Dynamic Game based Security framework in SDN-enabled Cloud Networking Environments

Ankur Chowdhary Sandeep Pisharody
Adel Alshamrani Dijiang Huang

School of Computing, Informatics and Decision Systems Engineering
Arizona State University, Tempe, AZ
<achaud16, spishar1, aalsham4, dhuang8>@asu.edu

ABSTRACT

SDN provides a way to manage complex networks by introducing programmability and abstraction of the control plane. All networks suffer from attacks to critical infrastructure and services such as DDoS attacks. We make use of the programmability provided by the SDN environment to provide a game theoretic attack analysis and countermeasure selection model in this research work. The model is based on reward and punishment in a dynamic game with multiple players. The network bandwidth of attackers is downgraded for a certain period of time, and restored to normal when the player resumes cooperation. The presented solution is based on Nash Folk Theorem, which is used to implement a punishment mechanism for attackers who are part of DDoS traffic, and reward for players who cooperate, in effect enforcing desired outcome for the network administrator.

Keywords

Software Defined Networking (SDN), Game Theory, Distributed Denial of Service (DDoS), Moving Target Defense (MTD), Cloud Systems

1. INTRODUCTION

Distributed Denial of Service (DDoS) is a major security problem affecting networks. Some recent cases include a massive DDoS attack on DNS provider Dyn in October 2016, and an attack on the website krebsonsecurity.com which was of magnitude 650 Gbps. The attackers leverage sophisticated botnets such as Leet to send massive traffic to the victim, thus overwhelming the limited capacity of the victim. Traditional networks usually incorporate mechanisms such as firewall, Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) to detect and counter such attacks. Since there are multiple entities (such as routers, firewall, switches etc.) involved in enforcing the security mechanism, and cooperative sharing of attack information between devices that have data and control plane embedded in a single device is hard; it becomes very difficult to detect and counter such attacks. In addition, each different device may have vendor specific command and control mechanism.

Software Defined Network (SDN) provides separation between data and control plane [13]. A logically centralized controller such as OpenDaylight (ODL) is used for taking control decisions such as routing, load balancing, firewall policies, IDS and Service Level Agreement (SLA) [14]. The data-plane, which is involved with traffic forwarding remains a part of devices such as switches and routers. The control decisions taken by control plane are enforced by devices.

In this work we model the DDoS attack as a dynamic game between the attacker and administrator. The attacker has a goal of targeting critical infrastructure by sending a huge volume of traffic through bots distributed inside or outside the target environment. While we can assume some of the bots are detected by the IDS based on signature match, modern DDoS botnets are very stealthy in nature. To this end, we introduce a game theoretic model which will help uncover entire botnet, and rate limit traffic from these malicious users/bots. The concept of reward and punishment which is used in game theoretic models to enforce cooperation between firms has been employed in this research work. To sustain mutually desirable outcomes, the agents/users with undesired behavior are punished. Various game theoretic approaches that can be used to model the system have been shown in the figure below.

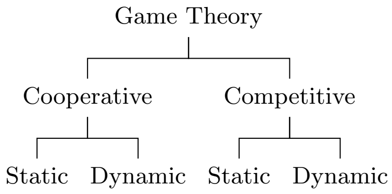

Figure 1: Game theory classification.

We consider our system to be a dynamic multi-player game. A single network administrator (SDN Controller in our case) is playing against multiple players, some of which are attackers. We deploy administrator's strategy in form of Openflow rules. Since some of the countermeasures deployed as part of defense strategy can conflict with some existing rules, in our previous work [15] we use a flow rule conflict detection and resolution algorithm to first detect and then

SDN-NFV Sec'17, March 22-24, 2017, Scottsdale, AZ, USA
© 2017 ACM. ISBN 978-1-4503-4908-6/17/03…$15.00
DOI: http://dx.doi.org/10.1145/3040992.3040998

eliminate flow rule conflicts. The attackers have some incentive of deviating from normal behavior at a particular instant but admin has control over network resource optimization, and he can counter attacker's move by limiting his available bandwidth in next time instance of play.

In Section 2, we discuss a motivating example and some key terminology. Section 3 introduces a reward-punishment model based on Nash Folk theorem. Section 4 consists of implementation and Section 5 discusses the evaluation of a game theoretic model on a small test environment. We discuss some related work in dealing with DDoS attacks in SDN and compare our model to them in Section 6. Finally, conclusion of this work and direction for the future are discussed in Section 7.

2. BACKGROUND

In this Section, we introduce some background concepts used in this research work.

Definition 1. *We define a N player extensive form repeated game G with perfect information between multiple players as* $G = \{N, A_i, u_i\}$ *where* $N = \{1, 2, \ldots n\}$ *denotes number of players,* $a_i \in A_i$ *is the action set available to player i.* $u_i : a_i \mapsto R_i$ *is the payoff function that maps actions to reward value R.*

Definition 2. *We consider that the game has been played to t periods of time, and define game history in an instance t as* $h_t = \{a_1, a_2, .., a_{t-1}\} = A^{t-1}$*. This denotes actions taken by a player until now.* $H_1 = \{\emptyset\}$*, and* $H = \sum_{t=1}^{\infty} H_t$*.*

A player i prefers an action a^t over b^t if $u_i(a^t) \geq_i^* u_i(b^t)$. The payoff profile for a player is considered feasible in this game, i.e. convex combination of payoff profiles of outcome of A, $s = \sum_{a \in A} \alpha_a u(a)$ such that $\sum_{a \in A} \alpha_a = 1$. Here s represents the strategy vector for a player.

We illustrate the game used in this work with an example of two players, who take turns to decide on an action. Assume the players P_1 and P_2 correspond to an attacker and administrator.

Definition 3. *The strategy vector of a player i,* s_i^* *is best response to strategy vector of all other players* s_{-i}^* *if* $u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*)$ *for all* s_i*. This vector is a Nash Equilibrium if the relation holds for all* s_i *and all i.*

For the two player example, we consider both players would try to play Nash Equilibrium value against each other. Let us consider utility in terms of network bandwidth for this game. P_1 has actions $a_1 = \{$*Cooperate, Defect*$\}$ and P_2 has action set $a_2 = \{$*Cooperate, Defect*$\}$. As long as any player behaves in a benign manner, the administrator will allow normal bandwidth to the player. If attack from some malicious node in a network is detected by administrator, he/she will play a strategy of Rate-Limiting the attacker's bandwidth available. Sample payoff matrix in normal form has been shown in the Figure 1, where B denotes the total network bandwidth.

Since we consider a dynamic game with perfect information, we need to consider an extensive form of the game. For this particular example, consider that the attacker chooses action a_2^1, that is *defect* against P_2 who assumes all users for network behave in a benign way and chooses a_2^1, i.e. *cooperate*. However, in second period $t = 2$ he/she checks the history H of P_1 defined above, observes the earlier *defect* action, and selects a_2^2 irrespective of action chosen by P_1. To punish the P_1 further, he chooses same action as $t = 2$ in period $t = 3$. The deviation from normal behavior is considered a *trigger strategy.*

		Player 2	
		a_2^1	a_2^2
Player 1	a_1^1	$(\frac{B}{2}, \frac{B}{2})$	$(\frac{3B}{4}, \frac{B}{4})$
	a_1^2	$(\frac{B}{4}, \frac{3B}{4})$	$(\frac{B}{5}, \frac{4B}{5})$

Table 1: Normal form representation of Attacker and administrator Payoff's

We define the minmax payoff in a Game before introducing definition of *Nash Folk theorem* which we use as the basis for our deterrence algorithm for malicious users.

Definition 4. *The minmax payoff value of a player is the lowest payoff value that can be forced upon a given player by all other players. This is denoted by value* $v_i = \min_{a_{-i} \in A_{-i}} \max_{a_i \in A_i} u_i(a_{-i}, a_i)$*.*

Since the future payoff is less desirable compared to the current payoff value, we use discount factor $\delta \in (0, 1]$. We use this variable to motivate the definition of the Nash Folk theorem.

Definition 5. *The Nash Folk Theorem for a an extensive form game states that payoff profile which was present for Nash Equilibrium denoted by* w_i *can be enforced upon a given player in the long term by punishing him for a given period of time. If the player* P_1 *deviates from good behavior, his opponents will use minmax strategy against him/her till the continuous reward value is no better than in the case where he never shows malicious behavior.* $w_i \geq v_i + \sum_{t=1}^{T} \delta^t \times min_{a_{-i} \in A_{-i}} max_{a_i \in A_i} u_i(a_{-i}^t, a_i^t)$*. Here* v_i *denotes defection payoff for attacker at* $t = 0$*. The second equation on right hand side i.e.* $min_{a_{-i} \in A_{-i}} max_{a_i \in A_i} u_i(a_{-i}^t, a_i^t)$ *shows that* P_1 *has been min-maxed by other players from* $t = \{1, T\}$ *if he defect at* $t = 0$*. The value for* T*, i.e. time periods for which punishment should be carried out can be derived by solving this linear inequality.*

Extensive form game tree for such a game is depicted in Figure 2 below. The P_1 represents the attacker and P_2 represent the network admin in this figure.

The path in green indicates the administrator's choice of action when the user behaves normally. We can see the payoff matrix from Table 1 and check average bandwidth at end of period $t = 2$, which will be $\frac{B}{2}$ since the user is cooperating at $t = \{0, 1\}$. In case user P_1 defects (behaves as an attacker), he/she will gain bandwidth $\frac{3B}{4}$ at $t = 1$, but the administrator P_2 will punish user at $t = 1$, resulting in a bandwidth $BW = \frac{B}{5}$ during the next time period. The resulting average BW will be $\frac{1}{2} \times \{\frac{3B}{4} + \frac{B}{5}\} = 0.475B$, which is lower than $0.75B$, the bandwidth had P_1 behaved in a cooperative fashion. The path in red shows attack countermeasure procedure followed by P_2. In the long term the attacker will be better off by behaving normally (sending no malicious traffic) if we deploy rate limiting mechanism using this scheme.

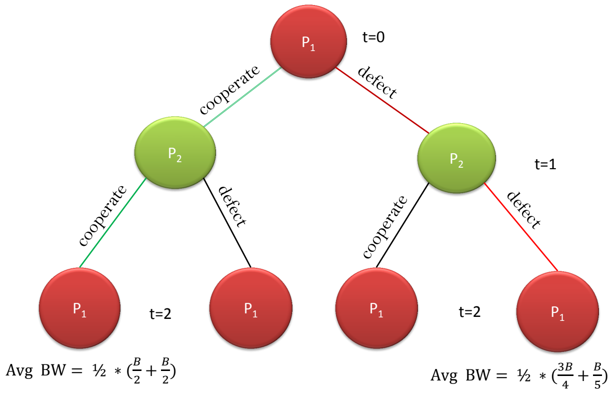

Figure 2: Extensive form of Dynamic Game

3. SYSTEM ARCHITECTURE & MODEL

The system architecture as shown in Figure 3 consists of ODL [9] based SDN platform. The southbound APIs are used for interacting with data-plane elements. We assume that the switches for our architecture are Openflow enabled. These switches interact with the hosts inside or outside the network. As can be seen, some hosts send normal traffic to switch while others act as part of a DDoS botnet. The SDN controller platform consists of several elements such as topology manager to perform any network topology reconfiguration, network config element to ensure persistence of the current network configuration.

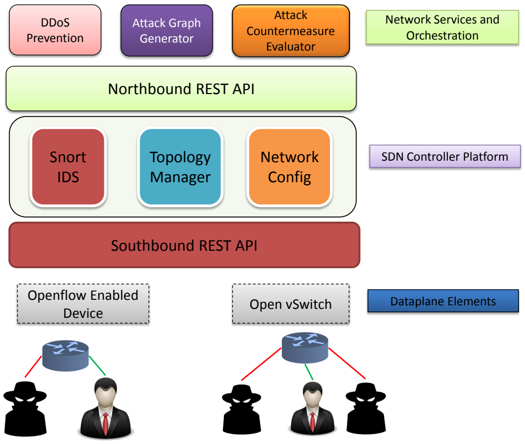

Figure 3: System Architecture

The main element from this layer used by us is Snort IDS [3]. We consider a signature based detection mechanism to collect and categorize network traffic as malicious or benign. This information is passed on to a network service and orchestration layers through northbound REST APIs. We consider three types of DDoS attacks in this work namely, SYN-Flood Attack, UDP Flood Attack, ICMP Flood Attack. The example below shows Snort signature used as a trigger for DDoS prevention mechanism based on the game theoretic approach.

```
alert tcp $HOME_NET any -> $HOME_NET 80
(flags: S; msg:"Possible TCP DoS";
flow: stateless; threshold: type both,
track by_src, count 70, seconds 10;
sid:10001;rev:1;)
```

Once Snort triggers the DDoS defense mechanism, the SDN flow table will be updated to rate limit the traffic from a particular source to destination in the home network.

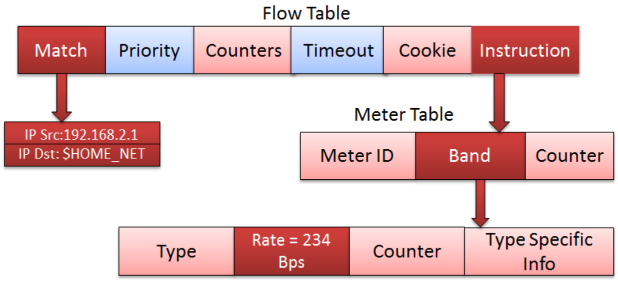

Figure 4: Traffic Rate Limiting in SDN

Figure 4 shows various entries for a given SDN flow table [1]. The match field is used for matching ingress port and packet headers. In this example, we have snort alert for IP Address `192.168.2.1`, which has been classified as attacker's IP. The `Instruction` field of flow table is added. The rate limit decided by the algorithm is set in `Rate` sub-field of Band Field in `Meter Table`. In the Figure 4 Once the period of punishment decided by the administrator is over for this particular IP address, the `Rate` sub-field will be updated to default traffic burst rate. We use the REST API to push these values continuously to ODL controller.

4. IMPLEMENTATION

System setup used and algorithmic details for Nash Folk theorem based DDoS Prevention are discussed in this section.

4.1 Openflow Rate Limiting Algorithm

The Openflow based rate limiting algorithm described in this section consists of two procedures. One procedure SET-RATE-LIMIT-METER is used for invoking meter with specified meterID in the corresponding flow table. By default the meter table is optional for a Flow Table. The host behaving normally will not face any decreased bandwidth. However, if the host is behaving maliciously and a trigger event based on Snort IDS alert is used to invoke meter with ID 1, the Bandwidth would be set to a value decided by the rate limit policy, depending upon cooperation or defection of malicious host in the current and subsequent periods. Lines 7-10 depict the preset values of bandwidth for host i P_i. The procedure NASH-FOLK-RATE-LIMIT based on the greedy approach loops through all flow tables in invoking meter with rate limiting threshold if matching source host is found in the list of malicious hosts from a Snort IDS (lines 11-17). The procedure of punishment is

Algorithm 1 SDN-DDoS-Rate-Limit-Algo

```
1:  procedure SET-RATE-LIMIT-METER(meterName, bandID, bandRate)
2:      $MeterName \leftarrow this.meterName$
3:      $mbh \leftarrow MeterBuilder.meterBandHeader()$
4:      $mbh.setBandID(this.bandID)$
5:      $mbh.setBandRate(this.bandRate)$
6:  procedure NASH-FOLK-RATE-LIMIT(
        )
7:      $u_i(coop, coop) \leftarrow B_c$ {Host $P_i$ cooperates}
8:      $u_i(coop, def) \leftarrow \{B_{cd} > B_c\}$ {Host $P_i$ defects}
9:      $u_i(def, coop)] \leftarrow \{B_{dc} < B\}$ {Host $P_i$ defected at $t_{k-1}$}
10:     $u_i(def, def) \leftarrow \{B_{dd} < B_{dc}\}$ {Host $P_i$ defected at $t_{k-1}$ and $t_k$ }
11:     for i ∈ [0,n-1] do
12:         $ft \leftarrow FlowTable_i$
13:         if ft.match.src_ip ∈ DDoSTrigger(src_ip) and $\sum_{t=0}^{k} \delta^t u_i(coop, coop) \leq u_i(coop, def) + \sum_{t=1}^{k} \delta^t u_i(def, \{coop, def\})$ then
14:             x ← ft.Instruction()
15:             x.SET-RATE-LIMIT-METER(”RLMeter”, 1, $u_i(det, coop)$)
16:         else
17:             x.SET-RATE-LIMIT-METER(”RLMeter”, 1, $u_i(det, det)$)
```

carried out for k instances of time where value of k is determined by equation $\sum_{t=0}^{k} \delta^t u_i(coop, coop) \leq u_i(coop, def) + \sum_{t=1}^{k} \delta^t u_i(def, \{coop, def\})$ in line 13. This linear equation ensures that defecting host is no better than for case of normal behavior at end of k periods of punishment.

5. EVALUATION

We used network simulator [2] and ODL controller on Ubuntu 16.04 OS. We conducted two experiments in our evaluation. The first experiment uses the algorithm proposed in Section 4 to deal with ICMP flood attacks. The second experiment uses the same algorithm for TCP SYN flood and UDP flood based attacks on a fat tree topology. The variation in topologies for both experiments is used to check the generality of our solution.

5.1 Experiment 1: ICMP Flood DDoS Attack on Linear Topology

In our first experiment we created a linear topology in mininet environment with the number of hosts varying from 50 to 500. The topology had a single layer of hosts, all connected to one switch. An example of linear topology can be seen in Figure 5.

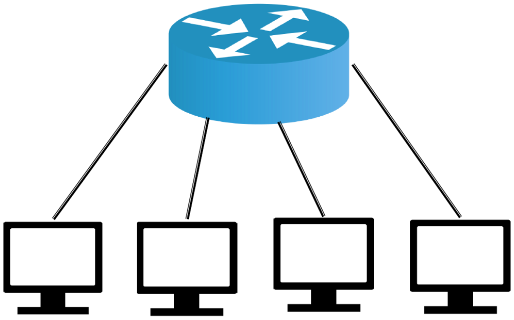

Figure 5: Linear Topology in SDN

We created an attack script in python, which uses multiprocessing to spawn shell for each host and send ICMP traffic of large packet sizes to a single host in the network. The traffic is port mirrored to a dummy port. The IDS intercepts the attack signature for ICMP flood DDoS attack and passes information to ODL controller. ODL application for DDoS mitigation decreases the traffic rate by a factor δ consecutively until the long term average for traffic is within normal traffic burst from a provided host. In this particular experiment we used the value of damping factor $\delta = 0.8$. This scheme punishes all the attacking hosts by degrading traffic throughput gracefully, instead of blocking the traffic entirely or rate limiting to a fixed value, which can affect the traffic from legitimate users.

Number of Attacking Hosts	ICMP Flood Traffic (Mb/s)	ICMP Traffic post Rate Limit(Mb/s)
50	39.49	1.33
100	79.85	2.70
200	163.69	5.54
300	241.17	8.122
400	321.96	10.83
500	467.16	15.69

Table 2: Number of Hosts vs ICMP Traffic at T=30s post attack.

The table shows that traffic burst at target for 100 hosts is 79.85 Mbps when there is no attack prevention mechanism to deal with DDoS attack. However, once the trigger for Rate Limit is set by IDS, the traffic decreases to 2.70 Mbps, a decrease of factor 30. Similarly, as the number of attacking hosts increase from 100 to 500, the throughput of DDoS attack increases from 79.58 Mbps to 467.16 Mbps, which shows a linear scaling in attack traffic. The Rate Limit (RL) algorithm quickly adapts to increased traffic and decreased corresponding traffic limit for 500 hosts to a value 15.69 Mbps. On comparing attack traffic and corresponding rate limited traffic for 500 hosts, we can observe a decrease by a factor of 29. The experiment shows a successful countermeasure using a game theoretic approach of punishing the attacker on a sufficiently large network.

5.2 Experiment 2: TCP/UDP Flood DDoS Attack on Fat Tree Topology

Most of the attacks faced by organizations target some DNS server to send large burst of TCP or UDP packets to target host. Also, since data centers follow fat tree topology architecture, we conducted experiment to test our algorithm on a fat tree topology using mininet, with depth=3, and fanout=3. We used a damping factor $\delta = 0.9$ for this experiment.

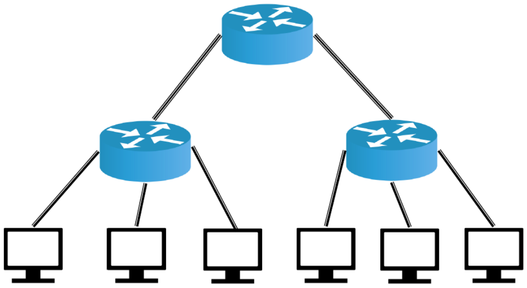

Figure 6: Fat Tree topology in SDN

In this experiment the normal allowed limit for TCP and UDP traffic set by SDN controller was 3.0 Mbps. We ran TCP SYN Flood DDoS attack on a topology of 64 hosts. The traffic decay once the rate limiting algorithm based attack countermeasure mechanism is triggered based on IDS alerts is plotted in red in Figure 7. Initially DDoS traffic is 157.03 Mbps, which clearly violates the permissible limit. The SDN controller pushes the flow to deal with this attack, and the traffic is reduced to 18.11 Mbps at t=10s. The traffic burst reaches a value 3.02 Mbps at t=50s, which is nearly equal to normal traffic rate allowed on this network.

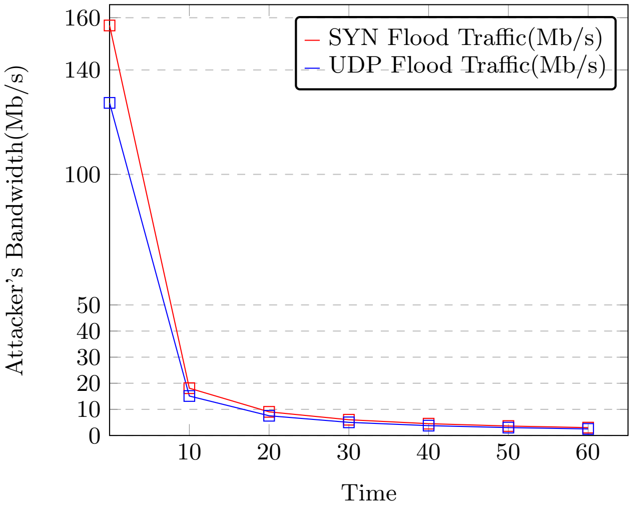

Figure 7: TCP and UDP Flood Attack Mitigation on fat tree topology

Similarly, the traffic pattern for a UDP flood attack starts off around 127.38 Mbps. The rate limit algorithm decreases this value to 15.12 Mbps at t=10s. Traffic rate is further reduced to 4.52 Mbps at t=40s. Finally traffic burst reaches a value 3.01Mbps at t=60s, and algorithm stops further enforcement of rate limit after t=60s. The IDS waits for further intrusion alerts at this point to notify the controller in case, malicious traffic is still sent by attacking hosts. Thus we can see from this experiment that the algorithm will take less than one minute to mitigate TCP and UDP based DDoS attacks on a sufficiently large network.

5.3 Complexity Analysis

The algorithmic complexity will depend upon the number of users in the system. In the worst case, all (N number of users) will behave maliciously. We consider n to be number of flow tables, and k to be upper bound on time for punishing a particular host. The complexity will be $O(N \times n \times k)$. The values n and k will be constant, so we get $c = n \times k$. The complexity will effectively be $O(N \times c) \sim O(N)$. Thus *Openflow Rate Limiting Algorithm* will have linear time complexity. The algorithm will be very fast with guaranteed termination.

6. RELATED WORK

We analysed several works that either use some intelligent framework to deal with active attacks in SDN or use some game theoretic model in network security. In [16], authors combine game theory and Machine Learning (ML) to model attacker's behavior in ML feature space. This work uses spam filtering as a target application to provide defense against current and future attacks. Random host mutation based on SDN platform has been used by Jafarian *et al.* [10] to map real and virtual IP addresses, and make reconnaissance hard for malicious hosts. Chung *et al.* [8] [7] have used proactive defense and countermeasure analysis framework in cloud network. The authors have used z Bayesian framework for attack analysis. Our work is based on Nash Equilibrium based attack model in dynamic games.

Braga *et al.* [5] have used pattern recognition based on Self Organizing Maps (SOM) to filter DoS attack traffic in NOX based Openflow network. The solution is lightweight compared to earlier works on the DoS attack detection based on KDD dataset. We have some concerns about the accuracy of pattern recognition in DDoS attack detection hence we have relied on signature based detection mechanism. FRESCO [17] which has been developed on top of NOX based SDN framework provides modular security to defend against network attacks. Bot Miner service module in FRESCO uses clustering mechanism to detect bots through network level flow analysis. Markov game based framework for two player zero sum game has been discussed by Alpcan *et al.* [4]. Their framework used Markov Chain based attack modeling to send information to IDS, so that the administrator can deal with active attacks.

Kampanakis *et al.* [12] have discussed obfuscation as a possible Moving Target Defense (MTD) strategy to deal with attacks in SDN environment. Authors have discussed OS fingerprinting and network reconnaissance as specific types of attacks in SDN. Random mutations of this nature may, however disrupt any active services and some cost-benefit analysis of MTD strategy is necessary. Another approach based on MTD solution for prevention of DoS attacks on SDN networks has been discussed by [11]. The authors propose the idea of moving secret proxies to new network locations using a greedy algorithm. This solution however is limited to malicious insiders in a network. A scalable attack graph approach to deal with system vulnerability based multi-hop attack has been discussed by Chowdhary *et al.* [6]. The authors use distributed hypergraph partitioning approach to construct an attack graph for a large system, however don't discuss the possible countermeasures to deal with active attacks such as DDoS, NTP amplification, etc., which we discuss as part of this work. In [18] authors use a game theoretic framework to deal with attacks against web applications. This work uses Stackelberg game to model attack and defense.

7. CONCLUSIONS

We analyzed a game theoretic framework based on reward and punishment mechanism which is used successfully in game theoretic modeling. Using a greedy algorithm we solved an optimization problem for rate limiting network bandwidth as a punitive mechanism for misbehaving players in a dynamic network game. The optimization algorithm used in this work, based on Nash Folk theorem, allowed us to degrade network bandwidth gracefully, without applying a static hard limit on network traffic. Our experimental work targeted DDoS attacks, specifically ICMP Flood, TCP SYN Flood, UDP Flood. The algorithm is able to deal with all these attacks based on alerts received from SDN controller. The framework proposed leveraged benefit of network optimization and programmability offered by SDN quite well, and the proposed algorithm can adapt well on varying topologies as demonstrated by Experiments 1 and 2 in Section 5.

We used the damping factor δ to be on higher side $\{0.8, 0.9\}$ in this experimental work to put more weight on future punishment based payoff to the attacker. The normal bandwidth which we used as a baseline for threshold bandwidth was selected by observing normal TCP, UDP traffic in a medium sized network for a time duration of about 10-15 minutes. Both these parameters can have an impact on final results and convergence time of the algorithm. We plan to study the impact of variation in these two parameters in the OpenStack based cloud as an extension of this work.

Our motivation in use of a signature based IDS was to deal with DDoS attacks whose signature can be easily identified. Most of anomaly detection methods we studied prior to the experimental setup of this work suffered from the issue false alarms. We plan to use Artificial Intelligence (AI) based algorithms to train our system for identified attacks and use anomaly detection along with signature based detection mechanism to construct a comprehensive attack mitigation solution as part of future work.

A limitation of our experimental work is the number of host subprocess we can spawn using the multiprocessing thread, which is currently limited to around 500. We plan to leverage a cloud framework based on OpenStack to deal with this scalability concern and analyze the impact of algorithm on dynamic attacks in a real cloud environment.

Acknowledgments

This research is supported by NSF Secure and Resilient Networking (SRN) Project (1528099) and NATO Science for Peace & Security Multi-Year Project (MD.SFPP 984425). S. Pisharody is supported by a scholarship from the NSF CyberCorps program (NSF-SFS-1129561). Adel Alshamrani is supported by King Abdul Aziz University, Jeddah, Saudi Arabia.

8. REFERENCES

[1] Openflow switch specification v 1.3.1. https://www.opennetworking.org/.

[2] Mininet Virtual Network https://www.mininet.org/, 2015.

[3] Snort IDS, https://www.snort.org/, 2017.

[4] T. Alpcan and T. Basar. An intrusion detection game with limited observations. In *Proceedings of the 12th Int. Symp. on Dynamic Games and Applications*, 2006.

[5] R. Braga, E. Mota, and A. Passito. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 408–415. IEEE, 2010.

[6] A. Chowdhary, S. Pisharody, and D. Huang. SDN based scalable MTD solution in cloud network. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, pages 27–36. ACM, 2016.

[7] C.-J. Chung. *SDN-based Proactive Defense Mechanism in a Cloud System*. PhD thesis, Arizona State University, 2015.

[8] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang. NICE: Network intrusion detection and countermeasure selection in virtual network systems. *Dependable and Secure Computing, IEEE Transactions on*, 10(4):198–211, 2013.

[9] L. Foundation. Opendaylight SDN controller. https://www.opendaylight.org/, 2017.

[10] J. H. Jafarian, E. Al-Shaer, and Q. Duan. Openflow random host mutation: Transparent moving target defense using software defined networking. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 127–132. ACM, 2012.

[11] Q. Jia, K. Sun, and A. Stavrou. MOTAG: Moving target defense against internet denial of service attacks. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE, 2013.

[12] P. Kampanakis, H. Perros, and T. Beyene. SDN-based solutions for moving target defense network protection. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, pages 1–6. IEEE, 2014.

[13] D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *proceedings of the IEEE*, 103(1):14–76, 2015.

[14] E. Z. Nick Feamster, Jennifer Rexford. The road to sdn: An intellectual history of programmable networks. In *Proceedings of the ACM SIGCOMM*, pages 87–98. ACM, 2014.

[15] S. Pisharody, A. Chowdhary, and D. Huang. Security policy checking in distributed SDN based clouds. In *2016 IEEE Conference on Communications and Network Security (CNS) (IEEE CNS 2016)*, Oct. 2016.

[16] K. G. Richard Colbaugh. Predictability oriented defense against adaptive adversaries. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 14–17. IEEE, 2012.

[17] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson. Fresco: Modular composable security services for software-defined networks. 2013.

[18] S. G. Vadlamudi, S. Sengupta, S. Kambhampati, M. Taguinod, Z. Zhao, A. Doupé, and G. Ahn. Moving target defense for web applications using bayesian stackelberg games. *CoRR*, abs/1602.07024, 2016.