VGuard: A Distributed Denial of Service Attack Mitigation Method using Network Function Virtualization

Carol J. Fung* and Bill McCormick†
*Computer Science Department, Virginia Commonwealth University, USA - email: cfung@vcu.edu
† Huawei Canada, Ottawa, Canada - email: bill.mccormick@huawei.com

***Abstract*—Distributed denial of service (DDoS) attacks have caused tremendous damage to ISPs and online services. They can be divided into attacks using spoofed IPs and attacks using real IPs (botnet). Among them the attacks from real IPs are much harder to mitigate since the attack traffic can be fabricated to be similar to legitimate traffic. The corresponding DDoS defence strategies proposed in past few years have not been proven to be highly effective due to the limitation of participating devices. However, the emergence of the next generation networking technologies such a network function virtualization (NFV) provide a new opportunity for researchers to design DDoS mitigation solutions.**

In this paper we propose VGuard, a dynamic traffic engineering solution based on prioritization, which is implemented on a DDoS virtual network function (VNF). The flows from the external zone are directed to different tunnels based on their priority levels. This way trusted legitimate flows are served with guaranteed quality of service, while attack flows and suspicious flows compete for resources with each other. We propose two methods for flow direction: the static method and the dynamic method. We evaluated the performance of both methods through simulation. Our results show that both methods can effectively provide satisfying service to trusted flows under DDoS attacks, and both methods have their pros and cons under different situations.

I. Introduction

Distributed denial-of-service (DDoS) attacks can cause tremendous damage to ISPs and online services, especially for those small or medium-sized organizations who lack the resources to withstand a high volume of DDoS traffic. Some recent incidents showed that DDoS attacks have significantly increased their strength. For example, the Spamhaus attack in 2013 [1] has generated 300 Gbps attack traffic. This number has been increased to 400Gbps in March 2014 [5], and again to 500Gbps in November 2014 [4].

There are two major types of DDoS techniques in the attack traffic: IP spoofing and real source IP attack. IP-spoofing is the type of attack where the source addresses are not the real IP address of the attacker. The DDoS attack based on real source IP addresses commonly utilizes compromised nodes in the Internet to launch the attack. For example, the attacker can use compromised bot nodes that are are under the control of bot masters. Many solutions [6], [7] have been proposed to mitigate DDoS attacks on real IPs. The capacity-based method limits the flow rate of each source based on their priority estimation, and the filter-based method blocks traffic which is identified as undesirable. However, the capacity-based solution introduces high overhead in computation and memory so it is not suitable to large scale DDoS attacks. On the other hand, the filter-based solution may also block new legitimate traffic when it is difficult to separate the attack traffic from the legitimate traffic. A new feasible solution is called upon to allow small to medium sized organizations to defend against DDoS attacks.

Network Functions Virtualization (NFV) is an emerging technology where network functions are implemented and provided in software, which runs on commodity hardware [12]. The functions are implemented in software and deployed in virtual machines. The virtual machines run on general purpose commodity hardware so that NFV does not only provide the benefit of elasticity, but also reduces the cost by running on commodity platforms like x86- or ARM-based servers instead of specialized hardware. This virtualized network makes the deployment of new network services much easier and less costly. At the same time, NFV also introduces an opportunity for DDoS detection and mitigation. The traditional methods of DDoS detection are limited by the computation capacity and flexibility of involved network functions such as switches and routers. The use of NFV opens a new opportunity to introduce more intelligent computation capability on software-based network functions.

In this work, we introduce a novel solution that involves the use of NFV to mitigate DDoS attacks. We utilize the flexibility of network functions and create two virtual tunnels, one high priority and the other low priority, for all external traffic to the DDoS targeted online service. Each flow is allocated to one of these two tunnels to reach the destination. A flow dispatching algorithm is used to manage the flows based on their priority level. Our experimental results show that our algorithm can effectively guarantee the quality of service for high priority flows under DDoS attack and at the same time allows the updating of priority level for new flows.

The contributions of this paper include: (1) To the best of our knowledge, this is the first proposed solution using network function virtualization to mitigate DDoS attacks. (2) We propose two priority-based dynamic flow dispatching methods and compare them with each other. (3) We evaluate

978-3-901882-77-7 © 2015 IFIP

our proposed solutions using simulation and demonstrate the effectiveness of our solutions.

The rest of this paper is organized as follows: Section II gives an overview of existing DDoS detection and mitigation solutions in the past. In Section III we briefly introduce network function virtualization. We propose an architectural design on how to utilize a virtual network function to mitigate DDoS attacks in Section IV and then incorporate a priority-based flow dispatching algorithm to handle the flows in section V. We evaluate the proposed solution in Section VI and finally we discuss and conclude this paper in Section VII and VIII correspondingly.

II. Related Work

DDoS attacks can be divided into two categories: attacks based on IP-spoofing and attacks based on real IP addresses, such as utilizing compromised nodes (bot-nodes). There are many proposed solutions to mitigate IP-spoofing attacks, such as BCP 38 [11] , IP traceback mechanisms [13], and packet marking and filtering mechanisms [20], [22]. For example, the BCP 38 standard [11] enforces ingress routers to verify that the packets from its administered network are using legitimate source IP addresses and filter traffic that uses IP addresses outside its subscribed range. Those solutions have been proved to be effective and can mitigate IP-spoofing attacks completely if they are adopted. However, our focus in this paper is to mitigate the other type of DDoS attack: the botnet DDoS attack.

Many million-node botnets have been discovered in the past, such as BredoLab [18] and the Ramnit botnet [3]. The bot masters can easily orchestrate a large scale real IP DDoS attack if they control a large number of bot-nodes in the Internet. Capability-based defense [6], [15], [23] and filter-based defense [7], [14], [16] are two major strategies on the receive side against the real IP DDoS attacks. The capability-based method attempts to limit the flow rate of each source by sending permission tokens to the selected sources to allow them to send packets. Sources with high privileges can easily get tokens and have guaranteed quality of service, while attacking sources will not receive tokens. The filter-based method tries to block traffic which is identified as undesirable. Attack sources can be filtered by ISPs when an incentive is provided to encourage cooperation among ISPs [7]. However, the capability-based solution introduces high over-head on computation and memory, so it is not suitable for large-scale DDoS attacks. On the other hand, the filter-based solution suffers from high false positive rates if it is difficult to separate the attack traffic from the legitimate traffic.

Our approach is to provide service differentiation to flows with different priority levels. It utilizes the flexibility of NFV design by implementing a virtualized network function that can be used to perform DDoS mitigation based on the priority of flows. The priority of flows can be estimated through history, known black-lists, and source geographical regions, etc. VGuard is light-weight and efficient compared to capability-based solutions. It does not only provide satisfying quality of service to high priority flows, but also provides the opportunity for low priority real users to enhance their priority level of their flows by accepting and passing a bot test. This way flows from bots will be eventually filtered and resources are saved for real users.

III. Background on Network Function Virtualization

Network function virtualization (NFV) is an emerging technology complementary to software defined networking (SDN) [12]. The main idea is to virtualize the hardware resources to provide high flexibility and reduced cost.

One of the primary benefits of NFV is the ability to replace expensive middleware boxes such as load balancers and firewalls running on proprietary hardware with a software based implementation running on commodity hardware or on virtual machines. Aside from the reduction in capital costs, service providers are interested in reducing operating costs in the same way as with server virtualization - increased agility in responding to user demands, reduced energy costs through consolidation of multiple services on the same physical equipment and reduced deployment times.

Performance has been a key concern when virtualizing network functions. Specialized equipment uses specialized hardware to minimize latency and ensure effective operation at line rate. It is well known that it's difficult to achieve stable high throughput on virtualized systems. [19] To address this performance problem, vendors are proposing the use of hardware acceleration (HA) built on field programmable gate arrays. The HA provides CPU offload for performance sensitive data plane processing ensuring that harder real time requirements can be met.

A layer 7 load balancer using this type of CPU offload was demonstrated by a group of equipment vendors and service providers at the SDN/Openflow world congress in 2014 and virtualization of HA is an active research topic [9]. Progress in these areas suggest that NFV will provide the flexibility and compute resources to enable applications such as VGuard.

IV. VGuard System Design

In VGuard, we proposethe use of a virtual network function (VNF) to perform DDOS attack mitigation. The architectural design of VGuard is shown in Figure 1. The main idea is to direct incoming traffic into two tunnels with traffic policing. For example, a DDoS attack may want to take down an online service by sending a large volume of requests to the server. The traffic will be filtered by a firewall first and then dispatched to different tunnels based on their priority level.

In a typical enterprise network, the internal network connects to the internet at the gateway shown in Figure 1, so that all external flows pass through this gateway. We propose configuring a service chain [8] from the enterprise gateway to a firewall VNF, and then to a DDOS mitigation VNF which dispatches requests through two tunnels to the online application server. While our proposal is focused on a NFV based implementation, these tunnels may be implemented

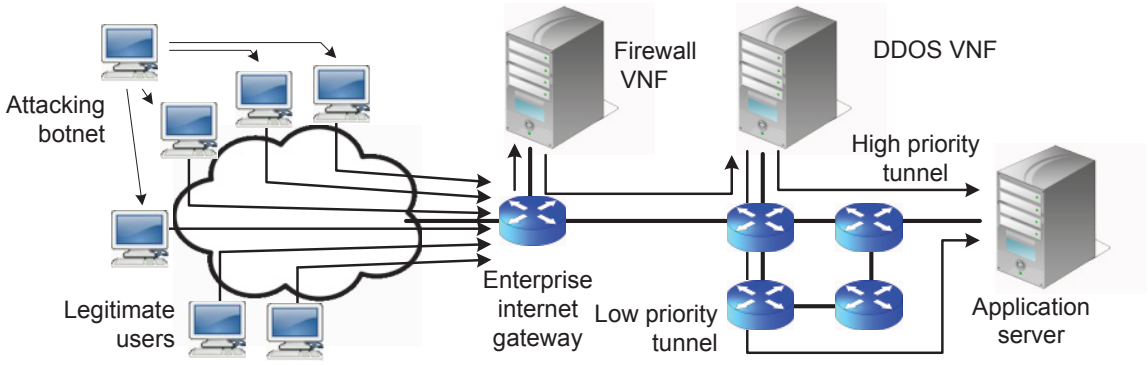


Fig. 1. Architectural Design of VGuard

using MPLS [17], GRE [10], or some other mechanism which supports traffic policing on a per tunnel basis. Our intent is to use one tunnel for traffic likely to be benign, and the second tunnel for traffic likely to be malicious.

Our scenario is an application-level DDoS attack from external peers to the online service. The attack sends a high volume of requests to the online service to consume its resources, such as CPU and memory, so that service to legitimate users is unavailable. Since the attack requests are mixed with benign requests, it is not desirable to simply filter excessive traffic to protect the server, as with this approach the legitimate requests will be also filtered. Our goal is to limit the request rate from untrusted sources to the server so that the server will not be overwhelmed, while requests from trusted sources are guaranteed to be served. The challenge is how to control the traffic through the DDOS mitigation VNF to achieve this goal.

In the VGuard design shown in Figure 1, requests from the external Internet will go through a firewall VNF where requests from known malicious sources ($p = 0$) will be filtered. The rest of the traffic to the application server will then be directed into two different tunnels, a high priority tunnel and a low priority tunnel, based on the priority level of the source IP address. The traffic dispatching algorithm is implemented on a DDoS VNF, which controls the routing of all flows to the online service.

Source IP address priority can be developed by the application service, which can distinguish benign IP addresses based, for example, on geographic location, prior activity records, or Captcha tests [2]. The DDoS VNF communicates with the application server on a regular basis to obtain an updated priority table of potential IPs.

Suppose the online application server maintains a list of potential source IP addresses, which contains the source IP addresses and their priority level $0 \leq p \leq 1$, where $p = 1$ means the IP is from trusted source, and $p = 0$ means the IP is absolutely malicious (such as IPs in the black list). Intermediate values of p define how likely the IP is to be benign. The DDOS mitigation VNF is to provide service differentiation so that benign flows will be served and malicious flows will be blocked. Flows with intermediate p values will be served based on the resource availability on the server. Details on flow dispatching will be described in the next section.

V. Priority-based Flow Dispatching

In this section, we develop the concept of using priority tunnels to handle DDoS attacks, then we describe two flow dispatching techniques to handle flows with different priority levels: a static-based approach where a static threshold is used for dispatching decisions, and a dynamic-based approach where the dispatching decision is based on the current active flows in both tunnels.

A. DDoS Mitigation using Priority

Flow with a priority of 1 are known to be benign and should always be admitted to the system. Flows with a priority of 0 are known to be malicious and should always be blocked by the firewall. However, most requests may not belong to either list – they could be benign or malicious.

DDoS attack flows have some typical features which make them distinct from most other flows. For example, they are typically flows which appear together with a large volume of other flows, their IPs are not familiar to the service, and they are bot traffic so that they cannot pass bot tests. They can be rated low priority flows using a flow prioritization algorithm [21]. However, flows satisfying these criteria are not necessary malicious. They may also be benign flows which happen to appear with the attack traffic for their first visit. Therefore, simply blocking these requests is not a good solution.

Our approach to mitigate DDoS attacks is to handle the service of flows given their priority, so that trusted flows will have guaranteed quality of service, while suspicious flows will be handled based on the availability of resources. As we discussed in Section IV, VGuard uses two priority tunnels to handle all traffic, one high-priority tunnel and one low-priority tunnel. In the next few subsections, we first analyze the utility function of the service, then we propose corresponding flow dispatching algorithms.

B. Utility Analysis

Let R_{max} denote the maximum request rate that the server can tolerate, and R_{norm} denote the normal request rate that the sever is expecting; We assume that the server is able to handle normal conditions, i.e., $R_{max} > R_{norm}$. Let $p_i \in [0, 1]$ denote the priority of flow i. The higher p_i is, the higher priority the flow is. As we discussed before, the DDOS VNF maintains two tunnels t_H, t_L to relay external requests to the online server. The tunnel t_H is a high priority tunnel so that flows with higher priority are directed to this channel. The tunnel t_L is a low priority tunnel for flows with lower probability of benign. The bandwidth capacity of t_H, t_L are limited to be C_H, C_L. Note that $C_H + C_L = R_{max}$ to guarantee the overall traffic will not overwhelm the server. Our task is to find a flow forwarding policy to satisfy the aforementioned criteria.

Suppose we use a simple threshold based flow dispatching mechanism to handle flows with different priorities. More specifically, the controller can forward the flows with higher priority than a threshold θ to t_H and the others to t_L. Given the data rate distribution function $f(x)$ which denote the packet rate distribution on all priority levels. Then we have:

$$r_L(\theta) = \int_0^{\theta} f(x)dx \text{ and } r_H(\theta) = \int_{\theta}^{1} f(x)dx \tag{1}$$

where r_L is the request rate that is forwarded to tunnel t_L and r_H is the request rate that is forwarded to the tunnel t_H. To model the satisfaction of users, we use the drop rate d_H, d_L to denote the probability that requests forwarded to these two tunnels are dropped due to overflow. Then we have:

$$d_L = 1 - min(1, C_L/r_L(\theta)) \tag{2}$$
$$d_H = 1 - min(1, C_H/r_H(\theta)) \tag{3}$$

In the above formulation we used min function since if the data rate is lower than the capacity, the drop rate should be 0; if the data rate is higher than the capacity then the drop rate is greater than 0.

To capture the quality of service, we need to quantify the satisfaction of service. We use a polynomial curve $h(x) = x^2$ to denote the satisfaction level of a flow i.e., if all data from that flow is served then the satisfaction of the flow is 1. The satisfaction of a flow i forwarded to tunnel t_L can be denoted by $s_L = (1 - d_L)^2 = min(1, (C_L/r_L)^2)$. Correspondingly, the satisfaction for a flow in tunnel t_H is $s_H = (1 - d_H)^2 = min(1, (C_H/r_H)^2)$. Our goal is to maximize the aggregated weighted satisfaction from all flows denoted by:

$$S = S_L + S_H = \sum_{\forall i, p_i < \theta} r_i p_i s_L + \sum_{\forall j, p_j \geq \theta} r_j p_j s_H$$
$$= \int_{u=0}^{\theta} uf(u)s_L du + \int_{v=\theta}^{1} vf(v)s_H dv$$

where r_i denotes the request rate of flow i and p_i is the priority of flow i.

Given R_{max}, $f(x)$, and C_H, C_L, our goal is to determine the optimal θ to maximize the overall satisfaction. Note that in the above equation, both s_L and s_H are functions of θ. Then we have:

$$\text{argmax}(\theta) = \max(S)$$
$$= \max(\int_0^{\theta} uf(u)s_L(\theta)du + \int_{\theta}^{1} vf(v)s_H(\theta)dv)$$

C. Static Flow Dispatching

To find the optimal θ, we discuss two scenarios:

1) if the total data rate is less than R_{max}, then we can find θ so that $r_L < C_L$ and $r_H < C_H \Rightarrow s_L = s_H = 1$. Therefore, $max(S) = \int_{x=0}^{1} xf(x)dx$.
2) if the total data rate is greater than R_{max}, we discuss the following conditions:

- if $r_H \leq C_H$, then we have

$$\frac{\partial}{\partial\theta}S = \theta f(\theta)s_L(\theta) + \int_0^{\theta} uf(u)s_L^{'}(\theta)du - \theta f(\theta)$$
$$= \theta f(\theta)(s_L(\theta) - 1) - \frac{2C_L^2}{r_L(\theta)^3}f(\theta)\int_0^{\theta} uf(u)du$$
$$< 0$$

- if $r_L \leq C_L$, then we have :

$$\frac{\partial}{\partial\theta}S = \theta f(\theta) + (-\theta f(\theta)s_H(\theta) + \int_{\theta}^{1} vf(v)s_H^{'}(\theta)dv)$$
$$= \theta f(\theta)(1 - s_H(\theta)) + \frac{2C_H^2}{r_H(\theta)^3}f(\theta)\int_{\theta}^{1} vf(v)dv$$
$$> 0$$

- If $r_H > C_H$ and $r_L > C_L$, then we have:

$$\frac{\partial}{\partial\theta}S = \theta f(\theta)s_L(\theta) + \int_0^{\theta} uf(u)s_L^{'}(\theta)du$$
$$+ (-\theta f(\theta)s_H(\theta) + \int_{\theta}^{1} vf(v)s_H^{'}(\theta)dv)$$
$$= f(\theta)(s_H(\theta)(2\theta_H - \theta) + s_L(\theta)(\theta - 2\theta_L))$$

where θ_H, θ_L represent the average priority in the high priority tunnel and the low priority tunnel. It is not difficult to see that when $s_H > s_L$, $\frac{\partial}{\partial\theta}S > 0$.

Therefore, when $r_H = C_H$, i.e., we set the threshold to be $\theta^* = F^{-1}(C_H)$, where F(x) is a cumulative function of f(x), so that the high priority tunnel is fully utilized to its capacity, we achieve the optimal overall satisfaction from all flows $max(S) = \int_0^{\theta^*} uf(u)s_L(\theta^*)du + \int_{\theta^*}^{1} vf(v)dv$.

D. Dynamic Flow Dispatching

The major limitations of the static flow dispatching is that the distribution of the flow priority function $f(x)$ is usually unknown and when the distribution changes, the "optimal" threshold may not be optimal anymore. Therefore, a flow dispatching method which does not rely on known flow priority distribution may be a good choice. In this subsection,

we propose a dynamic flow dispatching algorithm, which is described as follows:

The high priority tunnel needs guaranteed quality of service (QoS) and the low priority tunnel is best effort tunnel. In order to effectively utilize and distinguish the functionality of these two tunnels, we use the following algorithm to describe how to allocate flows to realize priority-based resource allocation.

The flow assignment described in Alg. 1 can be divided into the following scenarios:

1) when both tunnels are under-loaded, allocate flows to whichever tunnel has the lower utilization rate.
2) when the high priority tunnel is close to full, it enters selective mode, where only high priority flows are allowed to enter.
3) when the high priority tunnel is over loaded, all new flows are pushed to the low priority tunnel unconditionally.

Algorithm 1 Flow Assignment Algorithm

Notations :
t_H, t_L :High priority and low priority tunnels
τ_{max} :The maximum utilization that tunnel τ_H is allowed to have
τ_{norm} :The Threshold utilization that tunnel t_H enters selective phase. That is, only selected flows enter this tunnel.
U_H, U_L :The current utilization of the tunnels. i.e., the percentage that a tunnel is utilized.
//initialize
$t_L = t_H = \emptyset$
Event e **triggers only when a new flow** f **arrives:**
if $U_L < U_H$ **then**
  $t_L \leftarrow t_L \cup f$
else
  if $U_H < \tau_{norm}$ **then**
    $t_H \leftarrow t_H \cup f$
  else if $U_H > \tau_{max}$ **then**
    $t_L \leftarrow t_L \cup f$
  else
    //selectively add new flow to the high priority tunnel
    if $Priority(f) > averagePriority(t_H)$ **then**
      $t_H \leftarrow t_H \cup f$
    else
      $t_L \leftarrow t_L \cup f$
    end if
  end if
end if

In this algorithm, τ_{max} is the threshold to guarantee the quality of service in the high priority tunnel t_H. With the above flow allocation algorithm, we can allocate every new flow to destination tunnel.

VI. Evaluation

A. Experimental setup

We built a discrete event simulator using the Python simply package to assess the design of our system. Our simulation used two tunnels (t_H - high priority tunnel and t_L - low priority tunnel) both configured with 50 Mbps of bandwidth. We enforced traffic policing on each tunnel so that flow assignments in excess of 50 Mbps would result in discarded traffic. Flows had a bandwidth of 100 kbps with exponential inter-arrival times and durations. The mean flow duration was set to 10 seconds and the mean flow inter-arrival time was varied to adjust the total traffic intensity in the system. The data rate distribution function, $f(x)$, was uniform.

B. Selection of $\theta^\star$

Figure 2(a) shows the system satisfaction for different values of θ with a traffic intensity of 150 Mbps. From section 5.1, we would expect the optimal value of θ to be $\theta^\star = r_H^{-1}(C_H) = 0.67$ as shown in this figure. The satisfaction in the high priority tunnel increases until it is fully utilized and then begins to decrease as traffic is discarded.

C. Selection of system parameters

In Figure 2(b) we investigate how to choose τ_{max} to maximize system performance with a fixed traffic intensity of 150 Mbps. We find that the total satisfaction in the system is maximized as $\tau_{max} \to 1$. Choosing τ_{max} close to one allows the best use of the high priority channel and hence maximizes total satisfaction.

Experimentally we observe that a value of τ_{normal} slightly less than τ_{max} maximizes system satisfaction. If τ_{normal} is much smaller, the high priority tunnel is not fully utilized reducing system satisfaction. As τ_{normal} approaches τ_{max}, we find that the system rarely enters selective mode, which also reduces system satisfaction.

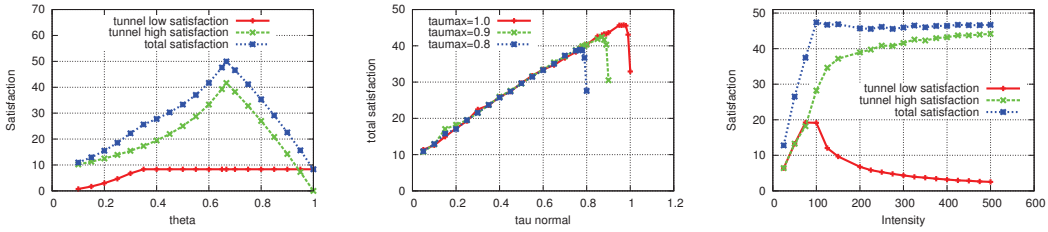
We selected $\tau_{max} = 1.0$ and $\tau_{normal} = 0.97$ for the remainder of the simulations.

D. System performance under varying traffic intensity

Figure 2(c) shows the system performance for our selected parameters under varying traffic loads. With a traffic intensity of less than 100 Mbps, we find that the system is lightly used and total satisfaction increases linearly with load. Satisfaction is maximized when traffic intensity reaches the system capacity of 100 Mbps. At this operating point the system is discarding little traffic and the system achieves a peak satisfaction of slightly under 50. As the system load continues to increase,the average priority in the high priority tunnel increases, increasing the satisfaction in the high priority tunnel. This effect is counteracted by the increasing discard rate in the low priority tunnel. The result is that satisfaction stays roughly constant at higher traffic intensities.

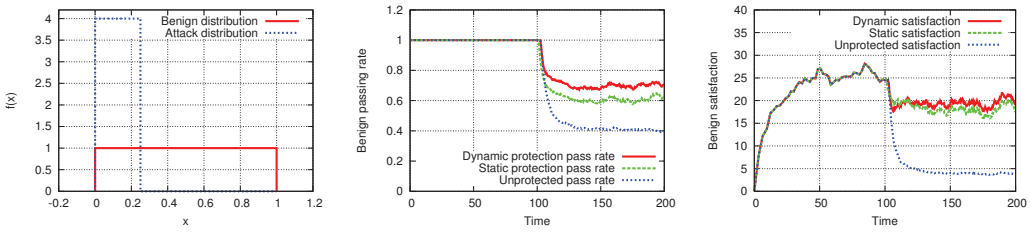
E. Robustness against DDoS attack

In order to see the effectiveness of VGuard against a DDoS attack, we simulate the DDoS attack flows with the distributions shown in Figure 3(a). The attack launches after 100 seconds on top of normal flows and we compare the quality of service of benign flows with and without VGuard protection. The system capacity is set to 100 Mbps shared equally between the high and low priority tunnels. We set the benign flow intensity at 50 Mbps and the attack intensity at 200 Mbps. For the static method, we choose $\theta_0 = 0.50$. Figure 3(b) and Figure 3(c) show that the packet passing rate and overall satisfaction of benign flows are much higher



(a) Selection of $\theta^{\star}$ (b) Selection of system parameters (c) Varying traffic intensity

Fig. 2. Parameter selection



(a) normal and DDoS attack flow distribution (b) Benign data passing rate under DDoS attack (c) Benign user satisfaction under DDoS attack

Fig. 3. Comparison of the impact from DDoS attacks with and without VGuard protection

when VGuard protection is in place, indicating that the impact from DDoS has been reduced significantly. As expected the static method only admits flows with a priority greater than 0.50 to the high priority tunnel, so that the high priority tunnel remains underutilized. The dynamic dispatching method adjusts to the new priority distribution with an increased utilization in the high priority tunnel, which results in the improved performance shown in the figure.

VII. Discussion

A. The choice of C_H and C_L

An important design choice is the selection of C_H and C_L. It is straightforward that we should set $C_H + C_L = R_{max}$ to serve as many requests as possible. The choice of C_H can be based on the history of the request rate to the server. For example, we can pick C_H to be the bandwidth of the normal request load and $C_L = R_{max} - C_H$ (note that R_{max} should be greater than C_H). One may argue that C_H should be set to R_{max} to serve the top priority requests only and drop all the rest. Our reasoning is that we wish to allow some (not overwhelming) low priority traffic to reach the server in order to investigate the nature of those requests by enabling further interaction with them. For example, the server can respond with a bot challenge back to a low priority source, such as Captcha [2]. If the source fails the challenge, the source is likely to be a bot and it can be added to the blacklist. On the other hand, if the source passes the challenge, it proves itself to be a real user, and its priority is increased correspondingly.

B. Static or Dynamic Dispatching

The major flaw of the static dispatching is its inflexibility as its performance drops more quickly than the dynamic model when the flow distribution changes suddenly. This method is a better choice if the normal traffic is predictable or stable. On the other hand, the dynamic method achieves less optimal results than the static model in the steady state case. However, it does not rely on knowledge of the flow distribution $f(x)$ and it out-performs the static model when $f(x)$ changes suddenly (Section VI-E).

C. DDoS Attack to the VNF

Our solution focuses on mitigating real IP-based DDoS attacks such as attacks from botnets. It limits the request volume to the server through a priority-based flow dispatching algorithm implemented on a virtual network function. A sufficiently intense DDOS attack may be able to overwhelm the VNF or even the enterprise gateway. We realize the limitations of our solution where an extremely high intensity DDoS attack can still impact the system since packets will drop at the gateway or the VNF's buffer queue before they get served by the dispatcher. However, our solution successfully protects the application server by limiting requests to it, and guarantees the QoS of trusted sources.

VIII. Conclusion

In this paper, we propose VGuard, a dynamic traffic engineering solution based on prioritization and implemented as a virtual network function (VNF) to perform DDoS mitigation. Two methods for flow direction were proposed in the paper: the static method and the dynamic method. We evaluated the performance of both methods through simulation. Our results show that both methods can effectively provide satisfying service to trusted flows under DDoS attacks, and we discussed the advantages and disadvantages of both methods under different situations.

REFERENCES

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

[2] Captcha: Telling humans and computers apart automatically. http://www.captcha.net/.

[3] Europol takedown of ramnit botnet frees 3.2 million pcs from cybercriminals' grasp. https://nakedsecurity.sophos.com/2015/02/27/europol-takedown-of-ramnit-botnet-frees-3-2-million-pcs-from-cybercriminals-grasp.

[4] Largest cyber-attack in history hits pro-hong kong protest websites. http://www.ibtimes.co.uk/largest-cyber-attack-history-hits-pro-hong-kong-protest-websites-1475876.

[5] Largest ever ddos cyber attack hits us and european victims. http://www.ibtimes.co.uk/largest-ever-ddos-cyber-attack-hits-us-european-victims-1435973.

[6] T. Anderson, T. Roscoe, and D. Wetherall. Preventing internet denial-of-service with capabilities. *ACM SIGCOMM Computer Communication Review*, 34(1):39–44, 2004.

[7] K. Argyraki and D. R. Cheriton. Scalable network-layer defense against internet bandwidth-flooding attacks. *IEEE/ACM Transactions on Networking (TON)*, 17(4):1284–1297, 2009.

[8] J. Blendin, J. Ruckert, N. Leymann, G. Schyguda, and D. Hausheer. Software-defined network service chaining. In *Third European Workshop on Software Defined Networks*, 2014.

[9] Z. Bronstein, E. Roch, J. Xia, and A. Molkho. Uniform handling and abstractor of NFV hardware accelerators. *to be published in IEEE Network*, 29(3), May-June 2015.

[10] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE). RFC 2784 (Proposed Standard), Mar. 2000. Updated by RFC 2890.

[11] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing (bcp 38). http://tools.ietf.org/html/rfc2827.

[12] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. Network function virtualization: Challenges and opportunities for innovations. *Communications Magazine, IEEE*, 53(2):90–97, Feb 2015.

[13] A. John and T. Sivakumar. Ddos: Survey of traceback methods. *International Journal of Recent Trends in Engineering*, 1(2):241–245, 2009.

[14] X. Liu, X. Yang, and Y. Lu. To filter or to authorize: Network-layer dos defense against multimillion-node botnets. *ACM SIGCOMM Computer Communication Review*, 38(4):195–206, 2008.

[15] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu. Portcullis: protecting connection setup from denial-of-capability attacks. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 289–300. ACM, 2007.

[16] T. Peng, C. Leckie, and K. Ramamohanarao. Protection from distributed denial of service attacks using history-based ip filtering. In *Communications, 2003. ICC'03. IEEE International Conference on*, volume 1, pages 482–486. IEEE, 2003.

[17] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), Jan. 2001. Updated by RFCs 6178, 6790.

[18] G. Tenebro. The bredolab files. *Symantec Corporation*, 2009.

[19] G. Wang and T. Ng. The impact of virtualization on network performance of amazon ec2 data center. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, March 2010.

[20] H. Wang, C. Jin, and K. G. Shin. Defense against spoofed ip traffic using hop-count filtering. *IEEE/ACM Transactions on Networking (ToN)*, 15(1):40–53, 2007.

[21] L. Wei and C. Fung. A request prioritizing algorithm for controller dos attacks in software defined networks. In *IEEE International Conference on Communications (ICC 2015)*. IEEE, 20015.

[22] A. Yaar, A. Perrig, and D. Song. Pi: A path identification mechanism to defend against ddos attacks. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 93–107. IEEE, 2003.

[23] X. Yang, D. Wetherall, and T. Anderson. Tva: a dos-limiting network architecture. *Networking, IEEE/ACM Transactions on*, 16(6):1267–1280, 2008.