Defending SDN-based IoT Networks Against DDoS Attacks Using Markov Decision Process

Jianjun Zheng
Computer Science Department
Texas Tech University
Lubbock, Texas, USA
Email: jianjun.zheng@ttu.edu

Akbar Siami Namin
Computer Science Department
Texas Tech University
Lubbock, Texas, USA
Email: akbar.namin@ttu.edu

***Abstract*—The emerging Internet of Things (IoT) has increased the complexity and difficulty of network administration. Fortunately, Software-Defined Networking (SDN) provides an easy and centralized approach to administer a large number of IoT devices and can greatly reduce the workload of network administrators. SDN-based implementation of networks, however, has also introduced new security concerns, such as increasing number of DDoS attacks. This paper introduces an easy and lightweight defense strategy against DDoS attacks on IoT devices in a SDN environment using Markov Decision Process (MDP) in which optimal policies regarding handling network flows are determined with the intention of preventing DDoS attacks.**

***Index Terms*—Internet of Things, DDoS, Markov Decision Process, Software-Defined Networking**

I. Introduction

In traditional network architecture, the data processing functionalities and the control logic governing transmission of flows are incorporated into network devices resulting in a distributed and highly-coupled system. Although this type of design is beneficial for performing reliable and efficient network operations, it can also create a heavy and expensive workload for administrators in order to maintain and update every network device. In particular, it is even a big hurdle for administrators when the complexity and scale of the underlying network increases. On the other hand, the emerging paradigm of Internet of Things (IoT) not only added more complexity to networks, but also introduced several attacks types and thus increased attack surfaces in the network. As an eye-opening fact, there was 600% increase in attacks against IoT devices between 2016 and 2017 [1].

Software-Defined Networking (SDN) [2] is an emerging network design aiming to solve the challenges in the current network architecture. It creates a flexible, programmable and secure network to meet the needs for more innovation and better security. In the software-defined networking architecture, the control logic in each network device is transfered into a centralized controller, leaving the network devices to only perform the data forwarding activities with respect to the flow-rules received from the centralized controller. The communication between the controller and the network switches is achieved through application programming interfaces (APIs). The most successful example of such an API is OpenFlow [3]. Each OpenFlow switch has multiple flow tables. Furthermore, in each table every row stores a packet-handling rule (i.e., flow-rule) defined by the controller. The controller also inserts and updates the flow-rules upon request from the switch.

The separation of the control logic from the data forwarding plane enables network administrators to manage and configure the entire network through a centralized controller and thus reduce the workload and at the same time increase the efficiency of their jobs. These features make SDN a good candidate for facilitate administration of IoT devices. On the other hand, the centralized design architecture of SDN also introduces new security concerns, such as the information leakage, DDoS attack, and more [4].

In SDN, each switch relies on the controller to make decisions on how to process the received data packets. Hence, any failure of the controller or the communication between the controller and the switch can make the whole network nonoperational. Distributed Denial of Service (DDoS) attack is a commonly used attack in SDN aiming to overload the controller with a large number of specially crafted flow requests [5] or IP packets [6] so that the controller becomes unresponsive to the legitimate requests from the network switches.

The mitigation strategies against DDoS attack has been discussed in several research papers. Most of the introduced approaches utilize flow analysis techniques to detect DDoS attacks and therefore alert the network administrators. The network administrators then create new mitigating flow rules, which will be sent to all network switches via the controller.

A possible challenge in the detection-based approaches of DDoS attacks is that the detection stage usually requires a large number of data packets to be analyzed to improve the detection accuracy. Another challenge is long response time due to the intensive and deep packet analysis of large amounts of data packets.

Inspired by [7] and [8], we propose to use Markov Decision Process to optimize the flow traffic in SDN based on flow traffic parameters and alert network administrators whenever the flow traffic needs to be optimized. Our approach does not require large amounts of data or deep packet analysis, so its response time is short and can detect and then alert the network administrators of potential DDoS attacks at the early stage.

The rest of the paper is organized as follows. In Section II

978-1-5386-5035-6/18/$31.00 ©2018 IEEE

we describe existing approaches to DDoS attacks in SDN. In Section III we introduce our MDP-based model. We present our simulation experiments and results in Section IV. Finally, Section V concludes the paper and presents future research opportunities.

II. Related Works

The mitigation of DDoS attacks in SDN has been studied using various approaches and techniques in which the effective detection of DDoS attacks is a key critical concern. Braga et al. [9] point out that the detection of DDoS attacks is slow and difficult because it requires a large amount of data to be analyzed to distinguish between the normal data packets and the attacker-crafted data packets. The authors [9] then propose a lightweight method to detect DDoS attacks by using some flow features stored in the flow tables of each switch. Since the controller has the direct access to the flow tables, it can retrieve the flow features in an efficient way and then use them to identify the abnormal flow traffic.

Fonseca et al. [6] argue that the idea of centralized design of SDN may not be resilient because of the issue of the single point of failures. Therefore, they propose to build a replication component called CPRecovery to improve the network resilience. When a primary controller is under an intensive attack, the replication component can seamlessly transit all information to a backup controller. However, as pointed out by Yao et al. [10], it may not be reliable to use multiple controllers to defend against DDoS attacks because it is possible that a DDoS attack on one controller can lead to the cascading failures of multiple controllers.

In this paper, we introduce a theoretical scheme based on Markov Decision Process to optimize the flow traffic for each controller by estimating the involved flow parameters. In the case of a DDoS attack, even at an early stage of the attack, the flow traffic would change and the MDP would be able to detect the abnormality in an efficient way and alert the network administrator to take necessary actions. The high efficiency and low overhead of our approach is beneficial to network administrators when they have to administer a great number of IoT devices connected to SDN.

III. Markov Decision Process Model

In this section, we formulate our approach as a discrete, finite-state, and finite-action Markov Decision Process (MDP) as a 4-tuple (S, A, P, R), where:

- S is a finite set of states.
- A is a finite set of actions.
- P is the probability of a state transitioning to a new state upon performing an action.
- R is the expected immediate reward received after state transition, due to the control action performed.

A. MDP States

Three parameters are selected [11] to formulate the finite set of states: 1) Used Flow Entry Size (F), 2) Flow Queue Size (Q), and 3) Transmitted Packets Count (T). The finite set of states in our MDP will be the Cartesian product of the finite sets of F, Q, and T:

$$S = F \times Q \times T \tag{1}$$

B. MDP Actions

At each fixed time period, the controller determines if the system would transit to the next state based on the current state.

$$A \in \begin{cases} 1 & \textit{Transit} \\ 0 & \textit{Stay} \end{cases} \tag{2}$$

C. Reward Function

The immediate reward, R, received after state transition is related to the three flow parameters F, Q, and T:

$$R = R_F + R_Q + R_T \tag{3}$$

Since each parameter has different impact on the reward, we use weighted reward function to take into account the impacts:

$$R = w_f R_F + w_q R_Q + w_t R_T \tag{4}$$

The Used Flow Entry Size is an indicator of the traffic load. A heavy traffic consumes more flow entries than a light traffic and thus a light traffic is preferable. We define the reward associated with the Unused Flow Entry Size (F) as follows:

$$R_F = \begin{cases} 1 & 0 \le f \le F_{min} \\ \frac{F_{max} - f}{F_{max} - F_{min}} & F_{min} < f < F_{max} \\ 0 & f \ge F_{max} \end{cases} \tag{5}$$

where:

- f is the unused flow entry size at the given state.
- F_{min} is the minimum acceptable unused flow entry size.
- F_{max} is the maximum acceptable unused flow entry size.

Similarly, we define the rewards associated with the Flow Queue Size (Q) and the Transmitted Packets Count (T):

$$R_Q = \begin{cases} 1 & 0 \le q \le Q_{min} \\ \frac{Q_{max} - q}{Q_{max} - Q_{min}} & Q_{min} < Q < Q_{max} \\ 0 & q \ge Q_{max} \end{cases} \tag{6}$$

where:

- q is the queue size at the given state.
- Q_{min} is the minimum acceptable flow queue size.
- Q_{max} is the maximum acceptable flow queue size.

$$R_T = \begin{cases} 0 & 0 \le t \le T_{min} \\ \frac{T_{max} - t}{T_{max} - T_{min}} & T_{min} < t < T_{max} \\ 1 & t \ge T_{max} \end{cases} \tag{7}$$

where:

- t is the transmitted packet count at the given state.
- T_{min} is the minimum acceptable transmitted packet count.
- T_{max} is the maximum acceptable transmitted packet count.

D. Bellman Optimality Equations

An *optimal policy* π^* is a control action $a \in A$ that generates the maximum expected total discounted reward and is expressed by *Bellman Optimality Equation* [12]:

$$V^*_{i+1}(s) = \max_{a \in A} \sum_{s' \in S} P(s, a, s')[R(s, a, s') + \gamma V^*_i(s')] \quad (8)$$

where:

- $P(s, \pi, s')$ is the transition probability starting from state s and ending at state s' after following policy π.
- $R(s, \pi, s')$ is the expected rewards received after state transition from s to s' after following policy π.
- γ is the discount factor.

E. Solving MDP

The optimal policy can be obtained by solving the MDP problem which is expressed in Equation (8). Commonly used methods to solve Equation (8) include dynamic programming, policy iteration, and value iteration. Due to its simplicity, in our approach we use the value iteration to solve the underlying MDP.

IV. Simulation Experiments

In this section, we presents the results of a simulation using combinations of the three flow parameters and analyze the impact of each parameter on the total expected discounted reward.

A. Experiments Parameters

The following parameters are used in the simulation:

- $F_{min} = 0$
- $F_{max} = 0.9$, 90% of the total flow entry size
- $Q_{min} = 0$
- $Q_{max} = 0.90$, 90% of the total flow queue size
- $T_{min} = 0.3$, 30% of the acceptable total transmitted packet count
- $T_{max} = 0.9$, 90% of the acceptable total transmitted packet count

We arbitrarily choose 4 values between the minimum and the maximum values of each parameter to formulate the finite sets:

- $F = \{0.35, 0.55, 0.75, 0.95\}$
- $Q = \{0.35, 0.55, 0.75, 0.95\}$
- $T = \{0.35, 0.55, 0.75, 0.95\}$

The Cartesian product of F, Q, and T will generate $4 \times 4 \times 4 = 64$ states, however, for simplicity purposes, in this simulation we arbitrarily choose 4 states from the 64 states as follows:

- $S_1 = \{f = 0.35, q = 0.35, t = 0.35\}$
- $S_2 = \{f = 0.55, q = 0.55, t = 0.55\}$
- $S_3 = \{f = 0.75, q = 0.75, t = 0.75\}$
- $S_4 = \{f = 0.95, q = 0.95, t = 0.95\}$

The MDP model with these 4 states is shown in Figure 1 where the solid lines represent the "Transit" action path from one state to another ones (including itself) with an equal transition probability (25%) and the dashed lines represent the "Stay" action path from one state back to itself with transition probability of 100%.

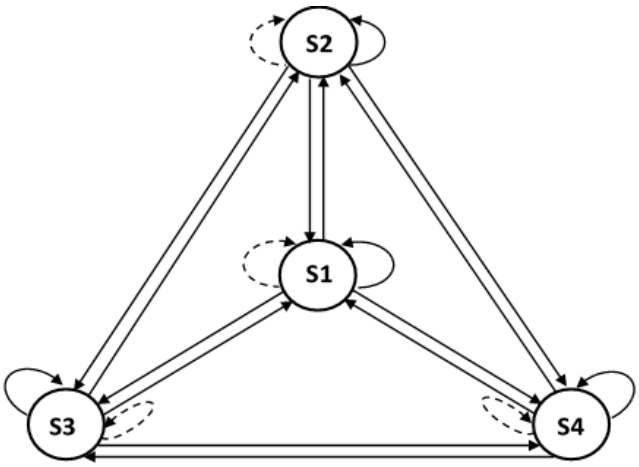

Fig. 1: The MDP model with 4 states.

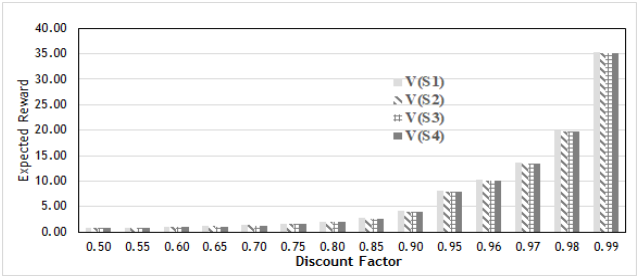

Fig. 2: Impact of discount factor γ on expected total reward.

B. Experiments Results

1) Discount Factor: The *discount factor* in MDP, denoted by $\gamma \in (0, 1)$, indicates the importance of the future rewards in comparison to the present ones. Smaller γ means the reward received in the future will be less significant than the present reward. This indicates that the reward should be collected sooner than later. Figure 2 depicts the impact of the discount factor on the total expected rewards. It shows that when the discount factor is smaller than 0.9, the total expected reward for each state does not increase much as the discount factor increases. However, when the discount factor is greater than 0.9, even a small increase in the discount factor will result in a big increase for the total expected rewards.

2) Reward Function Weight: Figure 3 depicts the impact of the reward weight of F, Q, and T on the expected total rewards. When the three weights are the same $w_f = w_q = w_t = \frac{1}{3}$, the impact on the expected total reward of each state is almost the same and this can be confirmed by Equation 4. The figure also shows that the larger w_t has bigger impact on expected total reward of the state S_4 than the expected total rewards of the other three states, but the larger w_f or w_q has bigger impact on the state S_1 than the other three states.

V. Conclusions and Future Work

In this paper, we presents a MDP-based optimal policy for detecting and mitigating potential DDoS attacks targeting IoT

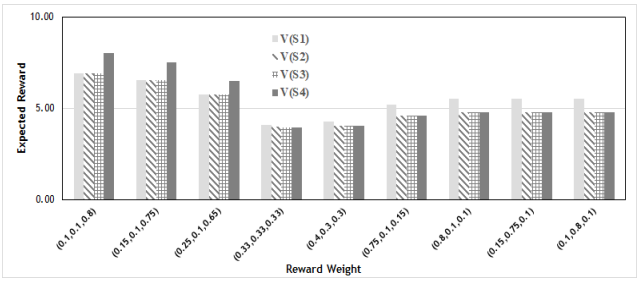

Fig. 3: Impact of reward weight on expected total reward.

devices on a Software-Defined Network. Our approach focuses on keeping the flow traffic optimized and alerting system administrators otherwise, therefore, it can detect potential DDoS attacks at the early stage and gives administrators enough time to mitigate the attacks quickly. Our simulation results show that, by adjusting the discount factor and the reward weight factor, administrators can control how the system transit from one state to another.

In the future research, we plan to include more flow parameters to formulate more realistic states. In addition, we also plan to implement machine learning technique to automate the detection and mitigation process.

ACKNOWLEDGEMENT

This project is funded in part by a grant (Awards No: 1516636 and 1564293) from National Science Foundation.

REFERENCES

[1] (2018) Internet security threat report by symantec corporation. [Online]. Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf

[2] N. McKeown. (2011) How sdn will shape networking. [Online]. Available: https://www.youtube.com/watch?v=c9-K5O_qYgA

[3] 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. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[4] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2317–2346, Fourthquarter 2015.

[5] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 165–166. [Online]. Available: http://doi.acm.org/10.1145/2491185.2491220

[6] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient openflow-based networking," in *2012 IEEE Network Operations and Management Symposium*, April 2012, pp. 933–939.

[7] A. Munir and A. Gordon-Ross, "An mdp-based application oriented optimal policy for wireless sensor networks," in *Proceedings of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS '09. New York, NY, USA: ACM, 2009, pp. 183–192. [Online]. Available: http://doi.acm.org/10.1145/1629435.1629461

[8] J. Zheng and A. Siami Namin, "A markov decision process to determine optimal policies in moving target," in *Proceedings of 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Toronto, ON, Canada: ACM, 2018. [Online]. Available: http://doi.acm.org/10.1145/3243734.3278489

[9] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *IEEE Local Computer Network Conference*, Oct 2010, pp. 408–415.

[10] G. Yao, J. Bi, and L. Guo, "On the cascading failures of multi-controllers in software defined networks," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, Oct 2013, pp. 1–2.

[11] ONF. (2015) Openflow switch specification version 1.5.1. [Online]. Available: https://www.opennetworking.org/software-defined-standards/specifications/

[12] R. E. Bellman, *Dynamic Programming*, reprint ed. Princeton University Press, 2010.