Dynamic DDoS Defense Resource Allocation using Network Function Virtualization

A.H.M. Jakaria[1], Bahman Rashidi[2], M. Ashiqur Rahman[1], Carol Fung[2], Wei Yang[2]
[1]Tennessee Technological University, TN, [2]Virginia Commonwealth University, VA
ajakaria42@students.tntech.edu, rashidib@vcu.edu, marahman@tntech.edu, cfung@vcu.edu, yangw3@vcu.edu

ABSTRACT

Distributed Denial of Service (DDoS) is one of the most common but destructive kind of cyberattacks. The DDoS attack traffic vastly includes the SYN packets. To handle excessive DDoS traffic without large impact on the benign traffic, it is important to allocate sufficient resource for the defense. With the support from Network Function Virtualization (NFV) technology, this can be done in a significantly lower lower cost, with higher flexibility. In this work, we propose a DDoS defense mechanism based on Network Function Virtualization platform, which is capable to balance with the attack load. Our proposed mechanism uses dynamic network agents to intercept packets when the system experiences DDoS attack. In particular, we focus on the mechanism that deploys virtual agents dynamically according to the attack strength. The corresponding load-balancing algorithm dynamically determines the number of agents to allocate for defending the attack traffic. Our simulation results demonstrate that the mechanism can successfully defeat the DDoS attacks with different attack intensities, while it incurs minimal performance degradation on the benign traffic and keeps the increase in the server's response time sufficiently low compared to that of a successful DDoS attack.

Keywords

Distributed Denial of Service Attacks; Network Defense; Cyber Security; Load Balancing; Network Functions Virtualization; Virtual Networks; Resource Management

1. INTRODUCTION

Distributed Denial of Service (DDoS) is one of the most devastating but frequently used cyberattacks. A huge volume of attack traffic can bring down online services and cause traffic jam to slow down ISP services. This type of attacks are very destructive to the small or medium-sized organizations which lack sufficient resources for high traffic volume. Some recent incidents show that DDoS attacks are becoming increasingly strong and frequent. For example, the Spamhaus attack in 2013 [1] has generated 300 Gbps attack traffic. This number has been become to 600 Gbps in January 2016 [10].

To handle DDoS attacks, sufficient resources shall be reserved to deal with the excessive traffic. For example, the SYN Flood attacks [13] can be handled using dedicated high-capacity firewalls or third-party services, such as Prolexic [2]. However, the firewall solution is not cost-effective because of expensive hardware devices, while the third-party service solution involves with privacy concerns due to the need of redirecting the traffic flows to the external filters. The emerging of Network Function Virtualization (NFV) technology in recent years has brought a new opportunity for DDoS defense solutions.

NFV is a new technology that promotes softwarized network functions. *i.e.*, firewalls and routers are provided through software running on commodity hardware [6]. NFV not only provides the benefit of elasticity, but also reduces the cost by replacing specialized hardware with general purpose computer servers, resulting in a much easier deployment and lower cost. Furthermore, NFV introduces new opportunities for DDoS detection and mitigation. The traditional methods of DDoS detection are limited by the computation capacity of the dedicated hardware, such as firewall and routers. The use of NFV results in much lower cost and higher flexibility in building a defense system against DDoS attacks. In this work, we leverage the capability of the NFV technology and provide a solution, named VFence, to defend DDoS attacks. We focus on the traffic management problem and propose a load-balancing algorithm to evenly distribute traffic load to multiple traffic handling agents to avoid overloading any agent. The algorithm is also energy-ware which avoids using excessive number of agents.

In particular, we present a DDoS defense solution in this paper, which mitigates DDoS attacks through a dynamic traffic filtering network. We utilize the flexibility of NFV to create DDoS traffic filtering agents dynamically to filter spoofed traffic to the targeted online service. External traffic is directed by a dispatcher to one of the agents for processing. Traffic with bogus source IPs or from known attacker IPs will be filtered. Traffic which passes the filtering process is then forwarded to the destination. We propose a dynamic load-balancing algorithm that dispatches packets to corresponding agents. Our experimental results demonstrate that VFence can effectively reduce the attack flow to the target server and mitigates DDoS attacks. The load balancing algorithm can effectively even out the workload on agents and avoid excessive number of agents to be used. The

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/10.1145/3040992.3041000

major contributions of this paper are two fold: (1) developing a dynamic energy-aware load-balancing algorithm to effectively utilize available virtual resource for DDoS defense; (2) providing a novel forwarding table design for the dispatcher that effectively decreases the forwarding table size during attacks.

The rest of this paper is organized as follows: Section 2 discusses the literature review on NFV-based DDoS defense and load balancing solutions. VFence framework is described in Section 3. The dispatching algorithm is presented in Section 4. The evaluation results of our model are presented in Section 5. Then, we conclude the paper in Section 6.

2. RELATED WORKS

A number of works are available in the literature that discuss NFV-based DDoS defense mechanisms. In this section, we briefly discuss existing works on utilizing NFV to defend DDoS attacks. We also review existing proposed load balancing algorithms for this defense.

VNGuard [5] is a framework designed by Deng *et al.* for effective provision and management of virtual firewalls to keep virtual networks (VNs) safe. This approach uses the features of NFV and SDN to define a high-level firewall policy language, finds optimal virtual firewall placements, and adapts virtual firewalls to VN changes. Fayaz *et al.* [7] proposed a flexible and elastic DDoS defense system, Bohatei, that presents the benefits of SDN and NFV in the context of DDoS defense. The authors use NFV capabilities to elastically alter the required scale (*e.g.*, 10 Gbps vs. 100 Gbps attacks) and type (*e.g.*, SYN proxy vs. DNS reflector defense) of DDoS defense mechanisms realized by network functions running on VMs.

A cloud-based architecture [14] was proposed for defense against DDoS attacks. VGuard [8] is a tool for dynamic traffic engineering based on prioritization, which is implemented on a virtual network function (VNF). Liyanage *et al.* introduced NFV-based security apps to protect the LTE architecture [12]. In our previous work [9], we have proposed a technique to filter out DDoS attack traffic by spoofing the TCP handshake with the attacker. This work further proposes a load balancing algorithm to complete the design of the dynamic agent deployment mechanism.

Le *et al.* [11] proposed a correlation-based load balancer to distribute traffic among multiple systems and protect networks against DDoS attacks. However, the challenge is to maintain the load balancing of the systems while minimizing the loss of correlation information due to distributing traffic. They addressed this problem by formalizing the load balancing problem as an optimization problem and then solving it using their own developed optimization algorithm called Benefit-based Load Balancing.

Belyaev *et al.* [4] proposed a load balancing solution to protect networks against DDoS attacks. In their work, they use SDN as a platform to balance incoming traffic to network. More specifically, they designed a two-phase load balancer that utilizes not only existing servers in the network but also other network devices. Their experimental results show that their solution increases the “survival” time during DDoS attacks.

Zinno *et al.* [15] proposed a mechanism that relocates the users’ resources including voice and data during a DDoS attacks. The mechanism is a load balancing algorithm, integrated in the architecture of LTE networks. They simulated an attack scenario using Network Simulator 3 (ns-3) and the results showed that if the algorithm is active, during a DDoS attack, communication services are always available for users.

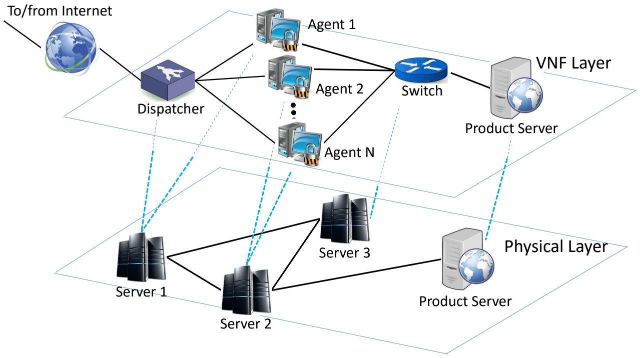


Figure 1: A topology illustration for NFV-based DDoS defense (VFence).

Akbar *et al.* [3] designed a scheme to protect Voice over IP (VoIP) networks against DDoS attacks. Their proposed scheme aimed to decrease the drain of computational resources of VoIP network and nodes. They use a load balancer to distribute incoming service requests. They achieved this by modifying an open source SIP proxy server called Kamailio.

Most of these these works, which have applied NFV for defending DDoS attacks and improving the packet processing capacity of VMs, did not talk about the implementation of a VNF that can balance the incoming packet load, so that it is easily possible to defend high volumes of DDoS attacks. A VM-based load balancing approach can significantly reduce the cost of high packet processing rate. Our proposed algorithms helps a dispatcher efficiently dispatch packets to a dynamic number of packet filtering virtual agents, which in turn defend a product server from DDoS attacks.

3. VFENCE FRAMEWORK

To handle high volume traffic from both benign sources and attackers, NFV technology provides a new opportunity for a flexible and low-cost solution that can be adopted by small to medium enterprises to defend against DDoS attacks. In this section, we present VFence, an NFV-based DDoS defense mechanism. We discuss the types of virtual network functions (VNFs) that are utilized in the design and elaborate on the work procedure of these virtual entities.

Figure 1 shows a topology overview of VFence. The defense network consists of a collection of dynamically allocated virtualized network functions (VNFs). A product server which provides online service to customers needs to be protected from cyberattacks. Several commodity servers are used to host virtual machines for several network functions such as dispatcher, switches, and agents. In the rest of this section, we briefly describe two major types of VNFs - dispatcher and agents, and then elaborate the DDoS attack scenarios and the dynamic defense strategy of VFence. The dispatcher is the gateway to the virtual filtering system and used to distribute incoming packets to agents. The agents can be created and removed dynamically based on demand, and are responsible for filtering out malicious traffic.

3.1 Dispatcher

The dispatcher is a VNF that intercepts all incoming external packets. It distributes the incoming packets to corresponding agents for handling them. The packets distribution is done based on an agent assignment algorithm (Section 4), which is executed according to a forwarding table. The table provides a mapping service from flows to handling agents. The forwarding table is updated whenever a flow is established, ends or expires.

One important function of the dispatcher is load balancing, where packets are delivered to corresponding agents to maintain moderate and balanced utilization rate on agents. The purposes of load balancing are two fold: First is to avoid using excessive number of agents. Fewer number of agents without performance penalty is preferred for a higher energy efficiency. Second is to have better performance when resource is limited. Balanced load on agents results in lower packet drop rate than the situation that some agent are overloaded and underloaded.

3.2 Agents

Agents are dynamically created packet handling agents. The purpose of agents is to filter packets that use fabricated source IP addresses. Furthermore, agents can also act as dynamic firewalls to block suspicious flows.

The agents verify the source IP addresses of all incoming SYN packets through a spoofed handshaking process [9]. After a successful handshaking with the source, the agent will initiate a handshaking with the destination and record the sequence numbers from both hand shakings for the flow. At the same time, the agent also notifies the dispatcher so that a new entry of established flow will be created for further forwarding. The rest of the packets in the flow will be forwarded to the destination after replacing their sequence numbers with the ones offered by the destination.

The agent can also run a rule-based firewall function to block known illegitimate flows. For example, whenever an agent receives a packet distributed by the dispatcher and no state of the flow is found on the agent, then the packet will be dropped. Otherwise run the flow identity against the firewall rules and drop the packet if a violation is detected.

Finally, agents also communicate with the dispatcher on a regular basis to report their utilization, the completion of new connections and the termination of connections. The dispatcher updates its forwarding tables accordingly based on the inputs from the agents.

3.3 DDoS Attack Scenario

During a SYN flood attack, the attacker generates a large volume of SYN packets with spoofed source IPs and send them to the victim, hoping to exhaust the resource from the target server. When the dispatcher receives a SYN packet from an attacker, it forwards the packet to a handling agent. The agent then initiates a *spoofed hand shaking process* by replying with a SYN-ACK with SYN cookies. Because the source address in the SYN packet is spoofed, the SYN-ACK will be sent to an invalid IP address or an incorrect source IP. The real attacker would never receive this SYN-ACK, which contains critical information to generate a correct ACK packet. As a result, the agent would never receive a correct ACK back from the attacker, and SYN flood attackers would not be able to establish a connection successfully with the agent. As a result, the attacker's traffic would not be forwarded to the product server. This process protects the server from DDoS attacks.

Even if the attacker is able to send a fabricated ACK to attempt to finish the handshaking, the agent verifies that it is not a valid ACK by checking its ACK number. It drops the packet. Any other type of packet whose source address is not on a white-list maintained by the agent will be dropped by it.

3.4 Dynamic Agents Deployment

The benefit of utilizing NFV for DDoS mitigation is that we can use dynamically created virtual machines to implement all network functions including dispatcher and agents. The agents can be deployed dynamically according to the demand. The dispatcher executes a dynamic agent assignment algorithm to distribute the incoming packets to the agents for filtering. The system will increase the number of agents when there is a huge flow of incoming packets.

When there is an increased number of incoming packets, the system will deploy new agents by creating more virtual machines. The dispatcher will then add the new agent into the list. For example, it will add the new agent ID into the agent list for the agent selection function to choose from. When the incoming traffic decreases, the dispatcher may turn off some agents to save energy. Before deleting an agent, the dispatcher updates its mapping table. The algorithms make sure that the dispatcher does not allocate new flows to the retiring agents.

4. LOADBALANCING ALGORITHM

In this section, we propose a load balancing algorithm for the dispatcher component that determines the packet handling agents so that the workload on agents are balanced with enhanced utilization rate of agents. Figure 2 shows an example of the routing table design on the dispatcher. The routing table is composed of two sections - entries for existing flows and entries for bucket list. The existing flow entries are for established flows which have completed the handshaking process. The bucket list are used to map the 5 tuple identities of new flows, which are in the process of handshaking, into a fixed number of buckets. The use of bucket list is necessary since during DDoS attacks, a huge number of SYN packets from various fabricated source IPs can easily overflow the routing tables. Through using the bucket list, the number of required entries on the dispatcher can be significantly reduced.

The process of assigning arriving packets to agents on the dispatcher is described in Algorithm 1. The load balancing process is presented in Algorithm 2.

As described in Algorithm 1, upon the arrival of a packet p_t, it first checks whether it is a handshaking packet (Line 2). If it is a handshaking packet, then use a hash function to find its corresponding bucket number (each bucket has a corresponding handling agent). The packet will be assigned to the agent which is responsible for the bucket (Line 3-4). Otherwise, if it is not a handshaking packet, it will first check whether it belongs to an existing flow which has finished handshaking (Line 7). If yes, then forward it to the agent which handles the flow, otherwise do not forward the packet.

The load balancing among agents is also handled by the dispatcher through adjusting the handling agent for flows. The dispatcher receives utilization reports from agents pe-

Algorithm 1 Agents Assignment on the Dispatcher

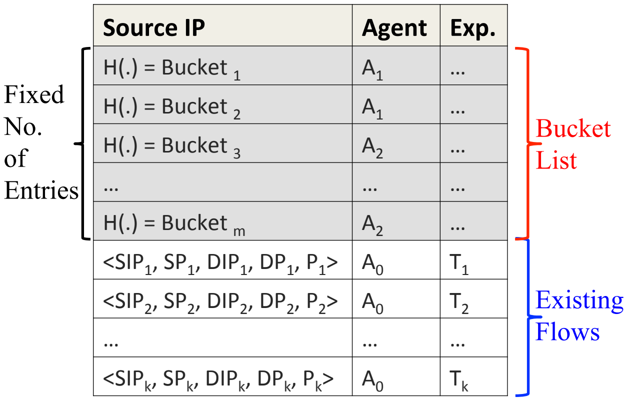
In this algorithm, each packet will be forwarded to a handling agent for processing. If the packet belongs to an existing flow which has finished handshaking, then forward the packet to the agent which handles the flow; if the packet is a handshaking packet, then find the bucket according to the 5 tuple signature and forward the packet to the corresponding agent which handles the bucket.
Input: Incoming packets p_t at time t; The forwarding table $\mathbf{R}$;
Output: Assigned agents to each packet p_t
Notations:
$H(.)$**:** the Hash function which maps a packet into a bucket number
$R_A(.)$**:** the agent given a entry of the routing table

```
upon event the arrival of an incoming packet $p_t$ do
    if $p_t$ is a SYN packet or an ACK packet then
        $B_t \Leftarrow H(p_t)$ // get the bucket number
        return $R_A(B_t)$ // the agent which handles the bucket
    end if
    $f \Leftarrow flow(p_t)$ // the 5 tuple flow definition for packet $p_t$
    if $isExistingFlow$ (f) then
        return $R_A(f)$ // the agent which handles the flow
    else
        return $null$ // packet will not be forwarded
    end if
```

Algorithm 2 Load Balancing and Maintenance Algorithm

This heuristic algorithm runs on the dispatcher to balance the workload on agents based on the current utilization of resources. Forwarding table is updated when a new agent assignment is processed.
Input: The forwarding table $\mathbf{R}$; The agents utilization rate $\mathbf{U}$.
Notations:
θ_o**:** the threshold indicating an agent is over utilized
θ_u**:** the threshold indicating an agent is under utilized

```
 1: // If agent i is over utilized, shift some load to others.
 2: upon event $\mathbf{U}_i > \theta_o$ do
 3:     k=$popBucket(i)$ // pop a bucket id that agent $i$ handles
 4:     l=$leastLoad$(U) // find the agent which has the least load
 5:     if $\mathbf{U}_l < \theta_u$ then
 6:         $updateAgent(k, l)$ // change the handling agent of bucket $k$ to agent $l$. Move some work load from agent $i$ to $l$.
 7:     else
 8:         m=$createNewAgent()$ // create a new agent
 9:         $updateAgent(k, m)$ // move the load to the new agent
10:     end if
11: // If agent j is under utilized, move some workload from others.
12: upon event $\mathbf{U}_j < \theta_u$ do
13:     $l$=$leastLoad$(U) // find the agent which has the least load
14:     if $j! = l$ then
15:         k=$popBucket(l)$ // pop a bucket id that agent $l$ is in charge
16:         $updateAgent(k, j)$ // update the handling agent of bucket $k$ to agent $j$. Move some workload from the least loaded agent.
17:         if $isEmpty(l)$ then
18:             $deleteAgent(l)$ // delete agent $l$ since it is empty
19:         end if
20:     end if
```



Source IP	Agent	Exp.
H(.) = Bucket $_1$	A_1	...
H(.) = Bucket $_2$	A_1	...
H(.) = Bucket $_3$	A_2	...
...	...	...
H(.) = Bucket $_m$	A_2	...
<SIP_1, SP_1, DIP_1, DP_1, P_1>	A_0	T_1
<SIP_2, SP_2, DIP_2, DP_2, P_2>	A_0	T_2
...	...	...
<SIP_k, SP_k, DIP_k, DP_k, P_k>	A_0	T_k

Fixed No. of Entries
Bucket List
Existing Flows

Figure 2: A Sample Forwarding Table for the Dispatcher.

riodically. Algorithm 2 is executed after every reporting cycle. When an agent is overloaded, the dispatcher will first look for an agent which is underloaded and shift one bucket workload to an underloaded agent (Line 5-6) if it is found. If no agent is underloaded, then a new agent will be created instead to share the load (Line 8-9). If an agent is underloaded and it is not the least loaded agent, then shift some load from the least loaded agent (Line 15-16). If the agent is empty after load shifting then remove the empty agent (Line 17-19).

5. EVALUATION

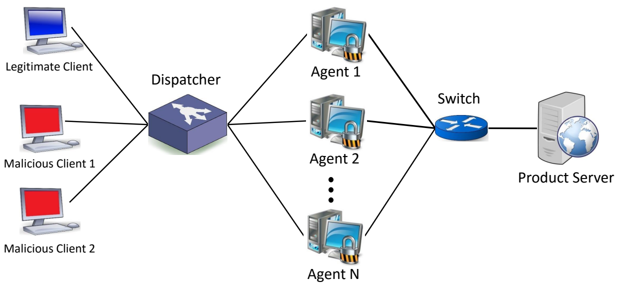
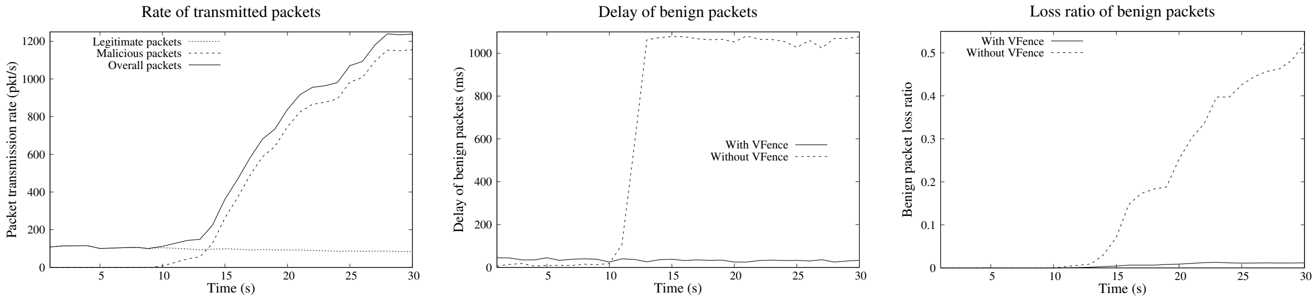


Figure 3: The simulation scenario.

We perform simulation to evaluate the proposed DDoS attack defense mechanism. We develop a Java program to simulate the SYN flood attack on a server based on a DDoS attack scenario. The program can simulate the effects of such an attack with or without the proposed defense mechanism which includes the dynamic agent allocation and load balancing algorithms. We perform the simulation on a machine running Windows 10 OS. The machine is equipped with an Intel Core i5 Processor and a 12 GB memory.

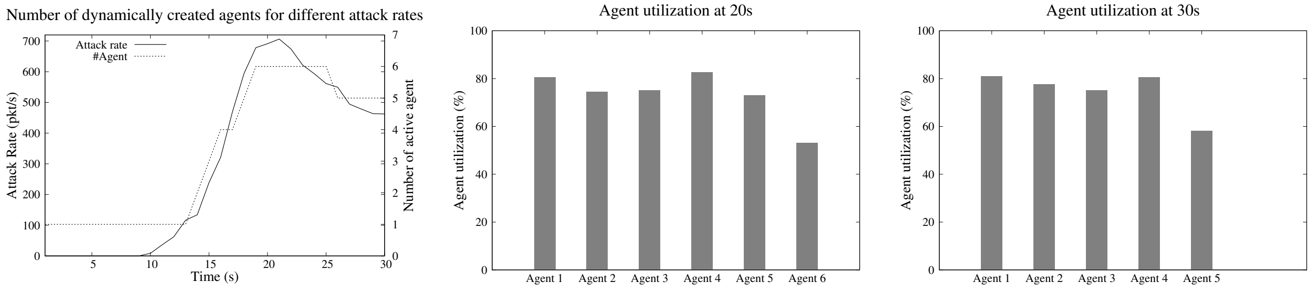
5.1 Experimental Setup

Figure 3 depicts the scenario typically used for the simulation. The network topology includes three clients, one dispatcher, several agents, one switch, and one server. One of the clients is taken as a legitimate client while the rest



(a) (b) (c)

Figure 4: (a) Packet transmission rate over simulation time, (b) Delay of benign packets reaching the server, (c) Benign packet loss over simulation time.



(a) (b) (c)

Figure 5: (a) Number of dynamically created agents over time along with malicious packet rate, (b) Agent utilization at 20s of simulation when attack intensity is high, (c) Agent utilization at 30s of simulation when attack intensity is comparatively low.

of the clients are considered as the malicious clients. The rogue clients generate the DDoS traffic. Figure 4(a) presents the traffic generated by the clients over the simulation time. The figure presents the packet transmission rates for the benign and malicious clients. It can be observed from the figure that until the 10^{th} second, the only client transmitting packets is the benign client. From the 10^{th} second, the malicious clients start transmitting increasing numbers of SYN packets to the server in order to launch a DDoS attack. The attack continues till the end of the simulation. The attack intensity of the malicious clients increases over time, and the attack packet transmission rate reaches up to 1150 pkt/s towards the end of the simulation, whereas the legitimate packet transmission rate remains almost constant, which is around 100 pkt/s throughout the entire simulation.

5.2 Results

We use two metrics to show the effectiveness of VFence and its load balancing techniques: the loss of legitimate packets in the case of an attack and average delay of these packets when they reach the server. We compare between two cases: one with a system running VFence and the other is a simple network without any major security measures.

5.2.1 Average Delay

Figure 4(b) describes the influence of the proposed mechanism on the average delay of benign packets. Delay is calculated as the time taken for a packet to reach the server from a client. When the attack traffic increases as time goes by, the average delay of the system without VFence increases drastically. At around 13 second, packets start to get dropped. The system with VFence, on the other hand, incurs a small delay throughout the entire simulation. It can be observed that before the attack starts, delay of the system with VFence is slightly greater than the system without VFence. The reason is that VFence imposes agents in the middle and these agents intercept the incoming packets and verify the legitimacy of all these packets. In spite of this small delay, VFence provides greater resiliency against DDoS attacks, which is the key in this research.

5.2.2 Benign Packet Loss Ratio

It can be observed from Figure 4(c) that when there is no attack traffic (during the first 10 seconds of the simulation), both the system with VFence and the system without VFence have the same low number of lost packets, which is basically due to low network congestion. However, when the incoming traffic (actually the attack traffic) significantly increases, the system without VFence starts losing packets. It takes around 2-3 seconds to fill up the buffer of the router. When the router queue is full, packets start to get dropped. The total number of dropped packets keep increasing almost linearly with time. As the figure suggests, more than half of the benign packets are lost during the entire simulation period. On the other hand, the system with VFence is able to keep the number of lost packets low. This is because VFence utilizes a dynamic number of agents to block the attack traffic and balance the load on the different agents. There is still a negligible amount of lost packets due to buffer size limitations of the agents and routers during severe DDoS attacks.

5.2.3 Number of Agents

The flexibility of the proposed load balancing algorithm can be observed from Figure 5. For this particular evaluation, we gradually increase the malicious traffic rate starting from the 10th second till the 21st second. After that, we decrease the attack intensity slightly to reduce the load on the agents. In Figure 5(a), we show both the attack traffic rate (pkt/s) and the number of active agents with respect to time. In a particular simulation scenario, it is observed that at 21st second, the attack traffic rate is around 700 pkt/s, while there are total six agents actively protecting the system. With the increase of the attack traffic, the number of agents also gradually increases to balance the loads. The number of agents keeps constant when the attack packet transmission rate is stable. Only one agent is deployed until the malicious clients start transmitting the attack traffic. To deal with the excessive number of attack packets, new agents are dynamically created and deployed to work. To maximize the utilization ratio of each agent and avoid the resource waste, the number of agents reduces when the attack traffic decreases. For example, after the 25th second of the simulation, one agent is removed from the system.

Figure 5(b) and Figure 5(c) present the snapshot of utilization percentage of each active agent at the 20th second and 30th second, respectively. For this experiment, we define the overutilization threshold to be 90%. It is observed that the agents are never overutilized, so that the system can perform optimally and prepare for worse situations. We keep the usage of agents below a threshold value as long as possible, and at the same time, we do not want to create too many agents for a given attack scenario. In both of the cases, the last agent is less loaded, as it is either a newly created agent or the agent is not being assigned for new packets to perform the verification.

6. CONCLUSION

In this paper, we propose a dynamic DDoS defense architecture using the NFV technology. A load balancing algorithm is used to balance the workload on virtual network agents that are dynamically created to handle the varying attack traffic. The use of fixed bucket list allows to control the number of needed entries in routing table under SYN flood attacks. Our evaluation results demonstrate that the proposed system can effectively defend against DDoS attacks and maintain reasonable utilization rate on involved agents. As a direction of our future work, we will evaluate the system based on emulated real traffic and measure the impact from the different parameter settings on the effectiveness, performance, and energy efficiency of the system.

7. REFERENCES

[1] Biggest internet attack in history threatens critical systems. http://www.ibtimes.co.uk/biggest-internet-attack-history-threatens-critical-infrastructure-450969.

[2] Prolexic routed. https://www.akamai.com/us/en/solutions/products/cloud-security/prolexic-routed.jsp.

[3] Akbar, A., Basha, S. M., and Sattar, S. A. Leveraging the sip load balancer to detect and mitigate ddos attacks. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)* (Oct 2015), pp. 1204–1208.

[4] Belyaev, M., and Gaivoronski, S. Towards load balancing in sdn-networks during ddos-attacks. In *2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC)* (Oct 2014), pp. 1–6.

[5] et al., J. D. Vnguard: An nfv/sdn combination framework for provisioning and managing virtual firewalls. In *NFV and SDN, IEEE Conference on* (2015), pp. 107–114.

[6] ETSI NFV ISG. Network Functions Virtualization White Paper 3: Network Operator Perspectives on Industry Progress. In *SDN and OpenFlow World Congress* (Oct. 2014).

[7] Fayaz, S. K., Tobioka, Y., Sekar, V., and Bailey, M. Bohatei: Flexible and elastic ddos defense. In *24th USENIX Security Symposium* (2015), USENIX, pp. 817–832.

[8] Fung, C. J., and McCormick, B. Vguard: A distributed denial of service attack mitigation method using network function virtualization. In *Network and Service Management (CNSM), 2015 11th International Conference on* (2015), IEEE, pp. 64–70.

[9] Jakaria, A. H. M., Yang, W., Rashidi, B., Fung, C., and Rahman, M. A. VFence: A Defense against Distributed Denial of Service attacks using Network Function Virtualization. In *STPSA, 11th IEEE International Workshop* (2016).

[10] Khandelwal, S. 602 gbps! this may have been the largest ddos attack in history. http://thehackernews.com/2016/01/biggest-ddos-attack.html.

[11] Le, A., Boutaba, R., and Al-Shaer, E. Correlation-based load balancing for network intrusion detection and prevention systems. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks* (New York, NY, USA, 2008), SecureComm '08, ACM, pp. 2:1–2:10.

[12] Liyanage, M., Ahmad, I., Ylianttila, M., Gurtov, A., Abro, A. B., and de Oca, E. M. Leveraging lte security with sdn and nfv. In *2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS)* (2015), pp. 220–225.

[13] Wang, H., Zhang, D., and Shin, K. G. Detecting syn flooding attacks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2002), vol. 3, IEEE, pp. 1530–1539.

[14] Yu, S., Tian, Y., Guo, S., and Wu, D. O. Can we beat ddos attacks in clouds? *IEEE Transactions on Parallel and Distributed Systems 25*, 9 (2014), 2245–2254.

[15] Zinno, S., Stasi, G. D., Avallone, S., and Ventre, G. A load balancing algorithm against ddos attacks in beyond 3g wireless networks. In *2014 Euro Med Telco Conference (EMTC)* (Nov 2014), pp. 1–6.