorded at different times, one could end up with many more bytes for a relatively small number of packets and vice versa. Such a discrepancy in the statistics would mislead the reinforcement learning agent and cause errors while it selects a reactive action. During the data preprocessing step, the data are filtered to form the features that will be used later for further computation.

3) Data Transformation

The information obtained so far lacks context. The context information allows the framework to understand the need for taking a reactive action. That is, the context information helps to derive some conclusion from the data. With respect to the framework, the context information summarizes the overall behavior of the network at a given point in time. The aim of this phase is to provide concrete evidence that the network is congested or is under an attack. This evidence contributes towards the intelligence of the proposed framework. The intelligence is 12 not only in taking the right action given a threat, but also on the right understanding of the network behavior. For example, if the framework scales up the bandwidth at a time when the network does not need more bandwidth to function, it defeats the purpose of an intelligent system. The context information is provided with the help of network congestion function and traffic classification. Network congestion function is one of the contributions of this research to identify the network behavior given limited information. Traffic classification is an extension to the network congestion function for obtaining protocol-specific congestion across the network.

4) Network Congestion Score

The very first requirement of a threat-aware response system is to know if there is any threat to the computer network. A network threat can be identified as an unusual behavior of the network, whether it is through the traffic or by the switches. This unusual behavior needs to be captured through some measure to determine if there is an active threat to the computer network. The network congestion function carries that purpose in the framework. This function returns a congestion score which helps the reinforcement learning agent distinguish between the normal and abnormal network behavior. That is, the congestion score becomes the ‘‘observation’’ and input to the reinforcement learning algorithm. Hence, congestion score allows the intelligence module to take reactive action in case of a threat and scale up the network resources when needed. The

network congestion score is defined as a function of throughput and network latency. The throughput of the network indicates the number of bytes or the rate of bytes being exchanged through the network per unit time. The network latency on the other hand indicates the round trip-time (RTT) that a packet would take for traveling across all the nodes of the network and returning to the origin. 13 That is, the packet will originate from the switch and hop onto each of the hosts present in the network and back to the switch. The following equation can be used to calculate network congestion score: Congestion Score = {□ $\triangle B$ B * 100}□ + {□ 1 k * n * $\triangle L$ L }□ (1) $\triangle B$ is the difference between the current speed of the port and the default speed of the port B, the current speed is the byte rate with which the port is transmitting data, and the default speed is the maximum speed with which the port can transmit data. The value of n indicates the total number of links in the local network and $\triangle L$ is the difference between the network latency and the ideal latency L of the local network. The total number of links helps to determine the ideal network latency.

B. *Reinforcement Learning Agent*

This section explains the reinforcement learning paradigm as applied to network security. This module is developed as a part of the RYU [15] SDN controller operating system. Formally speaking, a reinforcement learning problem consists of the following sub-elements:

- Policy
- Reward Signal
- Value function

The reinforcement learning problem is the selection of the right reactive action given the current network behavior. Let us go over each of the sub-elements of reinforcement learning. This approach not only provides an overview of reinforcement learning but also provides context information that facilitates the relationship between the concept of reinforcement learning and its practical implementation for network security.

C. *Threat Response*

The threat response module interacts with the ovsdb and OpenFlow southbound APIs of the SDN controller to deploy network configurations from the framework. The configurations are selected with the help of reinforcement learning and are deployed in real-time. As shown in Figure 1, the threat response consists of two sub-modules: bandwidth manager and flow rule update manager. Both of these sub-modules represent the actions which the reinforcement learning agent can take against a network threat. This module can be easily extended with more choices for actions against the network threat.

1) *Bandwidth Manager*

The bandwidth manager can be thought of as an interface which provides a southbound API to modify the bandwidth of the switch. The value of the bandwidth is determined by another reinforcement learning agent, making it possible for the system to scale the bandwidth up and down as needed. The state space and action 22 space for this agent are not so different from the agent that determines whether to update bandwidth or update a flow rule.

2) *Flow Rule Manager*

The flow rule manager is another interface which provides a southbound API to update flow rules on the switch. This API sends an OpenFlow message to the particular switch to indicate the update in the flow table. With the help of this API, 23 the framework can add, modify, and remove rules corresponding to various packets. There are many cases wherein just updating the bandwidth of the network does not solve the problem. For example, in the case of HTTP GET Request Flood, it is not the bandwidth which is exhausted in the network but the resources of the victim server. The action could be to limit or stop the incoming HTTP traffic to a particular switch or host in the network.

III. Results and Discussion

Apart from the iterative training of the reinforcement learning agent on various types of DoS attacks, the other aspect of evaluation is how quickly the reinforcement learning agent learns to select the optimal or the right action as per the standard strategy. For example, the standard strategy for a UDP flood might be increasing the bandwidth, but for a UDP flood with a higher intensity, it would be to moderate UDP traffic using a flow rule. Each attack is repeated ten times to get an estimate of the learning curve for the reinforcement learning agent. There is a lot of variance in the number of interactions made, and the reinforcement learning agent does not take too many interactions as well. The main reason for it is the number of actions in the framework; at present, the framework supports two major actions. The first one is bandwidth management in the computer network and the other one is flow rule management. There are different flow rules under the flow rule management action, and each of these flow rules are selected by looking at the protocol-specific 43 features obtained during traffic classification. It is possible that, with the increase in the number of actions, the reinforcement learning agent takes more interactions to arrive at the correct action for the particular scenario. The learning curve of the agent w.r.t. various network attacks. The last test which was carried out on the framework was one of bandwidth allocation. It is possible that at some point in time, the computer network is busier than usual. Maybe all the users were streaming live videos or playing games. This framework was developed with a major focus on handling different types of network attacks, but it also has an additional feature of bandwidth management. It can be configured to act like a resource allocation agent as well. Table 1 shows the various values of the bandwidth suggested by the reinforcement learning agent under different network congestion scenarios. Table01 – Evaluating Optimal Bandwidth Allocation Case

Network Congestion (%) Suggested Bandwidth(Mbps) 1 99.98 92-96 2 85.47 79-83 3 50.32 41-43 4 10.39 4-7 It is clear from all the results that were obtained, that the framework works reasonably well against various DoS attacks. However, there were challenges faced during the process to obtain such results. These challenges include simulation techniques for the various attacks to the deployment of the framework and the learning of the reinforcement learning agent. Overall, the agent counters the network attacks to a greater extent. Hence, it gives sufficient background on further implementation of an intelligent network security module with SDN.

TABLE I
EVALUATING OPTIMAL BANDWIDTH ALLOCATION

Case	Network Congestion(%)	Suggested Bandwidth(Mbps)
1	99.98	92-96
2	85.47	79-83
3	50.32	41-43
4	10.39	4-7

IV. CONCLUSION

This paper presented the overview of intelligent threat response.

ACKNOWLEDGEMENT

We would like to thank Mr. Pradeep Doss for his continuous guidance and support throughout my time as an under graduate student. I am greatly indebted to Mr. Pradeep Doss for all the technical concepts and research styles that I learned from her. Another person who had an impact on my work is Mr. Manikandan. I would like to thank him for his valuable insights and comments on the algorithmic approach for this work.

REFERENCES

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, ‘‘Openflow: Enabling innovation in campus networks,’’ SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69--74, Mar. 2008.
[2] L. H. Newman, “What we know about friday’s massive east coast internet outage,’’ Oct 2016.
[3] D. Bisson, ‘‘The 5 most significant ddos attacks of 2016,’’ Nov 2016.
[4] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. Cambridge, MA: MIT press, 1998, vol. 1, no. 1.
[5] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, ‘‘A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies,’’ Knowl. Eng. Rev., vol. 21, no. 2, pp. 97--126, Jun. 2006.