Towards autonomic DDoS mitigation using Software Defined Networking

Rishikesh Sahay, Gregory Blanc, Zonghua Zhang, Hervé Debar

Towards Autonomic DDoS Mitigation using Software Defined Networking

Rishikesh Sahay*‡, Gregory Blanc*‡, Zonghua Zhang†‡ and Hervé Debar*‡
*Institut Mines-Télécom, Télécom SudParis
91011 Evry, France
Email: {rishikesh.sahay,gregory.blanc,herve.debar}@telecom-sudparis.eu
†Institut Mines-Télécom, Télécom Lille
59650 Villeneuve-d'Ascq, France
Email: zonghua.zhang@telecom-lille.fr
‡CNRS UMR 5157 SAMOVAR
91011 Evry, France

***Abstract*—Distributed Denial of Service attacks (DDoS) have remained as one of the most destructive attacks in the Internet for over two decades. Despite tremendous efforts on the design of DDoS defense strategies, few of them have been considered for widespread deployment due to strong design assumptions on the Internet infrastructure, prohibitive operational costs and complexity. Recently, the emergence of Software Defined Networking (SDN) has offered a solution to reduce network management complexity. It is also believed to facilitate security management thanks to its programmability. To explore the advantages of using SDN to mitigate DDoS attacks, we propose a distributed collaborative framework that allows the customers to request DDoS mitigation service from ISPs. Upon request, ISPs can change the label of the anomalous traffic and redirect them to security middleboxes, while attack detection and analysis modules are deployed at customer side, avoiding privacy leakage and other legal concerns. Our preliminary analysis demonstrates that SDN has promising potential to enable autonomic mitigation of DDoS attacks, as well as other large-scale attacks.**

I. Introduction

Distributed Denial of Service (DDoS) attacks have extensively studied for more than two decades, but recent years can still see a surge in their growth. In particular, flooding-based attacks, such as UDP, TCP SYN and ICMP floods dominate the growth, and the target of such voluminous attacks is to deplete computing resources like CPU, memory and network bandwidth of victims by sending an overwhelming number of bogus packets. For example, in TCP SYN flood [14], the attacker overwhelms the victim with SYN packets by exhausting the connection table, while UDP and ICMP attacks mainly consume the network bandwidth by sending forged packets. Recent years have also seen an increase in multi-vector DDoS attacks [1]. One example is that of a large UDP flood combined with a slow HTTP GET flood [12], misleading victims to cope with the seemingly anomalous UDP flood, while the HTTP flood can slowly deplete the HTTP server computing resources.

To cope with DDoS attacks, tremendous efforts have been made from both academia and industry [26], [39]. However, few of the existing DDoS mitigation techniques have been considered for widespread deployment, primarily because of their implementation and deployment complexities, as well as prohibitive operational costs. One of the major reasons is that they usually require large network connection state tables to be maintained at routers or switches, resulting in extra storage and computational burdens. Also, some techniques like packet marking [4], [29] require a huge amount of packets to be monitored and collected, incurring additional processing overhead. More importantly, the operation of those techniques rely on the deployment of additional modules or devices, increasing deployment complexity. Overall, such strong design and deployment assumptions indicate a lack of *autonomic* properties, causing non-trivial labor cost and response latency. Despite early efforts on designing autonomic DDoS response [15], [33], their scalability and operational costs are questionable in the face of large-scale deployment, mainly due to the intensive collaboration and communication between different detection modules that must be installed in advance. Additionally, although an anomaly detection framework proposed in [40] can preserve some autonomic properties, its usability and effectiveness on DDoS mitigation in conventional networks has not been experimentally validated.

Therefore, it is desirable to make DDoS mitigation automated, lightweight, scalable, and easy-to-manage. While it is a fact that many design challenges still remain in the community, the recent emergence and rapid development of Software-Defined Networking (SDN) offer us an opportunity to re-examine and improve anti-DDoS designs, thanks to the decoupling of networking control plane and data plane, as well as the controller's programmability [25]. Since SDN controller allows to obtain a global view of the network states and achieve centralized network forensics [3], we envision that the major functionalities of DDoS mitigation schemes can be similarly implemented at the SDN controllers, as the framework proposed in [30]. As such, human intervention will not be necessarily required to manage and maintain the

SENT '15, 8 February 2015, San Diego, CA, USA

http://dx.doi.org/10.14722/sent.2015.23004

DDoS mitigation schemes. Also, as mitigation functions can be abstracted and integrated at the application layer of SDN, installation of specific devices is no more a compelling need. Meanwhile, the computational overhead at the routers and switches can be significantly reduced, as large connection states or flow tables can be migrated and handled by the SDN controller. More interestingly, some security applications or APIs can be deployed at the SDN controller, allowing ISPs to provide DDoS mitigation as an on-demand service to their customers [2].

This paper reports some preliminary results of our ongoing project, which aims to develop an autonomic DDoS mitigation framework by using SDN technologies. We firstly present a critical analysis of the existing anti-DDoS schemes in terms of autonomic properties. Then, we propose a generic SDN-based mitigation framework, with detailed illustration on major functional components. We further exemplify the application of our framework through a use case, with an objective to examine the feasibility of our proposal design, i.e., the ISPs and their customers can effectively collaborate to mitigate DDoS attacks based on the real-time communication between DDoS monitoring, analysis, detection, and reaction modulars.

II. Key Observations and Motivation

Our work starts with an evaluation of existing anti-DDoS mechanisms, which cover the entire security life-cycle from prevention and detection to characterization to response. The schemes we have investigated can be clustered into four categories: capability-based, congestion control, policy-based and traceback mechanisms. In the following, we will briefly discuss the four categories and analyze their limitations.

DDoS prevention and detection. The capability-based techniques, such as the ones reported in [19], [37], [38], usually assume that the sender and the receiver may need to establish a privileged channel for communication: the sender asks for the capability token from the receiver, and if the receiver agrees to establish the connection, it sends a capability token to the sender, who then embeds this token in the subsequent packets. Clearly, capability-based systems can prevent DDoS attacks from happening, but network-wide infrastructure support is necessary. The system may fail to work in the presence of any networking device failure. Another preventive anti-DDoS approach is intelligent congestion control, which aims to limit traffic rates based on the given thresholds. One typical example is Pushback [13], which alerts upstream routers of the victim to drop the attack traffic based on the congestion signatures that are generated beforehand, while only legitimate traffic is permitted to flow through the network. However, such filtering schemes also require additional devices to be deployed at every routers in the network. The whole system may come to halt if one device at the downstream router fails, since the congestion signature can not be propagated to the upstream routers.

DDoS reaction or mitigation. The work reported in [20], [28] show that some stateful DDoS mitigation policies can be specified to redirect DDoS traffic to middleboxes on demand when the customer experiences an attack. In particular, the dFence model proposed in [20] requires middleboxes to be deployed in the core network, and both directions of IP traffic should be intercepted. In addition, IP traceback mechanisms [29] have also been proposed to mitigate DDoS attacks.

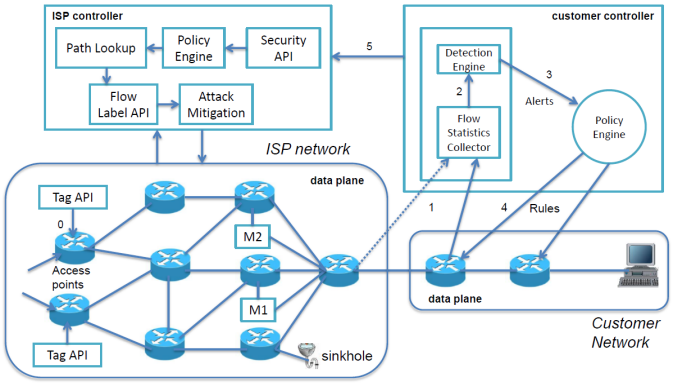

Fig. 1. Customer-oriented Collaborative DDoS Mitigation Framework: spanning from one customer network to (multiple) ISP networks.

The idea is to mark IP packets with some information like the router's ingress interface or its ID, which will then be used to reconstruct the path from the victim to the attack source. However, due to the fact that different paths may end up with the same Path ID [36], the false positive rate is quite high. Also, the generation and maintenance of marks introduce processing overhead at the routers, and the victim is required to collect a large number of packets to identify the attack path and initiate traceback process.

Our analysis shows that most of anti-DDoS mechanisms may perform well in terms of detection capability and mitigation efficiency, but the lack of *self-management* capability heavily deter their deployment and operation at large scale in practice. Thanks to the recent emergence of SDN paradigm, which is believed to be a promising solution to reduce the complexity of network management, we envision that DDoS mitigation schemes can be effectively implemented and deployed at SDN controllers, paving a way for ISPs and network customers to defend against DDoS attacks together by correlating and sharing the tasks of monitoring, analysis, detection and mitigation. While the widespread deployment of SDN technology in ISP networks is still at an early stage, we intend to investigate, demonstrate and verify the feasibility of using SDN to enhance DDoS mitigation.

III. Design Framework

A. Design assumptions and overview

Our framework is built on the following assumptions: (1) DDoS mitigation and traffic engineering are provided as an on-demand services to the customers, who need to subscribe to these services beforehand. That requires ISPs and customers to cooperate with each other to achieve DDoS mitigation on their agreements or willingness; (2) DDoS detection modules are running in the customer network and generate security alerts, so the customers can tailor their detection modules to particular attacks of concern; (3) both customer networks and ISPs have their own SDN controllers running and communicating in reliable and secure ways. As shown in Fig. 1, the framework is distributed across the ISP and customer networks, and the operational workflow (labeled with step No. in the figure) can be described as follows:

1) Traffic that enters in the ISP network is tagged by the access switches. Tag APIs runs at the access switches;
2) Flow statistics are periodically collected from SDN switches by a collector at the customer controller, OpenFlow exposes an API to obtain flow statistics, Statistics can be collected from both the customer and ISP networks, the latter requiring subscription to the service;
3) The anomaly detection engine take the flow statistics as input;
4) Threat alerts are generated in the presence of anomalous (either suspicious or malicious) traffic and passed to the policy engine, which in turn generates some policy rules corresponding to the alert;
5) The SDN controller of customer network enforces reaction polices at local routers;
6) Meanwhile, the anomalous flow information (actually a tag previously inserted at the ISP network) and corresponding countermeasure requests are sent by customer controller to ISP controller, which then changes the flow label information of the incoming flows of concern[1], and instructs access routers to label the relevant packets (identified by the tag provided by the customer – the generation of tags is detailed in Algorithm 2).
7) As a result of previous step, the identified suspicious or malicious traffic will be redirected to the corresponding middlebox (a mitigation example is given in Algorithm 1).

B. Major Framework Components

OpenFlow Switch. According to the OpenFlow specifications in [24], OpenFlow-enabled switches maintain flow tables to perform packet lookups and forwarding. Basically, flow entries consist of match fields, counters, and actions to be applied to the matching flows. Upon reception of the flow, OpenFlow switches perform a lookup operation in the flow table: if it does not have the entry for that flow then the flow information is forwarded to the controller.

Middlebox. Middleboxes are devices enforcing the security policies in order to mitigate attacks. In our framework, we have assumed that middleboxes may store and enforce different kinds of security policies that tackle different classes of DDoS attacks. Instead of versatile middleboxes, we may also deploy specialized ones that address a single class of attacks, therefore enforcing a single policy.

Monitoring Plane. It consists of the two following modules:

- *Flow Statistics Collector*, which collects the flow information from OpenFlow switches and forwards it to the *detection engine*. OpenFlow switches maintain counters for each flow table and flow entry. The customer controller can periodically requests to collect flow statistics of interest from the switches, to that end, some flow statistics collection techniques have already been experimented with SDN technology in [6], [21].
- *Detection Engine*, which takes the flow statistics from the *collector* as inputs and generates security alerts when anomalous flows are identified. The alerts then trigger the policy engine for incoming flows to be processed accordingly. In our mitigation technique it is out of scope to propose a flow statistics based detection method. For the flow statistics collection, we rely on some past proposal [10] which shows that OpenFlow and sFlow can be used for the collection of flow statistics and detection of anomalies in the traffic.

Policy Engine. On reception of an alert from the *detection engine*, it generates some rules to address the anomalous flow that has been identified. These rules are then stored in a lookup table for later enforcement. Finally, the controller deploys the rules to OpenFlow switches.

Security APIs. The framework allows ISPs to provide security functions to the customer through APIs at the controller, enabling on demand security as a service. Requests include the deployment of a middlebox to filter suspicious traffic, or blocking malicious traffic. The customer can also assign priority to the flows through these APIs, to provide preferential treatment to the legitimate traffic. The security services are only available to subscribed customers. This kind of security APIs have already been proposed in [2], [18]. This is also being currently discussed at the IETF under the non working group I2NSF [9] which aims at standardizing these security functions, prompting them to be available for widespread use in coming years.

Path lookup. Our framework also assumes that paths are pre-computed by the ISP. Similar to ETHANE [7], paths can be computed using an all-pairs shortest path algorithm [8]. If a link fails then the paths can be computed again. The path lookup component maintains a table of paths (from the ingress switch to the egress switch) sorted according to the quality of service provided by the paths, associated to unique labels. Paths are later assigned to flows based on the traffic class that they belong to [11]. For example, legitimate flows are assigned to high priority paths while suspicious flows are assigned to paths containing middleboxes (possibly longer in terms of hops). Finally, malicious flows are forwarded through paths which lead to a sinkhole. In our framework, the path lookup module takes inputs from the policy engine (required level of QoS and policy rules) and returns the paths that match.

Flow Label API. As aforementioned, flows which are not present in the flow table of the access switches are forwarded to the controller using the `PACKET-IN` message as described in the OpenFlow specification in [24], and they are installed in the flow table using the `FLOW-MOD` message according to the controller's centralized network policy. Leveraging the network end-to-end visibility provided by the SDN controller, flows are assigned a label which sets a path for the flow, from the ingress switch to the egress switch. The purpose of using the label is for fast switching and rerouting, as the switches can simply check the label and forward the flow to the next hop, instead of parsing the whole packet header. Additionally, it alleviates the load on the OpenFlow switches (except for ingress switches) by reducing the number of entries in their

[1]Note that when the flows arrive at the access routers in the ISP network, the routers will check the flow entry in their flow table. If the flow information does not exist, it will be forwarded to the ISP controller, which then assigns the label according to the policy in the network thanks to its end-to-end network visibility and centralized network policy.

flow table, to the label entries. In practice, the label can be assigned using the OpenFlow's `Push` action, rewriting the 12-bit VLAN ID field [24].

Attack Mitigation. Based on the pre-computed path associated to the anomalous flow, this module deploys middleboxes at given points in this path(identified by the label produced by the Flow Label API and then attached to the anomalous flow) before the anomalous traffic reaches the customer network. Based on the tags provided by the customer controller's detection engine, the ISP controller modifies the labels for the relevant flow entries, in order for these flows to be processed by middleboxes. An example of a mitigation algorithm is given in Alg. 1.

Algorithm 1 $Mitigation(alert) \rightarrow Action$

if alert.level is malicious **then**
  $controller.FlowTable \leftarrow modLabel(tag, malicious\_label)$
  return $Action(tag, Drop())$ // the drop policy is returned for the malicious tag
end if
if alert.level is suspicious **then**
  $controller.FlowTable \leftarrow modLabel(tag, suspicious\_label)$
  return $Action(tag, [Fwd(middlebox), Fwd(customer)])$
  // a set of forwarding policies is returned for the suspicious tag
  // and flow packets processed by the middlebox will be later forwarded to the customer network
end if

Tag API. This module is used to generate a unique hash tag. This action is performed at the access switches. This API will be provided as an application to be deployed at the switches using the "configuration apply" command. This API extracts the packet header, and uses the IP-4 tuple (source IP, destination IP, source port, destination port) as input and generates a unique tag to be inserted in the packet. This tag number is inserted in the source MAC address field using the `Push Tag` action [24]. A tag generation algorithm is given in Alg. 2. This algorithm should be deterministic, outputting a unique tag for a given flow. The tag helps to enforce a consistent end-to-end network policy, and it also provides a fine granularity to identify flows in a quick way. The tagging function is performed at the access switches(edge switch). The flows arriving at the access switches(edge switches) are tagged by the access switches itself. As proposed in the [5], our tagging function logic is encoded at the access switches(edge switches) to minimize the overhead at the controller.

IV. Use Case

The applicability of our technique is shown through an example described in Fig. 2. The scenario, although simple, involves all the components described in Section III-B and provides a full cycle of the framework's workflow. The components in the use case are switches `S1`, `S2`, `S3`, `S4`, and a middlebox, `M1`, attached to the switch `S3`. The switches `S2` and `S4` are egress switches in the ISP network forwarding the flows to the customer network. Let us consider two external hosts, one legitimate (denoted as `L`) and one malicious (`A`), that communicate with a host (`C`) at the customer network. This customer network is a client of the ISP on which Fig. 2 is centered. Both networks have deployed our proposed scheme. For the sake of brevity, we have omitted the controllers and their communication within the figure. We will focus on the mitigation process at the ISP network instead.

Algorithm 2 $TagAPI(packet) \rightarrow packet$

for each incoming packet p **do**
  $f \leftarrow Flow(p.IP_{src}, p.Port_{src}, p.IP_{dst}, p.Port_{dst})$
  if f is in switch.FlowTable **then**
    $p.VLANID \leftarrow label$
    $p.SourceMAC \leftarrow tag$
  else
    $PacketIN()$ // new flow is forwarded to the controller
    $new\_tag \leftarrow hash(f)$ // a new hash is generated at the access switch based on flow information
    $p.SourceMAC \leftarrow new\_tag$ // the new tag is inserted in the Source MAC field
    $p.VLANID \leftarrow controller.GenerateLabel(f)$ // the controller assigns a label to the new flow
    $FlowMod(f, new\_tag, p.VLANID)$ //flow is installed in the flow table of access switch
    $Send(new\_tag, controller)$ // Tag is forwarded to Controller
  end if
end for

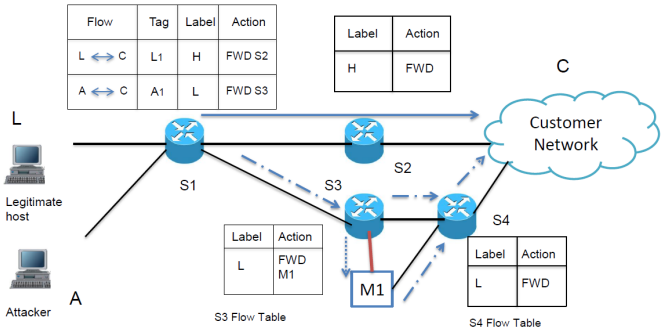

Fig. 2. An example scenario illustrating the application of Labels and Tags: action *FWD* means packet/flow forwarding.

The flows generated from both external hosts to host `C` have been installed in the flow table of the ingress switch `S1` the first time they have entered the ISP network. Following Alg. 2, `S1` has generated a tag (L_1 and A_1 for `L` and `A`, respectively) i.e, the hash computed over their IP 4-tuple and forwarded this tag and flow information to its controller. The controller, upon receiving new flows for which it does not have policy rules available, considers them as *legitimate* and attaches a label `H` to them, prompting switches to forward these flows' packets through a high quality path. This high quality path is denoted with a full arrow from `S1` to the customer network, and through switch `S2`. Flow table information of the switches `S1`, `S2`, `S3`, and `S4` are also maintained at the controller. The label information is communicated to `S1` by the controller. Finally, `S1` inserts the tag in the source MAC field, and the label in the VLAN ID field of the packet before forwarding it according to its path policy, that is the path computed for the assigned label.

As indicated in Fig. 2, the flow table of `S1` has already been modified, with the label assigned to `A` being now `L`, which indicates that it will be forwarded along a low quality path, usually reserved to suspicious flows. This is the consequence of an alert raised by the customer network for flows destined

to C and coming from A. The alert was issued by the customer network controller indicating to the ISP controller to treat tag A_1 as a *suspicious* flow. Upon receiving the alert, the ISP controller has modified its flow table, changing the label assigned to tag A_1 from H to L, prompting an update of the flow table of S1. Now, any packet from flow A_1 is switched based on label L, i.e., following a low quality path (represented using the dotted arrows in Fig. 2).

Switching the flow packets is done upon labels as described in Section III-B, and is possible, thanks to the OpenFlow ability to designate match fields in the packet's header. In our example, the actions in the flow tables of switches S1, S2, S3 and S4 are computed according to the *Policy engine* at the ISP's controller. Therefore, the suspicious flow A_1 is forwarded to S3 for further processing, according to its flow table. At S3, the *Attack Mitigation* component has deployed a middlebox (identified as M1) to clean the traffic. According to S3's flow table, packets labeled as L must be forwarded to M1. Note that other packets bearing different labels may reach S3 and hence be forwarded directly to S4, as indicated in Fig. 2. Once the traffic is cleaned through M1, packets are forwarded to S4, which in turn will forward them to the customer network. Meanwhile the traffic marked as H is directly sent to the customer network with a high quality of service, preserving legitimate hosts' experience, throughout the attack mitigation process. Additionally, when suspicious flows are processed by one or several middleboxes, these middleboxes may alter the packet header fields, which usually violate the security policy for these flows, as subsequent middleboxes may not be able to match the packets based on flow information. However, the tags we have proposed in this framework also help in correlating packets belonging to a given flow in a way that remains consistent throughout the network. The mitigation process stops upon request from the customer.

V. Discussions

The previous design descriptions and use case demonstrate that our proposed scheme preserve self-management properties that make it possible to achieve autonomic DDoS mitigation. In general, the SDN controllers make reactions on an event basis and dynamically adapt security policies to handle suspicious and malicious flows. Then the policy changes will eventually lead to the automated configuration of the OpenFlow switches. Also, the end-to-end visibility yielded at the controller allows to optimize the deployment of middleboxes and the computation of flow paths with different QoS levels. In particular, the path computation modular inherently provides failover when a link or a switch fails.

More specifically, the usage of tags and labels make it possible to achieve flexible and consistent packet switching: the labels convey the class of traffic being processed, aggregating different flows (or tags) under a single identifier which can be quickly switched throughout the network. Thanks to the global view of the network obtained by the SDN controller, label and action information can be precisely and quickly forwarded to the appropriate switches in order to rapidly modify the policy for a given tag.

In addition, the flow processing capability and scalability deserve careful consideration. In our framework, the SDN controller can install the rules in the OpenFlow switches on demand and can label the packets with VLAN-ID field without incurring too much overhead. Also, the migration of tagging function to access switches can reduce the processing overhead of SDN controller, making it more scalable. In fact, some existing work has shown that one single SDN controller is sufficiently scalable to handle huge amount of traffic, e.g., the NOX-MT and Beacon controllers can handle $50,000$ new flow requests per second, and the processing capability can be improved to handle $1.6M$ new flow requests per second with average response time of 2 milliseconds on a eight core machine [22].

VI. Related work and Open Issues

It has been recognized that the decoupling of data plane and control plane makes SDN as a promising solution to enable customizable security services [30] and to deal with DDoS attacks, few solid SDN-based solution has been proposed so far. Thanks to some early efforts on developing security functions and APIs, I2NSF [9] network group is undertaking their standardization. For instance, in [2], the authors assume that ISPs provide a security API to the customers to request for traffic filtering and rerouting. Then a victim can request multiple ISPs to trace back the attack, identify the attack source, and send commands to ISPs to mitigate the attacks. But no specific mitigation techniques are discussed in this architecture. In the prototype Drawbridge [18], the customers can subscribe to traffic engineering services provided by ISPs, based on the assumption that the customer's controller communicates with the ISP's controller which then enforces the rules to the SDN switches deployed at the ISP network. In this case, the ISP needs to share the its policy enforcement point(PEP) with the customer. Also, Brocade proposes a proprietary solution [17], which provides a web based user interface to the customer to request for traffic filtering: when the customer network experiences the attack, the packets of concern are simply dropped based on a threshold value. As this is a proprietary solution, the technique details about this proposal are unclear. In CenterTrack [34], traffic of interest is rerouted to some special tracking routers, which determine the ingress edge routers through which packets arrived in the network. The main drawback of this technique is the need to deploy special tracking routers. In NetFuse [35], a proxy device is deployed between the switches and the controller. The proxy monitors the network load, and instructs switches to reroute any overloading flows to NetFuse devices. In this case, NetFuse devices may be overloaded by the attacker. In our proposal, middleboxes are deployed by the ISP's controller only upon the customer's request, i.e., when a suspicious flow has been detected.

As the research on software defined networking is surging, many open issues arise in both networking and security domains. In particular, SDN controller, by design, serves as single point of failure, potentially attracting many attacks [16]. Its protection is therefore very challenging due to the broad attack surface, ranging from the southbound interface between controller and SDN switches to the vulnerable applications or services running at the application layer, to security policies enforced to the centralized controller through northbound APIs. OpenFlow [23] supports authentication using certificates and encryption to secure the connection between controller and

switches. To protect the SDN infrastructure from the attack through northbound APIs, the authentication and verification of security policies and rules is necessary before enforcing them to the networking devices in the data plane. In Rosemary [31], a robust and secure SDN controller has been designed. The main objective of the Rosemary's is to prevent applications from executing in such a way that will corrupt the SDN controller. The Rosemary controller can handle more than 10 million flow requests per second. In AVANT GUARD [32], some intelligence is added to the data plane to address the issue of DoS attacks between the data plane and control plane. The adversary could target our mitigation approach by implementing the traffic stream in such a way to cause the client to saturate the controller of the ISP with flow labeling logic. This issue can be addressed to some extent by providing the services to those clients who are subscribed to the mitigation services. Another important issue deals with conflicting rules [27], especially when multiple controllers are deployed in the same network. Although the current framework assumes one controller at the customer side and one controller at the ISP side (or one-to-one controller mode), it can be hopefully extended to one-to-many mode, i.e., one customer requests multiple ISPs to mitigate the attack, by appropriately resolving controller scalability and rule conflicts issues.

VII. Conclusion and Future work

This paper reported our ongoing effort on developing an autonomic DDoS mitigation mechanism by leveraging SDN paradigm to provide on-demand DDoS mitigation services to ISP network customers, ultimately allowing the ISPs and their customers to collaboratively thwart DDoS attacks. In particular, we demonstrated that SDN controller may facilitate ISPs to deploy appropriate DDoS mitigation techniques, e.g., differentiating legitimate traffic and redirecting suspicious traffic to pre-deployed middleboxes, based on the threat information provided by the customers and their particular concerns.

Our future work will be focused on enriching the framework, improving and implementing its major components: (1) we will further study the effectiveness of our DDoS mitigation framework with focuses on its scalability on handling a large number of mitigation requests from multiple customers, as well as the case that one customer sends mitigation requests to multiple ISPs. The response latency on redirecting suspicious traffic to the middlebox will be also studied; (2) we will continue improving the efficiency of traffic tagging and labeling techniques, as well as the fast deployment of tags-driven mitigation middleboxes; (3) we will develop prototypes and implement our schemes via testbed for performance evaluation.

Acknowledgment

This research has been supported by the Strategic International Collaborative R&D Promotion Project of the Ministry of Internal Affairs and Communication, Japan, and by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 608533 (NECOMA). The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the Ministry of Internal Affairs and Communications, Japan, or of the European Commission.

References

[1] Akamai, "Prolexic Quarterly Global DDoS Attack Report Q1 2014," Prolexic, Tech. Rep., 2014.

[2] A. Alwabel, M. Yu, Y. Zhang, and J. Mirkovic, "SENSS: Observe and Control Your Own Traffic in the Internet," in *Proceedings of the 2014 ACM Conference on SIGCOMM*. New York, NY, USA: ACM, 2014, pp. 349–350.

[3] A. Bates, K. Butler, A. Haeberlen, M. Sherr, and W. Zhou, "Let SDN Be Your Eyes: Secure Forensics in Data Center Networks," in *Proceedings of the NDSS Workshop on Security of Emerging Network Technologies (SENT)*, Feb. 2014.

[4] A. Belenky and N. Ansari, "On Deterministic Packet Marking," *Comput. Netw.*, vol. 51, no. 10, pp. 2677–2700, Jul. 2007.

[5] R. Bifulco and G. Karame, "Towards a richer set of services in software-defined networks," in *Proceedings of the NDSS Workshop on Security of Emerging Technologies (SENT)*, 2014.

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

[7] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking Control of the Enterprise," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 1–12, Aug. 2007.

[8] C. Demetrescu and G. F. Italiano, "A new approach to dynamic all pairs shortest paths," in *35th Annual ACM Symposium on Theory of Computing (STOC)*, 2003, pp. 159–166.

[9] L. Dunbar, M. Zarny, C. Jacquenet, and S. Chakrabarty, "Interface to Network Security Functions Problem Statement," Working Draft, IETF, Internet-Draft dunbar-i2nsf-problem-statement-01, September 2014. [Online]. Available: http://www.ietf.org/internet-drafts/draft-dunbar-i2nsf-problem-statement-01.txt

[10] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments," *Computer Networks*, vol. 62, no. 0, pp. 122 – 136, 2014.

[11] N. Hachem, H. Debar, and J. Garcia-Alfaro, "HADEGA: A novel MPLS-based mitigation solution to handle network attacks," in *31st IEEE International Performance Computing and Communications Conference (IPCCC)*, Dec 2012, pp. 171–180.

[12] R. Hansen, J. Kinsella, and H. Gonzalez, "Slowloris HTTP DoS," 2009.

[13] J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2002.

[14] M.-S. Kim, H.-J. Kong, S.-C. Hong, S.-H. Chung, and J. Hong, "A flow-based method for abnormal network traffic detection," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, vol. 1, Apr. 2004, pp. 599–612 Vol.1.

[15] G. N. Koutepas, F. Stamatelopoulos, and V. Maglaris, "Distributed Management Architecture for Cooperative Detection and Reaction to DDoS Attacks," *Journal of Network and Systems Management*, vol. 12, pp. 73–94, 2004.

[16] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 55–60.

[17] R. Krishnan and M. Durrani, "Real-time SDN Analytics for DDoS mitigation," 2014.

[18] J. Li, S. Berg, M. Zhang, P. Reiher, and T. Wei, "Drawbridge: Software-defined DDoS-resistant Traffic Engineering," in *Proceedings of the 2014 ACM Conference on SIGCOMM*. New York, NY, USA: ACM, 2014, pp. 591–592.

[19] X. Liu, X. Yang, and Y. Xia, "NetFence: preventing internet denial of service from inside out," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, Aug. 2010.

[20] A. Mahimkar, J. Dange, V. Shmatikov, H. Vin, and Y. Zhang, "dFence: Transparent Network-based Denial of Service Mitigation," in *Proceedings of the 4th USENIX Conference on Networked Systems Design Implementation (NSDI)*. Berkeley, CA, USA: USENIX Association, 2007, pp. 24–24.

[21] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting Traffic Anomaly Detection Using Software Defined Networking," in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, R. Sommer, D. Balzarotti, and G. Maier, Eds. Springer Berlin Heidelberg, 2011, vol. 6961, pp. 161–180.

[22] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[23] Open Networking Foundation, "SDN Security Considerations in the Data Center," ONF, Tech. Rep., 2012.

[24] ——, "OpenFlow Switch Specification Version 1.4.0," ONF, Tech. Rep., 2013.

[25] ——, "SDN Architecture Overview," ONF, Tech. Rep., 2013.

[26] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems," *ACM Comput. Surv.*, vol. 39, no. 1, Apr. 2007. [Online]. Available: http://doi.acm.org/10.1145/1216370.1216373

[27] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for openflow networks," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 121–126.

[28] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying Middlebox Policy Enforcement Using SDN," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 27–38, Aug. 2013.

[29] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 295–306, Aug. 2000.

[30] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson, "FRESCO: Modular Composable Security Services for Software-Defined Networks," in *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS)*, 2013.

[31] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, "Rosemary: A Robust, Secure, and High-performance Network Operating System," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2014, pp. 78–89.

[32] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-defined Networks," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. New York, NY, USA: ACM, 2013, pp. 413–424.

[33] D. Sterne, K. Djahandari, B. Wilson, B. Babson, D. Schnackenberg, H. Holliday, and T. Reid, "Autonomic Response to Distributed Denial of Service Attacks," in *Recent Advances in Intrusion Detection*, ser. LNCS. Springer Berlin Heidelberg, 2001, vol. 2212.

[34] R. Stone, "Centertrack: An ip overlay network for tracking dos floods," in *Proceedings of the 9th Conference on USENIX Security Symposium - Volume 9*, ser. SSYM'00, 2000, pp. 15–15.

[35] Y. Wang, Y. Zhang, V. Singh, C. Lumezanu, and G. Jiang, "Netfuse: Short-circuiting traffic surges in the cloud," in *Communications (ICC), 2013 IEEE International Conference on*, June 2013, pp. 3514–3518.

[36] 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*, May 2003, pp. 93–107.

[37] ——, "SIFF: a stateless Internet flow filter to mitigate DDoS flooding attacks," in *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, May 2004, pp. 130–143.

[38] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting Network Architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 241–252, Aug. 2005.

[39] S. T. Zargar, J. Joshi, and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[40] Z. Zhang, F. Naït-Abdesselam, P. Ho, and Y. Kadobayashi, "Toward cost-sensitive self-optimizing anomaly detection and response in autonomic networks," *Computers & Security*, vol. 30, no. 6-7, pp. 525–537, 2011.