SDN-Guard: DoS Attacks Mitigation in SDN Networks

Lobna Dridi, Mohamed Faten Zhani
Department of Software and IT Engineering
École de Technologie Supérieure (ÉTS), Montreal, Quebec, Canada
email: lobna.dridi.1@ens.etsmtl.ca, mfzhani@etsmtl.ca

***Abstract*—Software Defined Networking (SDN) has recently emerged as a new networking technology offering an unprecedented programmability that allows network operators to dynamically configure and manage their infrastructures. The main idea of SDN is to move the control plane into a central controller that is in charge of taking all routing decisions in the network. However, despite all the advantages offered by this technology, Deny-of-Service (DoS) attacks are considered a major threat to such networks as they can easily overload the controller processing and communication capacity and flood switch CAM tables, resulting in a critical degradation of the overall network performance. To address this issue, we propose in this paper SDN-Guard, a novel scheme able to efficiently protect SDN networks against DoS attacks by dynamically (1) rerouting potential malicious traffic, (2) adjusting flow timeouts and (3) aggregating flow rules. Realistic experiments using Mininet show that the proposed solution succeeds in minimizing by up to 32% the impact of DoS attacks on the controller performance, switch memory usage and control plane bandwidth and thereby maintaining acceptable network performance during such attacks.**

***Keywords*-Software-Defined Networking (SDN); Security; IDS attacks;**

I. Introduction

Software Defined Networking (SDN) is a new paradigm transforming the way IT networking infrastructures are managed, controlled and configured. The SDN perspective relies on the separation of the control plane (i.e., the network intelligence) from the data plane (i.e., packet forwarding). The control plane comes then under the responsibility of a centralized controller that takes all flow forwarding decisions in the network. The communication between the two planes is achieved through the OpenFlow protocol specified by the Open Networking Foundation (ONF) [1].

Typically, in an OpenFlow-based SDN network, when a switch receives a new packet, it checks first its flow table (i.e., called also TCAM table) to determine the output port (Fig. 1). If the packet does not match an existing entry in the flow table, a packet-in message is transferred by the switch to the controller inquiring a new flow rule entry. The controller decides of the routing path and instructs all the involved switches with the rules to handle this new flow. Each flow entry is characterized by a hard timeout after which the switch automatically deletes the entry. Fig. 1 illustrates a typical SDN deployment and the described steps.

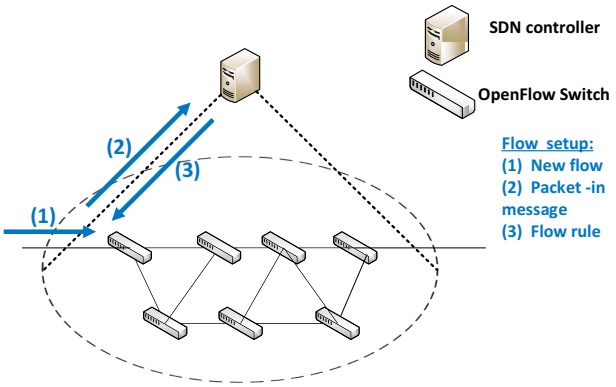

Fig. 1. Typical SDN deployment

As SDN technology is gaining momentum and several Internet providers are gradually embracing it, there is a growing attention to the security concerns that might arise when it comes to its real-world deployment [2]. In this context, Deny-of-Service Attacks (DoS) are the most popular and inevitable threats to SDN networks as they have always been to traditional networks. Usually, DoS attacks aim at overloading network links and the targeted servers by aggressively flooding them with the traffic until they fail to serve legitimate users.

Due to the centralized control of the SDN architecture, DoS attacks might have severe impacts on the network performance leading to cases where the entire control plane becomes completely crippled. More specifically, a DoS attack in an SDN network can lead to the following issues:

- Overloading the SDN controller: if the controller is overloaded, packet-in messages will be stuck in the controller's queue and no more routing decisions can be taken for the new incoming flows. In this case, flows with no forwarding entries will be stuck in the switches.
- Exhausting the control plane bandwidth (i.e., switch-to-controller bandwidth): this problem is tightly related to the previous one. However, in this case, the switch-to-controller links are congested,

978-1-5090-5093-2/16 $31.00 © 2016 IEEE
DOI 10.1109/CloudNet.2016.9

some packet-in messages might then be lost, which will delay the decision regarding the waiting flows.

- Switch TCAM memory overflow: DoS attacks can purposely create a large number of new flows that might saturate the flow forwarding tables of the switches. When this happens, switches are forced to constantly add and remove flow entries and to send more packet-in messages towards the controller.

In this paper, we propose SDN-Guard, a novel SDN application that protects SDN networks against DoS attacks and mitigate their impact on the SDN controller performance, the consumption of the control plane bandwidth and the switch TCAM usage. Unlike existing works, SDN-Guard is designed to mitigate simultaneously these issues by dynamically managing flow routes, rule entry timeouts and the aggregate flow rule entries based on the flow threat probability provided by an Intrusion Detection System (IDS).

The remaining of this paper is organized as follows. Section II provides an overview of previous works addressing DoS attacks in SDN networks. Section III presents the details of our proposed solution. Experimental results are then described in Section IV. We finally conclude in Section V.

II. Related Work

With the increasing adoption of SDN technology, a growing body of work is addressing its security issues and investigating how they can be mitigated. A survey on these issues is provided in [2]. However, in the following, we focus mainly on research efforts that have recently addressed DoS attacks in SDN networks [2], [6], [7], [3], [5], [4].

In this context, Shin et al. [6] and Kandoi et al. [7] have analyzed the impact of DoS attacks on the network performance and showed how such attacks may impact on several parameters like the control plane bandwidth (i.e., controller-switch channel), latency, switches flow tables and the controller performance. However, they do not provide any solution to address these issues.

L. Wei and C. Fung [3] proposed FlowRanger, a scheme that allows to detect and mitigate DoS attacks. FlowRanger is implemented at the controller side and consists of three components: (1) the trust management component that calculates a trust value for each packet-in message based on its source, (2) the queuing management component that places the message in the priority queue corresponding to its trust value and (3) the message scheduling component that process messages according to a weighted Round Robin strategy. FlowRanger can reduce the impact of DoS attacks on network performance by guaranteeing that legitimate flows are served first in the controller. However, unlike SDN-Guard, it does not prevent flooding the controller and switch CAM tables overload.

Dao et al. [4] present a solution to protect SDN networks against distributed DoS attacks based on IP filtering technique. The proposed scheme analyzes user behavior and uses it to assign the timeouts for the flow entries. Short timeout are assigned for malicious users flows and long timeouts are used for trusted ones. This solution forces entries of malicious traffic to be quickly removed from switches CAM tables. However, this may lead to new packet-in messages to be sent to the controller if the flow duration is higher than the set timeout. Furthermore, this solution drops all malicious traffic, which may be problematic for false positive flows.

Sahay et al. [5] leverage the centralized design and the programmability offered by SDN technology and propose a self-management scheme in which an ISP and its customers cooperate to mitigate DoS attacks. In this scheme, the ISP collects threat information provided by customers in order to use it to enforce security policies and update flow tables in the network accordingly. If a flow is considered legitimate by the customers, the ISP controller will tag it with a high priority value so that it takes a path with higher quality. However, if there is a doubt about the legitimacy of the flow, the ISP controller will assign a low priority to the flow and direct through the path designated for malicious flow. This proposal reduces the impact of the DoS attack on the network performance by balancing the load across different paths but it does not mitigate the risks of overloading the controller and flooding flow tables in the switches.

None of the aforementioned solutions is able to simultaneously reduce the controller load, the switch-to-controller bandwidth and also avoid flooding the CAM tables of the OpenFlow switches. In this work, we aim at achieving simultaneously these objectives in order to maintain an acceptable network performance during DoS attacks. Table I compares the proposed solution, SDN-Guard, to the existing ones with respect to their targeted performance objectives during DoS attacks.

III. Proposed solution

In this section, we present the design architecture of the proposed solution. We discuss its main components and major design rationales that were considered in order to achieve the targeted objectives.

A. Architecture Overview

As depicted in Fig. 2, SDN-Guard can be seen as an SDN application that can be plugged on top of any SDN controller. It consists of the following three modules:

- **Flow management module** is responsible for selecting the routing paths for each of the flows and deciding the hard timeout of their corresponding TCAM entries based on the threat probability of the flow to manage flows so as to mitigate the impact of the DoS attack.

- **Rule aggregation module** is in charge of aggregating flow entries of malicious traffic in order to reduce the number of entries used in the switches TCAMs.

- **Monitoring module** is responsible for permanently collecting multiple statistics about flows, switches and links (e.g., flow throughput, switch TCAM usage and link

TABLE I
SDN-GUARD VS. EXISTING SOLUTIONS

Approach	Objective: minimize the following metrics			
	Controller workload	Control plane bandwidth	Flow table usage	Network bandwidth usage
SDN-Guard	✓	✓	✓	✓
FlowRanger [3]	✓	✗	✗	✗
IP filtering approach [4]	✓	✓	✓	✗
Self-management scheme [5]	✗	✗	✗	✓

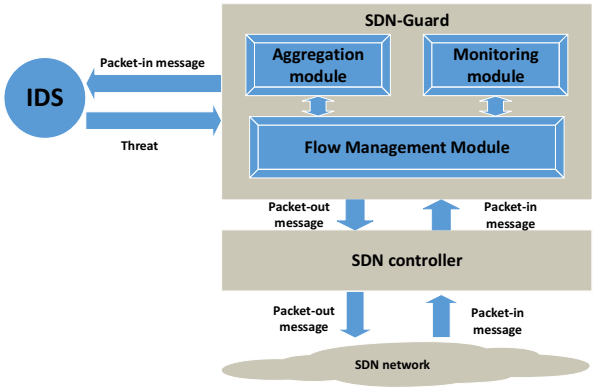

Fig. 2. Solution architecture

bandwidth usage) so that they can be used by other modules.

SDN-Guard constantly communicates with an Intrusion Detection System (IDS) that analyzes packet-in messages and informs SDN-Guard about the threat probability of each flow. It is worth noting that the IDS can be replaced by any other system able to accurately evaluate the flow threat probability such as the one used in [5]. It is worth noting that investigating the accuracy of such systems is out of the scope of this paper. However, studying the impact of the accuracy of the provided flow threat probability on SDN-Guard performance would be interesting and it is part of our future work.

B. Design Considerations

In order to mitigate the impact of DoS attack on the network and achieve the aforementioned objectives, the proposed solution is based on the following three design considerations:

1) Threat-based routing: In order to mitigate the impact of DoS attacks on bandwidth consumption and queuing delays, SDN-Guard redirects malicious traffic through the path having the least-utilized links in terms of bandwidth consumption and switch TCAMs. These two parameters are known to the flow management module thanks to the statistics collected by the monitoring module.

It is worth noting that the generated path using these parameters may not be the shortest path; however, it ensures a minimal impact of attack on the performance of legitimate flows. At the same time, malicious traffic will reach the destination (which is important in case of false positive malicious flows) where it can potentially be further analyzed by intrusion detection or prevention systems. We do not opt for dropping malicious traffic in order to make sure that false positive malicious flows get their chance to reach the destination, even with higher delays. As to the legitimate flows, they are always routed through the shortest paths between the source and the destination so as to ensure a minimal round trip time delays.

2) Timeout management: The flow management module assigns the timeout value of each of the flow rules according to the threat probability. As the switch will have to communicate with the controller whenever the hard timeout expires, a small hard timeout will result in much more communication traffic with the controller. This will not only increase the switch-to-controller bandwidth consumption but also overload the controller. Hence, if the incoming flow is considered malicious, SDN-Guard assigns its forwarding rules a high timeout. The rationale behind this decision is to ensure that the same flow does not trigger many communications between the switch and the controller.

3) Malicious flow rule aggregation: As malicious flows are assigned a large hard timeouts, such flow entries will remain for a long time in the TCAM table of switches. This might increase the number of used entries in the flow tables and might overload them. In order to address this issue, flow rules entries of malicious flows at a particular switch are automatically aggregated by the flow aggregation module if they have some shared properties (e.g., same source and destination) and forwarded to the same outgoing link.

C. Solution Design

When a switch receives a new flow that cannot be matched with any rule in its flow table, it inquires the controller for a rule in order to efficiently forward the flow to its destination. The packet-in messages are permanently sent to the IDS to analyze the flows and measure their threat probabilities. The threat probability is used by the flow management module to take a decision about the routing of each of the flows and the timeout for its corresponding entries in the switches' TCAMs. Two cases can be identified:

- Case 1: Malicious flow

TABLE II
FLOW MANAGEMENT DECISIONS

Flow type	Threat probability	Timeout	Path	Rule aggregation
Legitimate	low	default	shortest	optional
malicious	high	high	least-utilized links	mandatory

If the threat probability is above a predefined threshold, the flow is considered as malicious. In this case, the flow management module assigns a large hard timeout value to the flow rule and selects the least-utilized links in terms of bandwidth consumption and switch TCAMs to ensure that this flow does not compete with legitimate flows and impact their performance.

The aggregation module analyzes the generated rules for such malicious traffic and tries, when possible, to merge the rules in order to reduce their number, and thereby minimize the flow table usage.

- Case 2: Legitimate flow

When the flow threat probability is low, the flow is considered legitimate. The flow management module routes then the flow through the shortest path and assigns it a regular hard timeout value.

Table II summarizes the main decisions taken by SDN-Guard depending on the type of the received flow.

IV. EXPERIMENTAL RESULTS

In this section, we describe the experimental setup used to emulate an SDN network and the tools used to generate normal traffic and the DoS attacks. We then present and discuss the results obtained with and without SDN-Guard.

A. Experimental setup

We run our experiments on a server running Ubuntu 14.04 with a CPU Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz x 8 and 8 GB of RAM. To emulate the network topology, we used Mininet 2.3.0 [8] which creates a network of virtual hosts, switches, controllers, and links. OpenVSwitch (OVS) is used as the implementation for the OpenFlow switches [9].

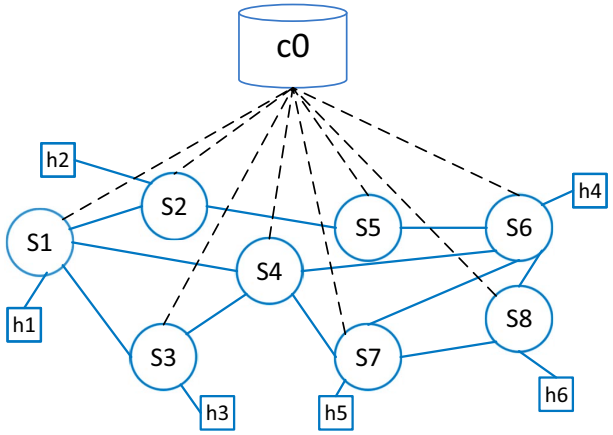

Fig. 3. Emulated topology

The created network is controlled by Floodlight 1.2 which is a widely-used java-based OpenFlow controller [10]. Furthermore, the communication between the switches and the Floodlight controller uses OpenFlow protocol version 1.3.

- **Emulated topology:** The emulated topology consists of six hosts and eight OpenFlow switches as shown in Fig. 3. Each switch is directly connected to the controller c_0 so that it can be instructed about forwarding decisions determined by SDN-Guard.

- **DoS attack scenario:** In our experiment, the host h_1 is considered as an attacker whereas the host h_6 is the server targeted by the DoS attack (Fig. 3). We first begin the experiment by sending normal traffic consisting of TCP flows from all sources to all destinations. The DoS attack starts afterward and lasts for ten minutes during which the server h_6 is flooded with a large number of new TCP flows. The traffic returns to a normal behavior later on and only a normal amount of TCP flows is sent to the targeted server. To launch the DoS attack, we send TCP traffic using *hping3* network tool [11] which floods the server h_6 with a large number of TCP SYN packets with different IP source addresses. Such traffic emulates a distributed DoS attack coming from multiple sources. In our experiments, we assume that the IDS accuracy in detecting flow threat probability is up to 70%, meaning that almost 30% of the malicious flows are wrongly considered legitimate.

B. Experimental results

In our study, we focus on the analysis of four parameters, namely: control incoming throughput, the average table size of the switches, the end-to-end throughput and the average Round Trip Time (RTT). To show the benefits of the proposed scheme, we run the same experiment two times: one time with the baseline routing algorithm and traffic management (i.e., without SDN-Guard) and another time with SDN-Guard activated.

- Controller incoming throughput:

To study the behavior of the controller during an attack, we analyze the throughput of packet-in messages received by the controller from all the switches in the network.
Fig. 4 shows the throughput of packet-in messages received by the floodlight controller requesting for new flow rules. It is clear that during the attack, there is a surge in the packet-in number received by the controller. However, compared with the baseline, we can see that SDN-Guard succeeds in reducing this throughput by up to 32%. This is mainly because SDN-Guard sets high hard timeouts to the forwarding rules associated with the malicious flow, and thereby significantly reduces the need to re-inquire the controller for new flow rules.

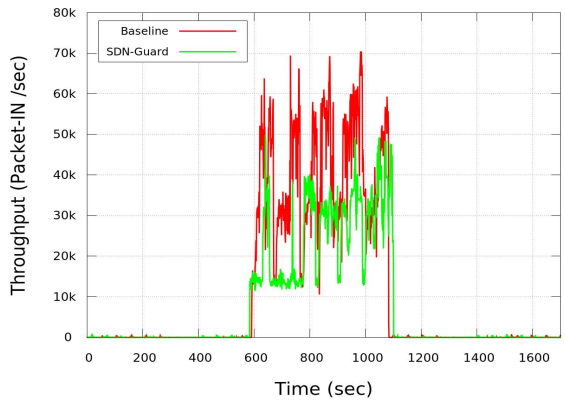

Fig. 4. Controller incoming throughput

- Average switch table size:

The switch table size can be easily flooded with DoS attacks due to the huge amount of new flows sent to switches, requiring a large number of flow rules to be stored.

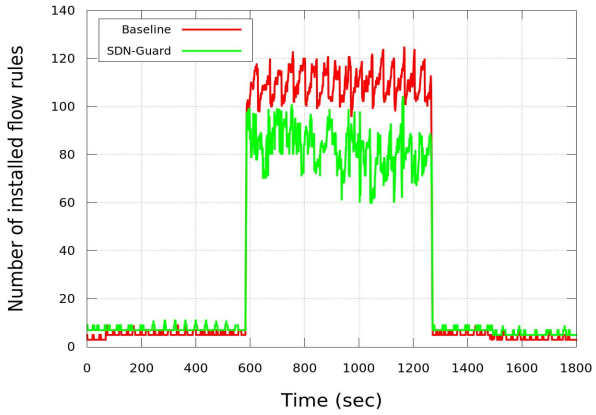

Fig. 5. Average table size of switch s_1

It is clearly shown in Fig. 5 that, with SDN-Guard, the number of flow rules in the table of the switch S_1 decreases by up to 26% compared to the baseline.
This is achieved because (i) the malicious traffic is forwarded through the least-utilized links in terms of bandwidth consumption and switch TCAMs, which means that flow entries will be inserted through switches of different paths (i.e., not only the switches of the shortest path), (ii) the aggregation module makes sure to minimize the number of flow entries of the malicious flow by aggregating them using common properties (e.g., same source and destination, same next hop). We also note that similar results were found for the other switches in the considered network.

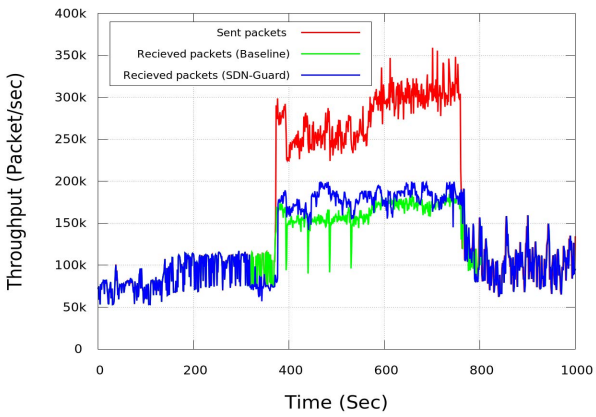

Fig. 6. Sent and received packet throughput

- Throughput from source to destination:

We also measure the source-to-destination throughput, which is the number of packets sent from the source h_1 and received at the destination h_6. Fig. 6 shows that, before the attack, there is almost no packet loss as the throughput of the sent packets is equal to the received one. However, during the attack, the received throughput is much less than the received showing a high loss rate. With SDN-Guard, the received throughput is relatively less affected by the attack and still higher than the baseline case (i.e., without SDN-Guard). By analyzing the results obtained during the attack, we find also that, with the baseline, there is 40% packet loss compared to 35% with our solution. This decrease in the packet loss is achieved by SDN-Guard because malicious traffic is balanced across the least-utilized links, which reduces congestion risks. Although this is not one of our main objectives but it can be considered as another benefit of SDN-Guard.

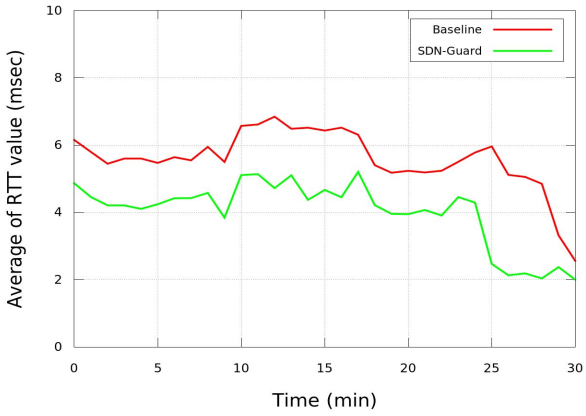

Fig. 7. Average RTT between h_1 and h_6

- Impact on average RTT:

Another important performance metric to be measured is the source-to-destination round trip time. We hence draw the average RTT value over time in order to study the impact of DoS attack on this parameter with and without using SDN-Guard. We can see from Fig. 7 that the average RTT value decreases by up to 23% when SDN-Guard is activated. This is because hard timeouts are set high for malicious traffic. Hence, the switches do not have to request new flow rules much often for the same flow. This eliminates the time needed to send the request to the controller and waiting the flow entry.

V. Conclusion

In this paper, we proposed SDN-Guard, a comprehensive SDN solution that is able to mitigate SDN-specific threats related to DoS attacks. Indeed, SDN-Guard is able to efficiently protect SDN networks against attacks by dynamically rerouting potential malicious traffic, adjusting flow timeouts and aggregating flow rules associated with the malicious traffic. The conducted experiments using Mininet showed that the SDN-Guard succeeds in minimizing the impact of DoS significantly reduces by up to 32% the controller incoming throughput and the control plane bandwidth and cuts down by up to 26% switch memory usage. Furthermore, we showed also that SDN-Guard reduces packet loss and average packet round trip time in the network during DoS attacks.

There are many promising directions we can pursue in the future. We are planning to evaluate the performance of the SDN-Guard for more realistic and larger-scale deployments. Another interesting avenue would be to further investigate the accuracy of intrusion detection systems in estimating flow threat probability and to study the impact of malicious flow accuracy on SDN-Guard performance.

References

[1] "Open Networking Foundation," https://www.opennetworking.org/, [Online; accessed 10-May-2016].

[2] S. Scott-Hayward and S. O'Callaghan, G.and Sezer, "SDN security: A survey," in *IEEE Conference on Network Softwarization (NetSoft)*, November 2013, pp. 1–7.

[3] L. Wei and C. Fung, "FlowRanger: A request prioritizing algorithm for controller DoS attacks in software defined networks," in *IEEE International Conference on Communications (ICC)*, 2015, pp. 5254–5259.

[4] N.-N. Dao, J. Park, M. Park, and S. Cho, "A feasible method to combat against DDoS attack in SDN network," in *International Conference on Information Networking (ICOIN)*, 2015, pp. 309–311.

[5] R. Sahay, G. Blanc, Z. Zhang, and H. Debar, "Towards autonomic DDoS mitigation using software defined networking," in *Network and Distributed System Security (NDSS) Symposium*, 2015.

[6] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, 2013, pp. 165–166.

[7] R. Kandoi and M. Antikainen, "Denial-of-service attacks in OpenFlow SDN networks," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 1322–1326.

[8] "Mininet Overview," http://mininet.org/overview/, [Online; accessed 26-May-2016].

[9] "OpenVswitch," http://openvswitch.org/, [Online; accessed 20-May-2016].

[10] "Floodlight controller," http://www.projectfloodlight.org/, [Online; accessed 10-March-2016].

[11] "Hping3: Network tools," http://linux.die.net/man/8/hping3/, [Online; accessed 10-May-2016].